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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Брайко Дмитра Віталійовича*
(ПІБ)

академічної групи *121-19-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка бота для MMO RPG World of Warcraft з використанням C#*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Бердник М.Г.*			
розділів:				
спеціальний	*проф. Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А*			
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-2* (група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка бота для ММО RPG World of Warcraft з використанням C#*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проектно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Брайко Д.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 111 с., 19 рис., 3 дод., 30 джерел.

Об'єкт розробки: бот для MMO RPG World of Warcraft.

Метою даної кваліфікаційної роботи є створення бота для MMO RPG World of Warcraft з використанням C#. Основний функціонал додатка полягає в можливості автоматично використовувати заклинання для облегшення гри чи покращення ігрових результатів.

У вступній частині кваліфікаційної роботи проведено аналіз та розглянуто сучасний стан проблеми, визначено конкретну мету дослідження та галузь її застосування, наведено обґрунтування актуальності обраної теми та деталізовано постановку задачі.

У першому розділі кваліфікаційної роботи було проведено детальний аналіз предметної галузі, розглянуто актуальність поставленої задачі та цілей розробки, сформульовано конкретну постановку завдання, а також визначено вимоги до програмної реалізації, використовуваних технологій та програмних засобів.

У другому розділі кваліфікаційної роботи було проаналізовано наявні рішення, обрані платформи для розробки, проведено проектування та розробка програми. Також була надана детальна інформація про роботу програми, включаючи опис алгоритму та структури її функціонування. В розділі було розглянуто процес виклику та завантаження програми, визначено вхідні та вихідні дані, а також надано характеристику параметрів технічних засобів.

У економічному розділі кваліфікаційної роботи була визначена трудомісткість розробленої інформаційної системи. Крім того, було проведено розрахунок вартості роботи зі створення програми та оцінено необхідний час для її реалізації.

Практичне значення полягає у розробці додатку зі зрозумілим інтерфейсом, який дозволяє користувачам безпечно грати у World of Warcraft використовуючи бота для автоматичного натискання клавіш у процесі гри.

Список ключових слів: КОМП’ЮТЕР, WOW, WORLDOFWARCRAFT, БОТ, АВТОРОТАЦІЯ, C, C#, LUA, KEYBOARDEMULATOR.

ABSTRACT

Explanatory note: 111 p., 19 fig., 3 appx., 30 sources.

Development object: The development of a bot for the MMO RPG World of Warcraft is considered using the C# programming language.

The purpose of this qualification work is to create a bot for MMO RPG World of Warcraft using C#. The main functionality of the application is the ability to automatically use spells to facilitate the game or improve game results.

In the introductory part of the qualification work, an analysis was carried out and the current state of the problem was considered, the specific purpose of the research and the field of its application were defined, the justification of the relevance of the chosen topic was given, and the statement of the problem was detailed.

In the first section of the qualification work, a detailed analysis of the subject area was conducted, the relevance of the task and development goals was considered, a specific statement of the task was formulated, and the requirements for software implementation, used technologies and software tools were determined.

In the second section of the qualification work, available solutions were analyzed, platforms were selected for development, design and development of the program was carried out. Detailed information about the operation of the program was also provided, including a description of the algorithm and structure of its operation. In the section, the process of calling and loading the program was considered, input and output data were defined, and the characteristics of the parameters of the technical means were provided.

In the economic section of the qualification work, the labor intensity of the developed information system was determined. In addition, the calculation of the cost of work on the creation of the program was carried out and the time required for its implementation was estimated.

The practical value is to develop an application with a clear interface that allows users to safely play World of Warcraft using a bot to automatically press keys while playing.

Keywords: PC, WOW, WORLDOFWARCRAFT, BOT, AUTOROTATION, C, C#, LUA, KEYBOARDEMULATOR.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

C - мова програмування загального призначення, яка відома своєю ефективністю, можливістю роботи з пам'яттю на низькому рівні та широким спектром застосувань.

C# - мова програмування, розроблена компанією Microsoft яка стала складовою платформи .NET.

Lua - легка скриптова мова програмування, яка має простий синтаксис, компактний розмір і високу швидкодію.

WoW (World of Warcraft) - це популярна масова багатокористувацька онлайн-гра, розроблена і випущена компанією Blizzard Entertainment.

Аддон (Addon) - в контексті комп'ютерних ігор відноситься до програмного розширення, яке додає додатковий функціонал до основної гри.

WoWAPI (World of Warcraft API) - це набір програмних інтерфейсів, який надає розробникам доступ до різноманітної інформації та функціоналу гри World of Warcraft.

Вступ

В сучасному світі комп'ютерні ігри стають все більш популярними і надихають людей на розробку нових інноваційних рішень. Особливе місце серед таких ігор займають MMO RPG, де гравцям пропонуються величезні віртуальні світи для дослідження і боротьби. Однак, у багатьох випадках геймери зіштовхуються з повторюваними діями, такими як натискання клавіш для виконання певних дій, що може стати втомлюючим і монотонним завданням.

З метою полегшення цього процесу і покращення геймплею, у даній кваліфікаційній роботі розглядається розробка бота для MMO RPG World of Warcraft, використовуючи мову програмування C#, C та Lua. Головною метою бота є автоматизація натискання клавіш для використання здібностей персонажа, що дозволить гравцеві зосередитись на стратегічних аспектах гри і використати свій час більш ефективно.

Програмне забезпечення драйвер на мові програмування C для імітації натискання клавіш. Застосування мови C# дозволяє розробити програму, яка встановлює драйвер, та відповідає за зв’язок драйверу з грою. Дана робота включає в себе аналіз поточної інформації в грі за допомогою аддона, розробленого на мові програмування Lua та WoWAPI, який змінює колір кількох пікселів на екрані для аналізу їх у програмі. Це дозволяє отримувати необхідні дані з гри, аналізувати їх і приймати відповідні рішення щодо натискання клавіш.

У кваліфікаційній роботі пропонується інноваційний підхід до полегшення геймплею в MMO RPG World of Warcraft. Застосування бота, розробленого на основі мови програмування C# та використання драйвера для імітації натискання клавіш, дозволить гравцям зосередитись на важливих аспектах гри, підвищуючи ефективність і задоволення від геймплею.

Розроблений бот з використанням C# і його інтеграція з грою дозволяють автоматизувати певні аспекти геймплею, що сприяє комфорту та задоволенню від гри. Пропонується внести новітні технології в галузь геймінгу, демонструючи

потенціал мови програмування C# та інтеграції з ігровими середовищами. Дослідження в цій галузі можуть бути корисними для розробників і гравців, які прагнуть полегшити геймплей і вдосконалити власні навички в MMO RPG World of Warcraft.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної області

MMO RPG (Massively Multiplayer Online Role-Playing Game) - це жанр комп'ютерних ігор, де гравці взаємодіють один з одним у величезному віртуальному світі. World of Warcraft (WoW) є однією з найпопулярніших MMO RPG грою, розробленою компанією Blizzard Entertainment. Гра має велику кількість активних гравців по всьому світу і пропонує широкий спектр можливостей для вирішення завдань, боротьби з ворогами та експлорації великого ігрового світу.

Ротація заклинань персонажа в MMO RPG відноситься до повторюваних дій, які гравці виконують для досягнення максимальної ефективності у бою. Цей процес вимагає постійного натискання певних клавіш у правильному порядку, щоб застосовувати навички та завдавати шкоду ворогам. Однак, така монотонна дія може бути дуже втомлюючою та відволікаючою для гравців.

Розробка бота для автоматизації ротації персонажа може спростити геймплей та полегшити завдання гравця. Використання мови програмування C# дозволяє розробити програмне забезпечення, яке може взаємодіяти з грою та імітувати натискання клавіш. Це дає можливість гравцям зосередитись на стратегічних аспектах гри, таких як планування та координація дій з командою, замість повторюваного натискання клавіш.

При розробці бота використовуються додаткові технології та інструменти для взаємодії з грою World of Warcraft. Наприклад, використання мови програмування Lua дозволяє створювати аддони для гри, які забезпечують можливість отримання інформації з гри та її аналізу. Ця інформація використовується ботом для прийняття рішень щодо натискання клавіш відповідно до поточного стану гри.

Окрім того, бот має можливість фільтрувати деякі клавіші, щоб уникнути нежаданих дій або конфліктів з іншими програмами або аддонами. Це забезпечує безпеку та стабільність роботи бота, дозволяючи гравцям використовувати його без обтяження.

Для зручного керування ботом в грі створюється маленька панель, де гравець може налаштовувати різні параметри для відповідної ситуації. Це дозволяє гравцям вибирати і налаштовувати різні стратегії та тактики залежно від поточних умов гри.

1.2. Призначення розробки та область застосування

Призначенням розробки бота для MMO RPG World of Warcraft з використанням мови програмування C# є спрощення геймплею гравців шляхом автоматизації ротації персонажа. Цей бот дозволяє гравцям зосередитись на важливих аспектах гри, таких як стратегія, планування та спілкування з командою, замість повторюваного натискання клавіш.

Область застосування розробленого бота охоплює MMO RPG World of Warcraft, яка є однією з найпопулярніших та широко відомих онлайн-ігр. Гравці, які активно беруть участь у цій грі та виконують повторювані дії під час боїв, зможуть скористатись перевагами, які надає розроблений бот.

Застосування даної розробки дозволить гравцям ефективніше виконувати завдання в грі, полегшити геймплей та забезпечити більше задоволення від грального процесу. Бот може бути корисним для гравців, які прагнуть підвищити свою продуктивність, оптимізувати ресурси та вдосконалити свої геймінгові навички.

Розробка бота на основі мови програмування C# та використання драйвера для імітації клавішних натискань може бути прикладом застосування технологій у галузі геймінгу.

Таким чином, розробка бота для MMO RPG World of Warcraft з використанням мови програмування C# має потенціал знайти застосування серед широкої геймерської спільноти, яка активно грає у цю гру. Гравці, які шукають ефективні способи полегшити геймплей та забезпечити більш комфортну ігральну досвід, можуть скористатись цим ботом для автоматизації ротації персонажа.

Додатково, розробка бота для MMO RPG World of Warcraft на базі мови програмування C# може викликати інтерес серед програмістів та розробників, які цікавляться розробкою програмного забезпечення для ігрових додатків. Цей проект може послужити прикладом використання програмування та інтеграції з ігровими середовищами для створення корисних інструментів для геймерів.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

- освітня програма 121 Інженерія програмного забезпечення;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023р;
- завдання на кваліфікаційну роботу на тему « Розробка бота для MMO RPG World of Warcraft з використанням C#».

1.4. Постановка завдання

Розробити бота для MMO RPG World of Warcraft з використанням C#.

Бот повинен працювати на будь-яких комп’ютерах, які підключені до інтернету та мають можливість запуску гри “World of Warcraft” .

Основним завданням розробки бота для MMO RPG World of Warcraft з використанням мови програмування C# є створення функціональності, що дозволить автоматизувати ротацію заклинань персонажа.

1.4.1. Функціональні особливості та можливості

Основним завданням розробки бота для MMO RPG World of Warcraft з використанням мови програмування C# є створення функціональності, що дозволить автоматизувати ротацію заклинань персонажа тобто послідовне натискання необхідних клавіш для виконання атак чи навичок відповідно до налаштувань та поточного стану гри для того щоб спростити геймплей для гравців.

1.4.2. Опис інтерфейсу користувача

1) Інтерфейс программи для встановлення та запуску:

- головне вікно з усіма налаштуваннями, кнопками закриття і згортання программи;
- кнопка Install – встановлює драйвер для емуляції клавіатури, перезапускає Windows якщо не увімкнений режим для тестування драйверів, встановлює сервіс для запуску драйверу;
- кнопка Uninstall – видаляє драйвер та сервіс який його запускає;
- NumericTextBox – текстове поле в яке можна ввести тільки цифри, відповідає за швидкість натискання клавіш(в мілісекундах);
- прапорець “Suspend when alt pressed” – перемикач який активує чи деактивує роботу программи при натисканні клавіши alt;
- прапорець “Suspend when ctrl pressed” – перемикач який активує чи деактивує роботу программи при натисканні клавіши ctrl;
- прапорець “Suspend when shift pressed” – перемикач який активує чи деактивує роботу программи при натисканні клавіши shift;

- кнопка Start – запускає роботу программи, перед цим на все вікно виводиться напис “Click any keyboard key to detect your keyboard” для того щоб программа могла визначити яка саме клавіатура зараз використовується у користувача;
- текстове поле “Clicks done count” – відображає скільки кліків зробила программа за час своєї роботи.

2) Інтерфейс аддону у грі:

- головне вікно яке можна рухати зажавжи праву кнопку миші. Воно має кнопки на яких відображені деякі з заклинань, використовуються для зміни пріорітетів залежно від ситуації в грі. Також мають tooltip з описом цієї кнопки (тобто при наведенні на кнопку з'являється маленьке вікно біля кнопки з текстом);
- маленке віконце з іконкою що відображає поточне заклинання що натискає программа;
- кнопка біля мінімапи - відкриває вікно з деякими налаштуваннями:
    - полоса з повзунком – відповідає за швидкість з якою аддон обновляє інформацію, при великій швидкості на слабих комп’ютерах можуть виникати зависання;
    - прапорець що скриває головне вікно з кнопками для зміни пріорітету заклинань;
    - прапорець що скриває вікно з іконкою поточного заклинання;
    - прапорець що відповідає за автоматичне відключення після виходу з режиму бою;
    - прапорець що відповідає за призупинення на короткий час, якщо користувач намагається натиснути одне з деяких заклинань.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Бот повинен працювати на будь-яких комп’ютерах, які підключені до інтернету та мають можливість запуску гри “World of Warcraft” та мають встановленний “.Net”.

1.5.2. Вимоги до інформаційної безпеки

Бот повинен забезпечувати безпеку гри та уникати викривання у застосунку заборонених методів.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для використання технічний засіб користувача повинен відповідати таким мінімальним технічним вимогам:

- мати операційну систему Windows 7-11 (64-bit);
- мати процессор 4 Cores, 3.0 GHz processor 4th Generation Intel® Core™ або AMD Ryzen™ Zen;
- мати відеокарту з підтримкою DirectX® і не менш ніж 4 гб відеопам’яті;
- мати не менше 8 ГБ оперативної пам'яті;
- мати не менше 120 ГБ вільного місця на жорсткому диску;
- мати можливість виходу у інтернет;
- мати мінімальну розподільну здатність екрану 1280x720.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для встановлення програми треба мати права адміністратора у системі Windows, щоб була можливість встановити драйвер. Необхідною вимогою до програмного забезпечення є встановлена гра World of Warcraft. Обов’язковою вимогою для функціонування програми є встановлений .Net Framework 3.5 або вище.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою даної кваліфікаційної роботи є створення бота для MMO RPG World of Warcraft з використанням C#. Основний функціонал додатка полягає в можливості автоматично використовувати заклинання для облегшення гри чи покращення ігрових результатів.

Функціональне призначення такого бота включає наступні аспекти:

1. Автоматичне виконання заклинань: бот повинен мати здатність виконувати певну послідовність заклинань, яка забезпечує оптимальну ротацію або дії героя в грі.

2. Керування ресурсами: бот повинен ефективно керувати ресурсами, такими як мана, енергія, які необхідні для виконання заклинань. Він повинен забезпечити оптимальне використання ресурсів для максимальної ефективності героя.

3. Налаштування та персоналізація: бот повинен мати можливості для налаштування його поведінки, наприклад, встановлення пріоритетів заклинань, визначення умов для використання певних заклинань, вибір ротацій або стратегій тощо. Користувач повинен мати можливість персоналізувати бота відповідно до своїх вподобань та потреб у грі.

Експлуатаційне призначення бота для автоматизації ротації заклинань у World of Warcraft визначається тим, що програма надає користувачеві наступні переваги та можливості:

1. Оптимізація геймплею - бот допомагає гравцеві виконувати складні ротації заклинань автоматично, забезпечуючи оптимальну дію героя. Це дозволяє гравцеві сконцентруватися на стратегічних аспектах гри, таких як тактика бою та прийняття рішень, замість повторного виконання однотипних дій.

2. Ефективність та точність - бот може виконувати послідовності заклинань з високою швидкістю та точністю, уникнувши помилок, які можуть стати людським фактором. Це дозволяє досягти максимальної продуктивності і результативності героя в грі.
3. Виконання монотонних завдань - бот здатен виконувати рутинні та повторювані завдання, такі як ротація заклинань, без втоми чи нудоти. Гравець може використовувати бота для автоматизації таких завдань, звільнивши себе від монотонної роботи.
4. Зростання продуктивності - бот може допомогти гравцю збільшити свою продуктивність шляхом ефективного використання ресурсів, оптимальної ротації заклинань та реагування на важливі події в грі. Це може позитивно позначитися на успішності гравця та досягненні його ігрових цілей.

Отже, функціональне призначення програми для автоматизації ротації заклинань у грі World of Warcraft полягає у наданні гравцю засобів для автоматичного виконання оптимальних ротацій заклинань та керування ресурсами. Експлуатаційне призначення полягає у полегшенні геймплею, зростанні продуктивності та ефективності гравця, а також у виконанні монотонних завдань, що дозволяє гравцеві більш повноцінно насолоджуватися грою.

2.2. Опис застосованих математичних методів

Під час проектування та розробки цього веб-додатка не використовувалися математичні методи або алгоритми. Розрахунки та обчислення, які використовувалися, обмежувалися простими арифметичними діями.

2.3. Опис використаної архітектури та шаблонів проектування

При проектуванні та розробці даного проекту використано архітектурний шаблон MVC (Model-View-Controller) та його варіацію, відому як MVVM (Model-View-ViewModel). У цьому шаблоні компонент ‘App’ виступає в ролі контролера, який керує взаємодією між моделлю (state) і представленням (відображенням).

MVC (Model-View-Controller) - це архітектурний шаблон проектування, який допомагає розділити компоненти додатку на три основні ролі: Модель (Model), Представлення (View) та Контролер (Controller). Кожна з цих ролей відповідає за свої відповідності в програмі та має чіткі обов'язки. Основна ідея MVC полягає в розділенні логіки програми, відображення даних та керування взаємодією користувача з додатком (рис. 2.1).

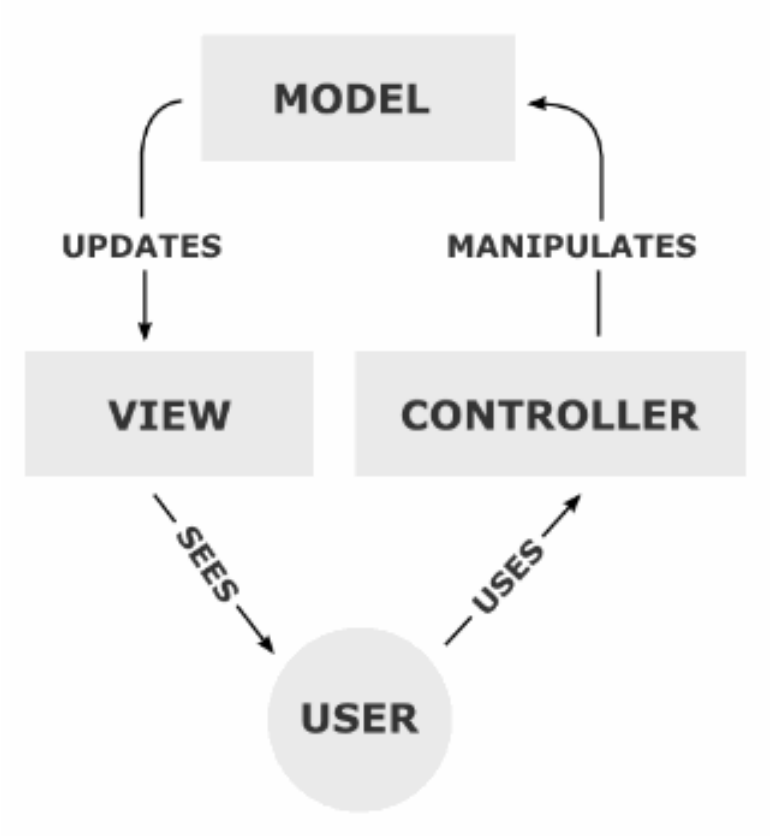

Рис. 2.1. Графічне представлення шаблону MVC

Основні компоненти MVC:

1. Модель (Model) - модель представляє дані та бізнес-логіку додатку. Вона відповідає за збереження, обробку та маніпулювання даними, виконання операцій з ними та відправку сповіщень про зміни даних. Модель функціонує незалежно від інших компонентів і може існувати самостійно.
2. Представлення (View) - представлення відповідає за візуалізацію даних та інтерфейс користувача. Воно отримує дані з моделі та відображає їх у відповідному форматі, щоб користувач міг взаємодіяти з додатком. Представлення також може генерувати події (наприклад, натискання кнопок) для обробки контролером.
3. Контролер (Controller) - контролер відповідає за обробку подій, взаємодію з користувачем та оновлення моделі та представлення. Він слухає події від представлення, виконує відповідні дії та змінює стан моделі або представлення. Контролер також може надсилати команди моделі для виконання розрахунків або оновлення даних.

В контексті C#, компоненти можна розглядати як комбінацію представлення та контролера, оскільки вони відповідають як за відображення даних, так і за обробку подій. Компоненти в C# (наприклад, класи, які успадковуються від Control або UserControl) можуть відображати дані з моделі (наприклад, властивості) і сприймати взаємодію користувача для подальшої обробки контролером (наприклад, обробники подій).

Переваги MVC:

- оптимальне розділення обов'язків - модульність MVC дозволяє чітко розподілити обов'язки між компонентами, сприяючи легкості розробки та підтримки коду;
- підвищена перевикористовуваність - шаблон MVC сприяє повторному використанню компонентів, оскільки модель та представлення можна використовувати у різних контекстах;

- забезпечення тестованості - компоненти моделі, представлення та контролера можуть бути тестовані окремо, спрощуючи процес тестування додатку.

Недоліки MVC:

- потенційна складність - реалізація MVC може потребувати додаткових зусиль і збільшення обсягу коду порівняно з менш складними шаблонами проектування;
- зайва залежність - некоректний розподіл обов'язків або злиття ролей можуть призвести до надмірної залежності між компонентами, ускладнюючи розробку та підтримку.

Загалом, MVC є потужним та ефективним шаблоном проектування, який надає незамінну підтримку для роботи зі складними додатками, а також забезпечує чітке розділення логіки, представлення та взаємодії з користувачем. Варто зауважити, що контекст використання та особливості конкретної технології чи фреймворку мають велике значення, оскільки реалізація MVC може відрізнятися в залежності від середовища розробки, в якому вона використовується. Тому ретельне вивчення та адаптація шаблону до потреб проекту є необхідними для досягнення найкращих результатів.

У даному проекті використовується "Event-Driven Architecture" (EDA) або "Реактивна архітектура" (рис. 2.2). Ця архітектура означає, що додаток реагує на вхідні події або зміни стану і виконує відповідні дії. У даному випадку, події це збір інформації з екрану. Коли відбувається збір інформації, додаток виконує необхідні дії на основі отриманих даних.

За такої архітектури, користувач може лише вмикати або вимикати додаток, що можна розглядати як зовнішню подію або команду, яка впливає на стан додатку і викликає відповідні дії.

У реактивній архітектурі підхід до проектування базується на обробці подій і зміни стану системи, що дозволяє створити реактивний, відгукнутий на зміни середовища додаток.

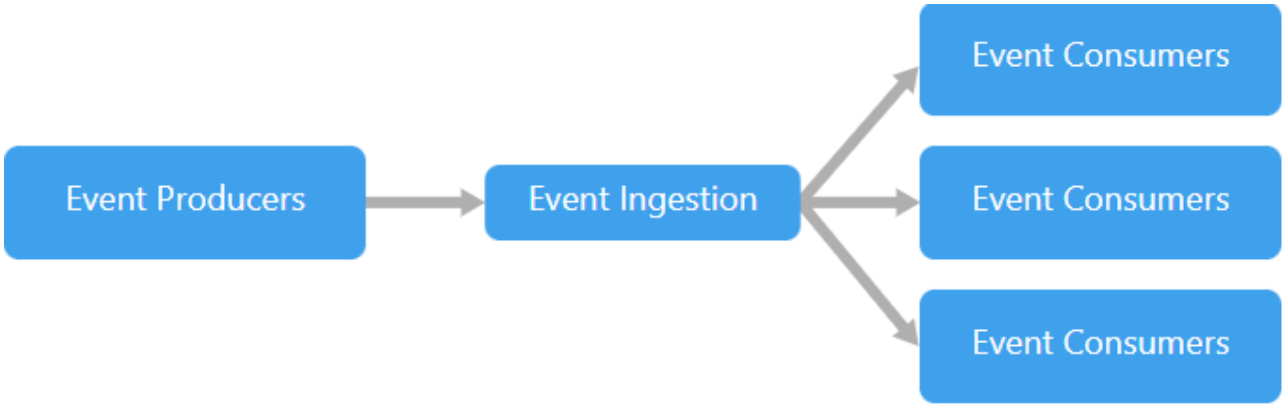

Рис. 2.2. Графічне представлення архітектури EDA

2.4. Опис використаних технологій та мов програмування

Технології та мови програмування, які були використані при створенні даного проекту:

- C;
- C#;
- Lua;
- WoW API (World of Warcraft API).

C є загально-призначеною мовою програмування, яка використовується для розробки системного програмного забезпечення, драйверів пристроїв, вбудованих систем та багатьох інших додатків. Мова C є компільованою мовою, що означає, що програми, написані на цій мові, спочатку перетворюються в машинний код за допомогою компілятора, а потім виконуються на відповідних платформах.

Основними концепціями мови C є функції, змінні, оператори та керуючі конструкції. Функції використовуються для групування коду та виконання певних операцій. Вони можуть приймати аргументи, виконувати певні дії та повертати значення. Змінні використовуються для зберігання даних, таких як числа, рядки або адреси пам'яті. Оператори використовуються для виконання

різних операцій, таких як присвоєння значень, обчислення арифметичних виразів та умовні переходи. Керуючі конструкції, такі як умовні оператори і цикли, використовуються для контролю потоку виконання програми.

Мова С має багато вбудованих функцій і бібліотек, які дозволяють взаємодіяти з операційною системою, обробляти введення та виведення, працювати зі строками, керувати пам'яттю та багато іншого. Крім того, в мові С можна створювати власні функції та типи даних, що дозволяє розширювати її можливості.

Мова С є однією з найпоширеніших мов програмування і має широке застосування у сфері розробки програмного забезпечення. Вона відома своєю ефективністю, близькістю до машинного коду та можливістю прямого доступу до апаратних ресурсів. С також є основою для багатьох інших мов програмування, таких як C++ та Objective-C.

Плюси мови програмування С:

- ефективність - С є мовою низького рівня, що дозволяє розробляти високоефективні програми. Вона має прямий доступ до пам'яті та низькорівневі операції, що дозволяє оптимізувати роботу з ресурсами та досягати високої швидкодії;
- портативність - код, написаний на мові С, може бути компільований та виконуваний на різних платформах без змін. Це дозволяє розробляти крос-платформові програми, які можуть працювати на різних операційних системах та апаратних платформах;
- близькість до апаратного забезпечення - мова С надає прямий доступ до апаратних ресурсів комп'ютера, таких як пам'ять, пристрої вводу-виводу та реєстри процесора. Це дає розробникам більший контроль над роботою програми та можливість оптимізувати взаємодію з апаратурою;
- велика спільнота та ресурси - С є однією з найпопулярніших мов програмування, тому є велика спільнота розробників, багато документації,

книг та онлайн-ресурсів. Це полегшує вивчення мови, розв'язання проблем та отримання підтримки.

Мінуси мови програмування С:

- складність - С є мовою з високим рівнем складності. Вона вимагає від розробників ретельного управління пам'яттю, урахування деталей низькорівневої роботи та ручного керування багатьма аспектами програми. Це може зробити розробку вимогливою до часу та підвищити ризик помилок;
- вразливість до помилок - оскільки С надає прямий доступ до пам'яті, неправильне використання цього доступу може призвести до помилок, таких як витоки пам'яті, переповнення буфера або невизначеність поведінки. Вимагається уважність та добре розуміння мови для уникнення таких проблем;
- відсутність автоматичного збирача сміття - мова С не надає вбудованого механізму автоматичного збирання сміття. Розробник повинен самостійно вручну керувати виділенням та звільненням пам'яті, що може бути причиною помилок, пов'язаних з управлінням пам'яттю;
- відсутність динамічної типізації - С використовує статичну типізацію, що означає, що типи змінних повинні бути визначені під час компіляції. Це може обмежити гнучкість та швидкість розробки порівняно з мовами з динамічною типізацією.

Для забезпечення безпеки та відповідності звичайній користувацькій взаємодії, при відправленні натискань клавіш використовуються наступні заходи:

- відправка натискання клавіш здійснюється від імені поточної клавіатури, що забезпечує нормальну поведінку системи та взаємодію з активними програмами;
- при відправці натискання клавіш, флаг LLMHF_INJECTED та LLKHF_LOWER_IL_INJECTED забезпечуються таким чином, щоб імітовані

натискання не розглядались як впроваджені зовнішніми джерелами, зменшуючи можливість виникнення конфліктів або проблем безпеки.

C# (C-Sharp) є об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft. Вона є частиною платформи .NET і використовується для розробки різноманітних додатків, включаючи веб-додатки, десктопні програми, мобільні додатки та багато іншого.

Основними концепціями мови C# є класи, об'єкти, методи, змінні та управління пам'яттю. Класи використовуються для опису об'єктів, що містять дані та пов'язані з ними методи. Об'єкти є екземплярами класів і можуть бути створені для доступу до їх функціональності. Методи використовуються для виконання дій або операцій над об'єктами. Змінні використовуються для зберігання даних, а їх типи можуть бути примітивними (наприклад, цілі числа, рядки) або власними (створеними користувачем). Управління пам'яттю відбувається автоматично за допомогою механізму збирача сміття, що дозволяє звільняти пам'ять, яку більше не використовується.

C# підтримує багато функцій, які спрощують розробку програм, такі як інтегрована обробка подій, обробка винятків, доступ до баз даних, робота зі зображеннями та багато іншого. Вона також має велику кількість бібліотек, які надають готовий функціонал для різних задач.

C# є однією з популярних мов програмування, особливо в контексті розробки додатків для платформи .NET. Вона має сучасний синтаксис, підтримку сильного типізації, винятковий контроль, а також багато інших функцій, що сприяють швидкій та надійній розробці програмного забезпечення. Плюси мови програмування C#:

- простота використання. C# має синтаксис, схожий на мови C і C++, що робить його відносно простим для вивчення та використання. Вона надає велику кількість вбудованих бібліотек та функцій, що полегшує розробку програм;

- широкі можливості розробки. C# використовується для розробки різноманітних програм, включаючи веб-додатки, настільні програми, мобільні додатки та ігри. Вона підтримує платформи Windows, macOS, Linux, Android та iOS;
- керування пам'яттю. C# використовує автоматичне управління пам'яттю за допомогою збирача сміття, що спрощує процес управління пам'яттю та дозволяє уникнути багатьох типичних помилок, пов'язаних з пам'яттю, таких як витоки пам'яті або використання недійсних посилань;
- багатопотоковість. C# надає потужні засоби для роботи з багатопотоковістю, що дозволяє створювати паралельні та асинхронні програми. Це особливо корисно для розробки програм, які вимагають обробки багатьох завдань одночасно;
- велика спільнота та підтримка. C# має велику та активну спільноту розробників, яка надає підтримку, документацію, форуми та багато інших ресурсів. Існує багато сторонніх бібліотек та фреймворків, які допомагають розширити можливості мови та спростити розробку.

Мінуси мови програмування C#:

- залежність від платформи. Хоча C# підтримує крос-платформову розробку за допомогою .NET Core або Mono, вона все ж залежить від платформи .NET Framework або .NET Core для виконання програм. Це може обмежувати переносимість деяких програм на інші платформи;
- високе споживання ресурсів. Деякі критики вказують на високе споживання ресурсів C# програмами порівняно з деякими іншими мовами. Використання великої кількості пам'яті та процесорного часу може стати проблемою в деяких обмежених середовищах;
- обмежена продуктивність для деяких завдань. Хоча C# є ефективною мовою програмування, для деяких високопродуктивних завдань, таких як розробка вбудованих систем або робота з дуже обмеженими ресурсами, можуть бути більш підходящі мови низького рівня;

- відсутність прямого доступу до апаратних ресурсів. Порівняно з мовами низького рівня, такими як C, C++ або Ассемблер, C# не надає прямого доступу до апаратних ресурсів комп'ютера. Для певних завдань, де важливий максимальний контроль над апаратурою, це може бути обмеженням.

C# - потужна та розширювана мова програмування з багатьма перевагами, але вона також має свої обмеження.

Lua є легковагою скриптовою мовою програмування, яка надає швидке виконання та простоту вбудовування. Вона часто використовується як вбудована мова для розширення функціональності інших програм, як самостійна мова для написання сценаріїв і прототипування або як мова для розробки інтерфейсу.

Основними концепціями мови Lua є таблиці, функції, змінні та контроль потоку. Таблиці є основною структурою даних в Lua і можуть використовуватись як асоціативні масиви, списки або об'єкти. Вони дозволяють зберігати дані та отримувати до них доступ за допомогою ключів. Функції використовуються для групування коду та виконання певних дій. Вони можуть бути оголошені, передані як аргументи, збережені в змінних та викликані. Змінні використовуються для зберігання даних, ім'я яких може бути присвоєне. Контроль потоку включає умовні вирази, цикли та обробку винятків для управління виконанням програми.

Lua має невеликий набір вбудованих функцій і бібліотек, але вона дуже розширювана та легко інтегрується з іншими мовами програмування. Lua може бути використана для автоматизації завдань, розробки ігор, реалізації сценаріїв інтерфейсу користувача та багатьох інших завдань.

Плюси мови Lua:

- простота вивчення. Lua має простий синтаксис та невеликий набір ключових слів, що робить її легкою для вивчення та розуміння. Вона може бути доступною навіть для початківців у програмуванні;
- швидкодія. Lua відома своєю високою швидкодією та ефективністю. Вона має компактний виконавчий код та мінімальні вимоги до ресурсів, що робить її

популярною для використання в вбудованих системах та областях, де швидкодія є важливою;

- легка інтеграція. Lua легко інтегрується з іншими мовами програмування, такими як C/C++, що дозволяє використовувати її як скриптову мову для розширення функціональності існуючих програм. Вона також має можливість виклику функцій з C/C++ та обміну даними між мовами;
- гнучкість. Lua надає гнучкість у розробці програм. Вона має динамічну типізацію, динамічне створення об'єктів та зручні механізми для маніпуляції зі структурами даних, такими як таблиці. Це дозволяє розробникам швидко експериментувати та реалізовувати складні логіки.

Мінуси мови Lua:

- обмежена екосистема. Lua має меншу екосистему порівняно з деякими іншими мовами програмування. Хоча вона має базові бібліотеки та модулі, вона може бути менш підходящою для деяких специфічних вимог та завдань;
- невелика спільнота розробників. В порівнянні з деякими іншими мовами програмування, спільнота розробників Lua менша. Це може призвести до меншої кількості ресурсів, документації та підтримки. Однак, є активні форуми та спільноти, де можна отримати допомогу та підтримку;
- відсутність стандартів. Lua не має жорстких стандартів, які б визначали певні практики розробки. Це може призвести до різних підходів у використанні мови та кодуванні, що ускладнює спільну роботу та обмін проектами між розробниками;
- відсутність вбудованої підтримки. Lua не має вбудованої підтримки для деяких функцій, таких як робота з графікою чи робота з базами даних. Хоча є деякі сторонні бібліотеки, використання таких функцій може вимагати додаткових зусиль та залежностей.

WoWAPI (World of Warcraft API) - це набір програмних інтерфейсів, які надають доступ до функціональності та даних гри World of Warcraft. Цей API розроблений і підтримується компанією Blizzard Entertainment, що дозволяє

розробникам створювати додатки, інтеграції та інструменти, які взаємодіють з ігровим світом World of Warcraft.

Основні можливості WoWAPI включають:

- загальна інформація про гру. API надає доступ до загальної інформації про гру, такої як назви реалми (серверів), категорії персонажів, фракції, раси, класи та інші загальні відомості про світ World of Warcraft;
- користувацькі дані персонажів. Розробники можуть отримувати інформацію про конкретного персонажа, таку як рівень, статистику, екіпірування, вміння, досягнення та інші дані, пов'язані з персонажем. Це дозволяє створювати додатки, які аналізують та відстежують прогрес персонажів гравців;
- данні про навколишнє середовище. WowAPI надає можливість отримувати дані про навколишнє середовище, такі як цілі (target), гравці (player), пети(pet), групи (group) та рейди(raid). Це дозволяє розробникам створювати додатки, які аналізують і взаємодіють з цими елементами гри, надаючи користувачам розширені функції та можливості взаємодії з ними;
- дані про предмети та заклинання. WowAPI дозволяє отримувати інформацію про предмети та заклинання у грі, включаючи їх властивості, статистику, вимоги та інші деталі. Це корисно для створення різноманітних додатків або аддонів, які надають гравцям інформацію про предмети та заклинання, а також допомагають в їх аналізі та використанні в грі. Аддони можуть аналізувати дані про заклинання, включаючи їхні ефекти, час відновлення та вимоги до використання. На основі цих даних, аддони можуть надавати рекомендації гравцям під час бою, підказувати оптимальний час використання заклинань або надавати важливі повідомлення про стан персонажа;

- розрахунок ігрових даних. API надає можливість розраховувати дані гри, такі як дамаг, здоров'я, мана та інші параметри, що дозволяє створювати розрахункові моделі та інструменти для аналізу геймплею та балансу гри.

WowAPI надає розробникам доступ до великого обсягу даних та функціональності гри World of Warcraft. Використовуючи цей API, розробники можуть створювати різноманітні додатки, веб-сайти та інструменти, які покращують досвід гравців і розширюють можливості світу World of Warcraft.

2.5. Опис структури програми та алгоритмів її функціонування

При створенні структури програмного застосунку було використано шаблон односторінкового додатка (Single-Window Application, SWA).

SWA є архітектурним підходом, при якому весь інтерфейс користувача відображається в одному вікні, де зазвичай містяться різні елементи, такі як меню, панелі інструментів, область вмісту та інші компоненти. Вікно може мінятися динамічно, залежно від дій користувача або від результатів виконання програми.

Структура інтерфейсу программного забезпечення на C# має такий вигляд:

1. MainForm.cs – файл який зв’язує контроль програми з інтерфейсом.
2. MainForm.Designer.cs – головний файл всього інтерфейсу вікна.
3. UserControl_CheckBox.cs – кастомний checkbox маючий кращий вигляд ніж представлений у звичайних елементах WinForms.
4. PanelMessageBox.cs – кастомний MessageBox який відкривається у вікні программи, у відмінку від звичайного.

Головним додатком у складі даного програмного забезпечення є Updater.exe, що реалізований на мові програмування C#. Використання такої назви додатку обумовлено потенційним захистом від виявлення анти-читами у грі. За звичайними алгоритмами перевірки процесів, анти-чит може реагувати на програми з очевидними назвами, такими як «WoW Bot» або «WoW Hack», і накладати заборони на користування грою.

У складі программи присутній класс DriverControl.cs, який відповідає за взаємодію з драйвером. У ньому реалізовані наступні функції:

- встановлення та видалення драйвера;
- увімкнення режиму тестування драйверів (Driver Test Mode), що дозволяє встановлювати непідписані драйвери Microsoft;
- встановлення та видалення сервісу, який запускає драйвер;
- перехоплення введення з поточної клавіатури драйвером для емуляції натискання клавіш з цієї клавіатури;
- надсилання драйверу інформації про клавішу, яку потрібно натиснути.

Для забезпечення зв'язку між драйвером і програмою, що реалізована на C#, використовується функція DeviceIoControl з бібліотеки kernel32. Ця функція є важливою складовою взаємодії з драйвером. Вона входить до складу бібліотеки kernel32 і надає можливість взаємодіяти з пристроями, що підтримують ввод/вивід через виклики відповідного драйвера.

Функція DeviceIoControl дозволяє виконувати різні операції з пристроями, включаючи відправку та отримання даних, керування параметрами пристрою та отримання стану пристрою. Це досягається шляхом передачі відповідних кодів операцій, параметрів та буферів даних через виклик функції.

Параметри DeviceIoControl можуть включати коди операцій, дескриптори пристроїв, вказівники на буфери даних та додаткові параметри, що визначають поведінку виклику функції. За допомогою цих параметрів програма на C# може передавати команди та отримувати результати взаємодії з драйвером.

Для отримання та обробки інформації з гри у программі використовується клас Core.cs. У цьому класі реалізовано отримання потрібної клавіші для натискання в грі за допомогою функції “Pixel.Get” (рис. 2.3).

```
Ссылок: 2
public class Pixel
{
    Ссылок: 2
    public static int Get(int x, int y)
    {
        IntPtr desk = GetDesktopWindow();
        IntPtr dc = GetWindowDC(desk);
        int a = (int)GetPixel(dc, x, y);
        ReleaseDC(desk, dc);
        return a;
    }
```

Рис. 2.3. Функція Pixel.Get

Завдяки функції Pixel.Get отримується колір двох пікселів (рис. 2.4): один відповідає за стан роботи бота в грі (увімкнено або вимкнено), а інший відповідає за те, яку клавішу треба натиснути. Для визначення клавіші за кольором був створений масив з відповідністю кольору та клавіші (рис. 2.5).

```
private KeyCode GetKeyToSend()
{
    int pixelColorLeft = Pixel.Get(0, 1439);
    int pixelColorRight = Pixel.Get(1, 1439);

    if (pixelColorRight != 0x00010101)
        return KeyCode.None;

    foreach(var _keyColor in keyColors)
    {
        if (_keyColor.color == pixelColorLeft)
            return _keyColor.key;
    }

    return KeyCode.None;
}
```

Рис. 2.4. Функція отримання кольорів двух пікселів для відправки клавіши

```
public static KeyColor[] keyColors = new KeyColor[]
{
    new KeyColor(0x00000001, KeyCode.Tilde),
    new KeyColor(0x00010001, KeyCode.Key1),
    new KeyColor(0x00020001, KeyCode.Key2),
    new KeyColor(0x00030001, KeyCode.Key3),
    new KeyColor(0x00000101, KeyCode.Key4),
    new KeyColor(0x00010101, KeyCode.Key5),
    new KeyColor(0x00020101, KeyCode.Key6),
    new KeyColor(0x00030101, KeyCode.Key7),
    new KeyColor(0x00000201, KeyCode.Key8),
    new KeyColor(0x00010201, KeyCode.Key9),
    new KeyColor(0x00020201, KeyCode.Key0),
    new KeyColor(0x00030201, KeyCode.Key10),
    new KeyColor(0x00000301, KeyCode.Key11),
    new KeyColor(0x00010301, KeyCode.Q),
    new KeyColor(0x00020301, KeyCode.W),
    new KeyColor(0x00030301, KeyCode.E),
    new KeyColor(0x00000002, KeyCode.R),
    new KeyColor(0x00010002, KeyCode.T),
    new KeyColor(0x00020002, KeyCode.Y),
    new KeyColor(0x00030002, KeyCode.U),
    new KeyColor(0x00000102, KeyCode.I),
    new KeyColor(0x00010102, KeyCode.O),
    new KeyColor(0x00020102, KeyCode.P),
    new KeyColor(0x00030102, KeyCode.A),
    new KeyColor(0x00000202, KeyCode.S),
    new KeyColor(0x00010202, KeyCode.D),
    new KeyColor(0x00020202, KeyCode.F),
    new KeyColor(0x00030202, KeyCode.G),
    new KeyColor(0x00000302, KeyCode.H),
    new KeyColor(0x00010302, KeyCode.J),
    new KeyColor(0x00020302, KeyCode.K),
    new KeyColor(0x00030303, KeyCode.L),
    new KeyColor(0x00000003, KeyCode.Z),
    new KeyColor(0x00010003, KeyCode.X),
    new KeyColor(0x00020003, KeyCode.C),
    new KeyColor(0x00030003, KeyCode.V),
    new KeyColor(0x00000103, KeyCode.B),
    new KeyColor(0x00010103, KeyCode.N),
    new KeyColor(0x00020103, KeyCode.M),
    new KeyColor(0x00030103, KeyCode.Tab),
    new KeyColor(0x00000203, KeyCode.F1),
    new KeyColor(0x00010203, KeyCode.F2),
    new KeyColor(0x00020203, KeyCode.F3),
    new KeyColor(0x00030203, KeyCode.F4),
    new KeyColor(0x00000303, KeyCode.F5),
    new KeyColor(0x00010303, KeyCode.F6),
    new KeyColor(0x00020303, KeyCode.F7),
    new KeyColor(0x00030303, KeyCode.F8)
};
```

Рис. 2.5. Масив з відповідністю кольору та клавіші

Після декількох перевірок на натиснуті модифікатори для клавіш, якщо перевірки пройдені успішно, викликається функція Click(driver, key, sleepTime). Цій функції передається посилання на драйвер, клавіша для натискання та час між натисканням та відпусканням клавіші. Час визначається за допомогою функції random.Next(min, max) для імітації натискання клавіші користувачем.

Для постійних перевірок пікселів у программі використовується клас System.Windows.Forms.Timer(). У цьому класі використовується цикл, в якому здійснюються постійні перевірки. Користувач має можливість вмикати та вимикати цей таймер, що впливає на роботу боту - коли таймер увімкнений, бот починає виконувати свою роботу, а коли таймер вимкнений, бот зупиняється.

Крім того, користувач має можливість впливати на швидкість натискання клавіш шляхом зміни параметра Interval таймера. Цей параметр визначає інтервал між перевірками пікселів та виконанням відповідних дій. Зміна значення параметра Interval дозволяє користувачеві налаштувати швидкість роботи боту відповідно до його потреб та вимог.

Таким чином, використання класу System.Windows.Forms.Timer дозволяє забезпечити постійну перевірку пікселів та взаємодію з ботом, а також надає користувачеві можливість контролювати роботу боту шляхом управління таймером та швидкістю натискання клавіш.

Структура драйверу на C має такий вигляд:

- main.h – використовується для включення заголовків та дефінювання прототипів функцій;
- main.c – головний файл з усіми функціями драйверу;
- public.h – використовується для дефінювання структур та параметрів.

Структура аддону на Lua має такий вигляд:

- _Helper.toc – файл що відповідає за загрузку всіх компонентів аддону;
- Core.lua – файл де відбувається основна логіка аддону та зв'язка всіх його компонентів;

- Tools.lua – функції для використання у логіці аддону, що отримують інформацію з гри та записують іх у таблиці у зручному форматі для подальшого використання;
- UITools.lua – функції які забезпечують створення інтерфейсу білш зручним;
- UI.lua – створення всього інтерфейсу аддону та забезпечення функціями для зв’язку з ним;
- SpellNames.lua – містить в собі таблицю з назвами здібностей та їхню інтерпретацію для зручної роботи з написання коду ротації, також має функцію що проводить занесення у таблицю заклинань;
- StopSpells.lua – модуль, що дозволяє під час роботи бота використовувати здібності власноруч за його бажанням;
- TimeToDie.lua – модуль, що вираховує час до смерті певної ігрової одиниці;
- Class\Hunter.lua – містить в собі загрузчик та вигрузчик певних наборів ротації відповідно до поточної спеціалізації классу “Мисливець” (Hunter);
- Class\Rogue.lua – містить в собі загрузчик та вигрузчик певних наборів ротації відповідно до поточної спеціалізації классу “Розбійник” (Rogue);
- Class\Hunter_Survival.lua – містить в собі модуль для классу “Мисливець” (Hunter) спеціалізації “Виживання” (Survival);
- Class\RogueSub.lua – містить в собі модуль для классу “Розбійник” (Rogue) спеціалізації “Прихованість” (Subtlety).

У файлі Core.lua, відбувається загрузка аддона та інформації, пов'язаної з ним. Перш за все, визначається таблиця, яка вже була визначена у додатку на C# і містить кнопки та відповідні їм кольори пікселів (рис. 2.6) та (рис. 2.7) які служать для звязку між програмою та грою.

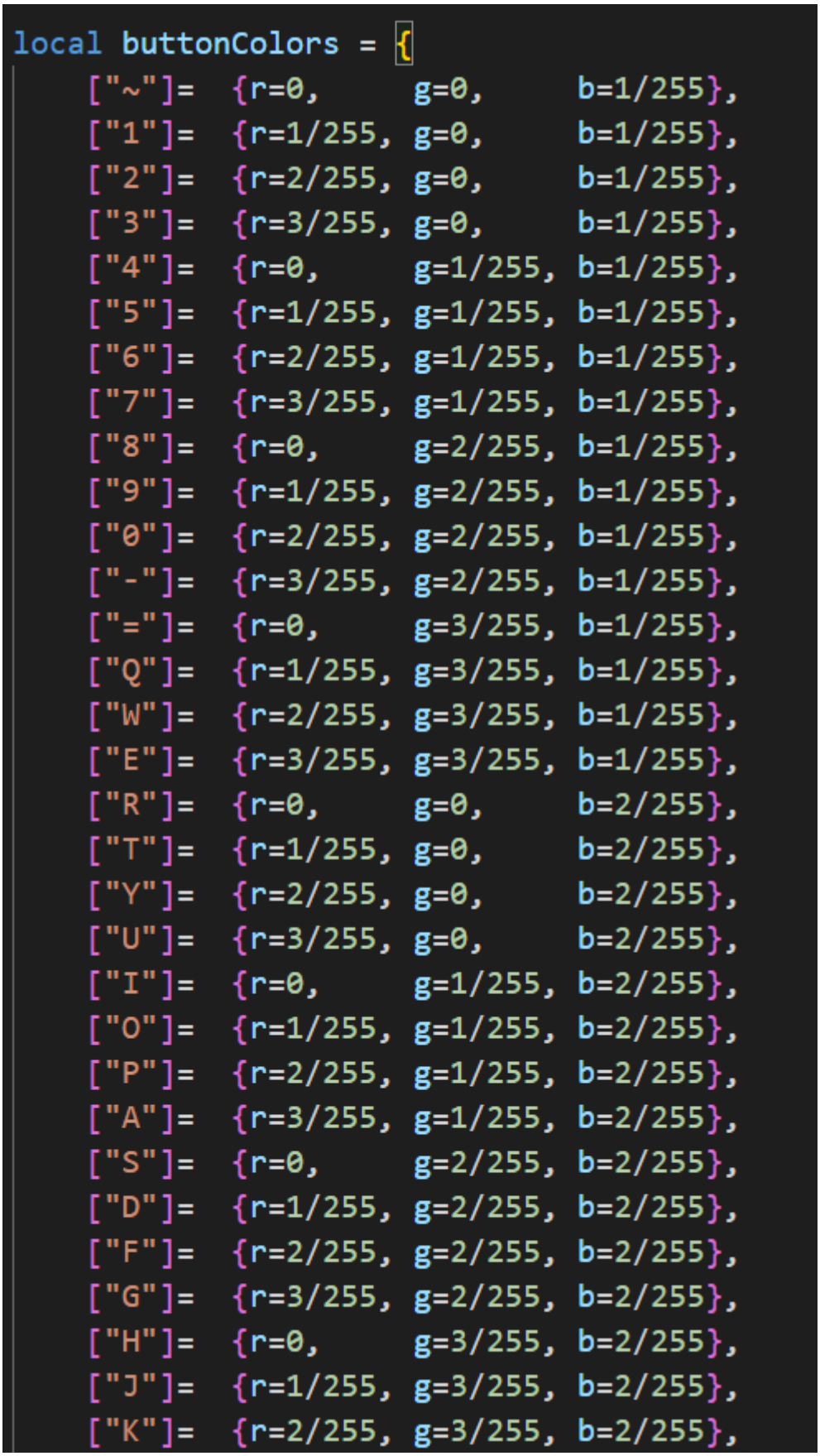

Рис. 2.6. Lua таблиця з кнопками яким відповідають кольора

```
    ["K"]=  {r=2/255, g=3/255, b=2/255},
    ["L"]=  {r=3/255, g=3/255, b=2/255},
    ["Z"]=  {r=0,     g=0,     b=3/255},
    ["X"]=  {r=1/255, g=0,     b=3/255},
    ["C"]=  {r=2/255, g=0,     b=3/255},
    ["V"]=  {r=3/255, g=0,     b=3/255},
    ["B"]=  {r=0,     g=1/255, b=3/255},
    ["N"]=  {r=1/255, g=1/255, b=3/255},
    ["M"]=  {r=2/255, g=1/255, b=3/255},
    ["Tab"]={r=3/255, g=1/255, b=3/255},
    ["F1"]= {r=0,     g=2/255, b=3/255},
    ["F2"]= {r=1/255, g=2/255, b=3/255},
    ["F3"]= {r=2/255, g=2/255, b=3/255},
    ["F4"]= {r=3/255, g=2/255, b=3/255},
    ["F5"]= {r=0,     g=3/255, b=3/255},
    ["F6"]= {r=1/255, g=3/255, b=3/255},
    ["F7"]= {r=2/255, g=3/255, b=3/255},
    ["F8"]= {r=3/255, g=3/255, b=3/255},
}

addon.activatedColor = { r = 1/255, g = 1/255, b = 1/255 }
addon.notActivatedColor = { r = 0, g = 0, b = 0 }
```

Рис. 2.7. Lua таблиця з кнопками яким відповідають кольора

Далі визначаються глобальні функції, які будуть використовуватися в цьому файлі. Для цього локальні функції отримують глобальне призначення (рис. 2.8).

```
57  local IsSpellKnown = IsSpellKnown
58  local IsSpellKnownOrOverridesKnown = IsSpellKnownOrOverridesKnown
59  local IsPassiveSpell = IsPassiveSpell
60  local GetMacroSpell = GetMacroSpell
61  local GetActionInfo = GetActionInfo
62  local GetBindingByKey = GetBindingByKey
63  local FindBaseSpellByID = FindBaseSpellByID
64  local GetSpellInfo = GetSpellInfo
65  local GetTime = GetTime
66  local GetSpellBookItemName = GetSpellBookItemName
67  local GetItemID = C_Item.GetItemID
68  local C_TimerAfter = C_Timer.After --(seconds, callback)
69
70  local Tools = addon.Tools
71  local UITools = addon.UITools
72  local SetFrameColor = addon.UITools.SetFrameColor
73  local SetPreclick = Tools.SetPreclick
74  local SetSpellTable = addon.SetSpellTable
```

Рис. 2.8. Початок файлу з призначенням функцій

Це зроблено з тією метою, щоб при виклику функції через глобальну таблицю не відбувався пошук серед великої кількості параметрів (понад 60 000). Замість цього, посилання на функцію зберігається у локальній змінній, що дозволяє викликати функцію без перебору таблиці. Після цього створюється фрейм, який реєструє визначені події в грі та приєднує обробник подій (рис. 2.9).

```
local EventHooks = {}
addon.EventHooks = EventHooks
addon.Frame = CreateFrame("Frame")
addon.Frame:RegisterEvent("PLAYER_ENTERING_WORLD")
addon.Frame:RegisterEvent("ACTIONBAR_SLOT_CHANGED")
addon.Frame:RegisterEvent("UPDATE_BONUS_ACTIONBAR")
addon.Frame:RegisterEvent("TRAIT_CONFIG_UPDATED")
addon.Frame:RegisterEvent("PLAYER_TALENT_UPDATE")
addon.Frame:SetScript("OnEvent", function(self, event, ...) if(EventHooks[event]) then   EventHooks[event](...) end end)
```

Рис. 2.9. Створення фрейму і реєстрація подій

Обробник подій у цьому фреймі відповідає за налаштування інтерфейсу, завантаження здібностей у таблицю, завантаження модуля, що відповідає поточному класу, додавання клавіш, що відповідають здібностям, та створення циклу для постійної обробки інформації.

Подія "PLAYER_ENTERING_WORLD" викликається один раз після того, як користувач зайшов у гру. Події "ACTIONBAR_SLOT_CHANGED" та "UPDATE_BONUS_ACTIONBAR" використовуються при зміні елементів на панелі здібностей. Обробники цих подій також змінюють здібності клавіш у таблиці. Події "PLAYER_TALENT_UPDATE" та "TRAIT_CONFIG_UPDATED" використовуються при зміні вмінь персонажу або його спеціалізації. Обробники цих подій завантажують або видаляють модулі для різних класів, а також оновлюють таблицю з інформацією про заклинання.

Наведемо опис однієї з ротацій персонажу - Hunter_Survival.lua. Файл починається з присвоєння глобальних функцій локальним змінним, як це зазвичай відбувається. Далі ініціалізується таблиця з необхідною інформацією, яка буде аналізуватись перед кожним оновленням бота. Ця таблиця містить імена та ідентифікатори заклинань, дебафів, бафів, талантів, а також інформацію для

модулю StopSpells.lua. Також створюється таблиця з основними параметрами для управління поточним модулем.

Після цього визначаються три функції. Перша функція відповідає за завантаження інтерфейсу для поточного модуля, друга - за оновлення інформації про заклинання, а третя - за вигрузку модуля.

Потім слідують кілька функцій, які використовуються для отримання інформації, а найважливіша з них - функція ротації, яка викликається при кожному оновленні бота. Ця функція побудована на системі пріоритетів, які змінюються в залежності від поточного стану гри та налаштувань користувача.

Для визначення найкращої ротації заклинань був використаний веб-сайт https://www.simulationcraft.org/, який використовується для симуляції гри та порівняння цих симуляцій з метою отримання найбільш ефективної стратегії гри.

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Враховуючи особливості розроблюваного програмного продукту, користувач не повинен надавати додаткові дані для користування додатком. В якості вихідних даних виступає реакція додатку на дії користувача з управління деякими пріорітетами.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для забезпечення користувачу запуску гри World of Warcraft розробники рекомендують мати такі мінімальні параметри комп’ютера:мати операційну систему Windows 7-11 (64-bit):

- процессор 4 Cores, 3.0 GHz processor 4th Generation Intel® Core™ або AMD Ryzen™ Zen;
- відеокарта з підтримкою DirectX® і не менш ніж 4 гб відеопам’яті;

- не менше 8 ГБ оперативної пам'яті;
- не менше 120 ГБ вільного місця на жорсткому диску SSD;
- мати можливість виходу у інтернет;
- мати мінімальну розподільну здатність екрану 1280x720;
- мати клавіатуру та мишу.

Ці рекомендації допоможуть з горем навпіл запустити World of Warcraft. Звичайно, якщо ви змогли запустити гру, то додаток також запуститься, бо його потреби мізерні у порівнянні з грою.

Технічні засоби, використані мною:

- центральний процесор (ЦП): AMD Ryzen 3600;
- відеоадаптер: AMD Radeon RX 6700 XT;
- відеопам'ять: 16 ГБ;
- накопичувач: 1 ТБ SSD;
- оперативна пам'ять: 64 ГБ;
- розподільну здатність екрану 2560x1440;
- клавіатура, миша.

2.7.2. Використані програмні засоби

При розробці цього проекту було використано наступні програмні засоби:

- Visual Studio 2022;
- Visual Studio Code;
- Windows Driver Kit (WDK).

Visual Studio Code (VS Code) - це безкоштовний, легкий у використанні текстовий редактор, розроблений компанією Microsoft. Він призначений для роботи з програмним кодом у різних мовах програмування та надає широкий спектр функцій, які сприяють розробці програмного забезпечення.

Основні особливості Visual Studio Code:

1. Підтримка різних мов програмування - VS Code надає підтримку для багатьох популярних мов програмування, таких як JavaScript, Python, C#, Java, HTML, CSS та інші. Це дозволяє розробникам працювати над різними типами проектів в одному редакторі.
2. Розширення та налаштування - VS Code дозволяє розширити його функціональність за допомогою розширень, які розробляються спільнотою. Розширення додають нові можливості, підтримку конкретних технологій, інструменти розробки та багато іншого. Крім того, користувачі можуть налаштовувати редактор залежно від своїх потреб.
3. Інтеграція з Git - VS Code має вбудовану підтримку системи контролю версій Git. Це дозволяє розробникам здійснювати коміти, переглядати та порівнювати зміни, створювати та об'єднувати гілки, працювати з репозиторіями та інше, все безпосередньо з редактора.
4. Розширені можливості пошуку та заміни - VS Code надає потужні інструменти для пошуку та заміни тексту у проекті. За допомогою регулярних виразів та різних параметрів пошуку, можна легко знаходити та замінювати текст у великих обсягах коду.
5. Відлагодження коду - редактор підтримує можливість відлагодження коду для різних мов програмування. За допомогою спеціальних розширень та налаштувань, розробники можуть встановлювати точки зупинки, переглядати значення змінних, виконувати кроки виконання коду та інше.
6. Live Share - функція, що дозволяє розробникам спільно працювати над проектом в реальному часі. За допомогою Live Share, ви можете запрошувати інших розробників до свого редактора, щоб вони могли бачити ваш код, робити зміни та спілкуватися.

Visual Studio Code є популярним вибором серед розробників завдяки своїй простоті використання, гнучкості та багатій екосистемі розширень. Він

доступний для різних операційних систем, включаючи Windows, macOS та Linux (рис. 2.10).

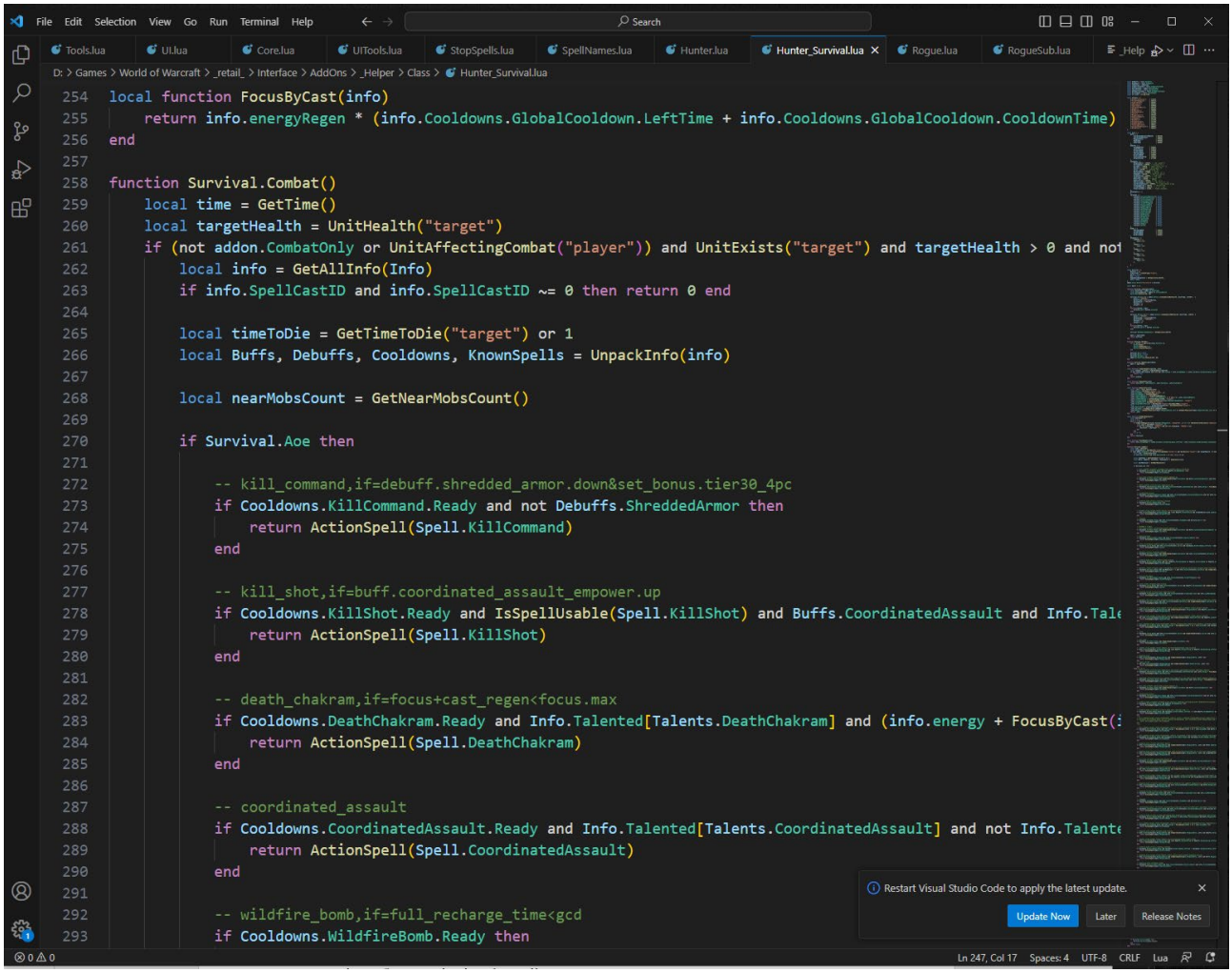

Рис. 2.10. Зовнішній вигляд VS Code

Visual Studio - це інтегроване середовище розробки (IDE) компанії Microsoft, призначене для розробки різноманітних програмних продуктів. Воно надає розробникам потужні інструменти та сервіси для створення програм на різних мовах програмування, включаючи C, C#, C++, Visual Basic, JavaScript, Python та інші.

Основні особливості Visual Studio включають:

- багатомовність. Visual Studio підтримує широкий спектр мов програмування, дозволяючи розробникам працювати з будь-якими

проектами незалежно від їх мовного стеку. Завдяки цьому, розробники можуть легко створювати програми для різних платформ і пристроїв;

- інтуїтивний інтерфейс. Visual Studio має зручний і простий у використанні інтерфейс, що дозволяє розробникам зосередитися на написанні коду. Інструменти редактора дозволяють швидко відстежувати синтаксичні помилки, автоматично доповнювати код, переглядати документацію та інші корисні функції;
- налагодження. Visual Studio надає потужні засоби для відлагодження програм, дозволяючи розробникам знайти та виправити помилки. Вона підтримує крок-за-кроком виконання коду, перегляд значень змінних, стеження за викликами функцій та багато іншого;
- керування версіями. Visual Studio інтегрується з системами керування версіями, такими як Git, що дозволяє розробникам відстежувати зміни у коді, спільно працювати з командою, вирішувати конфлікти та здійснювати інші операції, пов'язані з керуванням версіями;
- розширення. Visual Studio підтримує розширення, які дозволяють розробникам розширювати його функціональність та адаптувати середовище розробки до своїх потреб. Велика спільнота розробників постійно розробляє нові розширення, які додають додаткові можливості та інтеграцію з різними інструментами та сервісами.

Visual Studio є потужним інструментом для розробки програмного забезпечення, який дозволяє розробникам ефективно працювати над проектами будь-якої складності та масштабу. Він надає широкі можливості для підтримки різних мов програмування, відладки коду, керування версіями та багато іншого, що робить його популярним серед професіональних розробників програмного забезпечення (рис. 2.11).

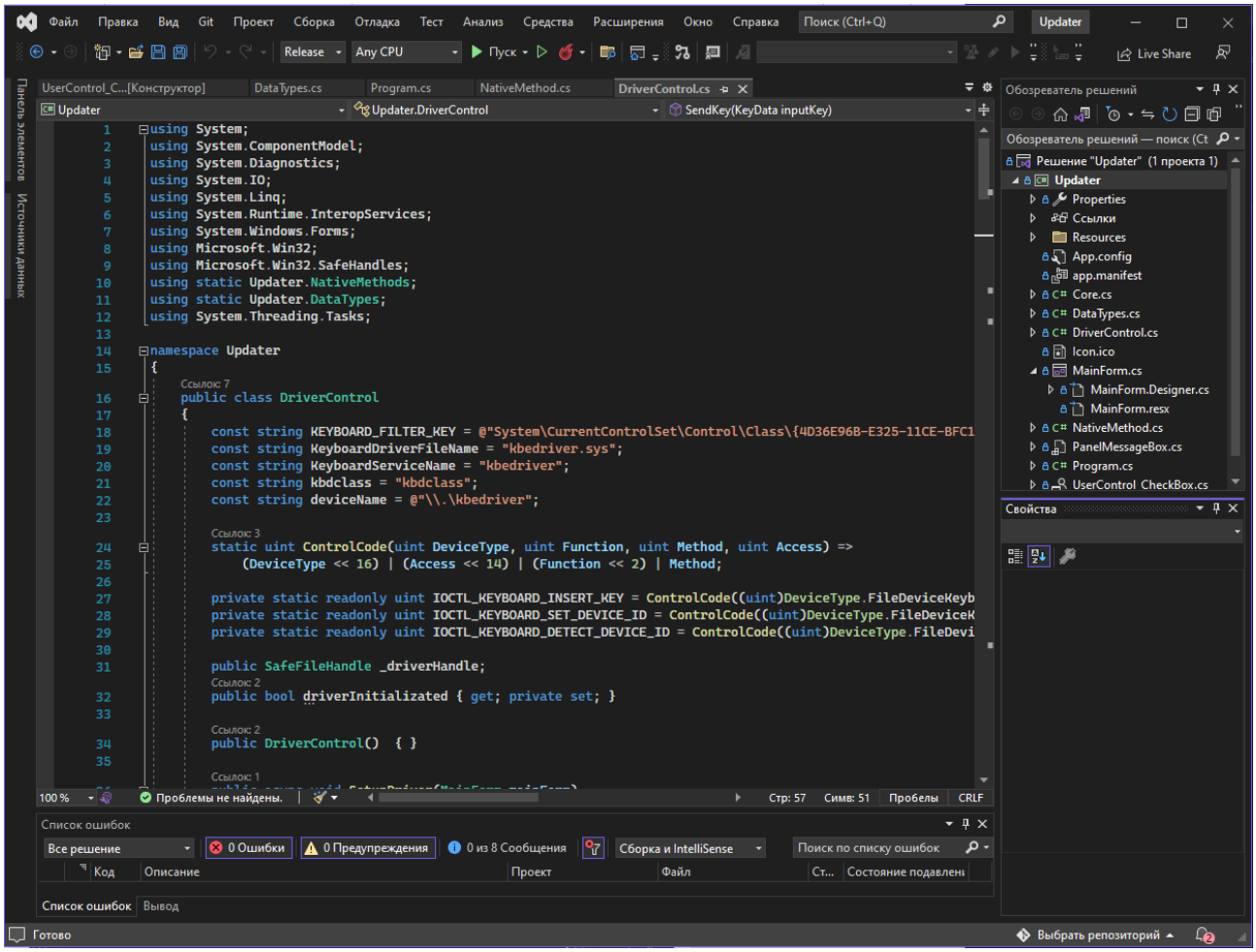

Рис. 2.11. Зовнішній вигляд Visual Studio 2022

Windows Driver Kit (WDK): Windows Driver Kit (WDK) є набором засобів для розробки драйверів для операційної системи Windows. Він надає розробникам необхідні компоненти та документацію для створення, відлагодження та тестування драйверів. WDK допомагає розробникам взаємодіяти з пристроями, забезпечує підтримку драйверів різних типів та пропонує інструменти для аналізу та відлагодження драйверів.

2.7.3. Виклик та завантаження програми

Розроблений додаток може бути доступний в мережі Інтернет за допомогою спеціалізованих сайтів з завантаження файлів.

Додаток гарантовано запускається на будь якій системі з Windows 7-11 та використовує не більш ніж 500 кб вільної пам’яті на жорсткому диску та не більш ніж 10 мб оперативної.

Таким чином, будь хто може встановити додаток, якщо в нього встановленна гра World of Warcraft.

2.7.4. Опис інтерфейсу користувача

Для початку роботи з додатком, необхідно запустити файл Updater.exe (рис. 2.12).

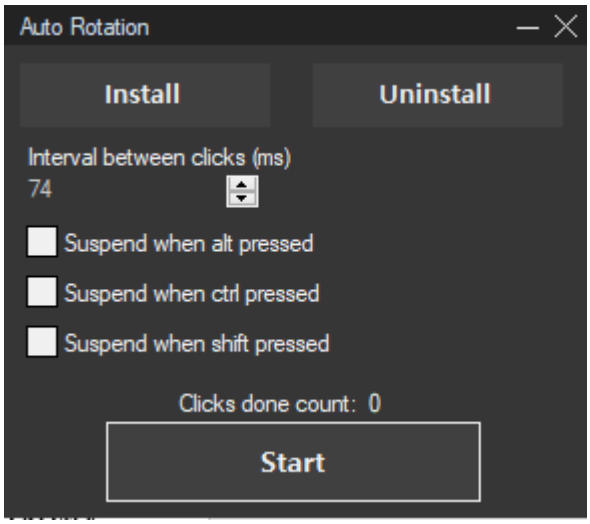

Рис. 2.12. Зовнішній вигляд Updater.exe

Після чого натиснути кнопку Install, якщо тестувальний режим для драйверів у віндовс не був увімкнений, він увімкне його та видасть повідомлення що треба перезапустити комп’ютер. Далі треба знов відкрити Updater.exe та продовжити установку. Після чого аддон _Helper треба помістити у папку з усіма іншими аддонами для World of Warcraft.

У додатку треба натиснути клавішу Start після чого вікно зміниться та попросить натиснути будь яку клавішу на клавіатуря для її ідентифікації (рис. 2.13).

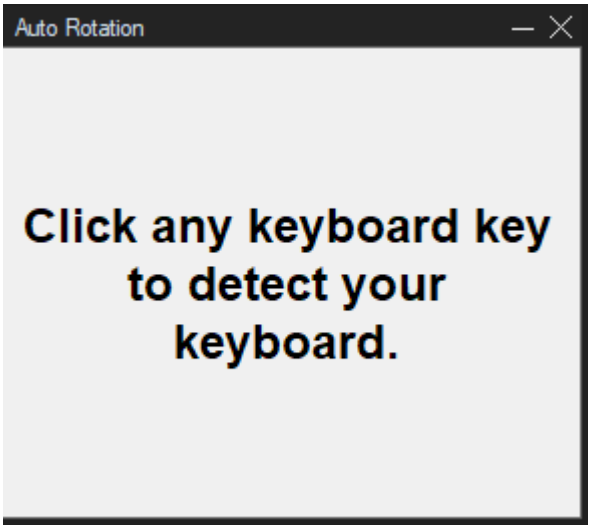

Рис. 2.13. Зовнішній вигляд при натисканні кнопки Start

Після ідентифікації клавіатури бот запуститься. У головному вікні є декілька параметрів для настроювання додатку для своїх вимог, а саме швидкість натискання клавіш і зупинка при натисканні обраних модифікаторів (рис. 2.14). Також на панелі можна побачити текст Clicks done count, він відображає скільки кліків зробила програма, що б можна було уявити скільки зусиль вона зберегла. Після 2-х годин геймплею може бути натиснуто клавіши 50000 разів або більше.

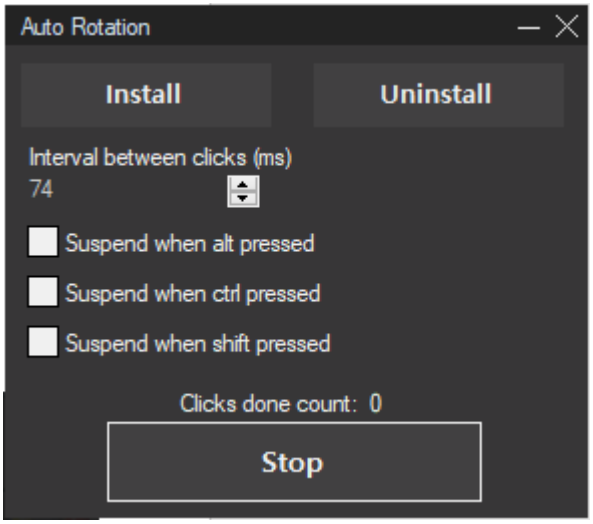

Рис. 2.14. Вікно управління

Головне вікно управління в грі містить у собі іконки заклинань які можна додати чи видалити з ротації (рис. 2.15). При наведенні на іконку можна прочитати що вона робить (рис. 2.16).

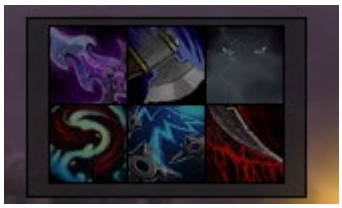

Рис. 2.15. Головне вікно управління у грі

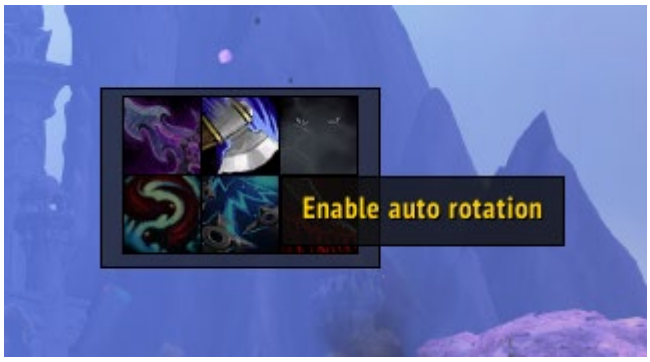

Рис. 2.16. Підказка при наведенні миші

Рис. 2.17. Налаштовані деякі з параметрів та іконка поточного заклинання

Рис. 2.18. Кнопка налаштувань біля мінімапи

Рис. 2.19. Вікно додаткових налаштувань при наведенні відображається опис налаштування

Таким чином, створений інтерфейс користувача є легким у освоєнні, та інтуїтивно зрозумілим, що забезпечує його зручність у використанні.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вихідні дані:

1. передбачуване число операторів програми – 1841;
2. коефіцієнт складності програми – 1.4;
3. коефіцієнт корекції програми в ході її розробки – 0.3;
4. годинна заробітна плата розробника – 132 грн/год (за версією сайта WorkUA) - https://www.work.ua/salary-dnipro-it/;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1.2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.3;
7. вартість машино-години ЕОМ – 18 грн/год (1 грн – е/е, 10 грн – ПЗ, 7 грн - амортизація).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,}$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q – передбачуване число операторів (2841);

C – коефіцієнт складності програми (1,4);

p – коефіцієнт корекції програми в ході її розробки (0,3).

$$Q = 1841 \cdot 1{,}4 \cdot (1 + 0{,}3) = 3351;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot K}, \text{ людино-годин}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

К – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1.3);

$$t_u = \frac{3351*1.2}{85*1.3} = 61.5 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K};$$

$$t_a = \frac{3351}{20 \cdot 1.3} = 128.88 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K};$$

$$t_n = \frac{3351}{25\cdot 1.3} = 103.1 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot K};$$

$$t_{отл} = \frac{3351}{5\cdot 1.3} = 515.5 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл};$$

$$t^{к}_{отл} = 1{,}2 \cdot 515.5 = 618.6 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до};$$

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_д = \frac{Q}{(15...20)\cdot K};$$

$$t_{дp} = \frac{3351}{20\cdot 1.3} = 128.88 \text{ людино-годин.}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0{,}75 \cdot t_{др};$$

$t_{до}$ = 0,75 · 128.88 = 96.66 людино-годин.

$t_{д}$ = 128.88 + 96.66 = 225.54 людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 61.5 + 128.88 + 128.88 + 515.5 + 225.54 = 1110.3 людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1110.3 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $Кпо$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$Кпо = З_{ЗП} + З_{МВ}, \text{ грн},$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн},$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата розробника становить 115 грн/год, то отримаємо:

$$З_{ЗП} = 1110.3 \cdot 132 = 146\ 559.6 \text{ грн}$$

Вартість машинного часу $З_{МВ}$, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_М \text{, грн,}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_М$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 515.5 \cdot 18 = 9270 \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$Кпо = 146\ 559.6 + 9270 = 155\ 829.6 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес.}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Витрати на створення програмного продукту:

$$T = \frac{1110.3}{1 \cdot 176} = 6.3 \text{ міс.}$$

Вартість розробки додатку становить 155 829 грн. Час розробки очікується приблизно 6.3 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Цей термін включає час, необхідний для дослідження, розробки алгоритму, дизайну і документування. Загальна кількістю людино-годин, яку буде витрачено на розробку – 1110.3.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка бота для MMO RPG World of Warcraft, використовуючи мову програмування C#.

В даній роботі було розглянуто проблему повторюваних дій гравців у MMO RPG іграх, зокрема в World of Warcraft, і була запропонована розробка бота, що автоматизує ці дії. Використані мови програмування C#, C та Lua дозволили ефективно реалізувати функціонал бота. Застосування розробленого бота дозволить геймерам зосередитись на важливих стратегічних аспектах гри, звільнивши їх від монотонних та втомлюючих завдань, пов'язаних з натисканням клавіш для використання заклинань персонажа. Це може сприяти поліпшенню геймплею та підвищенню задоволення від гри.

Результати кваліфікаційної роботи свідчать про можливість розробки ефективного бота для MMO RPG ігор, який може полегшити гравцям процес гри та забезпечити більш ефективне використання їх часу. Дана робота може служити основою для подальших досліджень та розробок в галузі автоматизації геймплею та розширення можливостей віртуальних світів MMO RPG ігор.

Для розробки бота використовувалися різноманітні програмні засоби, зокрема Visual Studio 2022, Visual Studio Code і Windows Driver Kit (WDK). Ці інструменти надали необхідні засоби для ефективного програмування, тестування та налагодження бота.

Розроблений додаток являє собою програму для запуску бота та аддон що реалізує інтерфейс управління ботом у грі. Завдяки використовуваним іконкам з гри, реалізується інтуїтивно зрозумілий інтерфейс.

Бот призначений для гри World of Warcraft надає можливість:

- використовувати здібності у повністю автоматичному режимі;
- змінювати пріорітети у ротації;
- впливати на швидкість оновлення бота;
- впливати на швидкість натискань клавіш;

- скривати інтерфейс боту у грі для безпечної зйомки екрану тощо;
- змінювати параметри при яких буде працювати бот.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення 1647.46 люд-год, підраховані витрати на створення програмного забезпечення - 352 048 грн і гаданий період розробки 9.3 міс при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jon Skeet. C# in Depth. — Manning Publications, 2019. — 528 pages.
2. Andrew Troelsen. Pro C# 7: With .NET and .NET Core. — Apress, 2017. — 1552 pages.
3. Eric Lippert. C# in a Nutshell: The Definitive Reference. — O'Reilly Media, 2019. — 1096 pages.
4. Jeffrey Richter. CLR via C#. — Microsoft Press, 2012. — 896 pages.
5. Joseph Albahari and Ben Albahari. C# 8.0 in a Nutshell: The Definitive Reference. — O'Reilly Media, 2019. — 1098 pages.
6. Jesse Liberty and Donald Xie. Programming C# 8.0: Build Cloud, Web, and Desktop Applications. — O'Reilly Media, 2019. — 800 pages.
7. Brian W. Kernighan and Dennis M. Ritchie. The C Programming Language. — Prentice Hall, 1988. — 288 pages.
8. Walter Oney. Programming the Microsoft Windows Driver Model. — Microsoft Press, 2003. — 656 pages.
9. Pavel Yosifovich. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. — Pearson, 2017. — 800 pages.
10. Pavel Yosifovich. Windows Internals, Part 2: System services, kernel debugging, and more. — Pearson, 2017. — 800 pages.
11. Brian Catlin. Windows Kernel Programming. — Addison-Wesley Professional, 1998. — 647 pages.
12. James T. Leiterman. Developing Windows NT Device Drivers: A Programmer's Handbook. — Microsoft Press, 1997. — 716 pages.
13. Roberto Ierusalimschy. Programming in Lua. — Lua.org, 2020. — 367 pages.
14. Paul Schuytema. Lua Programming Gems. — Lua.org, 2008. — 384 pages.
15. Kurt Jung and Aaron Brown. Lua for Programmers. — Lua.org, 2016. — 328 pages.
16. Terry Loveall and Michal Kula. Programming in Lua, Fourth Edition. — Lua.org, 2016. — 388 pages.

17. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. — Prentice Hall, 2008. — 464 pages.
18. Michael C. Feathers. Working Effectively with Legacy Code. — Prentice Hall, 2004. — 456 pages.
19. Sandi Metz. Practical Object-Oriented Design in Ruby: An Agile Primer. — Addison-Wesley Professional, 2012. — 272 pages.
20. Martin Fowler. Refactoring: Improving the Design of Existing Code. — Addison-Wesley Professional, 1999. — 431 pages.
21. Steve McConnell. Code Complete: A Practical Handbook of Software Construction. — Microsoft Press, 2004. — 960 pages.
22. Brian W. Kernighan and Rob Pike. The Practice of Programming. — Addison-Wesley Professional, 1999. — 267 pages.
23. Donald Knuth. The Art of Computer Programming. — Addison-Wesley Professional, 1968-present (multiple volumes).
24. Eric Freeman and Elisabeth Robson. Head First Design Patterns. — O'Reilly Media, 2004. — 694 pages.
25. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. — The MIT Press, 2009. — 1312 pages.
26. Steven S. Skiena. The Algorithm Design Manual. — Springer, 2008. — 730 pages.
27. WoWAPI URL: https://wowpedia.fandom.com/wiki/World_of_Warcraft_API
28. Ownedcore, useful about WoW: https://www.ownedcore.com/forums/world-of-warcraft/
29. Rotation spell priority https://www.simulationcraft.org/
30. Any helpful code https://github.com/

ДОДАТОК А

КОД ПРОГРАМИ

C Driver

main.c

```
#include "main.h"

//
// Collection object is used to store all the FilterDevice objects so
// that any event callback routine can easily walk thru the list and pick a
// specific instance of the device for filtering.
//
WDFCOLLECTION   FilterDeviceCollection;
WDFWAITLOCK     FilterDeviceCollectionLock;

#ifdef ALLOC_PRAGMA
#pragma alloc_text (INIT, DriverEntry)
#pragma alloc_text (PAGE, KbFilter_EvtDeviceAdd)
#pragma alloc_text (PAGE, KbFilter_EvtIoInternalDeviceControl)
#endif

//
// ControlDevice provides a sideband communication to the filter from
// usermode. This is required if the filter driver is sitting underneath
// another driver that fails custom ioctls defined by the Filter driver.
// Since there is one control-device for all instances of the device the
// filter is attached to, we will store the device handle in a global variable.
//

WDFDEVICE      ControlDevice = NULL;


NTSTATUS
DriverEntry(
                IN PDRIVER_OBJECT  DriverObject,
                IN PUNICODE_STRING RegistryPath
)
/*++

Routine Description:

                Installable driver initialization entry point.
                This entry point is called directly by the I/O system.

Arguments:

                DriverObject - pointer to the driver object

                RegistryPath - pointer to a unicode string representing the path,
                                        to driver-specific key in the registry.
```

```
Return Value:

                STATUS_SUCCESS if successful,
                STATUS_UNSUCCESSFUL otherwise.

--*/
{
                WDF_DRIVER_CONFIG          config;
                NTSTATUS                   status;

                //DebugPrint(("Keyboard Filter Driver Sample - Driver Framework Edition.\n"));
                //DebugPrint(("Built %s %s\n", __DATE__, __TIME__));

                //
                // Initialize driver config to control the attributes that
                // are global to the driver. Note that framework by default
                // provides a driver unload routine. If you create any resources
                // in the DriverEntry and want to be cleaned in driver unload,
                // you can override that by manually setting the EvtDriverUnload in the
                // config structure. In general xxx_CONFIG_INIT macros are provided to
                // initialize most commonly used members.
                //

                WDF_DRIVER_CONFIG_INIT(
                  &config,
                  KbFilter_EvtDeviceAdd
                );

                //
                // Create a framework driver object to represent our driver.
                //
                status = WdfDriverCreate(DriverObject,
                  RegistryPath,
                  WDF_NO_OBJECT_ATTRIBUTES,
                  &config,
                  WDF_NO_HANDLE); // hDriver optional
                if (!NT_SUCCESS(status)) {
                  //DebugPrint(("WdfDriverCreate failed with status 0x%x\n", status));
                }

                //
                // Since there is only one control-device for all the instances
                // of the physical device, we need an ability to get to particular instance
                // of the device in our KbFilter_EvtIoDeviceControl. For that we
                // will create a collection object and store filter device objects.
                // The collection object has the driver object as a default parent.
                //

                status = WdfCollectionCreate(WDF_NO_OBJECT_ATTRIBUTES,
                  &FilterDeviceCollection);
                if (!NT_SUCCESS(status))
```

```
        {
          //KdPrint(("WdfCollectionCreate failed with status 0x%x\n", status));
          return status;
        }

        //
        // The wait-lock object has the driver object as a default parent.
        //

        status = WdfWaitLockCreate(WDF_NO_OBJECT_ATTRIBUTES,
          &FilterDeviceCollectionLock);
        if (!NT_SUCCESS(status))
        {
          //KdPrint(("WdfWaitLockCreate failed with status 0x%x\n", status));
          return status;
        }

        return status;
}

NTSTATUS
KbFilter_EvtDeviceAdd(
        IN WDFDRIVER       Driver,
        IN PWDFDEVICE_INIT  DeviceInit
)
/*++
Routine Description:

        EvtDeviceAdd is called by the framework in response to AddDevice
        call from the PnP manager. Here you can query the device properties
        using WdfFdoInitWdmGetPhysicalDevice/IoGetDeviceProperty and based
        on that, decide to create a filter device object and attach to the
        function stack.

        If you are not interested in filtering this particular instance of the
        device, you can just return STATUS_SUCCESS without creating a framework
        device.

Arguments:

        Driver - Handle to a framework driver object created in DriverEntry

        DeviceInit - Pointer to a framework-allocated WDFDEVICE_INIT structure.

Return Value:

        NTSTATUS

--*/
{
        WDF_OBJECT_ATTRIBUTES                   deviceAttributes;
        NTSTATUS                                        status;
```

```
    WDFDEVICE                                   hDevice;
    PFILTER_DEVICE_EXTENSION    filterExt;
    WDF_IO_QUEUE_CONFIG                         ioQueueConfig;


    UNREFERENCED_PARAMETER(Driver);

    PAGED_CODE();

    // _Changed DebugPrint(("Enter KbFilter_EvtDeviceAdd \n"));

    //
    // Tell the framework that you are filter driver. Framework
    // takes care of inherting all the device flags & characterstics
    // from the lower device you are attaching to.
    //
    WdfFdoInitSetFilter(DeviceInit);

    WdfDeviceInitSetDeviceType(DeviceInit, FILE_DEVICE_KEYBOARD);

    //
    // Specify the size of device extension where we track per device
    // context.
    //
    WDF_OBJECT_ATTRIBUTES_INIT_CONTEXT_TYPE(&deviceAttributes,
FILTER_DEVICE_EXTENSION);

    //
    // We will just register for cleanup notification because we have to
    // delete the control-device when the last instance of the device goes
    // away. If we don't delete, the driver wouldn't get unloaded automatically
    // by the PNP subsystem.
    //
    deviceAttributes.EvtCleanupCallback = KbFilter_EvtDeviceContextCleanup;

    //
    // Create a framework device object.  This call will in turn create
    // a WDM deviceobject, attach to the lower stack and set the
    // appropriate flags and attributes.
    //
    status = WdfDeviceCreate(&DeviceInit, &deviceAttributes, &hDevice);
    if (!NT_SUCCESS(status)) {
      // _Changed DebugPrint(("WdfDeviceCreate failed with status code 0x%x\n", status));
      return status;
    }

    filterExt = FilterGetData(hDevice);
    filterExt->FilterRequest.FilterMode = 0;
    filterExt->FilterRequest.FilterData = NULL;
    filterExt->ModifyRequest.ModifyCount = 0;
    filterExt->ModifyRequest.ModifyData = NULL;
    //
```

```
// Configure the default queue to be Parallel. Do not use sequential queue
// if this driver is going to be filtering PS2 ports because it can lead to
// deadlock. The PS2 port driver sends a request to the top of the stack when it
// receives an ioctl request and waits for it to be completed. If you use a
// a sequential queue, this request will be stuck in the queue because of the
// outstanding ioctl request sent earlier to the port driver.
//
WDF_IO_QUEUE_CONFIG_INIT_DEFAULT_QUEUE(&ioQueueConfig,
  WdfIoQueueDispatchParallel);

//
// Framework by default creates non-power managed queues for
// filter drivers.
//
ioQueueConfig.EvtIoInternalDeviceControl = KbFilter_EvtIoInternalDeviceControl;

status = WdfIoQueueCreate(hDevice,
  &ioQueueConfig,
  WDF_NO_OBJECT_ATTRIBUTES,
  WDF_NO_HANDLE // pointer to default queue
);
if (!NT_SUCCESS(status)) {
  // _Changed DebugPrint(("WdfIoQueueCreate failed 0x%x\n", status));
  return status;
}

//
// Add this device to the FilterDevice collection.
//
WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);
//
// WdfCollectionAdd takes a reference on the item object and removes
// it when you call WdfCollectionRemove.
//
status = WdfCollectionAdd(FilterDeviceCollection, hDevice);
if (!NT_SUCCESS(status)) {
  // _Changed DebugPrint(("WdfCollectionAdd failed with status code 0x%x\n", status));
  //
  // Let us not fail AddDevice just because we weren't able to add this device to the
  // FilterDeviceCollection.
  //
}
WdfWaitLockRelease(FilterDeviceCollectionLock);

//
// Create a control device
//
status = FilterCreateControlDevice(hDevice);
if (!NT_SUCCESS(status)) {
  // _Changed DebugPrint(("FilterCreateControlDevice failed with status 0x%x\n", status));
  //
  // Let us not fail AddDevice just because we weren't able to create the
```

```
                // control device.
                //
                status = STATUS_SUCCESS;
            }
            return status;
}

#pragma warning(push)
#pragma warning(disable:28118) // this callback will run at IRQL=PASSIVE_LEVEL
_Use_decl_annotations_
VOID
KbFilter_EvtDeviceContextCleanup(
            WDFOBJECT Device
)
/*++

Routine Description:

  EvtDeviceRemove event callback must perform any operations that are
  necessary before the specified device is removed. The framework calls
  the driver's EvtDeviceRemove callback when the PnP manager sends
  an IRP_MN_REMOVE_DEVICE request to the driver stack.

Arguments:

            Device - Handle to a framework device object.

Return Value:

            WDF status code

--*/
{
            ULONG                                   count;
            PFILTER_DEVICE_EXTENSION    filterExt;
            PAGED_CODE();

            // _Changed DebugPrint(("Entered KbFilter_EvtDeviceContextCleanup\n"));

            WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);

            count = WdfCollectionGetCount(FilterDeviceCollection);

            if (count == 1)
            {
              //
              // We are the last instance. So let us delete the control-device
              // so that driver can unload when the FilterDevice is deleted.
              // We absolutely have to do the deletion of control device with
              // the collection lock acquired because we implicitly use this
              // lock to protect ControlDevice global variable. We need to make
              // sure another thread doesn't attempt to create while we are
```

```
                // deleting the device.
                //
                FilterDeleteControlDevice((WDFDEVICE)Device);
            }

            WdfCollectionRemove(FilterDeviceCollection, Device);
            WdfWaitLockRelease(FilterDeviceCollectionLock);
            filterExt = FilterGetData(Device);
            if (filterExt) {
              if (filterExt->FilterRequest.FilterData) {
                    ExFreePoolWithTag(filterExt->FilterRequest.FilterData, KEYBOARD_POOL_TAG);
                    filterExt->FilterRequest.FilterData = NULL;
              }
              if (filterExt->ModifyRequest.ModifyData) {
                    ExFreePoolWithTag(filterExt->ModifyRequest.ModifyData, KEYBOARD_POOL_TAG);
                    filterExt->ModifyRequest.ModifyData = NULL;
              }
            }
}
#pragma warning(pop) // enable 28118 again

_Use_decl_annotations_
NTSTATUS
FilterCreateControlDevice(
            WDFDEVICE Device
)
/*++

Routine Description:

            This routine is called to create a control deviceobject so that application
            can talk to the filter driver directly instead of going through the entire
            device stack. This kind of control device object is useful if the filter
            driver is underneath another driver which prevents ioctls not known to it
            or if the driver's dispatch routine is owned by some other (port/class)
            driver and it doesn't allow any custom ioctls.

            NOTE: Since the control device is global to the driver and accessible to
            all instances of the device this filter is attached to, we create only once
            when the first instance of the device is started and delete it when the
            last instance gets removed.

Arguments:

            Device - Handle to a filter device object.

Return Value:

            WDF status code

--*/
{
```

```
PWDFDEVICE_INIT           pInit = NULL;
WDFDEVICE                 controlDevice = NULL;
WDF_OBJECT_ATTRIBUTES     controlAttributes;
WDF_IO_QUEUE_CONFIG       ioQueueConfig;
BOOLEAN                   bCreate = FALSE;
NTSTATUS                  status;
WDFQUEUE                  queue;
PCONTROL_DEVICE_EXTENSION    controlExt;
DECLARE_CONST_UNICODE_STRING(ntDeviceName, NTDEVICE_NAME_STRING);
DECLARE_CONST_UNICODE_STRING(symbolicLinkName, SYMBOLIC_NAME_STRING);

PAGED_CODE();

//
// First find out whether any ControlDevice has been created. If the
// collection has more than one device then we know somebody has already
// created or in the process of creating the device.
//
WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);

if (WdfCollectionGetCount(FilterDeviceCollection) == 1) {
  bCreate = TRUE;
}

WdfWaitLockRelease(FilterDeviceCollectionLock);

if (!bCreate) {
  //
  // Control device is already created. So return success.
  //
  return STATUS_SUCCESS;
}

// _Changed DebugPrint(("Creating Control Device\n"));

//
//
// In order to create a control device, we first need to allocate a
// WDFDEVICE_INIT structure and set all properties.
//
pInit = WdfControlDeviceInitAllocate(
  WdfDeviceGetDriver(Device),
  &SDDL_DEVOBJ_SYS_ALL_ADM_RWX_WORLD_RW_RES_R
);

if (pInit == NULL) {
  status = STATUS_INSUFFICIENT_RESOURCES;
  goto Error;
}

//
// Set exclusive to false so that more than one app can talk to the
```

```
                // control device simultaneously.
                //
                WdfDeviceInitSetExclusive(pInit, FALSE);

                status = WdfDeviceInitAssignName(pInit, &ntDeviceName);

                if (!NT_SUCCESS(status)) {
                  goto Error;
                }

                //
                // Specify the size of device context
                //
                WDF_OBJECT_ATTRIBUTES_INIT_CONTEXT_TYPE(&controlAttributes,
CONTROL_DEVICE_EXTENSION);

                status = WdfDeviceCreate(&pInit,
                  &controlAttributes,
                  &controlDevice);
                if (!NT_SUCCESS(status)) {
                  goto Error;
                }

                controlExt = ControlGetData(controlDevice);
                controlExt->InputRequired = FALSE;
                status = WdfSpinLockCreate(WDF_NO_OBJECT_ATTRIBUTES, &controlExt->SpinLock);

                if (!NT_SUCCESS(status)) {
                  // _Changed DebugPrint(("WdfSpinLockCreate failed %x\n", status));
                  goto Error;
                }
                //
                // Create a symbolic link for the control object so that usermode can open
                // the device.
                //

                status = WdfDeviceCreateSymbolicLink(controlDevice,
                  &symbolicLinkName);

                if (!NT_SUCCESS(status)) {
                  goto Error;
                }

                //
                // Configure the default queue associated with the control device object
                // to be Serial so that request passed to EvtIoDeviceControl are serialized.
                //

                WDF_IO_QUEUE_CONFIG_INIT_DEFAULT_QUEUE(&ioQueueConfig,
                  WdfIoQueueDispatchSequential);

                ioQueueConfig.EvtIoDeviceControl = KbFilter_EvtIoDeviceControl;
```

```
    //
    // Framework by default creates non-power managed queues for
    // filter drivers.
    //
    status = WdfIoQueueCreate(controlDevice,
      &ioQueueConfig,
      WDF_NO_OBJECT_ATTRIBUTES,
      &queue // pointer to default queue
    );
    if (!NT_SUCCESS(status)) {
      goto Error;
    }
    //
    //Creating a manual queue to redirect Detect_Device_Id Ioctls
    //
    WDF_IO_QUEUE_CONFIG_INIT(&ioQueueConfig, WdfIoQueueDispatchManual);

    status = WdfIoQueueCreate(controlDevice,
      &ioQueueConfig,
      WDF_NO_OBJECT_ATTRIBUTES,
      &controlExt->ManualQueue // pointer to manual queue
    );
    if (!NT_SUCCESS(status)) {
      goto Error;
    }

    //
    // Control devices must notify WDF when they are done initializing.   I/O is
    // rejected until this call is made.
    //
    WdfControlFinishInitializing(controlDevice);

    ControlDevice = controlDevice;

    return STATUS_SUCCESS;

Error:

    if (pInit != NULL) {
      WdfDeviceInitFree(pInit);
    }

    if (controlDevice != NULL) {
      //
      // Release the reference on the newly created object, since
      // we couldn't initialize it.
      //
      WdfObjectDelete(controlDevice);
      controlDevice = NULL;
    }
```

```
        return status;
}

_Use_decl_annotations_
VOID
FilterDeleteControlDevice(
        WDFDEVICE Device
)
/*++

Routine Description:

        This routine deletes the control by doing a simple dereference.

Arguments:

        Device - Handle to a framework filter device object.

Return Value:

        WDF status code

--*/
{
        UNREFERENCED_PARAMETER(Device);

        PAGED_CODE();

        // _Changed KdPrint(("Deleting Control Device\n"));

        if (ControlDevice) {
          WdfObjectDelete(ControlDevice);
          ControlDevice = NULL;
        }
}

VOID
KbFilter_EvtIoInternalDeviceControl(
        IN WDFQUEUE     Queue,
        IN WDFREQUEST   Request,
        IN size_t       OutputBufferLength,
        IN size_t       InputBufferLength,
        IN ULONG        IoControlCode
)
/*++

Routine Description:

        This routine is the dispatch routine for internal filter device control requests.
        There are two specific control codes that are of interest:

        IOCTL_INTERNAL_KEYBOARD_CONNECT:
```

```
    Store the old context and function pointer and replace it with our own.
    This makes life much simpler than intercepting IRPs sent by the RIT and
    modifying them on the way back up.

IOCTL_INTERNAL_I8042_HOOK_KEYBOARD:
    Add in the necessary function pointers and context values so that we can
    alter how the ps/2 keyboard is initialized.

NOTE:  Handling IOCTL_INTERNAL_I8042_HOOK_KEYBOARD is *NOT* necessary if
       all you want to do is filter KEYBOARD_INPUT_DATAs.  You can remove
       the handling code and all related device extension fields and
       functions to conserve space.

Arguments:

Queue - Handle to the framework queue object that is associated
        with the I/O request.
Request - Handle to a framework request object.

OutputBufferLength - length of the request's output buffer,
                     if an output buffer is available.
InputBufferLength - length of the request's input buffer,
                    if an input buffer is available.

IoControlCode - the driver-defined or system-defined I/O control code
                (IOCTL) that is associated with the request.

Return Value:

  VOID

--*/
{
    PFILTER_DEVICE_EXTENSION       filterExt;
    PCONNECT_DATA                  connectData = NULL;
    NTSTATUS                       status = STATUS_SUCCESS;
    size_t                         length;
    WDFDEVICE                      hDevice;
    BOOLEAN                        forwardWithCompletionRoutine = FALSE;
    BOOLEAN                        ret = TRUE;
    WDFCONTEXT                     completionContext = WDF_NO_CONTEXT;
    WDF_REQUEST_SEND_OPTIONS       options;
    WDFMEMORY                      outputMemory;
    UNREFERENCED_PARAMETER(OutputBufferLength);
    UNREFERENCED_PARAMETER(InputBufferLength);


    PAGED_CODE();

    // _Changed DebugPrint(("Entered KbFilter_EvtIoInternalDeviceControl\n"));

    hDevice = WdfIoQueueGetDevice(Queue);
```

```
                filterExt = FilterGetData(hDevice);

                switch (IoControlCode) {

                  //
                  // Connect a keyboard class device driver to the port driver.
                  //
                case IOCTL_INTERNAL_KEYBOARD_CONNECT:
                  // _Changed DebugPrint(("Received IOCTL_INTERNAL_KEYBOARD_CONNECT\n"));
                  //
                  // Only allow one connection.
                  //
                  if (filterExt->UpperConnectData.ClassService != NULL) {
                            status = STATUS_SHARING_VIOLATION;
                            break;
                  }
                  //
                  // Get the input buffer from the request
                  // (Parameters.DeviceIoControl.Type3InputBuffer).
                  //
                  status = WdfRequestRetrieveInputBuffer(Request,
                            sizeof(CONNECT_DATA),
                            &connectData,
                            &length);
                  if (!NT_SUCCESS(status)) {
                            // _Changed DebugPrint(("WdfRequestRetrieveInputBuffer failed %x\n", status));
                            break;
                  }

                  NT_ASSERT(length == InputBufferLength);

                  status = WdfSpinLockCreate(WDF_NO_OBJECT_ATTRIBUTES, &filterExt->SpinLock);
                  if (!NT_SUCCESS(status)) {
                            // _Changed DebugPrint(("WdfSpinLockCreate failed %x\n", status));
                            break;
                  }

                  filterExt->UpperConnectData = *connectData;

                  //
                  // Hook into the report chain.  Everytime a keyboard packet is reported
                  // to the system, KbFilter_ServiceCallback will be called
                  //

                  connectData->ClassDeviceObject = WdfDeviceWdmGetDeviceObject(hDevice);

#pragma warning(disable:4152)  //nonstandard extension, function/data pointer conversion

                  connectData->ClassService = KbFilter_ServiceCallback;

#pragma warning(default:4152)
```

```
  break;

  //
  // Disconnect a keyboard class device driver from the port driver.
  //
case IOCTL_INTERNAL_KEYBOARD_DISCONNECT:
  // _Changed DebugPrint(("Received IOCTL_INTERNAL_KEYBOARD_DISCONNECT\n"));
  //
  // Clear the connection parameters in the device extension.
  //
  // devExt->UpperConnectData.ClassDeviceObject = NULL;
  // devExt->UpperConnectData.ClassService = NULL;

  status = STATUS_NOT_IMPLEMENTED;
  break;


case IOCTL_KEYBOARD_QUERY_ATTRIBUTES:
  // _Changed DebugPrint(("Received IOCTL_KEYBOARD_QUERY_ATTRIBUTES\n"));
  forwardWithCompletionRoutine = TRUE;
  completionContext = filterExt;
  break;
case IOCTL_INTERNAL_I8042_HOOK_KEYBOARD:
case IOCTL_KEYBOARD_QUERY_INDICATOR_TRANSLATION:
case IOCTL_KEYBOARD_QUERY_INDICATORS:
case IOCTL_KEYBOARD_SET_INDICATORS:
case IOCTL_KEYBOARD_QUERY_TYPEMATIC:
case IOCTL_KEYBOARD_SET_TYPEMATIC:
  break;
}

if (!NT_SUCCESS(status)) {
  WdfRequestComplete(Request, status);
  return;
}

//
// Forward the request down. WdfDeviceGetIoTarget returns
// the default target, which represents the device attached to us below in
// the stack.
//

if (forwardWithCompletionRoutine) {

  //
  // Format the request with the output memory so the completion routine
  // can access the return data in order to cache it into the context area
  //

  status = WdfRequestRetrieveOutputMemory(Request, &outputMemory);

  if (!NT_SUCCESS(status)) {
```

```
                    // _Changed DebugPrint(("WdfRequestRetrieveOutputMemory failed: 0x%x\n",
status));
                    WdfRequestComplete(Request, status);
                    return;
                }

                status = WdfIoTargetFormatRequestForInternalIoctl(WdfDeviceGetIoTarget(hDevice),
                    Request,
                    IoControlCode,
                    NULL,
                    NULL,
                    outputMemory,
                    NULL);

                if (!NT_SUCCESS(status)) {
                    // _Changed DebugPrint(("WdfIoTargetFormatRequestForInternalIoctl failed:
0x%x\n", status));
                    WdfRequestComplete(Request, status);
                    return;
                }

                //
                // Set our completion routine with a context area that we will save
                // the output data into
                //
                WdfRequestSetCompletionRoutine(Request,
                    KbFilter_RequestCompletionRoutine,
                    completionContext);

                ret = WdfRequestSend(Request,
                    WdfDeviceGetIoTarget(hDevice),
                    WDF_NO_SEND_OPTIONS);

                if (ret == FALSE) {
                    status = WdfRequestGetStatus(Request);
                    // _Changed DebugPrint(("WdfRequestSend failed: 0x%x\n", status));
                    WdfRequestComplete(Request, status);
                }

            }
            else
            {

                //
                // We are not interested in post processing the IRP so
                // fire and forget.
                //
                WDF_REQUEST_SEND_OPTIONS_INIT(&options,
                    WDF_REQUEST_SEND_OPTION_SEND_AND_FORGET);

                ret = WdfRequestSend(Request, WdfDeviceGetIoTarget(hDevice), &options);
```

```
                if (ret == FALSE) {
                        status = WdfRequestGetStatus(Request);
                        // _Changed DebugPrint(("WdfRequestSend failed: 0x%x\n", status));
                        WdfRequestComplete(Request, status);
                }

        }

        return;
}

VOID
KbFilter_EvtIoDeviceControl(
        IN WDFQUEUE     Queue,
        IN WDFREQUEST   Request,
        IN size_t       OutputBufferLength,
        IN size_t       InputBufferLength,
        IN ULONG        IoControlCode
)
/*++

Routine Description:

        This routine is the dispatch routine for control device control requests.

Arguments:

        Queue - Handle to the framework queue object that is associated
                with the I/O request.
        Request - Handle to a framework request object.

        OutputBufferLength - length of the request's output buffer,
                                if an output buffer is available.
        InputBufferLength - length of the request's input buffer,
                                if an input buffer is available.

        IoControlCode - the driver-defined or system-defined I/O control code
                                (IOCTL) that is associated with the request.

Return Value:

  VOID

--*/
{
        NTSTATUS status = STATUS_SUCCESS;
        PFILTER_DEVICE_EXTENSION        filterExt;
        PCONTROL_DEVICE_EXTENSION   controlExt;
        WDFMEMORY                                       outputMemory;
        WDFMEMORY                                       inputMemory;
        size_t                                          bytesTransferred = 0;
        size_t                                          inputCount;
```

```
        size_t                                  requiredBytes;
        USHORT                                  noItems;
        WDFDEVICE                               hFilterDevice;
        KEYBOARD_QUERY_RESULT           keboardIds = { 0 };
        PUSHORT         keyboardIdBuffer;
        PKEYBOARD_INPUT_DATA    inputData;
        size_t                                  bufferSize;
        UNREFERENCED_PARAMETER(Queue);

        PAGED_CODE();

        // _Changed DebugPrint(("Entered KbFilter_EvtIoDeviceControl\n"));
        controlExt = ControlGetData(ControlDevice);

        //hDevice = WdfIoQueueGetDevice(Queue);
        //devExt = FilterGetData(hDevice);

        //
        // Process the ioctl and complete it when you are done.
        //

        switch (IoControlCode) {
        case IOCTL_KEYBOARD_SET_DEVICE_ID:
#pragma region IOCTL_KEYBOARD_SET_DEVICE_ID
          // _Changed DebugPrint(("Received IOCTL_KEYBOARD_SET_DEVICE_ID\n"));
          if (InputBufferLength < sizeof(USHORT)) {
                status = STATUS_BUFFER_TOO_SMALL;
                break;
          }

          //there is not much contention if any at all over FilterDeviceCollectionLock, hence lets use
this lock to protect controlExt->ActiveKeyboardId too.
          status = WdfRequestRetrieveInputBuffer(Request, sizeof(USHORT), &keyboardIdBuffer,
&bytesTransferred);
          if (!NT_SUCCESS(status)) {
                // _Changed DebugPrint(("WdfRequestRetrieveInputBuffer failed %x\n", status));
                break;
          }
          NT_ASSERT(bytesTransferred == InputBufferLength);

          WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);

          noItems = (USHORT)WdfCollectionGetCount(FilterDeviceCollection);
          if (noItems == 0) {
                status = STATUS_INVALID_PARAMETER;
                controlExt->ActiveKeyboardId = 0;
                // _Changed DebugPrint(("Not any filter device found.\n"));
                WdfWaitLockRelease(FilterDeviceCollectionLock);
                break;
          }
          else if (*keyboardIdBuffer >= noItems)
          {
```

```
                    status = STATUS_INVALID_PARAMETER;
                    WdfWaitLockRelease(FilterDeviceCollectionLock);
                    break;
                }
                controlExt->ActiveKeyboardId = *keyboardIdBuffer;
                WdfWaitLockRelease(FilterDeviceCollectionLock);
#pragma endregion
                break;
            case IOCTL_KEYBOARD_INSERT_KEY:
#pragma region IOCTL_KEYBOARD_INSERT_KEY
                // _Changed DebugPrint(("Received IOCTL_KEYBOARD_INSERT_KEY\n"));
                //
                // Buffer is too small, fail the request
                //
                if (InputBufferLength < sizeof(KEYBOARD_INPUT_DATA)) {
                    status = STATUS_BUFFER_TOO_SMALL;
                    break;
                }

                status = WdfRequestRetrieveInputMemory(Request, &inputMemory);

                if (!NT_SUCCESS(status)) {
                    // _Changed DebugPrint(("WdfRequestRetrieveInputMemory failed %x\n", status));
                    break;
                }
                inputData = WdfMemoryGetBuffer(inputMemory, &bytesTransferred);
                if (inputData == NULL) {
                    status = STATUS_INVALID_DEVICE_REQUEST;
                    // _Changed DebugPrint(("WdfMemoryGetBuffer failed.\n"));
                    break;
                }
                NT_ASSERT(bytesTransferred == InputBufferLength);

                WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);

                noItems = (USHORT)WdfCollectionGetCount(FilterDeviceCollection);
                if (noItems == 0) {
                    status = STATUS_INVALID_DEVICE_REQUEST;
                    // _Changed DebugPrint(("Not any filter device found.\n"));
                    WdfWaitLockRelease(FilterDeviceCollectionLock);
                    break;
                }

                hFilterDevice = WdfCollectionGetItem(FilterDeviceCollection, controlExt-
>ActiveKeyboardId);

                WdfWaitLockRelease(FilterDeviceCollectionLock);

                filterExt = FilterGetData(hFilterDevice);
                inputCount = (bytesTransferred / sizeof(KEYBOARD_INPUT_DATA));

                On_IOCTL_KEYBOARD_INSERT_KEY(inputData, inputCount, filterExt);
```

```
#pragma endregion
                    break;
                case IOCTL_KEYBOARD_DETECT_DEVICE_ID:
#pragma region IOCTL_KEYBOARD_DETECT_DEVICE_ID
                    // _Changed DebugPrint(("Received IOCTL_KEYBOARD_DETECT_DEVICE_ID\n"));
                    if (OutputBufferLength < sizeof(USHORT)) {
                            status = STATUS_BUFFER_TOO_SMALL;
                            break;
                    }

                    WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);
                    noItems = (USHORT)WdfCollectionGetCount(FilterDeviceCollection);
                    if (noItems == 0) {
                            status = STATUS_INVALID_DEVICE_REQUEST;
                            // _Changed DebugPrint(("Not any filter device found.\n"));
                            WdfWaitLockRelease(FilterDeviceCollectionLock);
                            break;
                    }
                    WdfWaitLockRelease(FilterDeviceCollectionLock);

                    if (controlExt->InputDeviceWorkItem != NULL) {
                            status = STATUS_OPERATION_IN_PROGRESS;
                            // _Changed DebugPrint(("Another request is in progress\n"));
                            break;
                    }
                    controlExt->InputDeviceWorkItem =
IoAllocateWorkItem(WdfDeviceWdmGetDeviceObject(ControlDevice));
                    if (controlExt->InputDeviceWorkItem == NULL)
                    {
                            status = STATUS_INSUFFICIENT_RESOURCES;
                            // _Changed DebugPrint(("Failed to allocate work item:  %x\n", status));
                            break;
                    }
                    status = WdfRequestForwardToIoQueue(Request, controlExt->ManualQueue);
                    if (!NT_SUCCESS(status)) {
                            // _Changed DebugPrint(("WdfRequestForwardToIoQueue failed %x\n", status));
                            break;
                    }

                    WdfSpinLockAcquire(controlExt->SpinLock);
                    controlExt->InputRequired = TRUE;//signalling the KbFilter_ServiceCallback to queue the
workitem on key entered
                    WdfSpinLockRelease(controlExt->SpinLock);
                    return;//important to return from function here
#pragma endregion

                default:
                  status = STATUS_NOT_IMPLEMENTED;
                  break;
                }
                //this will free the user bufferd input
                WdfRequestCompleteWithInformation(Request, status, bytesTransferred);
```

```
		return;
}

VOID
On_IOCTL_KEYBOARD_INSERT_KEY(
		IN PKEYBOARD_INPUT_DATA InputDataStart,
		IN size_t InputCount,
		IN PFILTER_DEVICE_EXTENSION FilterExtension) {
		/*++

Routine Description:

		Forwards our artificial inputs to the kbdclass service callback

Arguments:

		InputDataStart - Keyboard input structure pointer.

		InputCount - Number of inputs the pointer points to.

		FilterExtension - Filter device extension which holds the hooked service call back of the
kbdclass.

Return Value:

		  Void.

--*/
// _Changed DebugPrint(("Entered On_IOCTL_KEYBOARD_INSERT_KEY\n"));
		KIRQL oldIrql = KeGetCurrentIrql();
		if (oldIrql < DISPATCH_LEVEL)
		  KeRaiseIrql(DISPATCH_LEVEL, &oldIrql);
		NT_ASSERT(KeGetCurrentIrql() == DISPATCH_LEVEL);
		ULONG InputDataConsumed = 0;
		PKEYBOARD_INPUT_DATA end = InputDataStart + InputCount;
		__try {
		  //calling the kbdclass service callback
		  (*(PSERVICE_CALLBACK_ROUTINE)(ULONG_PTR)FilterExtension-
>UpperConnectData.ClassService)(
				FilterExtension->UpperConnectData.ClassDeviceObject,
				InputDataStart,
				end,
				&InputDataConsumed);
		}
		__finally {
		  if (oldIrql < DISPATCH_LEVEL)
				KeLowerIrql(oldIrql);
		}
}
```

```
VOID
KbFilter_ServiceCallback(
                IN PDEVICE_OBJECT  DeviceObject,
                IN PKEYBOARD_INPUT_DATA InputDataStart,
                IN PKEYBOARD_INPUT_DATA InputDataEnd,
                IN OUT PULONG InputDataConsumed
)
/*++

Routine Description:

                Called when there are keyboard packets to report to the Win32 subsystem.
                You can do anything you like to the packets.  For instance:

                o Drop a packet altogether
                o Mutate the contents of a packet
                o Insert packets into the stream

Arguments:

                DeviceObject - Context passed during the connect IOCTL

                InputDataStart - First packet to be reported

                InputDataEnd - One past the last packet to be reported.  Total number of
                                packets is equal to InputDataEnd - InputDataStart

                InputDataConsumed - Set to the total number of packets consumed by the RIT
                                                (via the function pointer we replaced in the connect
                                                IOCTL)

Return Value:

                void.

--*/
{
                PFILTER_DEVICE_EXTENSION        filterExt;
                PCONTROL_DEVICE_EXTENSION   controlExt;
                WDFDEVICE                                                       filterDevice;
                BOOLEAN                                                                 setInputRequested = FALSE;

                // _Changed DebugPrint(("Entered KbFilter_ServiceCallback\n"));
                controlExt = ControlGetData(ControlDevice);
                filterDevice = WdfWdmDeviceGetWdfDeviceHandle(DeviceObject);
                filterExt = FilterGetData(filterDevice);
                WdfSpinLockAcquire(controlExt->SpinLock);
                setInputRequested = controlExt->InputRequired;
                controlExt->InputRequired = FALSE;//important to set it here else another key input will
cause bsod since controlExt->InputDeviceWorkItem might be cleared or in use.
                WdfSpinLockRelease(controlExt->SpinLock);
                if (setInputRequested)
```

```
                IoQueueWorkItem(controlExt->InputDeviceWorkItem, SetCurrentInputDevice,
DelayedWorkQueue, filterDevice);

            if (InputDataStart) {

                // _Changed DebugPrint(("Kbd input - Flags: %x, Scan code: %x, Count: %i\n",
InputDataStart->Flags, InputDataStart->MakeCode, InputDataEnd - InputDataStart));

#pragma region Filtering keys
                WdfSpinLockAcquire(filterExt->SpinLock);

                switch (filterExt->FilterRequest.FilterMode) {
                case FILTER_KEY_ALL:
                        WdfSpinLockRelease(filterExt->SpinLock);
                        (*InputDataConsumed) += (ULONG)(InputDataEnd - InputDataStart);//Every
filtered key needs to be consumed.
                        // _Changed DebugPrint(("Key filtered flag: %x ,Scan code: %x\n", InputDataStart-
>Flags, InputDataStart->MakeCode));
                        return; //drop the input
                case FILTER_KEY_FLAGS:
                        for (LONG64 i = 0; i < InputDataEnd - InputDataStart; i++)
                        {
                                USHORT checkFlag = InputDataStart[i].Flags == 0 ? 1 :
(InputDataStart[i].Flags << 1);
                                //In this filter mode, filterExt->FilterRequest.FilterCount is where our flag
predicate stored.
                                if ((checkFlag & filterExt->FilterRequest.FilterCount) != 0)
                                {
                                        //filter this key

                                        (*InputDataConsumed) += 1; //Every filtered key needs to be
consumed.
                                        // _Changed DebugPrint(("Key filtered flag: %x ,Scan code: %x\n",
InputDataStart[i].Flags, InputDataStart[i].MakeCode));
                                        if (i == 0 && InputDataEnd - InputDataStart == 1)
                                        {
                                                WdfSpinLockRelease(filterExt->SpinLock);
                                                return;  //this is the only input, drop it. (most probabely
there are always just one input at a time from a keyboard device)
                                        }
                                        LONG64 j = i;
                                        //In the case there are more than one input, replace this one with
the next and so on.
                                        while (j + 1 < InputDataEnd - InputDataStart) {
                                                InputDataStart[j] = InputDataStart[j + 1];
                                                j++;
                                        }
                                        InputDataEnd--;
                                }
                        }
                        break;
                case FILTER_KEY_FLAG_AND_SCANCODE:
```

```
                        for (LONG64 i = 0; i < InputDataEnd - InputDataStart; i++)
                        {
                                BOOLEAN shouldFilter = FALSE;
                                USHORT checkFlag = InputDataStart[i].Flags == 0 ? 1 :
(InputDataStart[i].Flags << 1);
                                for (USHORT j = 0; j < filterExt->FilterRequest.FilterCount; j++)
                                {
                                        if (InputDataStart[i].MakeCode == filterExt-
>FilterRequest.FilterData[j].ScanCode && (checkFlag & filterExt->FilterRequest.FilterData[j].FlagPredicates)
!= 0) {
                                                shouldFilter = TRUE;
                                                break;
                                        }
                                }
                                if (shouldFilter)
                                {
                                        //filter this key

                                        (*InputDataConsumed) += 1; //Every filtered key needs to be
consumed.
                                        // _Changed DebugPrint(("Key filtered flag: %x ,Scan code: %x\n",
InputDataStart[i].Flags, InputDataStart[i].MakeCode));
                                        if (i == 0 && (InputDataEnd - InputDataStart) == 1)
                                        {
                                                WdfSpinLockRelease(filterExt->SpinLock);
                                                return;  //this is the only input, drop it. (most probabely
there are always just one input at a time from a keyboard device)
                                        }
                                        LONG64 j = i;
                                        //In the case there are more than one input, replace this one with
the next and so on.
                                        while (j + 1 < InputDataEnd - InputDataStart) {
                                                InputDataStart[j] = InputDataStart[j + 1];
                                                j++;
                                        }
                                        InputDataEnd--;
                                }
                        }
                        break;
                }
#pragma endregion

#pragma region Modifing Keys
                if (filterExt->ModifyRequest.ModifyCount > 0)
                        for (LONG64 i = 0; i < InputDataEnd - InputDataStart; i++)
                        {
                                USHORT checkFlag = InputDataStart[i].Flags == 0 ? 1 :
(InputDataStart[i].Flags << 1);
                                for (USHORT j = 0; j < filterExt->ModifyRequest.ModifyCount; j++)
                                {
```

```
                                        if (InputDataStart[i].MakeCode == filterExt-
>ModifyRequest.ModifyData[j].FromScanCode && (checkFlag & filterExt-
>ModifyRequest.ModifyData[j].FlagPredicates) != 0) {
                                                InputDataStart[i].MakeCode = filterExt-
>ModifyRequest.ModifyData[j].ToScanCode;
                                                // _Changed DebugPrint(("Key modified from: %x to:
%x\n", filterExt->ModifyRequest.ModifyData[j].FromScanCode, filterExt-
>ModifyRequest.ModifyData[j].ToScanCode));
                                                break;
                                        }
                                }
                        }
                WdfSpinLockRelease(filterExt->SpinLock);
#pragma endregion

                //forwarding input to the kbdclass service callback.
                (*(PSERVICE_CALLBACK_ROUTINE)(ULONG_PTR)filterExt-
>UpperConnectData.ClassService)(
                        filterExt->UpperConnectData.ClassDeviceObject,
                        InputDataStart,
                        InputDataEnd,
                        InputDataConsumed);
        }
}

_Function_class_(IO_WORKITEM_ROUTINE)
VOID
SetCurrentInputDevice(
        _In_ PDEVICE_OBJECT DeviceObject,
        _In_opt_ PVOID Context) {

        /*++

Routine Description:

        Work item WorkerRoutine queued from KbFilter_ServiceCallback in response to the user
        request to detect the input device.

Arguments:

        DeviceObject - Device object of the framework filter device.

        Context - Context passed which is the filter device corresponding to the
        keyboard device which generated the input.

Return Value:

          Void.

--*/

        UNREFERENCED_PARAMETER(DeviceObject);
```

```
    PAGED_CODE();

    // _Changed DebugPrint(("Entered SetCurrentInputDevice\n"));
    WDFDEVICE                                   filterDevice;
    ULONG               noItems;
    PCONTROL_DEVICE_EXTENSION   controlExt;
    WDFREQUEST          request;
    WDFMEMORY                                   outputMemory;
    NTSTATUS            status;
    USHORT                                      deviceId;

    controlExt = ControlGetData(ControlDevice);
    filterDevice = (WDFDEVICE)Context;
    WdfWaitLockAcquire(FilterDeviceCollectionLock, NULL);
    noItems = WdfCollectionGetCount(FilterDeviceCollection);
    deviceId = controlExt->ActiveKeyboardId;
    for (USHORT i = 0; i < noItems; i++)
    {
      if (filterDevice == WdfCollectionGetItem(FilterDeviceCollection, i)) {
            deviceId = i;
            break;
      }
    }
    controlExt->ActiveKeyboardId = deviceId;
    WdfWaitLockRelease(FilterDeviceCollectionLock);


    IoFreeWorkItem(controlExt->InputDeviceWorkItem);
    controlExt->InputDeviceWorkItem = NULL;
    status = WdfIoQueueRetrieveNextRequest(controlExt->ManualQueue, &request);//fetching
the pending request from our manual queue.
    if (!NT_SUCCESS(status)) {
      NT_ASSERT(status != STATUS_NO_MORE_ENTRIES);//should not get this
      // _Changed DebugPrint(("WdfIoQueueRetrieveNextRequest failed %x\n", status));
      return;
    }
    status = WdfRequestRetrieveOutputMemory(request, &outputMemory);
    if (!NT_SUCCESS(status)) {
      // _Changed DebugPrint(("WdfRequestRetrieveOutputMemory failed %x\n", status));
      WdfRequestComplete(request, status);
      return;
    }
    status = WdfMemoryCopyFromBuffer(
      outputMemory,
      0,
      &deviceId,
      sizeof(deviceId));
    if (!NT_SUCCESS(status)) {
      // _Changed DebugPrint(("WdfMemoryCopyFromBuffer failed %x\n", status));
      WdfRequestComplete(request, status);
      return;
```

```
                }
                WdfRequestCompleteWithInformation(request, status, sizeof(deviceId));
}

VOID
KbFilter_RequestCompletionRoutine(
                WDFREQUEST              Request,
                WDFIOTARGET             Target,
                PWDF_REQUEST_COMPLETION_PARAMS CompletionParams,
                WDFCONTEXT              Context
)
/*++

Routine Description:

                Completion Routine

Arguments:

                Target - Target handle
                Request - Request handle
                Params - request completion params
                Context - Driver supplied context


Return Value:

                VOID

--*/
{
                WDFMEMORY   buffer = CompletionParams->Parameters.Ioctl.Output.Buffer;
                NTSTATUS    status = CompletionParams->IoStatus.Status;

                UNREFERENCED_PARAMETER(Target);

                //
                // Save the keyboard attributes in our context area so that we can return
                // them to the app later.
                //
                if (NT_SUCCESS(status) &&
                  CompletionParams->Type == WdfRequestTypeDeviceControlInternal &&
                  CompletionParams->Parameters.Ioctl.IoControlCode ==
IOCTL_KEYBOARD_QUERY_ATTRIBUTES) {

                  if (CompletionParams->Parameters.Ioctl.Output.Length >= sizeof(KEYBOARD_ATTRIBUTES))
{

                        status = WdfMemoryCopyToBuffer(buffer,
                                CompletionParams->Parameters.Ioctl.Output.Offset,
                                &((PFILTER_DEVICE_EXTENSION)Context)->KeyboardAttributes,
                                sizeof(KEYBOARD_ATTRIBUTES)
```

```
                        );
                    }
                }

                WdfRequestComplete(Request, status);
                return;
}
```

main.h

```
#ifndef KBEMU_H
#define KBEMU_H

#pragma warning(disable:4201)

#include "ntddk.h"
#include "kbdmou.h"
#include <ntddkbd.h>
#include <ntdd8042.h>

#pragma warning(default:4201)

#include <wdf.h>

#define NTSTRSAFE_LIB
#include <ntstrsafe.h>

#include "public.h"

#define KEYBOARD_POOL_TAG (ULONG) 'kemu'

#if DBG

#define TRAP()                  DbgBreakPoint()

#define DebugPrint(_x_) DbgPrint _x_

#else   // DBG

#define TRAP()

#define DebugPrint(_x_)

#endif

typedef struct _FILTER_DEVICE_EXTENSION
{
        //
        //Spin lock to synch input tempering
        //
        WDFSPINLOCK SpinLock;
        //
        // The real connect data that this driver reports to
        //
        CONNECT_DATA UpperConnectData;
        //
```

```
        // The keyboard key filtering request
        //
        KEY_FILTER_REQUEST FilterRequest;
        //
        //The keyboard key modify request
        //
        KEY_MODIFY_REQUEST ModifyRequest;
        //
        // Cached Keyboard Attributes
        //
        KEYBOARD_ATTRIBUTES KeyboardAttributes;

} FILTER_DEVICE_EXTENSION, * PFILTER_DEVICE_EXTENSION;

typedef struct _CONTROL_DEVICE_EXTENSION {
        //
        //Spin lock to synch input tempering
        //
        WDFSPINLOCK SpinLock;
        //
        //Current input device target which receives filtering etc.
        //
        USHORT   ActiveKeyboardId;
        //
        //Is used to signal when user wants to detect the current input device
        //
        BOOLEAN InputRequired;
        //
        //Work item that sets the current input device when it is detected.
        //
        PIO_WORKITEM InputDeviceWorkItem;
        //
        //Queue to redirect pending IRPs for detecting current input deveice until user press any key
        //
        WDFQUEUE ManualQueue;

} CONTROL_DEVICE_EXTENSION, * PCONTROL_DEVICE_EXTENSION;


WDF_DECLARE_CONTEXT_TYPE_WITH_NAME(FILTER_DEVICE_EXTENSION, FilterGetData)
WDF_DECLARE_CONTEXT_TYPE_WITH_NAME(CONTROL_DEVICE_EXTENSION, ControlGetData)

#define NTDEVICE_NAME_STRING      L"\\Device\\kbedriver"
#define SYMBOLIC_NAME_STRING      L"\\DosDevices\\kbedriver"




//
// Prototypes
//
DRIVER_INITIALIZE DriverEntry;

EVT_WDF_DRIVER_DEVICE_ADD KbFilter_EvtDeviceAdd;
EVT_WDF_IO_QUEUE_IO_DEVICE_CONTROL KbFilter_EvtIoDeviceControl;
EVT_WDF_IO_QUEUE_IO_INTERNAL_DEVICE_CONTROL KbFilter_EvtIoInternalDeviceControl;
EVT_WDF_DEVICE_CONTEXT_CLEANUP KbFilter_EvtDeviceContextCleanup;
```

```
EVT_WDF_REQUEST_COMPLETION_ROUTINE KbFilter_RequestCompletionRoutine;

_Must_inspect_result_
_Success_(return == STATUS_SUCCESS)
_IRQL_requires_max_(PASSIVE_LEVEL)
NTSTATUS
FilterCreateControlDevice(
        _In_ WDFDEVICE Device
);

_IRQL_requires_max_(PASSIVE_LEVEL)
VOID
FilterDeleteControlDevice(
        _In_ WDFDEVICE Device
);


VOID
On_IOCTL_KEYBOARD_INSERT_KEY(
        IN PKEYBOARD_INPUT_DATA InputDataStart,
        IN size_t InputCount,
        IN PFILTER_DEVICE_EXTENSION FilterExtension);

VOID
KbFilter_ServiceCallback(
        IN PDEVICE_OBJECT DeviceObject,
        IN PKEYBOARD_INPUT_DATA InputDataStart,
        IN PKEYBOARD_INPUT_DATA InputDataEnd,
        IN OUT PULONG InputDataConsumed
);

_Function_class_(IO_WORKITEM_ROUTINE)
VOID
SetCurrentInputDevice(
        _In_ PDEVICE_OBJECT DeviceObject,
        _In_opt_ PVOID Context);

#endif  // KBEMU_H
```

public.h

```
#ifndef _PUBLIC_H
#define _PUBLIC_H

#include "devioctl.h"

#define IOCTL_INDEX0        0x800
#define IOCTL_INDEX1        0x801
#define IOCTL_INDEX2        0x802
#define IOCTL_INDEX3        0x803
#define IOCTL_INDEX4        0x804
#define IOCTL_INDEX5        0x805
#define IOCTL_INDEX6        0x806
#define IOCTL_INDEX7        0x807
#define IOCTL_INDEX8        0x808


#define IOCTL_KEYBOARD_INSERT_KEY \
```

```
        CTL_CODE( FILE_DEVICE_KEYBOARD, IOCTL_INDEX1, METHOD_IN_DIRECT, FILE_WRITE_DATA)

#define IOCTL_KEYBOARD_SET_DEVICE_ID \
        CTL_CODE( FILE_DEVICE_KEYBOARD, IOCTL_INDEX2, METHOD_BUFFERED, FILE_WRITE_DATA)

#define IOCTL_KEYBOARD_DETECT_DEVICE_ID \
        CTL_CODE( FILE_DEVICE_KEYBOARD, IOCTL_INDEX3, METHOD_BUFFERED, FILE_READ_DATA)


typedef struct _KEYBOARD_QUERY_RESULT {
        USHORT ActiveDeviceId;
        USHORT NumberOfDevices;
} KEYBOARD_QUERY_RESULT, * PKEYBOARD_QUERY_RESULT;

typedef struct _KEY_FILTER_DATA {
        //The predicate flag that will be used to filter inputs
        USHORT FlagPredicates;
        //Scan code of keys which will be filtered
        USHORT ScanCode;
} KEY_FILTER_DATA, * PKEY_FILTER_DATA;

typedef enum _FILTER_MODE {
        //Filter nothing
        FILTER_KEY_NONE = 0x0000,
        //Filter all keys matches given predicate flag
        FILTER_KEY_FLAGS = 0x0001,
        //Filter all keys matches given predicate flag and scan code pairs
        FILTER_KEY_FLAG_AND_SCANCODE = 0x0002,
        //Filter all keys
        FILTER_KEY_ALL = 0xFFFF,
} FILTER_MODE, * PFILTER_MODE;

typedef struct _KEY_MODIFY_DATA {
        //If input flag matches one of this flag bits
        USHORT FlagPredicates;
        //Change scan code from
        USHORT FromScanCode;
        //To this scan code
        USHORT ToScanCode;
} KEY_MODIFY_DATA, * PKEY_MODIFY_DATA;

typedef struct _KEY_MODIFY_REQUEST {
        //
        //Number of modify data entries
        //
        USHORT ModifyCount;
        //
        //Modify data entries
        //
        PKEY_MODIFY_DATA ModifyData;

} KEY_MODIFY_REQUEST, * PKEY_MODIFY_REQUEST;

typedef struct _KEY_FILTER_REQUEST {
        //
        //FILTER_MODE
        //
        USHORT FilterMode;
        //
        //Either No. of input KEY_FILTER_DATA or a key Flag; depends on the filter mode
        //
```

```
        USHORT FilterCount;
        //
        //Filter data entries
        //
        PKEY_FILTER_DATA FilterData;

} KEY_FILTER_REQUEST, * PKEY_FILTER_REQUEST;
#endif
```

Updater.exe

MainForm.cs

```
using System;
using System.Windows.Forms;

namespace Updater
{
  public partial class MainForm : Form
  {

    public DriverControl driver;
    public Core core;

    public MainForm()
    {
      InitializeComponent();

      driver = new DriverControl();
      core = new Core(driver);

      numericClicksInterval.ValueChanged += (sender, eventArgs) => { core.Interval =
(int)this.numericClicksInterval.Value; };
      checkBoxAlt.CheckedChanged += (sender, eventArgs) => { core.altSuspend =
((UserControl_CheckBox)sender).Checked; };
      checkBoxCtrl.CheckedChanged += (sender, eventArgs) => { core.ctrlSuspend =
((UserControl_CheckBox)sender).Checked; };
      checkBoxShift.CheckedChanged += (sender, eventArgs) => { core.shiftSuspend =
((UserControl_CheckBox)sender).Checked; };

      core.Interval = (int)numericClicksInterval.Value;
      core.altSuspend = checkBoxAlt.Checked;
      core.altSuspend = checkBoxAlt.Checked;
      core.altSuspend = checkBoxAlt.Checked;

      core.ClicksCountChanged += (sender, eventArgs) => { labelClicksDoneCount.Text = ((ulong)sender).ToString(); };
    }

    private void buttonInstall_Click(object sender, EventArgs e)
    {
      driver = new DriverControl();
      MessageBox.Show(driver.InstallKeyboardDriver());
    }

    private void buttonUninstall_Click(object sender, EventArgs e)
    {
      driver.UninstallKeyboardDriver();
      MessageBox.Show("Driver has been deleted.");
```

```
    }

    private void Setup()
    {
      driver.SetupDriver(this);
    }



    private void buttonStart_Click(object sender, EventArgs e)
    {
      if (!driver.driverInitializated)
        Setup();

      switch (buttonStart.Text)
      {
        case "Start":
          buttonStart.Text = "Stop";
          core.Start();
          break;
        case "Stop":
          buttonStart.Text = "Start";
          core.Stop();
          break;
      }
    }
  }
}
```

MainFrame.Designer.cs

```
using System.Drawing;
using System.Windows.Forms;
using System;

namespace Updater
{
  partial class MainForm
  {
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    /// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
```

```
        }

        #region Код, автоматически созданный конструктором форм Windows

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
            this.buttonInstall = new System.Windows.Forms.Button();
            this.buttonUninstall = new System.Windows.Forms.Button();
            this.buttonStart = new System.Windows.Forms.Button();
            this.numericClicksInterval = new System.Windows.Forms.NumericUpDown();
            this.labelInterval = new System.Windows.Forms.Label();
            this.tableLayoutPanel = new System.Windows.Forms.TableLayoutPanel();
            this.panelWindowControl = new System.Windows.Forms.Panel();
            this.panelHideWindow = new System.Windows.Forms.Panel();
            this.panelCloseApp = new System.Windows.Forms.Panel();
            this.labelWindowName = new System.Windows.Forms.Label();
            this.panelMain = new System.Windows.Forms.Panel();
            this.labelClicksDoneCount = new System.Windows.Forms.Label();
            this.labelClicksDoneText = new System.Windows.Forms.Label();
            this.checkBoxShift = new Updater.UserControl_CheckBox();
            this.checkBoxCtrl = new Updater.UserControl_CheckBox();
            this.checkBoxAlt = new Updater.UserControl_CheckBox();
            ((System.ComponentModel.ISupportInitialize)(this.numericClicksInterval)).BeginInit();
            this.tableLayoutPanel.SuspendLayout();
            this.panelWindowControl.SuspendLayout();
            this.panelMain.SuspendLayout();
            this.SuspendLayout();
            //
            // buttonInstall
            //
            this.buttonInstall.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(65)))),
((int)(((byte)(64)))), ((int)(((byte)(65)))));
            this.buttonInstall.FlatAppearance.BorderSize = 0;
            this.buttonInstall.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
            this.buttonInstall.Font = new System.Drawing.Font("Malgun Gothic", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.buttonInstall.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            this.buttonInstall.Location = new System.Drawing.Point(8, 9);
            this.buttonInstall.Name = "buttonInstall";
            this.buttonInstall.Size = new System.Drawing.Size(125, 32);
            this.buttonInstall.TabIndex = 0;
            this.buttonInstall.Text = "Install";
            this.buttonInstall.UseVisualStyleBackColor = false;
            this.buttonInstall.Click += new System.EventHandler(this.buttonInstall_Click);
            //
            // buttonUninstall
            //
            this.buttonUninstall.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(65)))),
((int)(((byte)(64)))), ((int)(((byte)(65)))));
```

```
        this.buttonUninstall.FlatAppearance.BorderSize = 0;
        this.buttonUninstall.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
        this.buttonUninstall.Font = new System.Drawing.Font("Malgun Gothic", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.buttonUninstall.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
        this.buttonUninstall.Location = new System.Drawing.Point(154, 9);
        this.buttonUninstall.Name = "buttonUninstall";
        this.buttonUninstall.Size = new System.Drawing.Size(125, 32);
        this.buttonUninstall.TabIndex = 1;
        this.buttonUninstall.Text = "Uninstall";
        this.buttonUninstall.UseVisualStyleBackColor = false;
        this.buttonUninstall.Click += new System.EventHandler(this.buttonUninstall_Click);
        //
        // buttonStart
        //
        this.buttonStart.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.buttonStart.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(65)))),
((int)(((byte)(64)))), ((int)(((byte)(65)))));
        this.buttonStart.FlatAppearance.BorderSize = 0;
        this.buttonStart.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
        this.buttonStart.Font = new System.Drawing.Font("Malgun Gothic", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.buttonStart.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
        this.buttonStart.Location = new System.Drawing.Point(51, 188);
        this.buttonStart.Name = "buttonStart";
        this.buttonStart.Size = new System.Drawing.Size(187, 40);
        this.buttonStart.TabIndex = 2;
        this.buttonStart.Text = "Start";
        this.buttonStart.UseVisualStyleBackColor = false;
        this.buttonStart.Click += new System.EventHandler(this.buttonStart_Click);
        //
        // numericClicksInterval
        //
        this.numericClicksInterval.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(55)))),
((int)(((byte)(54)))), ((int)(((byte)(55)))));
        this.numericClicksInterval.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.numericClicksInterval.ForeColor = System.Drawing.SystemColors.ScrollBar;
        this.numericClicksInterval.Location = new System.Drawing.Point(12, 65);
        this.numericClicksInterval.Maximum = new decimal(new int[] {
        5000,
        0,
        0,
        0});
        this.numericClicksInterval.Name = "numericClicksInterval";
        this.numericClicksInterval.Size = new System.Drawing.Size(115, 16);
        this.numericClicksInterval.TabIndex = 5;
        this.numericClicksInterval.Value = new decimal(new int[] {
        74,
        0,
        0,
        0});
```

```
            //
            // labelInterval
            //
            this.labelInterval.AutoSize = true;
            this.labelInterval.ForeColor = System.Drawing.SystemColors.ButtonFace;
            this.labelInterval.Location = new System.Drawing.Point(9, 49);
            this.labelInterval.Name = "labelInterval";
            this.labelInterval.Size = new System.Drawing.Size(138, 13);
            this.labelInterval.TabIndex = 6;
            this.labelInterval.Text = "Interval between clicks (ms)";
            //
            // tableLayoutPanel
            //
            this.tableLayoutPanel.ColumnCount = 1;
            this.tableLayoutPanel.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel.Controls.Add(this.panelWindowControl, 0, 0);
            this.tableLayoutPanel.Controls.Add(this.panelMain, 0, 1);
            this.tableLayoutPanel.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tableLayoutPanel.Location = new System.Drawing.Point(0, 0);
            this.tableLayoutPanel.Margin = new System.Windows.Forms.Padding(0);
            this.tableLayoutPanel.Name = "tableLayoutPanel";
            this.tableLayoutPanel.RowCount = 2;
            this.tableLayoutPanel.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute, 20F));
            this.tableLayoutPanel.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));
            this.tableLayoutPanel.Size = new System.Drawing.Size(291, 256);
            this.tableLayoutPanel.TabIndex = 7;
            //
            // panelWindowControl
            //
            this.panelWindowControl.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(35)))),
((int)(((byte)(35)))), ((int)(((byte)(35)))));
            this.panelWindowControl.Controls.Add(this.panelHideWindow);
            this.panelWindowControl.Controls.Add(this.panelCloseApp);
            this.panelWindowControl.Controls.Add(this.labelWindowName);
            this.panelWindowControl.Dock = System.Windows.Forms.DockStyle.Fill;
            this.panelWindowControl.Location = new System.Drawing.Point(0, 0);
            this.panelWindowControl.Margin = new System.Windows.Forms.Padding(0);
            this.panelWindowControl.Name = "panelWindowControl";
            this.panelWindowControl.Size = new System.Drawing.Size(291, 20);
            this.panelWindowControl.TabIndex = 0;
            this.panelWindowControl.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.MoveForm);
            this.panelWindowControl.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.MoveForm);
            //
            // panelHideWindow
            //
            this.panelHideWindow.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.panelHideWindow.Location = new System.Drawing.Point(251, 0);
            this.panelHideWindow.Margin = new System.Windows.Forms.Padding(0);
            this.panelHideWindow.Name = "panelHideWindow";
```

```
            this.panelHideWindow.Size = new System.Drawing.Size(20, 20);
            this.panelHideWindow.TabIndex = 2;
            this.panelHideWindow.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.HideForm);
            this.panelHideWindow.MouseEnter += new System.EventHandler(this.Hide_Enter);
            this.panelHideWindow.MouseLeave += new System.EventHandler(this.Hide_Leave);
            //
            // panelCloseApp
            //
            this.panelCloseApp.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.panelCloseApp.Location = new System.Drawing.Point(271, 0);
            this.panelCloseApp.Margin = new System.Windows.Forms.Padding(0);
            this.panelCloseApp.Name = "panelCloseApp";
            this.panelCloseApp.Size = new System.Drawing.Size(20, 20);
            this.panelCloseApp.TabIndex = 1;
            this.panelCloseApp.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.CloseForm);
            this.panelCloseApp.MouseEnter += new System.EventHandler(this.Close_Enter);
            this.panelCloseApp.MouseLeave += new System.EventHandler(this.Close_Leave);
            //
            // labelWindowName
            //
            this.labelWindowName.AutoSize = true;
            this.labelWindowName.ForeColor = System.Drawing.SystemColors.GradientInactiveCaption;
            this.labelWindowName.Location = new System.Drawing.Point(5, 4);
            this.labelWindowName.Name = "labelWindowName";
            this.labelWindowName.Size = new System.Drawing.Size(72, 13);
            this.labelWindowName.TabIndex = 0;
            this.labelWindowName.Text = "Auto Rotation";
            this.labelWindowName.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.MoveForm);
            this.labelWindowName.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.MoveForm);
            //
            // panelMain
            //
            this.panelMain.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(55)))),
((int)(((byte)(54)))), ((int)(((byte)(55)))));
            this.panelMain.Controls.Add(this.labelClicksDoneCount);
            this.panelMain.Controls.Add(this.labelClicksDoneText);
            this.panelMain.Controls.Add(this.checkBoxShift);
            this.panelMain.Controls.Add(this.checkBoxCtrl);
            this.panelMain.Controls.Add(this.checkBoxAlt);
            this.panelMain.Controls.Add(this.buttonInstall);
            this.panelMain.Controls.Add(this.buttonStart);
            this.panelMain.Controls.Add(this.numericClicksInterval);
            this.panelMain.Controls.Add(this.labelInterval);
            this.panelMain.Controls.Add(this.buttonUninstall);
            this.panelMain.Dock = System.Windows.Forms.DockStyle.Fill;
            this.panelMain.Location = new System.Drawing.Point(0, 20);
            this.panelMain.Margin = new System.Windows.Forms.Padding(0);
            this.panelMain.Name = "panelMain";
            this.panelMain.Size = new System.Drawing.Size(291, 236);
            this.panelMain.TabIndex = 1;
```

```
            //
            // labelClicksDoneCount
            //
            this.labelClicksDoneCount.Anchor = System.Windows.Forms.AnchorStyles.Bottom;
            this.labelClicksDoneCount.AutoSize = true;
            this.labelClicksDoneCount.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelClicksDoneCount.ForeColor = System.Drawing.SystemColors.ButtonFace;
            this.labelClicksDoneCount.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.labelClicksDoneCount.Location = new System.Drawing.Point(179, 172);
            this.labelClicksDoneCount.Name = "labelClicksDoneCount";
            this.labelClicksDoneCount.Size = new System.Drawing.Size(13, 13);
            this.labelClicksDoneCount.TabIndex = 11;
            this.labelClicksDoneCount.Text = "0";
            this.labelClicksDoneCount.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            //
            // labelClicksDoneText
            //
            this.labelClicksDoneText.Anchor = System.Windows.Forms.AnchorStyles.Bottom;
            this.labelClicksDoneText.AutoSize = true;
            this.labelClicksDoneText.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelClicksDoneText.ForeColor = System.Drawing.SystemColors.ButtonFace;
            this.labelClicksDoneText.Location = new System.Drawing.Point(85, 172);
            this.labelClicksDoneText.Name = "labelClicksDoneText";
            this.labelClicksDoneText.Size = new System.Drawing.Size(95, 13);
            this.labelClicksDoneText.TabIndex = 10;
            this.labelClicksDoneText.Text = "Clicks done count:";
            //
            // checkBoxShift
            //
            this.checkBoxShift.AutoSize = true;
            this.checkBoxShift.BackgroundColor = System.Drawing.Color.FromArgb(((int)(((byte)(55)))),
((int)(((byte)(54)))), ((int)(((byte)(55)))));
            this.checkBoxShift.BorderColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            this.checkBoxShift.BorderSize = 1;
            this.checkBoxShift.CheckBoxSize = 16;
            this.checkBoxShift.CheckBoxText = "Suspend when shift pressed";
            this.checkBoxShift.Checked = true;
            this.checkBoxShift.CheckedColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            this.checkBoxShift.Location = new System.Drawing.Point(8, 137);
            this.checkBoxShift.Name = "checkBoxShift";
            this.checkBoxShift.Size = new System.Drawing.Size(168, 19);
            this.checkBoxShift.TabIndex = 9;
            this.checkBoxShift.TextColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            //
            // checkBoxCtrl
            //
            this.checkBoxCtrl.AutoSize = true;
            this.checkBoxCtrl.BackgroundColor = System.Drawing.Color.FromArgb(((int)(((byte)(55)))),
((int)(((byte)(54)))), ((int)(((byte)(55)))));
            this.checkBoxCtrl.BorderColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
```

```
            this.checkBoxCtrl.BorderSize = 1;
            this.checkBoxCtrl.CheckBoxSize = 16;
            this.checkBoxCtrl.CheckBoxText = "Suspend when ctrl pressed";
            this.checkBoxCtrl.Checked = true;
            this.checkBoxCtrl.CheckedColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            this.checkBoxCtrl.Location = new System.Drawing.Point(8, 112);
            this.checkBoxCtrl.Name = "checkBoxCtrl";
            this.checkBoxCtrl.Size = new System.Drawing.Size(163, 19);
            this.checkBoxCtrl.TabIndex = 8;
            this.checkBoxCtrl.TextColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            //
            // checkBoxAlt
            //
            this.checkBoxAlt.AutoSize = true;
            this.checkBoxAlt.BackgroundColor = System.Drawing.Color.FromArgb(((int)(((byte)(55)))),
((int)(((byte)(54)))), ((int)(((byte)(55)))));
            this.checkBoxAlt.BorderColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))),
((int)(((byte)(0)))), ((int)(((byte)(0)))));
            this.checkBoxAlt.BorderSize = 1;
            this.checkBoxAlt.CheckBoxSize = 16;
            this.checkBoxAlt.CheckBoxText = "Suspend when alt pressed";
            this.checkBoxAlt.Checked = true;
            this.checkBoxAlt.CheckedColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            this.checkBoxAlt.Location = new System.Drawing.Point(8, 87);
            this.checkBoxAlt.Name = "checkBoxAlt";
            this.checkBoxAlt.Size = new System.Drawing.Size(160, 19);
            this.checkBoxAlt.TabIndex = 7;
            this.checkBoxAlt.TextColor = System.Drawing.Color.FromArgb(((int)(((byte)(240)))),
((int)(((byte)(240)))), ((int)(((byte)(240)))));
            //
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(291, 256);
            this.ControlBox = false;
            this.Controls.Add(this.tableLayoutPanel);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            this.Icon = Updater.Properties.Resources.Icon;
            this.Name = "MainForm";
            this.Text = "Updater";
            this.Load += new System.EventHandler(this.FormLoad);
            ((System.ComponentModel.ISupportInitialize)(this.numericClicksInterval)).EndInit();
            this.tableLayoutPanel.ResumeLayout(false);
            this.panelWindowControl.ResumeLayout(false);
            this.panelWindowControl.PerformLayout();
            this.panelMain.ResumeLayout(false);
            this.panelMain.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion
```

```
#region MyCode

private Point MouseHook;
private Pen Pen_ShiteSmoke = new Pen(Color.WhiteSmoke, 1);


void FormLoad(object sender, EventArgs e)
{
   this.DrawClose(false);
   this.DrawHide(false);
}

void MoveForm(object sender, MouseEventArgs e)
{
   if (e.Button != MouseButtons.Left) MouseHook = e.Location;
   Location = new Point(Location.X - MouseHook.X + e.Location.X, Location.Y - MouseHook.Y +
e.Location.Y);
}

void CloseForm(object sender, MouseEventArgs e)
{
   if (e.Button == MouseButtons.Left)
   {
      Close();
   }
}
void Close_Enter(object sender, EventArgs e) { DrawClose(true); }
void Close_Leave(object sender, EventArgs e) { DrawClose(false); }

void HideForm(object sender, MouseEventArgs e)
{
   if (e.Button == MouseButtons.Left)
   {
      WindowState = FormWindowState.Minimized;
   }
}
void Hide_Enter(object sender, EventArgs e) { DrawHide(true); }
void Hide_Leave(object sender, EventArgs e) { DrawHide(false); }

void DrawHide(bool mouseInRange)
{
   Bitmap bitmap = new Bitmap(panelHideWindow.Width, panelHideWindow.Height);
   Graphics graphics = Graphics.FromImage(bitmap);
   graphics.FillRegion(new SolidBrush(mouseInRange ? Color.FromArgb(80, 80, 86) :
Color.FromArgb(35, 35, 35)), new Region(new Rectangle(0, 0, bitmap.Width, bitmap.Height)));
   Cursor.Current = mouseInRange ? Cursors.Hand : Cursors.Default;
   graphics.DrawLine(Pen_ShiteSmoke, new Point((panelCloseApp.Width / 2) - 5,
(panelCloseApp.Height / 2) + 1), new Point((panelCloseApp.Width / 2) + 5, (panelCloseApp.Height / 2) +
1));
   panelHideWindow.BackgroundImage = bitmap;
}
void DrawClose(bool mouseInRange)
{
```

```
            Bitmap bitmap = new Bitmap(panelCloseApp.Width, panelCloseApp.Height);
            Graphics graphics = Graphics.FromImage(bitmap);
            graphics.FillRegion(new SolidBrush(mouseInRange ? Color.FromArgb(200, 50, 50) :
Color.FromArgb(35, 35, 35)), new Region(new Rectangle(0, 0, bitmap.Width, bitmap.Height)));
            Cursor.Current = mouseInRange ? Cursors.Hand : Cursors.Default;
            graphics.DrawLine(Pen_ShiteSmoke, new Point((panelCloseApp.Width / 2) - 6,
(panelCloseApp.Height / 2) - 5), new Point((panelCloseApp.Width / 2) + 4, (panelCloseApp.Height / 2) +
5));
            graphics.DrawLine(Pen_ShiteSmoke, new Point((panelCloseApp.Width / 2) + 4,
(panelCloseApp.Height / 2) - 5), new Point((panelCloseApp.Width / 2) - 6, (panelCloseApp.Height / 2) +
5));
            panelCloseApp.BackgroundImage = bitmap;
        }

        #endregion

        private System.Windows.Forms.Button buttonInstall;
        private System.Windows.Forms.Button buttonUninstall;
        private System.Windows.Forms.Button buttonStart;
        private System.Windows.Forms.NumericUpDown numericClicksInterval;
        private System.Windows.Forms.Label labelInterval;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel;
        private System.Windows.Forms.Panel panelWindowControl;
        private System.Windows.Forms.Panel panelHideWindow;
        private System.Windows.Forms.Panel panelCloseApp;
        private System.Windows.Forms.Label labelWindowName;
        public System.Windows.Forms.Panel panelMain;
        private UserControl_CheckBox checkBoxAlt;
        private UserControl_CheckBox checkBoxCtrl;
        private UserControl_CheckBox checkBoxShift;
        private Label labelClicksDoneText;
        private Label labelClicksDoneCount;
    }
}
```

DriverControl.cs

```
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using Microsoft.Win32;
using Microsoft.Win32.SafeHandles;
using static Updater.NativeMethods;
using static Updater.DataTypes;
using System.Threading.Tasks;

namespace Updater
{
    public class DriverControl
    {
```

```
    const string KEYBOARD_FILTER_KEY = @"System\CurrentControlSet\Control\Class\{4D36E96B-
E325-11CE-BFC1-08002BE10318}";
    const string KeyboardDriverFileName = "kbedriver.sys";
    const string KeyboardServiceName = "kbedriver";
    const string kbdclass = "kbdclass";
    const string deviceName = @"\\.\kbedriver";

    static uint ControlCode(uint DeviceType, uint Function, uint Method, uint Access) =>
      (DeviceType << 16) | (Access << 14) | (Function << 2) | Method;

    private static readonly uint IOCTL_KEYBOARD_INSERT_KEY =
ControlCode((uint)DeviceType.FileDeviceKeyboard, 0x801, (uint)MemoryPassMode.MethodInDirect,
(uint)FileAccessMode.FileWriteAccess);
    private static readonly uint IOCTL_KEYBOARD_SET_DEVICE_ID =
ControlCode((uint)DeviceType.FileDeviceKeyboard, 0x802, (uint)MemoryPassMode.MethodBufferred,
(uint)FileAccessMode.FileWriteAccess);
    private static readonly uint IOCTL_KEYBOARD_DETECT_DEVICE_ID =
ControlCode((uint)DeviceType.FileDeviceKeyboard, 0x803, (uint)MemoryPassMode.MethodBufferred,
(uint)FileAccessMode.FileReadAccess);

    public SafeFileHandle _driverHandle;
    public bool driverInitializated { get; private set; }

    public DriverControl()  { }

    public async void SetupDriver(MainForm mainForm)
    {
      _driverHandle = CreateFile(deviceName, FileAccessMask.GenericRead |
FileAccessMask.GenericWrite, FileShareMode.Read,
        IntPtr.Zero, CreationDisposition.OpenExisting, CreateFileFlags.None, IntPtr.Zero);
      if (_driverHandle.IsInvalid)
        throw new Win32Exception(Marshal.GetLastWin32Error());

      ushort deviceId = 0;
      using (var panelMessageBox = new PanelMessageBox("Click any keyboard key to detect your
keyboard.", mainForm.panelMain))
        deviceId = await KeyboardDetectDeviceId();

      KeyboardSetActiveDevice(deviceId);
      driverInitializated = true;
    }

    public void SendKey(KeyData inputKey)
    {
      var inputData = new KEYBOARD_INPUT_DATA[] {
        new KEYBOARD_INPUT_DATA
        {
          UnitId = 0,
          MakeCode = (ushort)inputKey.keyCode,
          Flags = (KeyboardKeyState)inputKey.keyState,
        }
      };
      KeyboardInsertKeys(inputData);
    }

    public void SendKey(KeyData[] inputKeys)
```

```
        {
            if (inputKeys.Length == 0)
                return;
            KEYBOARD_INPUT_DATA[] inputData = new KEYBOARD_INPUT_DATA[inputKeys.Length];
            for (int i = 0;  i < inputKeys.Length; i++)
            {
                inputData[i] = new KEYBOARD_INPUT_DATA
                {
                    UnitId = 0,
                    MakeCode = (ushort)inputKeys[i].keyCode,
                    Flags = (KeyboardKeyState)inputKeys[i].keyState,
                };
            }
            KeyboardInsertKeys(inputData);
        }

        private void KeyboardInsertKeys(KEYBOARD_INPUT_DATA[] inputKeys)
        {
            if (inputKeys == null)
                throw new ArgumentNullException(nameof(inputKeys));

            if (inputKeys.Length == 0)
                return;
            if (!DeviceIoControl(_driverHandle, IOCTL_KEYBOARD_INSERT_KEY, inputKeys,
(uint)(inputKeys.Length * Marshal.SizeOf(typeof(KEYBOARD_INPUT_DATA))), IntPtr.Zero, 0, out uint
bytesReturned, IntPtr.Zero))
            {
                throw new Win32Exception(Marshal.GetLastWin32Error());
            }
        }

        public bool DriverInstalled()
        {
            return File.Exists(GetDriverInstallPath());
        }

        private  string GetInstallerDirectory()
        {
            return AppDomain.CurrentDomain.BaseDirectory;
        }

        public void KeyboardSetActiveDevice(ushort deviceId)
        {
            if (!DeviceIoControl(_driverHandle, IOCTL_KEYBOARD_SET_DEVICE_ID, ref deviceId,
sizeof(ushort), IntPtr.Zero, 0, out uint bytesReturned, IntPtr.Zero))
            {
                throw new Win32Exception(Marshal.GetLastWin32Error());
            }
        }

        public async Task<ushort> KeyboardDetectDeviceId()
        {
            ushort deviceId = 0;
            await Task.Run(() => {
                if (!DeviceIoControl(_driverHandle, IOCTL_KEYBOARD_DETECT_DEVICE_ID, IntPtr.Zero,
0, out deviceId, sizeof(ushort), out uint bytesReturned, IntPtr.Zero))
```

```
            {
                throw new Win32Exception(Marshal.GetLastWin32Error());
            }
        });

        return deviceId;
    }


    public string GetDriverInstallPath()
    {
        return Path.Combine(Environment.SystemDirectory, "drivers", KeyboardDriverFileName);
    }

    public string InstallKeyboardDriver()
    {
        try
        {

            string filePath = Environment.Is64BitOperatingSystem ? Path.Combine(GetInstallerDirectory(),
"Sys64", KeyboardDriverFileName)
                : Path.Combine(GetInstallerDirectory(), "Sys86", KeyboardDriverFileName);
            FileInfo file = new FileInfo(filePath);
            if (!file.Exists && !DriverInstalled())
                return $"'{filePath}' not found. Installing keyboard driver failed.";
            string destPath = GetDriverInstallPath();

            file.CopyTo(destPath, true);

            if (!File.Exists(destPath))
                return $"Could not copy driver to system directory. Installing keyboard driver failed. Copy
{KeyboardDriverFileName} in {destPath} and try again.";
            if (!IsTestModeEnabled())
            {
                EnableDriverTestMode();
                return "Reboot your PC";
            }
            if (StartService(KeyboardServiceName, destPath))
            {
                SetRegistryKeyValue(KEYBOARD_FILTER_KEY, kbdclass, KeyboardServiceName);
                return "Successfully installed  keyboard driver.";
            }
            return "Could not create service. Installing keyboard driver failed.";
        }
        catch (Exception ex)
        {
            return $"Installing keyboard driver failed. Error message: {ex.Message}";
        }
    }

    public bool UninstallKeyboardDriver()
    {

        string destPath = GetDriverInstallPath();
        try
        {
```

```
            if (File.Exists(destPath))
                File.Delete(destPath);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        DeleteService(KeyboardServiceName);

        RemoveRegistryKeyValue(KEYBOARD_FILTER_KEY, kbdclass, KeyboardServiceName);

        return true;
    }

    bool StartService(string serviceName, string servicePath)
    {
        using (Process p = new Process())
        {
            p.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            p.StartInfo.CreateNoWindow = true;
            p.StartInfo.FileName = "sc.exe";
            p.StartInfo.UseShellExecute = false;
            p.StartInfo.RedirectStandardOutput = true;
            p.StartInfo.RedirectStandardError = true;
            p.StartInfo.Verb = "runas";
            p.StartInfo.Arguments = $"create {serviceName} type= kernel binPath= {servicePath}
DisplayName= {serviceName}";
            try
            {
                if (p.Start())
                {
                    while (!p.StandardOutput.EndOfStream)
                    {
                        string line = p.StandardOutput.ReadLine();
                        Console.WriteLine(line);
                    }
                    while (!p.StandardError.EndOfStream)
                    {
                        string line = p.StandardOutput.ReadLine();
                        Console.WriteLine(line);
                    }
                    if (p.ExitCode == 0 || p.ExitCode == 1073)
                        return true;
                    Console.WriteLine($"sc.exe exited with code {p.ExitCode}");
                    return false;

                }
                return false;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                return false;
            }
        }
    }
```

```
bool DeleteService(string serviceName)
{
    using (Process p = new Process())
    {
        p.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        p.StartInfo.CreateNoWindow = true;
        p.StartInfo.FileName = "sc.exe";
        p.StartInfo.UseShellExecute = false;
        p.StartInfo.RedirectStandardOutput = true;
        p.StartInfo.RedirectStandardError = true;
        p.StartInfo.Verb = "runas";
        p.StartInfo.Arguments = $"delete {serviceName}";
        try
        {
            if (p.Start())
            {
                while (!p.StandardOutput.EndOfStream)
                {
                    string line = p.StandardOutput.ReadLine();
                    Console.WriteLine(line);
                }
                while (!p.StandardError.EndOfStream)
                {
                    string line = p.StandardOutput.ReadLine();
                    Console.WriteLine(line);
                }
                if (p.ExitCode == 0 || p.ExitCode == 1060)
                    return true;//deleted or not found
                Console.WriteLine($"sc.exe ended with code {p.ExitCode}");
                return false;
            }
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
            return false;
        }
    }
}


private void SetRegistryKeyValue(string regKey, string classService, string ourServiceName)
{
    var value = Registry.LocalMachine.OpenSubKey(regKey, true);
    var upperFiltersCurrent = (string[])value.GetValue("UpperFilters");
    upperFiltersCurrent = upperFiltersCurrent.Where(v => v != ourServiceName).ToArray();//make sure
our service name does not exists
    string[] upperFiltersFinal = new string[upperFiltersCurrent.Length + 1];
    int currentIdnex = 0;
    for (int i = 0; i < upperFiltersCurrent.Length; i++)
    {
        if (upperFiltersCurrent[i] == classService)
        {
            upperFiltersFinal[currentIdnex] = ourServiceName;
```

```
                upperFiltersFinal[++currentIdnex] = classService;
            }
            else
                upperFiltersFinal[currentIdnex] = upperFiltersCurrent[i];
            currentIdnex++;
        }
        value.SetValue("UpperFilters", upperFiltersFinal, RegistryValueKind.MultiString);
        value.Close();
    }

    private void RemoveRegistryKeyValue(string regKey, string classService, string ourServiceName)
    {
        var value = Registry.LocalMachine.OpenSubKey(regKey, true);
        var upperFiltersCurrent = (string[])value.GetValue("UpperFilters");
        string[] upperFiltersFinal = upperFiltersCurrent.Where(v => v != ourServiceName).ToArray();

        value.SetValue("UpperFilters", upperFiltersFinal, RegistryValueKind.MultiString);
        value.Close();
    }

    public void EnableDriverTestMode()
    {

        var proc = new ProcessStartInfo()
        {
            Verb = "runas",
            UseShellExecute = true,
            WorkingDirectory = @"C:\Windows\System32",
            FileName = @"C:\Windows\System32\cmd.exe",
            Arguments = "/c bcdedit / set testsigning on",
            WindowStyle = ProcessWindowStyle.Hidden,
        };

        Process.Start(proc);
    }

    public bool IsTestModeEnabled()
    {
        var systemStartOptions =
(string)Registry.GetValue(@"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control",
"SystemStartOptions", "");
        return systemStartOptions.Contains("TESTSIGNING");
    }
  }
}
```

PanelMessageBox.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace Updater
{
  public class PanelMessageBox : Panel
  {
    private Label label;
    public PanelMessageBox(string message, Control parent)
    {
      this.BackColor = SystemColors.Control;
      this.BorderStyle = BorderStyle.FixedSingle;
      this.Size = new Size(300, 150);
      this.Dock = DockStyle.Fill;

      label = new Label();
      label.Text = message;
      label.Font = new Font("Arial", 18, FontStyle.Bold);
      label.AutoSize = false;
      label.Dock = DockStyle.Fill;
      label.TextAlign = ContentAlignment.MiddleCenter;
      this.Controls.Add(label);

      parent.Controls.Add(this);
      parent.Controls.SetChildIndex(this, 0);
    }
  }
}
```

Core.cs

```
using System;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Updater.DataTypes;
using static Updater.NativeMethods;

namespace Updater
{
  public class Core
  {
    public static KeyColor[] keyColors = new KeyColor[]
    {
      new KeyColor(0x00000001, KeyCode.Tilde),
      new KeyColor(0x00010001, KeyCode.Key1),
      new KeyColor(0x00020001, KeyCode.Key2),
      new KeyColor(0x00030001, KeyCode.Key3),
      new KeyColor(0x00000101, KeyCode.Key4),
      new KeyColor(0x00010101, KeyCode.Key5),
      new KeyColor(0x00020101, KeyCode.Key6),
      new KeyColor(0x00030101, KeyCode.Key7),
      new KeyColor(0x00000201, KeyCode.Key8),
      new KeyColor(0x00010201, KeyCode.Key9),
      new KeyColor(0x00020201, KeyCode.Key0),
      new KeyColor(0x00030201, KeyCode.Key10),
      new KeyColor(0x00000301, KeyCode.Key11),
      new KeyColor(0x00010301, KeyCode.Q),
      new KeyColor(0x00020301, KeyCode.W),
```

```
    new KeyColor(0x00030301, KeyCode.E),
    new KeyColor(0x00000002, KeyCode.R),
    new KeyColor(0x00010002, KeyCode.T),
    new KeyColor(0x00020002, KeyCode.Y),
    new KeyColor(0x00030002, KeyCode.U),
    new KeyColor(0x00000102, KeyCode.I),
    new KeyColor(0x00010102, KeyCode.O),
    new KeyColor(0x00020102, KeyCode.P),
    new KeyColor(0x00030102, KeyCode.A),
    new KeyColor(0x00000202, KeyCode.S),
    new KeyColor(0x00010202, KeyCode.D),
    new KeyColor(0x00020202, KeyCode.F),
    new KeyColor(0x00030202, KeyCode.G),
    new KeyColor(0x00000302, KeyCode.H),
    new KeyColor(0x00010302, KeyCode.J),
    new KeyColor(0x00020302, KeyCode.K),
    new KeyColor(0x00030303, KeyCode.L),
    new KeyColor(0x00000003, KeyCode.Z),
    new KeyColor(0x00010003, KeyCode.X),
    new KeyColor(0x00020003, KeyCode.C),
    new KeyColor(0x00030003, KeyCode.V),
    new KeyColor(0x00000103, KeyCode.B),
    new KeyColor(0x00010103, KeyCode.N),
    new KeyColor(0x00020103, KeyCode.M),
    new KeyColor(0x00030103, KeyCode.Tab),
    new KeyColor(0x00000203, KeyCode.F1),
    new KeyColor(0x00010203, KeyCode.F2),
    new KeyColor(0x00020203, KeyCode.F3),
    new KeyColor(0x00030203, KeyCode.F4),
    new KeyColor(0x00000303, KeyCode.F5),
    new KeyColor(0x00010303, KeyCode.F6),
    new KeyColor(0x00020303, KeyCode.F7),
    new KeyColor(0x00030303, KeyCode.F8)
};

public int _interval = 100;
public int sleepTimeUpperValue = 72;
public int sleepTimeLowerValue = 34;


public event EventHandler ClicksCountChanged;
public ulong clicksCount = 0;
public ulong ClicksCount
{
    get { return clicksCount; }
    set {
        clicksCount = value;
        ClicksCountChanged?.Invoke(ClicksCount, new EventArgs());
    }
}

public bool altSuspend = true;
public bool ctrlSuspend = true;
public bool shiftSuspend = true;

private DriverControl driver;
private Random random;
private System.Windows.Forms.Timer timer;
```

```
        public int Interval {
            get { return _interval; }
            set {
                _interval = value;
                timer.Interval = value;
            }
        }

        public Core(DriverControl driver)
        {
            this.driver = driver;
            random = new Random();
            timer = new System.Windows.Forms.Timer();
            timer.Interval = Interval;
            timer.Tick += (arg1, arg2) => { Next(); };
        }


        public void Start()
        {
            timer.Start();
        }

        public void Stop()
        {
            timer.Stop();
        }
        private KeyCode GetKeyToSend()
        {
            int pixelColorLeft = Pixel.Get(0, 1439);
            int pixelColorRight = Pixel.Get(1, 1439);

            if (pixelColorRight != 0x00010101)
                return KeyCode.None;

            foreach(var _keyColor in keyColors)
            {
                if (_keyColor.color == pixelColorLeft)
                    return _keyColor.key;
            }

            return KeyCode.None;
        }

        private void Next()
        {
            if ((shiftSuspend && (GetAsyncKeyState(Keys.ShiftKey) & 0x8000) != 0) || (altSuspend &&
(GetAsyncKeyState(Keys.Menu) & 0x8000) != 0) || (ctrlSuspend && (GetAsyncKeyState(Keys.ControlKey) & 0x8000) !=
0))
                return;

            KeyCode key = GetKeyToSend();
            if (key == KeyCode.None)
                return;

            Click(driver, key, random.Next(sleepTimeLowerValue, sleepTimeUpperValue));
```

```
            ClicksCount++;
        }

        public static async void Click(DriverControl driver, KeyCode key, int sleepTime)
        {
            await Task.Run(() =>
            {
                driver.SendKey(new KeyData()
                {
                    keyCode = key,
                    keyState = KeyState.KEY_DOWN,
                });
                Thread.Sleep(sleepTime);
                driver.SendKey(new KeyData()
                {
                    keyCode = key,
                    keyState = KeyState.KEY_UP,
                });
            });
        }

    }


    public class KeyColor
    {
        public int color;
        public KeyCode key;

        public KeyColor() { }
        public KeyColor(int color, KeyCode key)
        {
            this.color = color;
            this.key = key;
        }
    }

    public class Pixel
    {
        public static int Get(int x, int y)
        {
            IntPtr desk = GetDesktopWindow();
            IntPtr dc = GetWindowDC(desk);
            int a = (int)GetPixel(dc, x, y);
            ReleaseDC(desk, dc);
            return a;
        }

        public static System.Drawing.Color Color(int intColor)
        {
            return System.Drawing.Color.FromArgb(255, (intColor >> 0) & 0xff, (intColor >> 8) & 0xff, (intColor >> 16) &
0xff);
        }
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна_робота_ Брайко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота_ Брайко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Брайко.pptx	Презентація кваліфікаційної роботи