Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Розробка навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ1911: _____ _____ /Дмитро КАНАРЕЙКІН/
(посада) (підпис)

Керівник: _____ _____ /Вадим АНДРЮЩЕНКО/
(посада) (підпис)

Нормоконтролер: _____ _____ /Світлана ВОЛКОВА/
(посада) (підпис)

Засвідчую, що у цій роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Bachelor's Thesis
bachelor

on the topic: «Development of educational software to demonstrate the use of the AutoHotKey language in working with the Windows graphical interface»

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ1911: ___ _____ /Dmytro KANEREIKIN/
(посада) (підпис)

Scientific Supervisor: ___ _____ /Vadym ANDRIUSHCHENKO /
(посада) (підпис)

Normative controller: ___ _____ / Svitlana VOLKOVA/
(посада) (підпис)

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: бакалавр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ червня 2023 р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Бакалавр_____

студенту Канарейкіну Дмитру Олександровичу_____

1. Тема дипломної роботи: Розробка навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом.
Керівник роботи: Андрющенко Вадим Олександрович
затверджені наказом 1196 ст від 05.12. 2022 року

2. Строк подання студентом роботи 19.06.2023 року

3. Вихідні дані до дипломної роботи:

4. Зміст пояснювальної записки (перелік питань до розробки):
   4.1. Аналіз предметної галузі;
   4.2. Огляд програмної реалізації;
   4.3. Огляд застосунку;
   4.4. Тестування;

5. Перелік демонстраційного матеріалу:

5.1. доповідь;

5.2. презентація;

5.3. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Постановка задачі	01.03.23 – 05.03.23	10%
2	Огляд літератури	05.03.23 – 25.03.23	15%
3	Визначення вимог до програми	25.03.23 – 01.04.23	28%
4	Технічне завдання	01.04.23 – 05.04.23	34%
5	Узгодження та затвердження ПЗ	05.04.23 – 10.04.23	48%
6	Розробка та програмування логіки програми	10.04.23 – 01.05.23	66%
7	Відлагодження програми	01.05.23 – 05.05.23	75%
8	Розробка демонстраційних матеріалів	06.05.23 – 10.06.23	89%
9	Подання кваліфікаційної роботи до кафедри	19.06.23	99%
10	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	26.06.23	100%

Студент _____ /Дмитро КАНАРЕЙКІН /

Керівник роботи _____ /Вадим АНДРЮЩЕНКО /

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра:

X с., 10 рис., 2 дод., 15 джерела.

Предмет розробки –програмне забезпечення, що дозволяє користувачам ефективно вивчати та практикувати використання мови AutoHotKey для автоматизації та взаємодії з графічним інтерфейсом.

Мета роботи – метою даної роботи є створення навчального програмного забезпечення, яке надає користувачам зручну та ефективну платформу для вивчення та демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом. Розроблене програмне забезпечення надає користувачам можливість швидкого доступу та взаємодії з різноманітними елементами графічного інтерфейсу за допомогою скриптів AutoHotKey.

Галузь застосування – розроблене навчальне програмне забезпечення розширює доступ до знань та навичок, пов'язаних з роботою з графічними інтерфейсами. Воно надає користувачам, які бажають набути навичок використання мови AutoHotKey.

Методи дослідження – дослідження включає аналіз сучасних методів автоматизації графічних інтерфейсів, вивчення різних інструментів і технологій програмування, а також проектування, розробку та тестування навчального програмного забезпечення.

Ключові слова: АВТОМАТИЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ, ГРАФІЧНИЙ ІНТЕРФЕЙС, МОВИ ПРОГРАМУВАННЯ, НАВЧАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, AUTOHOTKEY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

Автоматизація – процес автоматизації повторюваних завдань або операцій за допомогою програмного забезпечення або скриптів, що зменшує ручну працю та підвищує ефективність;

API – Application Programming Interface – інтерфейс прикладного програмування – набір протоколів та інструментів, які дозволяють різним програмним додаткам спілкуватися та обмінюватися інформацією один з одним;

AutoHotKey – мова сценаріїв, що використовується для автоматизації та налаштування завдань в операційних системах Windows, зокрема в роботі з графічними інтерфейсами;

GUI – Graphical User Interface – графічний інтерфейс – візуальний інтерфейс, який дозволяє користувачам взаємодіяти з програмними додатками за допомогою графічних елементів, таких як кнопки, меню та вікна;

IDE – Integrated Development Environment – інтегроване середовище розробки – програмне забезпечення, яке надає комплексні інструменти та функції для розробки програмного забезпечення, включаючи редагування коду, налагодження та тестування;

Мова сценаріїв – мова програмування, спеціально розроблена для написання сценаріїв або автоматизованих завдань. Зазвичай має спрощений синтаксис і орієнтована на швидку розробку та автоматизацію;

Навчальне програмне забезпечення – програмне забезпечення, призначене для надання інструкцій, керівництва та практики в певній навичці або предметі;

Скрипт – набір інструкцій або команд, написаних у програмі;

Гарячі клавіші – комбінації клавіш, які при натисканні викликають певні дії або команди.

ВСТУП

У сучасну цифрову епоху графічні інтерфейси користувача (GUI) стали невід'ємною частиною програмних додатків, надаючи користувачам візуальний та інтерактивний спосіб взаємодії з технологіями. Вміння ефективно працювати з графічними інтерфейсами та автоматизувати завдання в них набуває все більшого значення в різних сферах. Метою даної кваліфікаційної роботи є розробка навчального програмного забезпечення, яке демонструє практичне застосування мови AutoHotKey в роботі з графічними інтерфейсами, надаючи можливість користувачам покращити свої навички автоматизації роботи з графічними інтерфейсами. Розробка цього навчального програмного забезпечення зумовлена зростаючим попитом на досвід автоматизації графічних інтерфейсів. Багато організацій у різних галузях визнають цінність автоматизації повторюваних завдань, оптимізації процесів і підвищення загальної ефективності. Надаючи користувачам комплексний навчальний ресурс, це навчальне програмне забезпечення дозволить їм набути необхідних навичок для ефективної навігації та маніпулювання графічними інтерфейсами за допомогою мови AutoHotKey.

Однією з ключових переваг використання AutoHotKey є її універсальність і гнучкість в автоматизації взаємодії з графічними інтерфейсами. Завдяки інтуїтивно зрозумілій мові сценаріїв AutoHotKey дозволяє користувачам створювати власні сценарії автоматизації, які можуть виконувати широкий спектр завдань, від простих до складних. Використовуючи можливості AutoHotKey, користувачі можуть автоматизувати повторювані дії, взаємодіяти з елементами графічного інтерфейсу і значно підвищити свою продуктивність.

Метою даної кваліфікаційної роботи є розробка зручного та інтерактивного навчального програмного забезпечення, яке ефективно демонструє можливості AutoHotKey в автоматизації графічного інтерфейсу. Це програмне забезпечення надаватиме покрокові інструкції, практичні приклади та практичні вправи, які

допоможуть користувачам зрозуміти синтаксис AutoHotKey, функції та найкращі практики автоматизації графічного інтерфейсу.

Очікуваний вплив цієї кваліфікаційної роботи є багатогранним. По-перше, вона забезпечить користувачів цінними навичками, які стають все більш затребуваними на ринку праці. Вміння автоматизувати графічний інтерфейс за допомогою AutoHotKey підвищить їхню придатність до працевлаштування, особливо на посадах, які вимагають ефективної автоматизації програмного забезпечення та оптимізації процесів. Крім того, люди, які покладаються на додатки з графічним інтерфейсом у своїх повсякденних завданнях, таких як введення даних або тестування програмного забезпечення, отримають вигоду від можливостей автоматизації AutoHotKey, які заощаджують час і зменшують кількість помилок.

Крім того, це навчальне програмне забезпечення зробить свій внесок у ширшу сферу розробки та автоматизації програмного забезпечення, сприяючи використанню AutoHotKey як потужного інструменту для взаємодії з графічним інтерфейсом. Воно слугуватиме ресурсом для осіб, зацікавлених у вивченні автоматизації графічного інтерфейсу, надаючи їм міцну основу для подальшого вивчення та розширення своїх навичок автоматизації.

Автоматизація набуває все більшого значення у сучасному світі, оскільки компанії та приватні особи прагнуть впорядкувати свої процеси та підвищити ефективність. Графічні інтерфейси користувача (GUI) відіграють вирішальну роль у програмних додатках, надаючи користувачам візуальний та інтерактивний спосіб взаємодії з технологіями. Автоматизація графічних інтерфейсів дозволяє користувачам автоматизувати повторювані завдання, зменшити ручну працю та підвищити продуктивність.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Огляд мов автоматизації та написання сценаріїв

Мови автоматизації та сценаріїв відіграють важливу роль у сучасному світі, дозволяючи окремим особам та організаціям впорядкувати свої робочі процеси, зменшити ручну працю та підвищити ефективність. Ці мови дають змогу автоматизувати повторювані завдання, взаємодіяти з програмними додатками та маніпулювати різними системами і процесами. У цьому розділі ми надамо огляд мов автоматизації та сценаріїв, зосередившись на їхньому значенні у сфері автоматизації графічного інтерфейсу.

1.1.1 Вступ до мов сценаріїв

Скриптові мови – це мови програмування, призначені для написання сценаріїв, тобто послідовностей інструкцій, які автоматизують певні завдання або операції. На відміну від компільованих мов, мови сценаріїв інтерпретуються під час виконання, що забезпечує більшу гнучкість і простоту використання. Їх часто використовують для автоматизації повторюваних завдань, взаємодії з програмними додатками через API (інтерфейси прикладного програмування) та маніпулювання даними. Мови сценаріїв широко застосовуються в різних сферах, включаючи системне адміністрування, веб-розробку, аналіз даних і, що особливо важливо, автоматизацію графічних інтерфейсів. Завдяки своєму стислому і зрозумілому синтаксису, мови сценаріїв забезпечують доступну точку входу для людей, які хочуть автоматизувати завдання без необхідності глибоких знань програмування [5], [9].

У контексті автоматизації графічного інтерфейсу мови сценаріїв пропонують зручні засоби для взаємодії з графічними елементами, такими як кнопки, меню та вікна, у програмних додатках. Використовуючи скриптові можливості цих мов, користувачі можуть автоматизувати завдання на основі графічного

інтерфейсу, такі як заповнення форм, вилучення даних і тестування інтерфейсу [8].

AutoHotKey – одна з таких мов сценаріїв, яка чудово підходить для автоматизації графічного інтерфейсу. Її простий, але потужний синтаксис, обширна документація та активна підтримка спільноти роблять її популярним вибором серед ентузіастів автоматизації. AutoHotKey дозволяє користувачам створювати сценарії автоматизації, які відтворюють дії миші та клавіатури, маніпулюють вікнами та елементами керування, а також взаємодіють з елементами графічного інтерфейсу в точний та ефективний спосіб.

Крім того, мови сценаріїв часто містять вбудовані бібліотеки або фреймворки, які полегшують автоматизацію графічного інтерфейсу. Ці бібліотеки пропонують заздалегідь визначені функції та методи для взаємодії з компонентами графічного інтерфейсу, спрощуючи процес автоматизації та зменшуючи кількість необхідного коду. Такі бібліотеки дозволяють користувачам зосередитися на конкретній логіці автоматизації, а не на низькорівневих деталях маніпуляцій з графічним інтерфейсом.

Таким чином, мови сценаріїв слугують цінними інструментами для автоматизації завдань і маніпулювання елементами графічного інтерфейсу в програмних додатках. Вони забезпечують гнучкий і доступний підхід до автоматизації, дозволяючи користувачам підвищити свою продуктивність, зменшити ручні зусилля і досягти більшої ефективності. AutoHotKey, зокрема, виділяється як мова сценаріїв, яка чудово справляється з автоматизацією графічного інтерфейсу, дозволяючи користувачам з легкістю автоматизувати широкий спектр завдань. У наступних розділах ми заглибимося в особливості та можливості AutoHotKey, досліджуючи, як його можна ефективно використовувати для автоматизації графічного інтерфейсу.

1.1.2 Роль автоматизації у розробці програмного забезпечення

Автоматизація змінила правила гри у світі розробки програмного забезпечення. Вона змінила спосіб створення, тестування та розгортання додатків. Автоматизуючи різні завдання та процеси, розробники можуть впорядкувати свій робочий процес і досягти більшої ефективності та продуктивності. Однією з ключових переваг автоматизації в розробці програмного забезпечення є підвищення ефективності. Ручні завдання, які повторюються і займають багато часу, можна автоматизувати, звільнивши час розробників, щоб зосередитися на більш важливих аспектах проекту. Наприклад, такі завдання, як генерація коду, компіляція та розгортання, можуть бути автоматизовані, що дозволяє розробникам швидко виконувати ітерації та випускати оновлення програмного забезпечення [11], [15].

Автоматизувавши ці процеси, розробники можуть значно зменшити ймовірність людської помилки. Завдання, що виконуються вручну, схильні до помилок, але автоматизація забезпечує послідовність і точність протягом усього процесу розробки. Це призводить до вищої якості програмного забезпечення та меншої кількості помилок, оскільки автоматизовані процеси слідують заздалегідь визначеним правилам та інструкціям. Автоматизація також відіграє життєво важливу роль у масштабуванні зусиль з розробки програмного забезпечення. Зі зростанням складності та розміру проектів, керувати ними вручну стає дедалі складніше. Завдяки автоматизації розробники можуть працювати з більшими кодовими базами та ефективніше керувати залежностями. Наприклад, фреймворки для автоматизованого тестування можуть виконувати тести в різних середовищах і конфігураціях, гарантуючи, що програмне забезпечення працює належним чином у різних сценаріях [7].

1.2 Вступ до AutoHotKey

AutoHotKey – це потужна мова сценаріїв, яка широко використовується для автоматизації та кастомізації в операційній системі Windows. Вона надає зручну платформу для автоматизації завдань, створення гарячих клавіш і маніпулювання графічними інтерфейсами користувача (GUI). За своєю суттю AutoHotKey призначений для спрощення процесу автоматизації повторюваних завдань. Незалежно від того, чи це автоматизація введення даних, керування вікнами додатків або взаємодія з елементами графічного інтерфейсу, AutoHotKey пропонує гнучку та інуїтивно зрозумілу мову сценаріїв для ефективного виконання цих завдань [2], [10].

Однією з важливих особливостей AutoHotKey є простота використання. Синтаксис простий і нагадує природну мову, що робить її доступною як для початківців, так і для досвідчених програмістів. Скрипти AutoHotKey записуються у звичайні текстові файли з розширенням «.ahk», які можна редагувати за допомогою будь-якого текстового редактора. Ця простота дозволяє користувачам швидко освоїти основи мови і почати створювати свої сценарії автоматизації. AutoHotKey надає широкий спектр вбудованих функцій і команд, які можна використовувати для взаємодії з різними аспектами операційної системи і графічними інтерфейсами. Ці функції дозволяють користувачам виконувати такі завдання, як надсилання натискань клавіш, клацань миші та команд керування певним вікнам або програмам. Крім того, AutoHotKey підтримує змінні, цикли, умовні оператори та інші конструкції програмування, що дозволяє створювати більш просунуті сценарії автоматизації.

Однією з ключових переваг AutoHotKey є можливість створювати гарячі клавіші та гарячі рядки. Гарячі клавіші – це комбінації клавіш, які при натисканні викликають певні дії або команди. Наприклад, ви можете призначити гарячу клавішу для відкриття часто використовуваної програми або виконання низки завдань за допомогою однієї комбінації клавіш. З іншого боку, гарячі рядки автоматично замінюють заздалегідь визначені скорочення на повнотекстові

розширення. Ця функція особливо корисна для швидкої вставки часто використовуваних фраз, фрагментів коду або підписів електронної пошти.

AutoHotKey також пропонує широку підтримку автоматизації графічного інтерфейсу. За допомогою команд і функцій графічного інтерфейсу користувачі можуть створювати власні діалогові вікна, вікна, меню та інші графічні елементи. Це дозволяє розробляти зручні інтерфейси для сценаріїв автоматизації. Крім того, AutoHotKey надає інструменти для маніпулювання елементами керування графічним інтерфейсом, такими як кнопки, прапорці та випадаючі меню, забезпечуючи точну взаємодію та контроль над програмами на основі графічного інтерфейсу. AutoHotKey може похвалитися великою і активною спільнотою користувачів, які діляться скриптами, надають підтримку і сприяють розвитку мови. Офіційний форум AutoHotKey та інші онлайн-ресурси слугують цінними довідниками та джерелами натхнення як для початківців, так і для досвідчених користувачів. Колективні знання спільноти та готовність допомогти роблять AutoHotKey привабливим вибором для тих, хто шукає рішення для автоматизації.

1.2.1 Історія та передумови створення AutoHotKey

AutoHotKey має багату історію, яка сягає початку 2000-х років. Коріння AutoHotKey можна простежити до мови сценаріїв під назвою AutoIt, яку розробив Джонатан Беннетт. AutoIt набула популярності серед користувачів Windows завдяки своїм можливостям автоматизації та простоті використання. Однак, оскільки попит на більш просунуті функції зростав, Кріс Маллетт, член спільноти AutoIt, взявся за розробку нової мови сценаріїв на основі AutoIt. У 2003 році Кріс Маллетт випустив AutoHotKey як мову сценаріїв з відкритим вихідним кодом. AutoHotKey була побудована на фундаменті, закладеному AutoIt, але з деякими покращеннями та додатковими можливостями. Основною метою було надати користувачам більш потужну і гнучку мову сценаріїв,

спеціально розроблену для автоматизації та налаштування в середовищі Windows [14].

AutoHotKey швидко завоювала популярність серед користувачів завдяки своїй надійності, простоті та активній підтримці спільноти. З роками мова продовжувала розвиватися, додаючи нові функції та вдосконалення, засновані на відгуках користувачів та внесках спільноти. Розвиток AutoHotKey відбувається завдяки пристрасті та відданості спільноти, а регулярні оновлення та нові випуски гарантують, що мова залишається актуальною та відповідає потребам користувачів.

1.2.2 Особливості та можливості AutoHotKey

AutoHotKey пропонує широкий спектр функцій і можливостей, які роблять його кращим вибором для автоматизації та налаштування завдань у Windows. Давайте розглянемо деякі з його ключових можливостей.

Перш за все, AutoHotKey надає зручну мову сценаріїв, яку легко вивчити та використовувати. Синтаксис інтуїтивно зрозумілий, схожий на природну мову, що дозволяє користувачам швидко засвоїти основи і почати створювати свої сценарії автоматизації. Крім того, обширна документація та онлайн-ресурси, доступні для AutoHotKey, ще більше полегшують користувачам вивчення та початок роботи з мовою [13].

Однією з особливостей AutoHotKey є можливість створювати гарячі клавіші та гарячі рядки. Гарячі клавіші дозволяють користувачам визначати комбінації клавіш, які запускають певні дії або команди. Ця функція особливо корисна для автоматизації повторюваних завдань або виконання складних послідовностей дій за допомогою простої комбінації клавіш. З іншого боку, гарячі рядки дозволяють користувачам визначати абревіатури, які автоматично розгортаються в повнотекстові заміни. Ця функція економить час і зусилля, дозволяючи користувачам швидко вставляти часто використовувані фрази або фрагменти коду [1], [12].

AutoHotKey відмінно справляється з автоматизацією графічного інтерфейсу, надаючи повний набір команд і функцій для взаємодії з графічними інтерфейсами користувача. Користувачі можуть створювати власні діалогові вікна, вікна, меню та інші елементи графічного інтерфейсу, що дозволяє розробляти зручні інтерфейси для своїх сценаріїв автоматизації. Можливість маніпулювати елементами керування графічного інтерфейсу, такими як кнопки, прапорці та випадаючі меню, дозволяє здійснювати точний контроль і взаємодію з додатками на основі графічного інтерфейсу.

1.3 GUI у розробці програмного забезпечення

Графічний інтерфейс користувача (GUI) відіграє вирішальну роль у розробці програмного забезпечення, формуючи спосіб взаємодії користувачів з додатками та впливаючи на їхній загальний досвід. Графічний інтерфейс забезпечує візуальний та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам взаємодіяти з програмним забезпеченням за допомогою графічних елементів, таких як кнопки, меню та вікна. У цьому розділі ми розглянемо важливість графічного інтерфейсу у користувацькому досвіді та обговоримо проблеми і міркування, які слід враховувати при розробці графічного інтерфейсу [3], [4].

1.3.1 Важливість графічного інтерфейсу

Графічний інтерфейс – це фундаментальний аспект дизайну програмного забезпечення, оскільки він безпосередньо впливає на взаємодію з користувачем. Добре розроблений графічний інтерфейс підвищує зручність, ефективність і задоволеність користувачів, які взаємодіють з програмним забезпеченням [6]. Ось кілька ключових причин, чому графічний інтерфейс важливий для користувацького досвіду:

– покращена зручність використання: добре продуманий графічний інтерфейс спрощує взаємодію між користувачами і програмним забезпеченням, полегшуючи користувачам навігацію, розуміння і роботу з програмою. Інтуїтивно зрозумілі та візуально привабливі інтерфейси допомагають користувачам швидко зрозуміти функціональність і можливості програмного забезпечення, скорочуючи час навчання та покращуючи загальну зручність використання;

– підвищення ефективності: графічні інтерфейси дозволяють користувачам ефективно виконувати завдання, надаючи чіткі візуальні підказки, впорядковані робочі процеси та легкий доступ до часто використовуваних функцій. Користувачі можуть переміщатися по програмі, взаємодіяти з різними елементами і виконувати завдання швидше і без особливих зусиль, що призводить до підвищення продуктивності;

– візуальне представлення: графічні інтерфейси використовують графічні елементи, такі як піктограми, зображення та візуальний зворотній зв'язок для передачі інформації, що полегшує користувачам інтерпретацію та розуміння даних. Візуальне представлення допомагає у візуалізації даних, прийнятті рішень та передачі складної інформації у більш зручному для сприйняття форматі;

– інтерактивність та зворотній зв'язок: графічні інтерфейси сприяють інтерактивному спілкуванню між користувачами та програмним забезпеченням, дозволяючи користувачам вносити дані, отримувати зворотній зв'язок та реагувати на підказки. Інтерактивні елементи, такі як кнопки, прапорці та повзунки, дозволяють користувачам керувати програмою та отримувати негайний зворотній зв'язок, створюючи більш цікавий та чуйний користувацький досвід;

– брендинг та естетика: графічні інтерфейси сприяють брендингу та естетиці програмних додатків. Візуально привабливий і послідовний інтерфейс підсилює ідентичність бренду, формує довіру користувачів і покращує загальне сприйняття програмного забезпечення. Продумані графічні інтерфейси з увагою

до візуальних елементів, кольорових схем і типографіки створюють позитивний емоційний зв'язок з користувачами.

1.4 Інтеграція AutoHotKey з графічним інтерфейсом

AutoHotKey пропонує численні можливості для інтеграції з графічним інтерфейсом (GUI). У цьому розділі ми розглянемо випадки використання та переваги застосування AutoHotKey для автоматизації GUI, а також наведемо приклади скриптів AutoHotKey, які демонструють його можливості у взаємодії з елементами GUI.

1.4.1 Варіанти використання та переваги AutoHotKey

Інтеграція AutoHotKey з графічним інтерфейсом відкриває широкий спектр можливостей використання і надає ряд переваг для автоматизації графічного інтерфейсу. Одним з основних варіантів використання є автоматизація повторюваних завдань, що виконуються в додатках на основі графічного інтерфейсу. AutoHotKey дозволяє користувачам створювати сценарії, які можуть імітувати взаємодію користувача з елементами графічного інтерфейсу, такими як кнопки, меню та вікна. Автоматизувавши ці завдання, користувачі можуть заощадити час і зусилля, підвищити продуктивність і зменшити ризик помилок.

Ще однією значною перевагою використання AutoHotKey для автоматизації графічного інтерфейсу є його здатність підвищувати ефективність робочого процесу. Автоматизуючи складні послідовності взаємодій з графічним інтерфейсом, користувачі можуть впорядкувати свій робочий процес і досягти послідовних і точних результатів. Наприклад, можна створити скрипт для автоматизації завдань введення даних, де скрипт вводить дані в різні поля у формі на основі графічного інтерфейсу, усуваючи необхідність ручного введення і мінімізуючи ймовірність помилок при введенні даних.

Крім того, AutoHotKey пропонує гнучкість і можливості налаштування для автоматизації роботи з графічним інтерфейсом. Користувачі можуть створювати сценарії, пристосовані до їхніх конкретних потреб та вподобань. AutoHotKey надає повний набір команд і функцій, які дозволяють користувачам маніпулювати елементами графічного інтерфейсу, отримувати інформацію з вікон графічного інтерфейсу і виконувати дії на основі певних умов.

1.4.2 Приклади скриптів AutoHotKey

Щоб краще зрозуміти можливості AutoHotKey в автоматизації GUI, давайте розглянемо кілька прикладів скриптів AutoHotKey, які демонструють його використання для взаємодії з елементами GUI. Почнемо з прикладу натискання кнопки (лістинг 1.1).

Лістинг 1.1 – приклад натискання кнопки

```
Button_Click:
GuiControl, Click, Button1
Return
```

У цьому прикладі скрипт ідентифікує кнопку з назвою «Button1» у вікні графічного інтерфейсу і імітує натискання на цю кнопку. Це може бути корисно для автоматизації завдань, які вимагають від користувача натискання певних кнопок у програмі з графічним інтерфейсом.

Тепер розглянемо приклад тексту до текстового поля (лістинг 1.2).

Лістинг 1.2 – Приклад введення тексту до текстового поля

```
SetText:
GuiControl, Focus, TextField
SendInput, Це приклад тексту.
Return
```

У цьому прикладі сценарій фокусується на текстовому полі з назвою "TextField" і вводить до нього текст "Це приклад тексту.". Це може бути використано для автоматизації завдань введення даних, де користувачеві потрібно ввести певний текст у текстові поля графічного інтерфейсу.

Також одна з можливостей є отримання інформації з вікна. Приклад поданий у лістингу 1.3.

Лістинг 1.3 – приклад отримання інформації з вікна

```
GetWindowText:
WinGetText, WindowText, Notepad
MsgBox, Вміст вікна Notepad: %WindowText%
Return
```

Ці приклади підкреслюють простоту і ефективність скриптів AutoHotKey у взаємодії з елементами графічного інтерфейсу. Використовуючи різні команди і функції AutoHotKey, користувачі можуть створювати сценарії, які автоматизують широкий спектр взаємодій з графічним інтерфейсом, від простого натискання кнопок до складних маніпуляцій з даними.

1.5 Використання AutoHotKey для начального застосунку

Розробка навчального програмного забезпечення для демонстрації взаємодії AutoHotKey і графічного інтерфейсу використовує потужність і гнучкість самого AutoHotKey. AutoHotKey слугує основною мовою сценаріїв та фреймворком для створення програмного забезпечення, що дозволяє безперешкодно інтегрувати елементи графічного інтерфейсу та можливості автоматизації. Ось деякі ключові аспекти використання AutoHotKey в процесі розробки:

– мова сценаріїв: синтаксис AutoHotKey та можливості написання сценаріїв роблять його ідеальним вибором для розробки навчального програмного забезпечення. Він пропонує просту, але потужну мову сценаріїв, яка дозволяє розробникам писати стислий і читабельний код. AutoHotKey надає багатий набір

вбудованих функцій і команд, спеціально розроблених для автоматизації роботи з графічним інтерфейсом, що робить його добре придатним для створення програмного забезпечення, орієнтованого на взаємодію з графічним інтерфейсом;

– маніпуляції з графічним інтерфейсом: AutoHotKey надає широку підтримку для маніпуляцій з графічним інтерфейсом, дозволяючи розробникам створювати, змінювати та взаємодіяти з графічними інтерфейсами користувача програмно. За допомогою AutoHotKey розробники можуть динамічно створювати елементи графічного інтерфейсу, визначати їх властивості та ефективно реагувати на дії користувача;

– програмування на основі подій: подієво-керована модель AutoHotKey відіграє вирішальну роль у розробці навчального програмного забезпечення. Вона дозволяє розробникам писати код, який реагує на певні події, викликані діями користувача, такі як натискання кнопок або вибір меню. Використовуючи підхід AutoHotKey, заснований на подіях, навчальне програмне забезпечення може надавати зворотній зв'язок у реальному часі та реагувати на взаємодію користувачів у середовищі графічного інтерфейсу. Це підвищує інтерактивність та залучення користувачів до навчального програмного забезпечення;

– можливості автоматизації: можливості автоматизації AutoHotKey є центральним елементом навчального програмного забезпечення. Це дозволяє розробникам автоматизувати повторювані завдання та імітувати взаємодію користувача з елементами графічного інтерфейсу. Навчальна програма використовує AutoHotKey для демонстрації різних сценаріїв автоматизації, таких як натискання кнопок, введення тексту і переміщення по меню. Демонструючи ці можливості автоматизації, користувачі можуть зрозуміти потенціал AutoHotKey для впорядкування власних робочих процесів і підвищення ефективності.

2 ОГЛЯД ПРОГРАМНОЇЇ РЕАЛІЗАЦІЇ

2.1 Використання розробленного застосунку xPeek

Перед початком розробки навчального застосунку створення інтерфейсів за допомогою AutoHotKey був створений застосунок xPeek. Код цього застосунку буде використаний для навчальних екранів з прикладами коду та результатами цього коду. Додаток xPeek є інструментом для ентузіастів та гравців League of Legends, які хочуть аналізувати та оцінювати силу і склад своїх команд. Завдяки інтуїтивно зрозумілому інтерфейсу та широкому спектру функцій, xPeek є рішенням для побудови команди, прийняття стратегічних рішень та оцінки ефективності. Розглянемо можливості та особливості програми xPeek більш детально:

– управління складом команди: xPeek дозволяє користувачам створювати та керувати складом своєї команди, обираючи героїв з величезного списку персонажів League of Legends. Користувачі можуть вибирати героїв як для своєї команди, так і для команди суперника, створюючи збалансований і конкурентоспроможний матч. Додаток пропонує зручний інтерфейс, де користувачі можуть переглянути імена героїв, їхні позиції та сильні сторони;

– візуалізація героїв: додаток надає інтерфейс, де користувачі можуть бачити рисунки героїв та їхні позиції, що дозволяє швидко та легко переглянути склад команди;

– рейтинг сили: xPeek включає в себе систему рейтингу сили у вигляді мечів, яка оцінює силу окремих героїв на різних етапах гри. Ця система враховує ефективність героїв на початку, в середині та наприкінці гри, надаючи користувачам цінну інформацію про загальну силу та ефективність їхньої команди. Присвоюючи ранги кожному герою на основі їхніх показників, xPeek дозволяє користувачам приймати обґрунтовані рішення щодо складу та стратегії команди.

Приклади роботи застосунку xPeek подані нижче на рисунках 2.1, 2.2, 2.3.

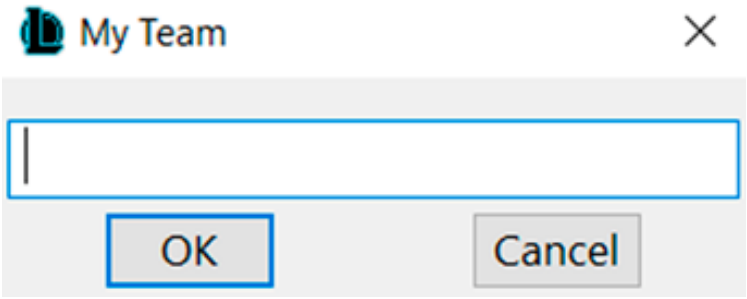

Рисунок 2.1 – Форма створення команди

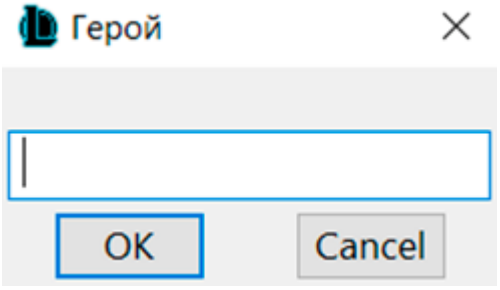

Рисунок 2.2 – Вікно вибору героя

Рисунок 2.3 – Головний інтерфейс застосунку xPeek

2.2 Застосунок для навчання створення інтерфейсів

2.2.1 Мета навчального застосунку

Навчальне програмне забезпечення для AutoHotKey має подвійну мету: надати користувачам комплексну навчальну платформу для написання сценаріїв AutoHotKey та продемонструвати практичну реалізацію першого додатку – xPeek для League of Legends (LoL). Основна мета навчального програмного забезпечення – дати можливість користувачам вивчити і освоїти мову сценаріїв AutoHotKey. Воно пропонує зручний інтерфейс і структуровану навчальну програму, яка проводить користувачів через основи AutoHotKey. Користувачі можуть вивчати різні концепції, синтаксис та найкращі практики, пов'язані з написанням сценаріїв на AutoHotKey.

Однак, що відрізняє це навчальне програмне забезпечення від інших, так це його інтеграція з додатком xPeek. xPeek призначений для аналізу вибору героя в популярній грі League of Legends. Він надає інформацію та рекомендації щодо оптимального вибору героя, допомагаючи гравцям приймати стратегічні рішення в грі. Навчальне програмне забезпечення використовує можливості AutoHotKey, щоб продемонструвати, як працює код програми xPeek. За допомогою інтерактивних екранів або покрокових демонстрацій користувачі можуть спостерігати за виконанням фрагментів коду і бачити результати в режимі реального часу. Цей практичний досвід дозволяє користувачам зрозуміти основну логіку та функціональність додатку Hero Peek.

Фрагменти коду, продемонстровані в навчальному програмному забезпеченні, охоплюють широкий спектр функціональних можливостей, таких як маніпулювання графічними елементами, відтворення звуків, зчитування даних з файлів та обробка користувацького вводу. Користувачі можуть спостерігати, як ці фрагменти коду використовуються в додатку xPeek для досягнення конкретних цілей, таких як відображення зображень героїв, керування графічним інтерфейсом та надання звукового зворотного зв'язку.

Така інтеграція між навчальним програмним забезпеченням і додатком xPeek забезпечує цілісний навчальний процес. Користувачі не лише отримують знання про сценарії AutoHotKey, але й бачать їх практичне застосування в реальних умовах. Вони можуть пов'язати концепції, вивчені в навчальному програмному забезпеченні, з реальним і відповідним сценарієм використання, що посилює їх розуміння і заохочує до творчого вирішення проблем.

Навчальний застосунок поєднує комплексні навчальні матеріали з практичною демонстрацією виконання коду в додатку чPeek. Така інтеграція не лише покращує розуміння користувачами скриптів AutoHotKey, але й надихає їх застосовувати набуті знання у власних проектах.

Надаючи користувачам можливість вивчати AutoHotKey і демонструючи його реальне застосування, навчальне програмне забезпечення пропонує цінний навчальний досвід. Незалежно від того, чи є користувачі програмістами-початківцями, завзятими гравцями League of Legends, чи просто цікавляться автоматизацією та кастомізацією, цей комбінований підхід забезпечує потужну платформу для набуття навичок, підвищення продуктивності та розкриття творчого потенціалу.

2.2.2 Огляд коду навчального застосунку

Цей підпункт містить огляд коду, використаного в навчальному додатку. Навчальний додаток слугує застосунком для вивчення сценаріїв AutoHotKey, одночасно демонструючи його практичну реалізацію в додатку XPeek. Структура коду і логіка навчального додатку дають цінну інформацію про те, як AutoHotKey можна ефективно використовувати для автоматизації завдань, маніпулювання графічними елементами, обробки даних, введених користувачем, і взаємодії із зовнішніми ресурсами. Цей огляд має на меті дати уявлення про внутрішню роботу навчальної програми, створюючи підґрунтя для більш глибокого вивчення її кодових компонентів. Далі буде розглянутий код роботи навчального застосунку. Повний програмний код поданий у додатку Б.

Розглянемо програмний код ініціалізації змінних для обробки звуку та зображень. Код поданий у лістингу 2.1.

Лістинг 2.1 – Програмний код ініціалізації змінних для обробки звуку та зображень

```
img_line_num = 1
file_name = start.txt

start = source\sounds\start.mp3
error = source\sounds\error.mp3
trash = source\sounds\trash.mp3
success = source\sounds\success.mp3

sound_up = %start%
sound_down = 0
show_hide = 0
```

Перший рядок «img_line_num = 1» ініціалізує змінну з назвою img_line_num і присвоює їй значення 1. Ця змінна може бути використана для відстеження номера рядка зображення у програмі. Другий рядок «file_name = "start.txt"» ініціалізує змінну з назвою file_name і присвоює їй значення «start.txt». Ця змінна, найімовірніше, являє собою ім'я файлу, з яким працюватиме програма.

Далі йдуть чотири рядки, які визначають шляхи до різних звукових файлів. Ці звукові файли зберігаються у каталозі «source\sounds». Нижче наведено змінні та відповідні їм шляхи до звукових файлів:

– start - шлях до звукового файлу «start.mp3»;

– error - шлях до звукового файлу «error.mp3»;

– trash - шлях до звукового файлу «trash.mp3»;

– success - шлях до звукового файлу «success.mp3».

Після цього у нас є ще три рядки, які присвоюють значення іншим змінним:

– sound_up має значення %start%. Це заповнювач або посилання на змінну start, яку ми визначили раніше. Відсоткові знаки % можуть вказувати на те, що це спеціальне значення або синтаксис, який використовується у програмі;

– sound_down встановлено у 0, що означає, що їй присвоєно нульове значення;

– show_hide також дорівнює 0, що означає, що їй також присвоєно нульове значення.

Ці змінні можна використовувати у програмі для керування поведінкою програми, наприклад, відтворення певних звукових файлів, маніпулювання файлами із заданою назвою або налаштування видимості певних елементів.

Розглянемо програмний код налаштування графічного інтерфейсу (лістинг 2.2).

Лістинг 2.2 – Програмний код налаштування графічного інтерфейсу та відображення зображень при взаємодії з мишею

```
   FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
   Gui, +AlwaysOnTop +LastFound +Owner -Caption -Border
   Gui, Add, Picture, x0 y0 w1119 h657 vMyPicture, source\screen\%img_name%
   GUI, Show, x218 y115 h657 w1119
   OnMessage(0x201, "WM_LBUTTONDOWN")
   WM_LBUTTONDOWN()
   {
      PostMessage, WM_NCLBUTTONDOWN := 0xA1, HTCAPTION := 2
   }
   Return
```

Код містить декілька команд та змінних:

– FileReadLine: ця команда використовується для читання певного рядка з файлу, але не використовується у цьому фрагменті коду;

– img_name: ця змінна зберігає ім'я файлу зображення. На неї посилаються далі у коді;

– джерело\ім'я файлу\%ім'я_файлу%: це шлях до файлу, який включає значення змінної ім'я_файлу. Вказує на місцезнаходження конкретного файлу;

– «%img_line_num%»: ця змінна використовується у команді графічного інтерфейсу далі у коді. Це номер рядка, пов'язаного із зображенням.

Код також встановлює вікно графічного інтерфейсу користувача (GUI) з певними атрибутами за допомогою команди Gui. Ці атрибути включають +AlwaysOnTop (вікно залишається поверх інших вікон), +LastFound (вікно є останнім знайденим скриптом), +Owner (скрипт стає власником вікна) і -Caption -Border (вікно не має рядка заголовка або межі).

Елемент зображення додається до вікна графічного інтерфейсу за допомогою команди Gui, Add, Picture, в якій вказується положення (x0 y0) і розміри (w1119 h657) зображення. Елементу зображення присвоюється ім'я змінної MyPicture, а шлях до файлу зображення «source\screen\%img_name%»використовується як джерело зображення.

Потім вікно графічного інтерфейсу відображається за допомогою команди GUI, Show з певними розмірами і положенням (x218 y115 h657 w1119). Код визначає функцію WM_LBUTTONDOWN і пов'язує її з повідомленням 0x201 (натиснута ліва кнопка миші) за допомогою команди OnMessage.

У функції WM_LBUTTONDOWN команда PostMessage використовується для імітації неклієнтського натискання лівої кнопки миші (WM_NCLBUTTONDOWN := 0xA1) в області підписів вікна (HTCAPTION := 2). Нарешті, інструкція Return вказує на завершення роботи сценарію.

Далі йде код для вибору та відображення зображень при натисканні на клавіші NumPad (лістинг 2.3).

Лістинг 2.3 – Програмний код вибору та відображення зображень при натисканні на клавіші NumPad

```
User
NumPad1::
file_name = start.txt
img_line_num = 1
sound_up = %start%
sound_down = 0
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl, , MyPicture, source\screen\%img_name%
return

NumPad2::
file_name = show_pick.txt
img_line_num = 1
sound_up = 0
sound_down = 0
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl, , MyPicture, source\screen\%img_name%
return
```

При натисканні NumPad1:

– змінній file_name присвоюється значення «start.txt»;

– змінній img_line_num присвоюється значення 1;

– змінній sound_up присвоюється значення «%start%»;

– змінній sound_down присвоюється значення 0.

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\%img_name%».

При натисканні NumPad2:

– змінній file_name присвоюється значення «show_pick.tx»;

– змінна img_line_num дорівнює 1;

– змінні sound_up та sound_down дорівнюють 0.

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою source\screen\%img_name%. Оператор return вказує на кінець блоку кожної гарячої клавіші.

Отже, натискання NumPad1 встановить певні значення для змінних і оновить вікно графічного інтерфейсу зображенням на основі вмісту файлу «start.txt», а натискання NumPad2 зробить те ж саме, але з файлом «show_pick.txt». Ці гарячі клавіші дають змогу змінювати зображення у вікні графічного інтерфейсу на основі даних, введених користувачем.

Після цього йде код додавання зображень героїв при натисканні клавіш додавання та віднімання використовуючи NumPad (лістинг 2.4).

Лістинг 2.4 – Програмний код додавання зображень героїв при натисканні клавіш додавання та віднімання на NumPad

```
NumPadAdd::
file_name = addmyhero.txt
img_line_num = 1
sound_up = %success%
sound_down = %error%
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl, , MyPicture, source\screen\%img_name%
return

NumPadSub::
file_name = addyouhero.txt
```

Продовження лістингу 2.4

```
   img_line_num = 1
   sound_up = %success%
   sound_down = %error%
   FileReadLine, img_name, source\file name\%file_name%,
%img_line_num%
   GuiControl, , MyPicture, source\screen\%img_name%
   return
```

Фрагмент коду визначає гарячі клавіші для клавіш NumPadAdd і NumPadSub. Він встановлює різні дії для кожної клавіші.

При натисканні NumPadAdd:

– змінна file_name набуває значення «addmyhero.txt»;

– змінна img_line_num встановлюється в 1;

– змінній sound_up присвоюється значення «%success%»;

– змінній sound_down присвоюється значення «%error%».

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\%img_name%».

При натисканні NumPadSub:

– змінна file_name набуває значення «addyouhero.txt»;

– змінній img_line_num присвоюється значення 1;

– змінній sound_up присвоюється значення «%success%»;

– змінній sound_down присвоюється значення «%error%».

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення,

розташованим за адресою «source\screen\%img_name%».Оператор return вказує на кінець блоку кожної гарячої клавіші.

Таким чином, натискання NumPadAdd встановить певні значення для змінних і оновить вікно графічного інтерфейсу зображенням на основі вмісту файлу «addmyhero.txt». Аналогічно, натискання NumPadSub зробить те саме, але з файлом «addyouhero.txt». Ці гарячі клавіші дозволяють змінювати зображення у вікні графічного інтерфейсу на основі введення користувачем, супроводжуючи його відповідними звуковими ефектами, заданими у змінних sound_up і sound_down.

Розглянемо програмний код видалення зображень героїв та ворогів при натисканні клавіш додавання та віднімання.

Лістинг 2.5 – Програмний код видалення зображень героїв та ворогів при натисканні Shift + NumPad додавання та віднімання

```
   +NumPadAdd::
   file_name = del_hero.txt
   img_line_num = 1
   sound_up = 0
   sound_down = %trash%
   FileReadLine,     img_name,     source\file     name\%file_name%,
%img_line_num%
   GuiControl, , MyPicture, source\screen\%img_name%
   return

   +NumPadSub::
   file_name = del_enemy.txt
   img_line_num = 1
   sound_up = 0
   sound_down = %trash%
   FileReadLine,     img_name,     source\file     name\%file_name%,
%img_line_num%
   GuiControl, , MyPicture, source\screen\%img_name%
   return
```

При натисканні Shift + NumPadAdd:

– змінній file_name присвоюється значення «del_hero.txt»;

– змінній img_line_num присвоюється значення 1;

– змінній sound_up присвоюється значення 0;

– змінній sound_down присвоюється значення «%trash%»;

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\%img_name%».

При натисканні Shift + NumPadSub:

– змінна file_name набуває значення «del_enemy.txt»;

– змінній img_line_num присвоюється значення 1;

– змінній sound_up присвоюється значення 0;

– змінній sound_down присвоюється значення «%trash%»;

Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\%img_name%».Оператор return вказує на кінець блоку кожної гарячої клавіші.

Таким чином, комбінація клавіш Shift + NumPadAdd встановить певні значення для змінних і оновить вікно графічного інтерфейсу зображенням на основі вмісту файлу «del_hero.txt». Аналогічно, натискання Shift + NumPadSub зробить те саме, але з файлом «del_enemy.txt». Ці гарячі клавіші дозволяють змінювати зображення у вікні графічного інтерфейсу на основі введення користувачем, супроводжуючи його звуковим ефектом, заданим у змінній sound_down, який представляє дію видалення або відкидання об'єкта.

Розглянемо код навігації зображеннями та відтворення звуку (лістинг 2.6).

Лістинг 2.6 – Програмний код навігації зображеннями та відтворення звуку

```
Right::
img_line_num++
FileReadLine, img_name, source\file name\%file_name%,
%img_line_num%
GuiControl, , MyPicture, source\screen\%img_name%
Return

Left::
if img_line_num = 1
return
img_line_num--
FileReadLine, img_name, source\file name\%file_name%,
%img_line_num%
GuiControl, , MyPicture, source\screen\%img_name%
Return

Up::
SoundPlay, %sound_up%
return

Down::
SoundPlay, %sound_down%
return
```

Фрагмент коду містить гарячі клавіші для клавіш зі стрілками «вправо», «вліво», «вгору» і «вниз». Давайте розберемо кожну частину.

При натисканні клавіші зі стрілкою праворуч img_line_num збільшується на 1. Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи оновлене значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення,

розташованим за адресою «source\screen\%img_name%». Оператор Return вказує на кінець блоку гарячих клавіш.

При натиснутій клавіші зі стрілкою вліво код перевіряє, чи img_line_num дорівнює 1. Якщо це так, гаряча клавіша повертається без виконання жодних дій. Якщо img_line_num не дорівнює 1, він зменшується на 1. Команда FileReadLine зчитує певний рядок з файлу, розташованого за адресою «source\ім'я_файлу\%ім'я_файлу%», використовуючи оновлене значення img_line_num. Отриманий рядок зберігається у змінній img_name. Команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\%img_name%». Оператор Return вказує на кінець блоку гарячих клавіш.

При натисканні клавіші зі стрілкою вгору Команда SoundPlay відтворює звуковий файл, вказаний у змінній sound_up. Оператор Return вказує на кінець блоку гарячих клавіш. При натисканні клавіші зі стрілкою вниз команда SoundPlay відтворює звуковий файл, вказаний у змінній sound_down. Оператор Return вказує на кінець блоку гарячих клавіш.

Отже, натискання клавіші зі стрілкою праворуч збільшує значення img_line_num, зчитує відповідний рядок з файлу та оновлює зображення у вікні графічного інтерфейсу. Натискання клавіші зі стрілкою вліво зменшує значення img_line_num (якщо воно ще не дорівнює 1), зчитує відповідний рядок і оновлює зображення у вікні графічного інтерфейсу. Натискання клавіші зі стрілкою вгору відтворює звук, вказаний у параметрі sound_up, а натискання клавіші зі стрілкою вниз відтворює звук, вказаний у параметрі sound_down.

Останньою частиною програмного коду є код перемикання видимості зображень на клавіші множення NumPad (лістинг 2.7).

Лістинг 2.7 – Програмний код перемикання видимості зображень на клавіші множення NumPad

```
NumPadMult::
show_hide := show_hide + 1
if (show_hide = 1)
{
  GuiControl, , MyPicture, source\screen\hide_show.png
  Gui, Show, %possgui%
}
else if (show_hide = 2)
{
  Gui, Cancel
  show_hide = 0
}
return

GuiClose:
ExitApp
```

Фрагмент коду містить гарячу клавішу для клавіші NumPadMult, а також мітку з назвою GuiClose.

При натисканні NumPadMult змінна show_hide збільшується на 1. Код перевіряє значення show_hide за допомогою інструкції if-else. Якщо show_hide дорівнює 1, команда GuiControl оновлює елемент MyPicture у вікні GUI файлом зображення, розташованим за адресою «source\screen\hide_show.png». Команда Gui, Show показує вікно графічного інтерфейсу за допомогою параметра «%possgui%». Якщо show_hide дорівнює 2, команда Gui, Cancel скасовує вікно графічного інтерфейсу. Змінна show_hide при цьому скидається до 0. Оператор Return вказує на кінець блоку гарячих клавіш.

Мітка GuiClose спрацьовує, коли вікно графічного інтерфейсу закривається. Команда ExitApp викликається для завершення роботи скрипта.

Отже, натискання клавіші NumPadMult збільшує змінну show_hide і виконує різні дії залежно від її значення. Якщо show_hide дорівнює 1, то у вікні графічного інтерфейсу буде показано зображення, задане параметром MyPicture. Якщо show_hide дорівнює 2, вікно графічного інтерфейсу скасовується і show_hide скидається до 0. Закриття вікна графічного інтерфейсу викликає мітку GuiClose, яка завершує скрипт за допомогою команди ExitApp.

2.2.3 Результати огляду коду

Огляд коду показав ефективне використання можливостей та функцій AutoHotKey, демонструючи майстерність розробників у використанні можливостей цієї скриптової мови. Використання змінних, операцій з файлами, елементів керування графічним інтерфейсом та гарячих клавіш демонструє всебічне розуміння можливостей AutoHotKey. Код демонструє модульний підхід, з окремими розділами, присвяченими різним функціональним можливостям навчальної програми. Такий модульний дизайн покращує можливість повторного використання коду та його підтримку, полегшуючи додавання нових функцій або модифікацію існуючих у майбутньому.

Крім того, чіткість коду та широке використання коментарів сприяють його зрозумілості, що дозволяє іншим розробникам розуміти проект та ефективно співпрацювати над ним. Продумані домовленості щодо іменування та послідовний стиль кодування покращують читабельність, полегшуючи відстеження потоку роботи програми.

3 ОГЛЯД ЗАСТОСУНКУ

3.1 Керування застосунком

В навчальному застосунку було реалізовано кілька ключових елементів керування для покращення взаємодії з користувачем та навігації. Давайте розглянемо ці елементи управління та їхні функції докладніше.

Для навігації по екранних формах і переходу від однієї екранної форми до іншої ви можете просто використовувати клавіші зі стрілками вліво і вправо на клавіатурі. Натиснувши клавішу зі стрілкою вліво, ви можете повернутися до попередньої екранної форми, тоді як клавіша зі стрілкою вправо дозволяє перейти до наступної екранної форми. Така інтуїтивно зрозуміла схема управління забезпечує плавний і безперешкодний перегляд.

Ще одним важливим елементом управління є клавіша «+» на клавіатурі NumPad. При натисканні вона запускає показ екранних форм, спеціально розроблених для додавання персонажа до програми. Ці екранні форми проведуть вас через процес створення та налаштування нового героя.

На клавіатурі NumPad можна знайти клавішу «-«, яка слугує для швидкого доступу до екранних форм для додавання ворожих персонажів. Ці екранні форми містять покрокові інструкції з додавання грізних супротивників у додаток, що дозволить вам створювати складні сценарії та перевіряти силу ваших героїв.

Якщо потрібно тимчасово приховати додаток, то це можна зробити натиснувши клавішу «*» на клавіатурі NumPad. Ця зручна функція дозволяє швидко перемикатися між додатком та іншими завданнями, не закриваючи його повністю, забезпечуючи легку багатозадачність і безперебійну роботу користувача.

Тепер давайте розглянемо функції, які викликаються цифровими клавішами на цифровій клавіатурі. Натискання клавіші «2» відкриває набір екранних форм, які представляють інтерфейс, де демонструються процес створення інтерфейсу з створеними героями та їх характеристиками.. Таке візуальне представлення

дозволяє отримати повне уявлення про ваших героїв та їхні сильні сторони, що допоможе вам у розробці стратегії та створенні команди.

З іншого боку, натискання клавіші «1» на цифровій клавіатурі викликає екранні форми, які є на початку роботи застосунку. Ці вступні екранні форми задають контекст, надаючи огляд призначення та функціональних можливостей програми, допомагаючи вам зрозуміти роботу застосунку з правильної сторони.

Ці ключові елементи керування слугують інтуїтивно зрозумілими клавішами швидкого доступу, що дозволяють вам легко переміщатися по освітньому додатку та отримувати доступ до певних функцій та інформації у зручний для вас час. Розуміючи та ефективно використовуючи ці елементи керування, ви зможете отримати максимальну віддачу від процесу навчання та повністю зануритися у світ взаємодії AutoHotKey та графічного інтерфейсу.

3.2 Огляд екранних форм навчального застосунку

Навчальний додаток надає користувачам унікальну можливість заглибитися в тонкощі розробки коду, пропонуючи екранні форми, які демонструють фрагменти коду та їх попередній перегляд. Ці екранні форми слугують безцінним ресурсом для розуміння основних процесів, пов'язаних зі створенням програми xPeek.

Кожен фрагмент коду супроводжується попереднім переглядом, який візуально демонструє результат виконання коду. Це дозволяє користувачам спостерігати за ефектами коду і глибше зрозуміти, як різні компоненти взаємодіють для досягнення бажаного результату. Попередній перегляд коду надає користувачам візуальне представлення графічного інтерфейсу програми, що дозволяє їм побачити практичне застосування коду. Частина екранних форм навчального застосунку будуть розглянуті далі.

3.2.1 Екранна форма зміни значка у треї

Екранна форма (рисунок 3.1) призначена для практичного навчання шляхом представлення фрагмента коду, який відповідає за зміну значку в треї програми.

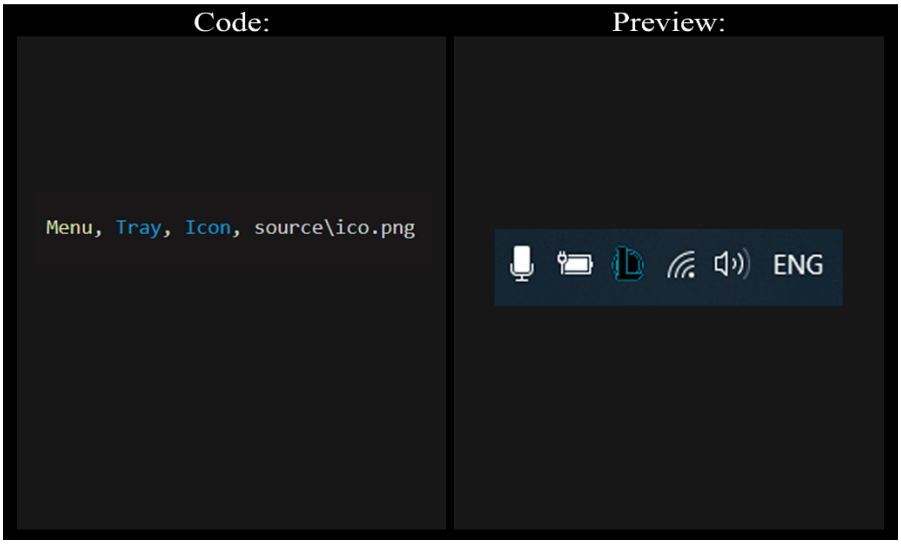

Рисунок 3.1 – Екранна форма зміни значка у треї

3.2.2 Екранна форма створення героя

На рисунку 3.2 зображена екранна форма завдяки якій користувачі можуть вивчити код, що відповідає за створення нового користувача в додатку xPeek. Вони можуть вивчити покроковий процес збору вхідних даних користувача, їх перевірки та зберігання у відповідних структурах даних. Супровідний результат демонструє успішне створення нового користувача, надаючи користувачам візуальне підтвердження функціональності коду.

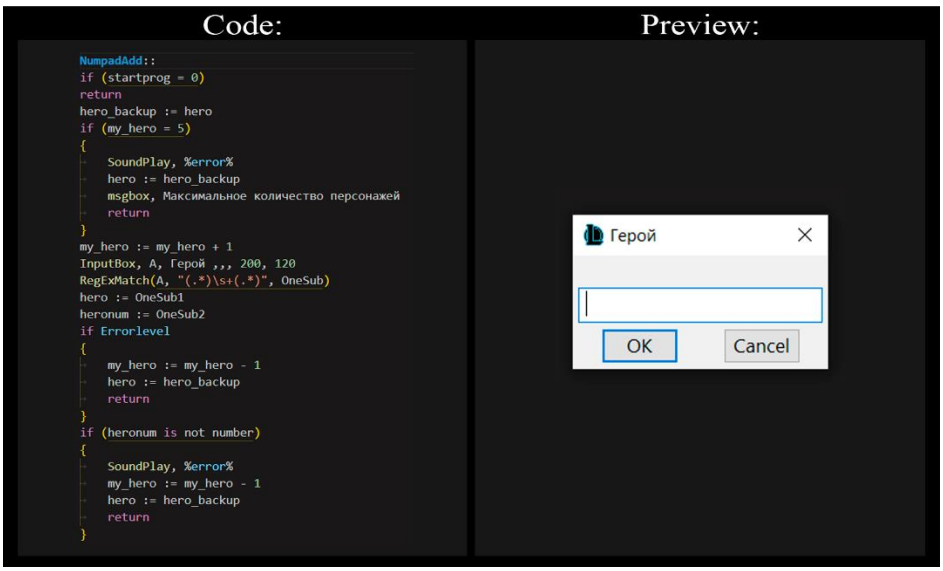

Рисунок 3.2 – Екранна форма процесу створення героя

3.2.3 Екранна форма створення команди

На екранній формі (рисунок 3.3) зображений код, який обробляє створення нової команди. Користувачі можуть спостерігати логіку створення команди. Відображений результат дозволяє користувачам візуалізувати результат виконання коду.

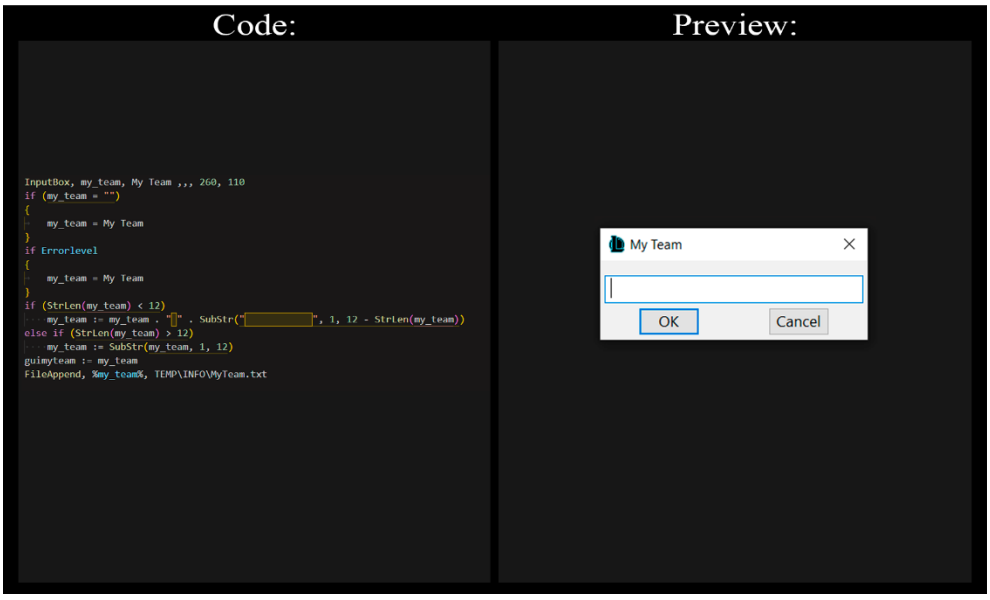

Рисунок 3.3 – Екранна форма процесу створення команди

3.2.4 Екранна форма з відображенням процесу вибору героїв

На цій екранній формі представлено код, що відповідає за відображення обраних героїв у додатку xPeek. Користувачі можуть вивчити логіку, за якою відбувається пошук і представлення обраних героїв, що дозволяє їм зрозуміти основний механізм. Відображений результат демонструє успішний рендеринг обраних героїв, надаючи користувачам візуальне представлення їхнього вибору.

Рисунок 3.4 – Екранна форма з відображенням процесу вибору героїв

3.2.5 Екранна форма з відображенням створення головного інтерфейсу

На цій екранні формі (рисунок 3.5) представлено фрагмент коду, що відповідає за рендеринг головного інтерфейсу додатку xPeek. Користувачі можуть дослідити код, який створює зручний графічний інтерфейс, що дозволяє їм зрозуміти структуру та компоненти програми. Результат, що відображається,

демонструє візуально привабливий і функціональний інтерфейс основного додатка, надаючи користувачам огляд його макета і дизайну.

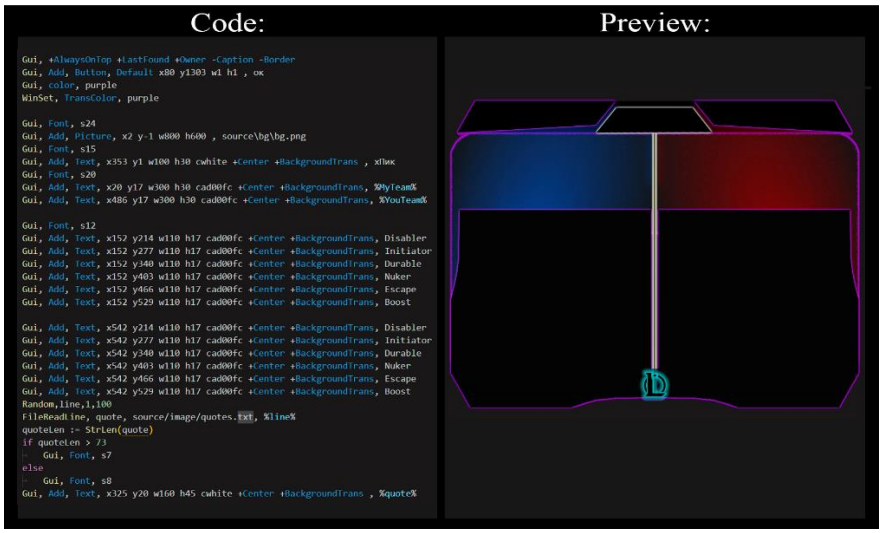

Рисунок 3.5 – Екранна форма процесу створення головного інтерфейсу

3.2.6 Екранна форма заповнення характеристик героїв

Ця екранна форма (рисунок 3.6) ілюструє фрагмент коду, який обробляє процес заповнення характеристик персонажів у додатку xPeek.

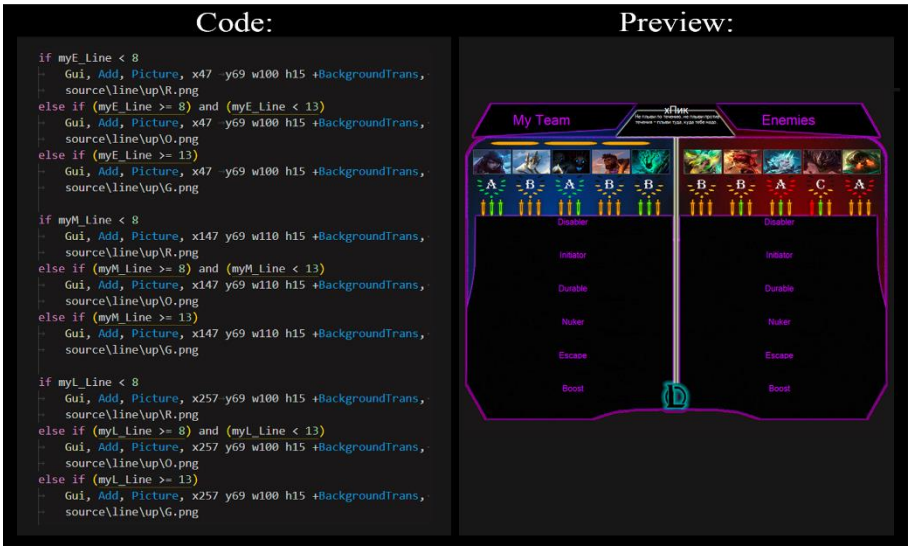

Рисунок 3.6 – Екранна форма заповнення характеристик героїв

Користувачі можуть спостерігати за кодом, що відповідає за збір та зберігання таких даних, як імена, позиції та атрибути персонажів. Відображений результат демонструє, як поля введення та мітки динамічно оновлюються введеною інформацією, дозволяючи користувачам візуалізувати деталі персонажів у реальному часі.

3.2.7 Екранна форма з фінальним інтерфейсом застосунку xPeek

На рисунку 3.7 зображена екранна форма де показано код, що відповідає за візуалізацію фінального інтерфейсу програми xPeek. Вона демонструє, як всі герої, їхні відповідні ранги, сила команди та індивідуальна сила героя відображаються в інтерфейсі. Користувачі можуть бачити імена героїв, їхні мініатюри та позиції. Відображений результат ілюструє процес створення інтерфейсу, який полегшує розуміння складу героїв у додатку xPeek.

Рисунок 3.7 – Екранна форма створення фінального інтерфейсу застосунку xPeek

4 ТЕСТУВАННЯ

4.1 Вступ до тестування

Тестування програмного забезпечення відіграє вирішальну роль у забезпеченні якості, надійності та продуктивності програмних додатків. Це систематичний процес виявлення дефектів, помилок і вразливостей у програмному забезпеченні з метою надання кінцевим користувачам надійного продукту без помилок. У цій статті ми розглянемо різні підходи до тестування програмного забезпечення, висвітлимо їхні сильні сторони, обмеження та найкращі практики. Розуміючи різні методології тестування, команди розробників можуть приймати обґрунтовані рішення про те, який підхід обрати, виходячи з їхніх конкретних вимог та обмежень проекту.

4.2 Види тестування

Далі будуть розглянуті різні типи підходів до тестування програмного забезпечення. Кожен підхід має свої унікальні переваги, обмеження та найкращі практики. Розуміючи ці різні методології, команди розробників можуть приймати обґрунтовані рішення про те, який підхід обрати, виходячи з їхніх конкретних вимог та обмежень проекту.

Почнемо з ручного тестування. Це традиційний підхід, коли тестувальники виконують тестування вручну. Вони слідують заздалегідь визначеним тестовим кейсам або досліджують програмне забезпечення інтуїтивно, щоб виявити дефекти і проблеми. Ручне тестування забезпечує гнучкість і адаптивність. Тестувальники можуть застосовувати свої знання предметної області, креативність і навички критичного мислення, щоб виявити потенційні проблеми. Це також дозволяє отримувати зворотній зв'язок в режимі реального часу і негайно повідомляти про помилки, сприяючи ефективній співпраці між тестувальниками і розробниками. Однак ручне тестування може забирати багато

часу, бути повторюваним і схильним до людських помилок. Воно найкраще підходить для тестування на ранніх стадіях, юзабіліті-тестування та сценаріїв, де людське судження та інтуїція є важливими.

Наступним є автоматизоване тестування, яке передбачає використання спеціалізованих інструментів і фреймворків для автоматичного виконання заздалегідь визначених тестових кейсів. Такий підхід спрощує процес тестування, збільшує покриття тестів і підвищує загальну ефективність. Автоматизовані тести можна запускати багаторазово, забезпечуючи узгодженість результатів і знижуючи ризик людської помилки. Це особливо корисно для регресійного тестування, тестування продуктивності та сценаріїв з великою кількістю завдань, що повторюються. Автоматизоване тестування також підтримує практики безперервної інтеграції та безперервної доставки. Однак воно вимагає попередніх інвестицій в інструментарій, розробку сценаріїв і підтримку. Автоматизоване тестування може не підходити для певних типів тестування, які вимагають людського судження, або для додатків з вимогами, що часто змінюються.

Іншим підходом є модульне тестування, яке фокусується на тестуванні окремих одиниць коду в ізоляції. Це гарантує, що кожна одиниця коду функціонує за призначенням і відповідає очікуваним специфікаціям. Зазвичай модульні тести пишуться самими розробниками і можуть бути автоматизовані за допомогою фреймворків для модульного тестування. Модульне тестування допомагає виявляти помилки на ранніх стадіях циклу розробки, сприяє підвищенню якості коду і підтримує рефакторинг коду. Воно також сприяє швидшому налагодженню, локалізуючи проблеми в конкретних блоках коду. Однак саме по собі модульне тестування не може гарантувати коректність роботи всієї системи, оскільки не враховує взаємодію між різними модулями та інтеграцію зовнішніх залежностей.

Далі йде інтеграційне тестування, цей підхід перевіряє взаємодію та інтерфейси між різними компонентами, модулями або підсистемами програмного додатку. Він зосереджений на тому, щоб переконатися, що

інтегрована система функціонує правильно і працює відповідно до очікувань. Інтеграційні тести можуть проводитися на різних рівнях, наприклад, тестування інтеграції API або тестування інтеграції бази даних, залежно від архітектури та вимог програми. Інтеграційне тестування допомагає виявити проблеми, пов'язані з потоком даних, протоколами зв'язку та інтероперабельністю системи. Воно також перевіряє правильність загальної поведінки та функціональності програми. Однак інтеграційне тестування може бути складним і трудомістким, вимагаючи ретельного планування, налаштування тестового середовища та координації між різними командами розробників.

Також є системне тестування, яке передбачає тестування всієї програмної системи в цілому. Воно оцінює відповідність системи заданим вимогам і бізнес-цілям. Системне тестування перевіряє наскрізну функціональність, взаємодію з користувачами та поведінку системи в різних сценаріях. Воно охоплює функціональне тестування, тестування продуктивності, тестування безпеки та юзабіліті-тестування, серед іншого. Системне тестування гарантує, що програмне забезпечення відповідає бажаним стандартам якості та надійно працює у призначеному для нього середовищі. Зазвичай його проводять спеціальні команди тестування або незалежні фахівці із забезпечення якості, забезпечуючи неупереджену оцінку готовності програмного забезпечення до випуску.

4.3 Сценарії тестування застосунку

Далі будуть розглянута низка сценаріїв для тестування навчального додатку. Тестування навчального додатку має важливе значення для забезпечення їхньої функціональності, зручності використання та ефективності у наданні освітнього контенту. Представлені тут сценарії охоплюють різні аспекти програми, включаючи її початкове налаштування, навігацію, відображення контенту, взаємодію та загальний користувацький досвід. Тестуючи ці сценарії, ми прагнемо виявити будь-які потенційні проблеми,

перевірити очікувану поведінку і переконатися, що додаток функціонує безперебійно, підтримуючи безперебійний і цікавий навчальний процес. Давайте зануримося в сценарії тестування цього освітнього додатку і дослідимо його різні функціональні можливості та очікувану поведінку.

Сценарій: відкриття програми та відображення початкової екранної форми.

Етапи тестування:

– запустіть програму;

– переконайтеся, що вікно програми з'являється з початковою екранною формою. Очікувана поведінка: вікно програми відкривається без помилок і відображає початкову екранну форму;

– переконайтеся, що зображення на екранній формі відповідає очікуваній поведінці. Очікувана поведінка: зображення на початковій екранній формі відповідає очікуваному вмісту;

Сценарій: перехід до різних екранних форм за допомогою клавіш зі стрілками.

Тестові кроки:

– відкрийте програму і переконайтеся, що відображається початкова екранна форма;

– натисніть клавішу зі стрілкою вправо і переконайтеся, що відображається наступна екранна форма. Очікувана поведінка: при натисканні клавіші зі стрілкою вправо програма переходить до наступної екранної форми в послідовності;

– натисніть клавішу зі стрілкою вліво і переконайтеся, що відображається попередня екранна форма. Очікувана поведінка: при натисканні клавіші зі стрілкою вліво програма переходить до попередньої екранної форми в послідовності;

– повторіть кроки для навігації по екранним формам і перевірте точність переходів між екранними формами. Очікувана поведінка: щоразу після

натискання клавіш зі стрілками програма точно переходить до відповідної наступної або попередньої екранної форми.

Сценарій: відображення екранних форм для додавання героя.

Етапи тестування:

– відкрийте програму і переконайтеся, що відображається початкова екранна форма;

– натисніть клавішу «+» на цифровій клавіатурі клавіатури;

– переконайтеся, що відображається екранна форма для додавання героя. Очікувана поведінка: при натисканні клавіші «+» програма переходить на екранну форму, де користувач може побачити реалізацію додавання нового героя;

– повторіть процес з іншими екранними формами, щоб переконатися в послідовній поведінці. Очікувана поведінка: при натисканні клавіш «Вліво» та «Вправо» програма відображає відповідні екранні форми з відповідним вмістом, пов'язаним з персонажем.

Сценарій: відображення екранних форм для додавання ворога.

Етапи тестування:

– відкрийте програму і переконайтеся, що відображається початкова екранна форма;

– натисніть клавішу «0» на цифровій клавіатурі клавіатури;

– переконайтеся, що відображається екранна форма з реалізацією коду для додавання ворога. Очікувана поведінка: при натисканні клавіші «-» програма переходить на екранні форми, де користувач може побачити реалізацію додавання ворога;

– переконайтеся, що зображення на екранній формі відповідає очікуваній поведінці. Очікувана поведінка: зображення, що відображається на екранній формі «Додати ворога», відповідає очікуваному вмісту, пов'язаному з ворогом;

– повторіть процес з іншими екранними формами, щоб переконатися в послідовності поведінки. Очікувана поведінка: щоразу, коли натискається

клавіша «-», програма відображає відповідні слайд з відповідним контентом, пов'язаним з ворогом.

Сценарій: приховати і показати додаток.

Етапи тестування:

– відкрийте додаток і переконайтеся, що відображається початкова екранна форма;

– натисніть клавішу «*» на цифровій клавіатурі клавіатури;

– переконайтеся, що вікно програми приховано. Очікувана поведінка: при натисканні клавіші «*» вікно програми буде приховано від користувача;

– натисніть клавішу «*» ще раз і переконайтеся, що вікно програми з'явилося. Очікувана поведінка: при повторному натисканні клавіші «*» вікно програми знову стає видимим;

– переконайтеся, що слайд залишається незмінним до і після приховування програми. Очікувана поведінка: Незалежно від видимості вікна програми, відображувана екранна форма залишається незмінним.

Сценарій: закриття програми.

Етапи тестування:

– відкрийте програму і перевірте, чи відображається початкова екранна форма;

– закрийте вікно програми;

– переконайтеся, що програма успішно завершує роботу без помилок або проблем. Очікувана поведінка: при закритті програма має завершити роботу плавно, без жодних повідомлень про помилки або неочікуваної поведінки.

ВИСНОВКИ

Розробка навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом була значною частиною цієї кваліфікаційної роботи. Метою цього проекту було створення комплексної та зручної платформи, що дозволяє ефективно вивчати та практично застосовувати AutoHotKey для автоматизації завдань та взаємодії з графічними інтерфейсами.

При аналізі предметної області було досліджена важливість автоматизації та скриптових мов у розробці програмного забезпечення. Мови сценаріїв відіграють життєво важливу роль в автоматизації повторюваних завдань і підвищенні продуктивності. AutoHotKey, зокрема, має багату історію і пропонує потужні функції та можливості, які роблять її цінним інструментом для автоматизації графічного інтерфейсу. Інтеграція AutoHotKey з графічним інтерфейсом була ключовим аспектом цього проекту. Були обговорені переваги та варіанти використання AutoHotKey в цьому контексті, а також надали приклади скриптів AutoHotKey для демонстрації його практичного застосування. Можливість взаємодії з графічними інтерфейсами користувача має важливе значення для покращення користувацького досвіду та впорядкування робочого процесу при розробці програмного забезпечення.

Додаток xPeek, розроблений в рамках цього проекту, слугував основою для демонстрації використання AutoHotKey та взаємодії з графічним інтерфейсом. Його мета полягала в тому, щоб надати користувачам практичний досвід навчання, дозволяючи їм досліджувати і практикувати концепції AutoHotKey в змодельованому середовищі. Додаток пропонував такі функції, як створення та управління героями, командами та елементами інтерфейсу, демонструючи потенціал AutoHotKey в реальних сценаріях.

Крім того, був розроблений окремий навчальний додаток, який зосереджувався на створенні графічних інтерфейсів. Ця програма надавала покрокове керівництво по створенню інтерфейсів за допомогою AutoHotKey, що

дозволило користувачам глибше зрозуміти тонкощі, пов'язані з проектуванням та реалізацією елементів графічного інтерфейсу.

Аналіз коду навчального додатку виявив добре структуровану та організовану реалізацію, що забезпечує ясність і легкість розуміння. Для демонстрації коду та відповідних результатів в навчальному застосунку були представлені різні екрани, що дозволило користувачам візуалізувати результати різних процесів. Ці екрани охоплювали такі важливі аспекти, як зміна іконки трею, створення нових героїв і команд, вибір героїв, рендеринг основного інтерфейсу і заповнення характеристик героїв. Кожен екран давав уявлення про логіку коду та його вплив на функціональність програми.

Процес розробки навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом стала цінним ресурсом для осіб, які прагнуть набути навичок автоматизації графічного інтерфейсу. Програмне забезпечення пропонує комплексну навчальну платформу, що поєднує теоретичні знання з практичними прикладами і дозволяє користувачам досліджувати можливості AutoHotKey у змодельованому середовищі. Завдяки впровадженню xPeek і навчальних додатків, користувачі можуть отримати практичний досвід і розвинути навички використання AutoHotKey для автоматизації завдань і взаємодії з графічними інтерфейсами. Проект успішно досягнув своїх цілей, сприяючи поглибленню знань і навичок у сфері автоматизації графічних інтерфейсів і мов сценаріїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Automate The Planet. Be more efficient with autohotkey scripts. Automate The Planet. URL: https://www.automatetheplanet.com/be-more-efficient-with-autohotkey-scripts/ (дата звернення: 09.06.2023).

2. Computer Hope. AutoHotKey. Computer Hope. URL: https://www.computerhope.com/jargon/a/autohotkey.htm (дата звернення: 09.06.2023).

3. Heavy.AI. Graphical user interface definition. Heavy.AI. URL: https://www.heavy.ai/technical-glossary/graphical-user-interface (дата звернення: 09.06.2023).

4. Indeed. What is a GUI (graphical user interface)? Definition, elements and benefits. Indeed. URL: https://www.indeed.com/career-advice/career-development/gui-meaning (дата звернення: 09.06.2023).

5. Javatpoint. What is scripting Language?. Javatpoint. URL: https://www.javatpoint.com/what-is-a-scripting-language (дата звернення: 09.06.2023).

6. Johnson K. What is a GUI and why is it important?. WikiJob. URL: https://www.wikijob.co.uk/industry/it-technology/what-is-a-gui (дата звернення: 09.06.2023).

7. Kaholo. The role of automation in software development lifecycle. Kaholo. URL: https://kaholo.io/blog/the-role-of-automation-in-software-development-lifecycle/ (дата звернення: 09.06.2023).

8. Kimberly Forsythe Lexi Demers. What is scripting language?. Career Karma. URL: https://careerkarma.com/blog/what-is-a-scripting-language/ (дата звернення: 09.06.2023).

9. Motiso D. What is a scripting language? (with types and advantages). Indeed. URL: https://www.indeed.com/career-advice/career-development/what-is-scripting-language (дата звернення: 09.06.2023).

10. Nicholas Xuan Nguyen. Autohotkey: the ultimate guide. AtaLearning. URL: https://adamtheautomator.com/autohotkey/ (дата звернення: 09.06.2023).

11. Scalerandi D. Is automation the key to the future of software development?. BairesDevBlog. URL: https://www.bairesdev.com/blog/automation-the-key-to-software-development/ (дата звернення: 09.06.2023).

12. Terrence D. Automate all the things: an autohotkey primer for developers. Visual Studio Magazine. URL: https://visualstudiomagazine.com/articles/2015/06/01/autohotkey-primer.aspx (дата звернення: 09.06.2023).

13. Wate Y. AutoHotkey: a perfect tool to automate tasks on windows. Techpp. URL: https://techpp.com/2020/12/09/autohotkey-windows-automation-guide/ (дата звернення: 09.06.2023).

14. Wikipedia. AutoHotKey. Wikipedia. URL: https://en.wikipedia.org/wiki/AutoHotkey (дата звернення: 09.06.2023).

15. Wissen. The role of automation in software development. Wissen. URL: https://www.wissen.com/blog/the-role-of-automation-in-software-development (дата звернення: 09.06.2023).

ДОДАТКИ

ДОДАТОК А

ЗАТВЕРДЖЕНО

44165850.94106 -01-ЛЗ

Програмне забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows

Технічне завдання

44165850.94106-01

Листів 12

2023

44165850.94106-01

ЗМІСТ

1. ВВЕДЕННЯ

Додаток для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows.

Основна термінологія:

Графічний інтерфейс – є способом взаємодії користувача з комп'ютерною системою чи програмою за допомогою графічних елементів, таких як вікна, кнопки, меню, текстові поля та інші візуальні компоненти. GUI надає зручний та інтуїтивно зрозумілий спосіб управління програмою шляхом використання миші, клавіатури або інших пристроїв введення.

Скриптова мова – мова яка є високорівневою програмною мовою, призначеною для автоматизації завдань або виконання певних дій. Вона зазвичай використовується для створення скриптів, які виконуються інтерпретатором без необхідності компіляції.

Причина виникнення продукту – присутність аналогів, які все ж таки не спрямовані на задовільнення функціональних потреб замовника.

Область застосування – застосунок розроблений спеціально для людей, які бажають навчитися створювати графічні інтерфейси за допомогою мови програмування AutoHotkey. AutoHotkey є потужним скріптовим мовою, яка дозволяє автоматизувати завдання на комп'ютері, включаючи створення інтерактивних вікон, кнопок, меню та інших елементів графічного інтерфейсу.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є навчальний план для спеціальності 121 Інженерія програмного забезпечення, затверджений ректором Українського державного університету науки і технологій 30.06.19 р.

Тема дипломної роботи – «Розробка навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows».

Керівник – доцент Андрющенко В.О.

3.ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – розробка навчального програмного забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows.

Навчальне програмне забезпечення, створене для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows, має наступні функціональні призначення:

– вивчення синтаксису та основ мови AutoHotKey: програмне забезпечення надає навчальні матеріали, приклади коду та пояснення, що допомагають користувачам ознайомитися з синтаксисом мови AutoHotKey та основними поняттями, необхідними для роботи з графічним інтерфейсом Windows;

– візуалізація дій та результатів: програмне забезпечення надає можливість візуалізувати дії, які виконуються з використанням мови AutoHotKey, на графічному інтерфейсі Windows. Це допомагає користувачам краще розуміти, як код AutoHotKey взаємодіє з різними елементами графічного інтерфейсу та які результати можуть бути досягнуті.

Експлуатаційне призначення: навчити користувачів використовувати мову AutoHotKey для розробки графічних інтерфейсів на платформі Windows.

.

4. ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Вимоги до характеристик наступні:

– забезпечення можливості створення та відображення графічних інтерфейсів Windows з використанням мови AutoHotKey;

– демонстрація основних елементів графічного інтерфейсу, таких як кнопки, поля введення, випадаючі списки та інші;

– можливість взаємодії з графічними елементами за допомогою скриптів AutoHotKey;

– надання зразків коду та прикладів використання різних функцій AutoHotKey для роботи з графічним інтерфейсом.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

– забезпечення стабільності роботи програмного продукту під час створення, відображення та взаємодії з графічним інтерфейсом;

– контроль вхідних та вихідних даних для забезпечення коректної роботи програмного продукту;

– захист від несанкціонованого копіювання програмного забезпечення.

4.3 Умови експлуатації

Умови експлуатації наступні:

– підтримка роботи на операційних системах Windows XP, 7, 8, 10, 11;

– наявність платформи AutoHotKey для виконання скриптів;

Обслуговування не потрібне.

Для роботи із ПЗ достатньо однієї людини, що має досвід роботи із ПК та ознайомлена із керівництвом користувача.

4.4 Вимоги до складу і параметрів технічних засобів

Склад технічних засобів:

– процесор з тактовою частотою 1 ГГЦ або вище.

– доступ до мережі Інтернет;

– 512 мб. місця на накопичувачі; - 4096 мб. оперативної пам’яті;

– клавіатура;

– миша№

– монітор.

4.5 Вимоги до інформаційної і програмної сумісності

Програма має функціонувати під управлінням ОС Windows Xp\7\8\10\11.

На системі має бути встановлений AutoHotKey 2.0.

4.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних). На упаковці повинно бути вказана назва продукту, номер версії (якщо вона змінювалась), мінімальні системні вимоги. На зворотній стороні упаковки вказується розробник та його юридична адреса.

4.7 Вимоги до транспортування і зберігання

Транспортування повинне забезпечувати збереження програмного продукту, його цілісність і запобігання несанкціонованого доступу до нього. Програмний виріб міститься на хмарному носії, переданий на флешці.

5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача.

Програмна документація повинна відповідати вимогам ДСТУ [1].

6. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

В табл. 1 приведені стадії та етапи розробки.

Таблиця 1. Стадії та етапи розробки

Етапи розробки	Строки виконання
Обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	01.03.23 – 10.04.23
Розробка і узгодження ТЗ	
Розробка та програмування логіки програми	10.04.23-06.05.23
Відлагодження програми	

7. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Проводиться перевірка коректного виконання програмою закладених у ній функцій, тобто здійснюється функціональне тестування програми. Також здійснюється візуальна перевірка інтерфейсу програми. Строки проведення випробувань обговорюються додатково.

Контроль виконання здійснює керівник розробки доц. Андрющенко В. О.

8. БІБЛІОГРАФІЧНИЙ СПИСОК

1. AutoHotKey. AutoHotKey documentation. AutoHotKey. URL: https://www.autohotkey.com/docs/v2/..

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
Анатолій РАДКЕВИЧ
30.06.23

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДЕМОНСТРАЦІЇ ВИКОРИСТАННЯ МОВИ AUTOHOTKEY В РОБОТІ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ WINDOWS

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

4416580.94106-01-ЛЗ

Представники підприємства-розробника
Завідувач кафедри
Вадим ГОРЯЧКІН
30.06.23

Керівник розробки
Вадим АНДРЮЩЕНКО
30.06.23

Виконавець
Дмитро КАНАРЕЙКІН
30.06.23

Нормоконтролер
Світлана ВОЛКОВА
30.06.23

2023

ДОДАТОК Б

ЗАТВЕРДЖЕНО

44165850.94106 -01 12 01-ЛЗ

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДЕМОНСТРАЦІЇ ВИКОРИСТАННЯ МОВИ AUTOHOTKEY В РОБОТІ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ WINDOWS

Текст програми

44165850.94106-01 12 01

Листів 8

2023

ЗМІСТ

АНОТАЦІЯ

Документ 44165850.94103-01 12 01 “ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДЕМОНСТРАЦІЇ ВИКОРИСТАННЯ МОВИ AUTOHOTKEY В РОБОТІ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ WINDOWS Текст програми” входить до складу програмної документації на додаток, що реалізують програмне забезпечення для демонстрації використання мови AutoHotKey в роботі з графічним інтерфейсом Windows.

У даному документі представлений текст програми. Програма написана на мові AutoHotKey. Об'єм пам'яті, що займають програми комплексу, складає 10 Мб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі MS WINDOWS XP/7/8/8.1/10/11.

КОД ПРОГРАМИ

main.ahk:

```
img_line_num = 1
file_name = start.txt

start = source\sounds\start.mp3
error = source\sounds\error.mp3
trash = source\sounds\trash.mp3
success = source\sounds\success.mp3

sound_up = %start%
sound_down = 0
show_hide = 0

FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
Gui, +AlwaysOnTop +LastFound +Owner -Caption -Border
Gui, Add, Picture, x0 y0 w1119 h657 vMyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\%img_name%
GUI, Show, x218 y115 h657 w1119
OnMessage(0x201, "WM_LBUTTONDOWN")
WM_LBUTTONDOWN()
{
   PostMessage, WM_NCLBUTTONDOWN := 0xA1, HTCAPTION := 2
}
Return

1::
file_name = start.txt
img_line_num = 1
sound_up = %start%
sound_down = 0
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
```

```
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return

2::
file_name = show_pick.txt
img_line_num = 1
sound_up = 0
sound_down = 0
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return

NumPadAdd::
file_name = addmyhero.txt
img_line_num = 1
sound_up = %success%
sound_down = %error%
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return

NumPadSub::
file_name = addyouhero.txt
img_line_num = 1
sound_up = %success%
sound_down = %error%
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return

+NumPadAdd::
file_name = del_hero.txt
```

```
img_line_num = 1
sound_up = 0
sound_down = %trash%
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return

+NumPadSub::
file_name = del_enemy.txt
img_line_num = 1
sound_up = 0
sound_down = %trash%
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
return


Right::
img_line_num++
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
Return

Left::
if img_line_num = 1
return
img_line_num--
FileReadLine, img_name, source\file name\%file_name%, %img_line_num%
GuiControl,                          ,                          MyPicture,
C:\Users\sundunchan\Desktop\diploma_order\source\screen\%img_name%
Return

Up::
```

```
SoundPlay, %sound_up%
return

Down::
SoundPlay, %sound_down%
return

NumPadMult::
show_hide := show_hide + 1
if (show_hide = 1)
{
GuiControl, , MyPicture, source\screen\hide_show.png
Gui, Show, %possgui%
}
else if (show_hide = 2)
{
Gui, Cancel
show_hide = 0
}
return

GuiClose:
ExitApp
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
Анатолій РАДКЕВИЧ
30.06.23

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ДЕМОНСТРАЦІЇ ВИКОРИСТАННЯ МОВИ AUTOHOTKEY В РОБОТІ З ГРАФІЧНИМ ІНТЕРФЕЙСОМ WINDOWS

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
4416580.94106-01 12 01-ЛЗ

Представники підприємства-розробника
Завідувач кафедри
Вадим ГОРЯЧКІН
30.06.23

Керівник розробки
Вадим Андрющенко
30.06.23

Виконавець
Дмитро КАНАРЕЙКІН
30.06.23

Нормоконтролер
Світлана ВОЛКОВА
30.06.23

2023