**Форма № Н-6.01у**

Кременчуцький національний університет імені Михайла Остроградського

(повне найменування вищого навчального закладу)

Кафедра автоматизації та інформаційних систем

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

**(РОБОТА)**

з дисципліни «Сучасні мови об’єктно-орієнтованого програмування»

(назва дисципліни)

на тему Розробка WPF додатку «Гра 2048»

Студента \_2\_ курсу КН–22–2 групи

Ступінь вищої освіти «Бакалавр»

(бакалавр, магістр)

Спеціальність 122 – «Комп’ютерні науки»

Освітньо-професійна програма

«Комп’ютерні науки»

Панцюк К В

(прізвище та ініціали)

Керівник старший викладач кафедри АІС

\_\_\_\_\_\_\_ Бельська В. Ю.\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_. Оцінка: ЄКTС \_\_\_\_

Члени комісії  . В. Ю. Бельська

(підпис) (ініціали та прізвище)

І. Г. Оксанич

(підпис) (ініціали та прізвище)

В. В. Найда

(підпис) (ініціали та прізвище)

м. Кременчук 2023 рік

**Форма № Н-9.01у**

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра автоматизації та інформаційних систем

Дисципліна «Сучасні мови об’єктно-орієнтованого програмувння»

Освітній ступінь \_\_ «Бакалавр»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність 122 – «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітня програма «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс 2 група \_\_КН-22-2 семестр \_\_3\_\_

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Панцюк Кирило Віталійович \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема роботи: Розробка WPF додатку «Гра 2048»
2. Термін здачі студентом роботи 1 грудня 2023 р
3. Вихідні дані до роботи: save.dat (містить серіалізовані дані на момент останнього збереження) , statistics.dat (зберігає статистику гри ), statisticsUser.dat (зберігає статистику гри з іменем користувача та кількість набраних очок протягом гри)
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

постановка завдання, аналіз технічного завдання, розробка алгоритму роботи програми, опис моделі даних, структура програмного забезпечення, функціональна схема та інтерфейс програми.

1. Перелік графічного матеріалу:
2. Дата видачі завдання: 1 жовтня 2023 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  пор. | Назва етапів курсового проекту | Терміни виконання етапів проекту | Вказівки та  зауваження викладача (з зазначенням дати консультації) | Оцінювання етапів проекту | | |
| за націо- нальною шкалою | за шкалою ЄКТС | кількість балів |
| 1 | Етап 1  Аналіз предметної області | 01.10.23–  25.10.23 |  |  | | |
| 2 | Етап 2  Створення моделі даних | 16.10.23–  28.10.23 |  |  | | |
| 3 | Етап 3  Розробка віконного інтерфейсу та створення основного програмного коду додатку | 29.10.23–  24.11.23 |  |  | | |
| 4 | Етап 4  Тестування програмного коду | 16.10.23  29.11.23 |  |  | | |
| 5 | Етап 5  Оформлення пояснювальної записки | 25.11.23–  31.11 .23 |  |  | | |
| 6 | Етап 9  Захист | 01.12.23 |  |  | | |
|  | Разом | 8 тижнів |  |  | | |

Студент

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ В. Ю. Бельська \_\_\_\_

(підпис) (ініціали та прізвище)

«1» жовтня 2023 р.

РЕФЕРАТ

Курсова робота містить 43 сторінок, 2 розділи, 16 рисунків, 1 таблиця, 5 використаних джерел.

Об’єкт розробки – WPF додаток «2048».

Мета: створення багатовіконного Wpf додатку, що реалізує гру «2048».

Під час виконання завдання, поставленого на курсову роботу, було виконано аналіз алгоритму гри «2048». Визначено основні функціональні та не функціональні вимоги до додатку, що створюється. Побудована модель даних, що дозволить вирішити наступні задачі: розробка алгоритму занесення користувача в таблицю статистики ,підрахунку найкращої та поточної суми очок , розробка алгоритму можливості продовжити гру тощо.

В якості мови програмування для створення Wpf додатку використовувалася мова с# та середовище програмування Visual Studio 22

Результатом виконання всіх етапів є застосунок, який може заносити користуваців до таблиці статистики, проводити гру, надає можливість продовжити гру , переглянути таблицю статистики .

ОБ’ЄКТ, ІНТЕРФЕЙС, КЛАС,ООП

**Зміст**

[Вступ 2](#_Toc151681430)

[1 Опис предметної областІ 3](#_Toc151681431)

[1.1 У кожного студента буде індивідуальна тема для висвітлення 3](#_Toc151681432)

[1.2 Аналіз технічного завдання на роботу 5](#_Toc151681433)

[1.2.1 Функціональні вимоги 5](#_Toc151681434)

[1.2.2 Нефункціональні вимоги 5](#_Toc151681435)

[1.3 Опис алгоритму основних задач/підзадач у роботі 6](#_Toc151681436)

[1.3.1 Структура програмного забезпечення 7](#_Toc151681437)

[Висновки до розділу 7](#_Toc151681438)

[2 Опис розробки програмного забезпечення 8](#_Toc151681439)

[2.1 Структура програмного забезпечення 8](#_Toc151681440)

[2.2 Опис роботи програми 11](#_Toc151681441)

[2.3 Функціональна схема програми 12](#_Toc151681442)

[2.4 Опис інтерфейсу програми 13](#_Toc151681443)

[Висновки до розділу 16](#_Toc151681444)

[ВИСНОВКИ 17](#_Toc151681445)

[СПИСОК ЛІТЕРАТУРИ 18](#_Toc151681446)

[Додаток А XAML розмітки вікон 19](#_Toc151681447)

[Додаток Б C# коди класів вікон 28](#_Toc151681448)

[Додаток B C# основні класи 33](#_Toc151681449)

[Додаток Г Тестування додатку 38](#_Toc151681450)

ВСТУП

Технологія WPF (Windows Presentation Foundation) є складовою частиною платформи .NET і надає потужну систему для створення графічних інтерфейсів.

WPF широко використовується для розробки програмного забезпечення з графічним інтерфейсом, зокрема в мові програмування C#.

У порівнянні з традиційним WinForms, де візуалізацію елементів керування обробляли User32 та GDI+, WPF базується на DirectX. Головною особливістю рендерингу графіки у WPF є використання графічного процесора на відеокарті для візуалізації як кнопок, так і складних 3D-моделей, що дозволяє використовувати апаратне прискорення графіки.

WPF використовує декларативну розмітку XAML, що ґрунтується на XML. Це дозволяє створювати гарні графічні інтерфейси через декларативне оголошення інтерфейсу або код на керуючих мовах, таких як C#.

Платформа WPF надає різноманітні можливості для створення різних додатків, включаючи анімації, binding, стилі, шаблони та широкий вибір елементів.

Тому WPF є відмінним вибором для створення гри "2048", де використання анімацій, ресурсів, стилів та біндингу дозволяє створити привабливий та зрозумілий інтерфейс. І використання вбудованих класів та інтерфейсів спрощує реалізацію логіки гри.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 У кожного студента буде індивідуальна тема для висвітлення

Спадкування в .NET - потужний механізм повторного використання коду та розширення функціональності класів. Розглянемо його роботу більш детально.

**Як працює спадкування?**

Уявімо, є базовий клас "Фігура". Він описує спільні властивості для всіх фігур - координати та метод переміщення:

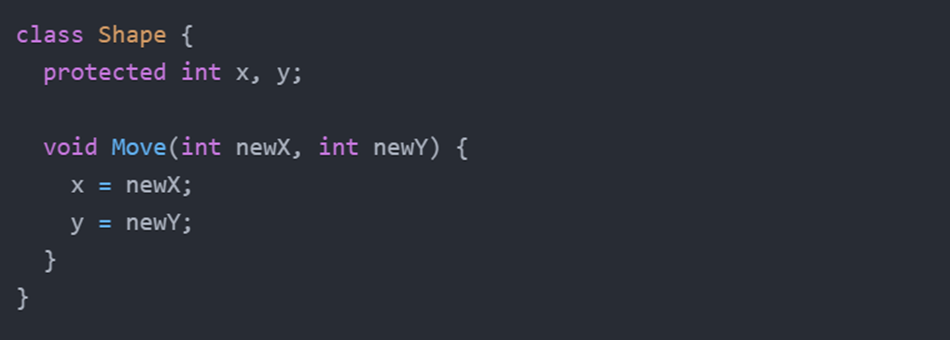


Рисунок 1.1.1– Приклад створення базового класу «Фігура»

Тепер створимо клас "Коло", який успадковує від "Фігура":

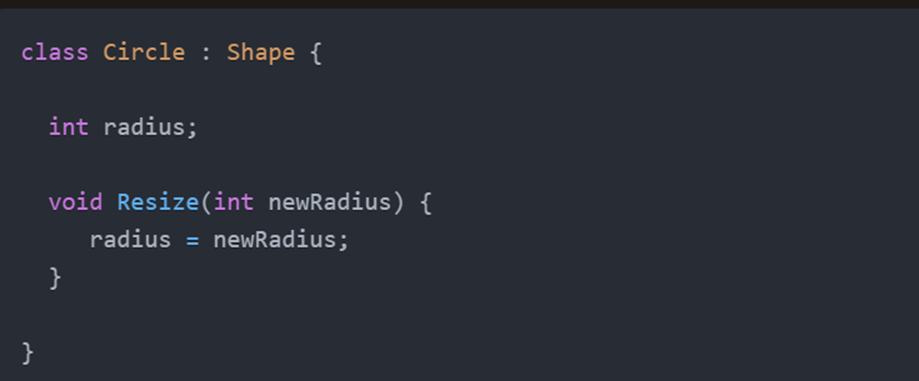


Рисунок 1.1.2– Приклад створення класу «Коло»

Клас Коло автоматично отримує поля x, y та метод Move() з базового класу Фігура. Також він може додати щось своє, як поле radius та метод Resize().

Тобто при спадкуванні похідний клас базується на базовому класі, розширюючи та деталізуючи його.

**Абстрактний клас та інтерфейс**

Абстрактний клас визначає базову поведінку. Інтерфейс – лише можливості

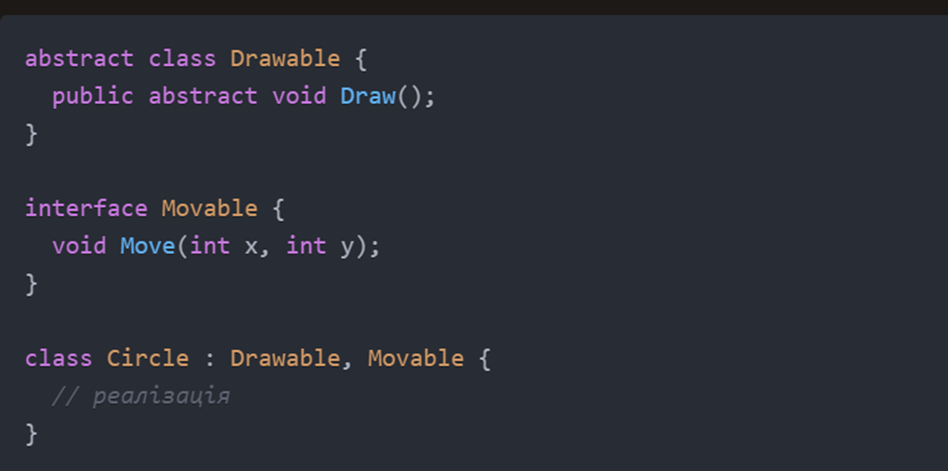


Рисунок 1.1.3 – Приклад успадкування класом «Коло» метода Draw() та реалізацією інтерфейсу Movable

Отже, у результаті розгляду принципів спадкування та відмінностей абстрактних класів і інтерфейсів в .NET, можна зробити наступні висновки:

Спадкування дозволяє будувати ієрархії класів, коли похідний клас базується на базовому класі, успадковуючи та розширюючи його функціональність.

Абстрактний клас визначає базову поведінку, яку по-своєму реалізовують похідні класи. Він може містити абстрактні члени.

Інтерфейс вказує лише можливості, які клас має реалізувати. Інтерфейс містить тільки сигнатури.

Головна відмінність в тому, що абстрактний клас дає базову реалізацію і бере участь у спадкуванні. А інтерфейс лише вказує "договір", якого клас має дотримуватися.

Отже, спадкування, абстрактні типи та інтерфейси - це базові та вкрай важливі механізми ООП в .NET для побудови гнучких ієрархій класів та структур даних.

1.2 Аналіз технічного завдання на роботу

Мета курсового проєкту – Створення багатовіконного WPF додатку що реалізує гру «2048».

Під час розробки моделей даних та графічного інтерфейсу були висунуті наступні вимоги:

**1.2.1 Функціональні вимоги**

1. Можливість грати в гру.
2. Можливість занесення результатів гравця в таблицю результатів
3. Можливість перегляду статистики користувачів.
4. Можливість збереження даних користувачів
5. Можливість реєстрації користувача за нікнеймом з збереженням статистики гри

**1.2.2 Нефункціональні вимоги**

1. Зовнішній інтерфейс користувача має бути реалізованим за допомогою створення вікон в середовищі WPF додатку.
2. Додаток має бути багатовіконним(якщо інший варіант, то вказуєте на це).
3. Використання патерну MVVM тa прив’язки даних .
4. Усі дані повинні зберігатися у бінарному форматі.

1.3 Опис алгоритму основних задач/підзадач у роботі

Розробка алгоритмів функціонування для гри "2048" передбачає створення складної моделі роботи програми як єдиної системи. На даному етапі необхідно врахувати потенційні обмеження, які можуть виникнути в майбутньому та завадити коректному розширенню функціоналу гри.

Ігрове поле являє собою квадратну сітку розміром 4х4 клітинки. На початку гри 2 випадкові клітинки містять числа 2 або 4.

Гравець може керувати полем, натискаючи клавіші вгору, вниз, ліворуч або праворуч. При цьому всі числа зсуваються відповідним чином.

Якщо дві однакові клітинки зіштовхуються під час руху, вони об'єднуються в одну клітинку з числом, що дорівнює сумі цих двох чисел.

Кожен хід, після зсуву і об'єднання клітинок, у випадково обраній пустій клітинці з'являється нове число 2 або 4.

Мета гри - поєднати якомога більше однакових клітинок, щоб отримати велике число 2048.

Гра закінчується програшем, якщо на полі немає жодних можливих ходів (всі клітинки заповнені і немає однакових для об'єднання).

Очки рахуються як сума всіх чисел на полі. Чим більше очок, тим краще.

Результат гри (очки) можна занести до таблиці рекордів для порівняння з іншими гравцями.

**1.3.1 Структура програмного забезпечення**

Висновки до розділу

У першому розділі висвітлено основні принципи об'єктно-орієнтованого програмування (ООП), визначена розширена постановка задачі, проведений аналіз предметної області та сформульовані вимоги до функціональних можливостей програми. Розроблено план розробки програми та описано функціональності, які мають бути втілені в програмі. Також розглянуто технічне завдання та розроблений алгоритм роботи програмного продукту.

2 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Структура програмного забезпечення

Структура програмного забезпечення включає в себе комплекс усіх використаних модулів програми, які утворюють працездатну систему. Кожен модуль має реалізовувати функції програми, спрямовані на вирішення основних завдань проєкту. Застосунок розроблено з використанням модуля Windows Presentation Foundation (WPF) на основі мови програмування C#.

На рис. 2.1 зображено оглядач рішень програмного застосунку. В табл.2.1 представлено опис модулів(класів) проекту курсової

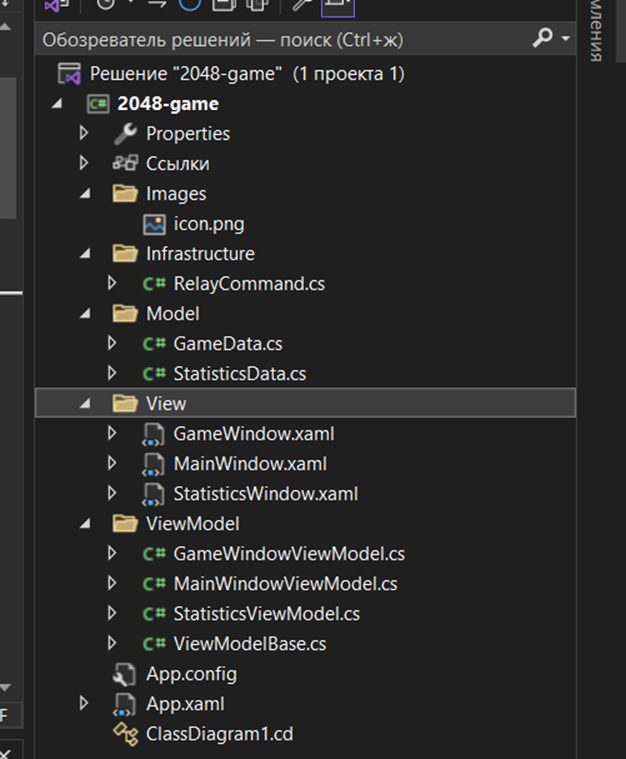
****

Рисунок 2.1 – Оглядач рішень

Таблиця 2.1 – Призначення модулів програми

|  |  |
| --- | --- |
| **Назва модуля** | **Призначення модуля** |
| MainWindow.xaml.cs | Клас, головного вікна програми |
| MainWindow.xaml | Головне вікно, що містить меню гри |
| GameWindow.xaml.cs | Клас вікна гри |
| GameWindow.xaml | Розмітка вікна гри |
| StatisticsWindow.xaml.cs | Клас вікна статистики |
| StatisticsWindow.xaml | Вікно статистики що містить таблицю з полями Нік та Кількість очок |
| GameData.cs | Клас містить данні про структуру та стан гри для полальшого використання в програмі |
| StatisticsData.cs | Клас служить для збереження різних гравців у грі та їх досягнення у грі |
| RelayCommand.cs | Клас представляє реалізацію інтерфейсу ICommand для обробки команд в інфраструктурі гри |
| GameWindowViewModel.cs | Клас взаємодіє з інтерфейсом гри, обробляючи вхідні дані, оновлюючи стан гри та взаємодіючи з файловою системою для збереження та відновлення даних гравця |
| MainWindowViewModel.cs | Клас взаємодіє з інтерфейсом головного вікна, надаючи можливість користувачу створювати нову гру або продовжувати попередню. Перевірка наявності файлу збережених даних забезпечує можливість продовження тільки тоді, коли є щось для відновлення. |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| **Назва модуля** | **Призначення модуля** |
| StatisticsViewModel.cs | Цей клас дозволяє інтерфейсу коректно відображати та оновлювати статистику гравця в грі. |
| ViewModelBase.cs | Цей клас надає базовий функціонал, який може бути спільно використаний для всіх відображувальних моделей у грі. |

Під час створення програми було створено 3 вікна. Діаграма класів вікон зображена на рис. 2.2. Всі зображені класи неявно реалізують інтерфейс INotifyPropertyChanged та наслідують клас Window.

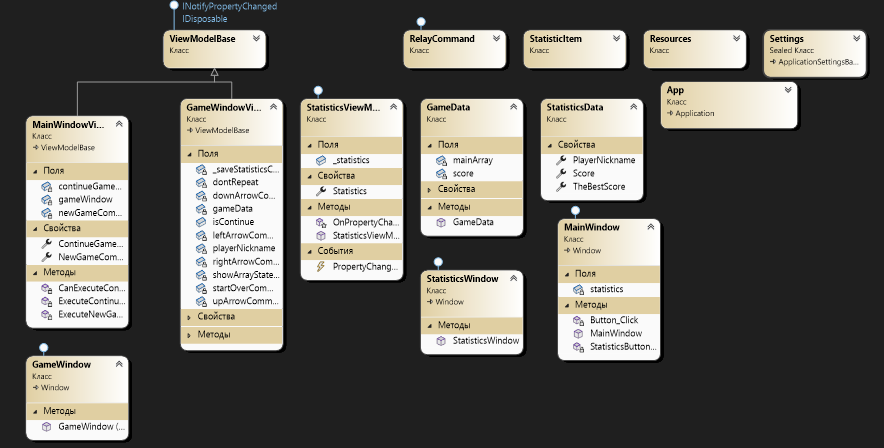


Рисунок 2.2 – Діаграма класів вікон

Файли з XAML розміткою знаходяться у додатку А, C# класи вікон у додатку Б, основні C# класи у додатку В.

2.2 Опис роботи програми

При завантаженні програми з'являється ігрове вікно, на якій можна перейти до гри , вийти з програми, продовжити гру та переглянути статистику . На вікні розміщена кнопка "Нова гра", яка обнуляє рахунок та починає гру спочатку. Рахунок розташований під написом "Рахунок гри " і відображає кількість балів, накопичених під час гри. Управління грою здійснюється за допомогою стрілок на клавіатурі.

Ігрове поле представляє собою квадрат, який складається з 16 клітин. Із кожним ходом випадковим чином у вільній клітці з'являється плитка з числом "2" або "4". При досягненні числа "2048" виникає вікно з повідомленням про перемогу, а у випадку, коли неможливо здійснити наступні кроки, виводиться вікно із повідомленням про поразку.

**2.3 Функціональна схема програми**

Детальний опис й пояснення процесів окремих функцій розробляємої системи утворюють функціональну схему програми(рис. 2.4).

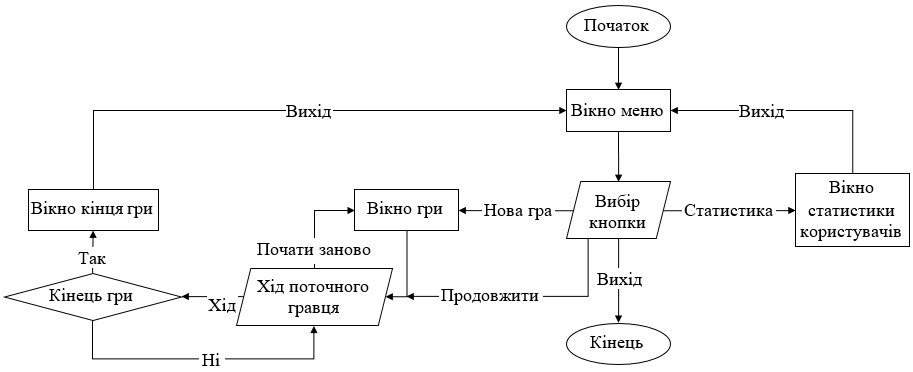


Рисунок 2.4 – Функціональна схема програмного коду

2.4 Опис інтерфейсу програми

При розробці програми створено зручний інтерфейс користувача. Опишемо основні вікна програми:

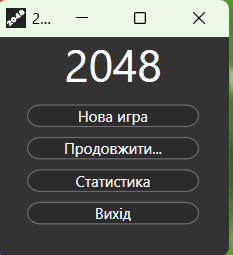


Рисунок 2.5 – Інтерфейс вікна MainWindow

Вікно меню (рисунок 2.5) містить 4 кнопок:

1. При натисканні кнопки "Нова гра", користувач потрапляє у нове ігрове вікно, де починається нова гра.
2. При натисканні кнопки "Продовжити...", користувач потрапляє у вікно зі збереженою раніше грою, яку можна продовжити.
3. При натисканні кнопки "Статистика", користувач потрапляє у вікно зі статистикою своїх ігор та результатами інших гравців.
4. При натисканні кнопки "Вихід", програма закривається.

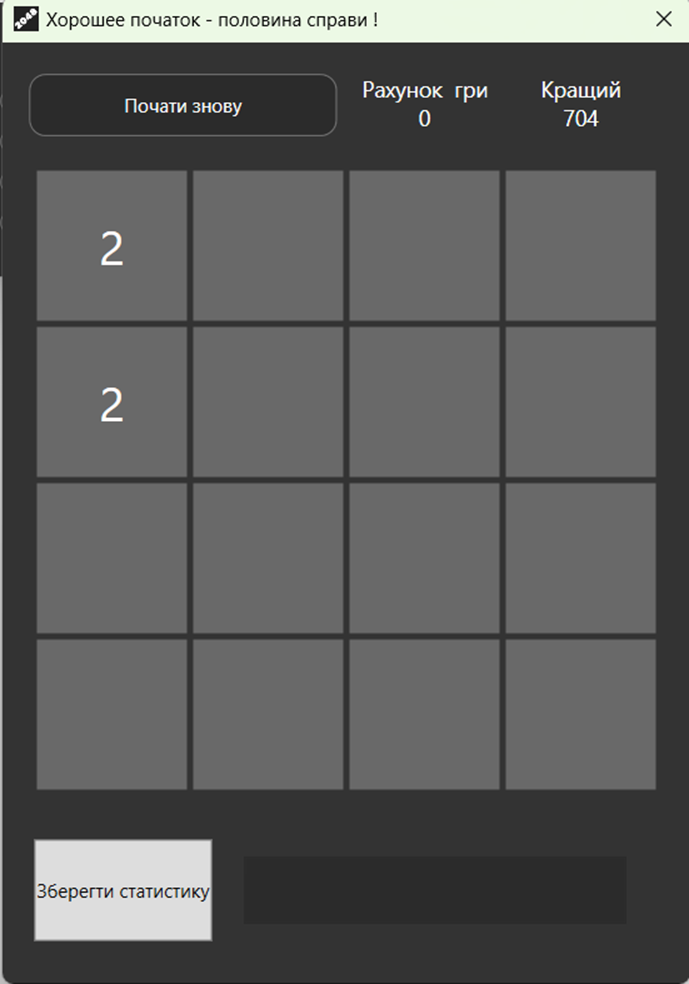


Рисунок 2.6 – Вікно гри

Вікно гри (рисунок 2.6) містить :

1. Вікно містить ігрове поле 4x4, де відображаються числа з масиву гри. Оновлення чисел відбувається при зміні відповідних властивостей GameData в ViewModel.
2. Реалізовано керування стрілками клавіатури для виконання ходів гравцем.
3. Відображаються поточний рахунок гри та кращий рахунок (найбільший за всю історію).
4. Є кнопки почати гру знову та зберегти статистику гри.
5. При натисканні Tab відображається поточний стан масиву гри (для розробників).
6. Перед збереженням статистики користувач вводить свій нікнейм у текстове поле.
7. Закриття вікна, призводить до потрапляння на головне выкно програми

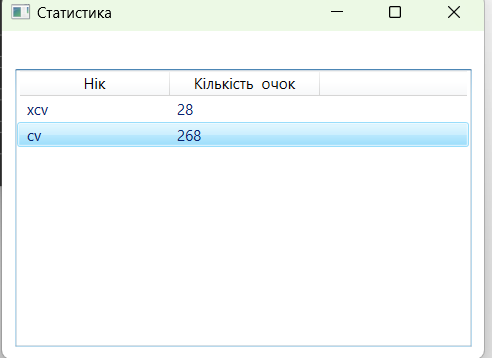


Рисунок 2.7 – Відображення вікна статистики

Висновки до розділу

У другому розділі розглянуті: ієрархія класів, ініціалізація даних, функціональна схема програми та опис основних функціональних вікон програми, їх призначення та властивості.

Крім того, розроблено інтерфейс програми і детально описано керівництво користувача системи.

ВИСНОВКИ

В процесі виконання курсового проекту було виявлено, що обрана тема є надзвичайно актуальною для новачків, оскільки розробка ігор може стати захоплюючим завданням, яке дозволяє поєднати творчість і програмування. Описаний загальний алгоритм розробки гри базується на обробці подій, оновленні стану гри, перевірці на завершення гри, оновленні графічного відображення та повторенні циклу.

Об'єктно-орієнтоване програмування відіграє важливу роль у розробці гри, дозволяючи моделювати об'єкти гри як окремі сутності з власними властивостями та методами. Це сприяє розподілу відповідальностей та функціональності між різними класами, що дозволяє незалежно працювати над різними частинами гри, зберігаючи чистоту коду та зменшуючи його складність.

Під час створення курсової роботи були враховані базові принципи паттерну MVVM, проведений аналіз предметної області. Мова програмування C# та технологія WPF були вибрані для виконання проекту, оскільки вони ідеально підходять для розробки програм з використанням принципів об'єктно-орієнтованого програмування.

Ключовим етапом було планування. Перед початком розробки важливо детально проробити функціональність, структуру та взаємодію різних компонентів гри. Це допомагає уникнути зайвого перекодування та проблем з організацією коду..

СПИСОК ЛІТЕРАТУРИ

1. Інтернет ресурс https://metanit.com/

2. Документація Microsoft - https://docs.microsoft.com/

3. Інтернет ресурс C# Corner - https://www.c-sharpcorner.com/

4. Інтернет ресурс W3Schools - https://www.w3schools.com/

5. https://uk.wikipedia.org/wiki/Windows\_Presentation\_Foundation

Додаток А  
XAML розмітки вікон

**Файл GameWindow.xaml**

<Window x:Class="\_2048\_game.View.GameWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_2048\_game.ViewModel"

mc:Ignorable="d"

Title="GameWindow" SizeToContent="WidthAndHeight" ResizeMode="NoResize"

Icon="/2048-game;component/Images/icon.png" Background="#333333" Height="700" Width="477">

<Window.DataContext>

<local:GameWindowViewModel/>

</Window.DataContext>

<Window.InputBindings>

<KeyBinding Key="Left" Command="{Binding LeftArrowCommand}"/>

<KeyBinding Key="Right" Command="{Binding RightArrowCommand}"/>

<KeyBinding Key="Up" Command="{Binding UpArrowCommand}"/>

<KeyBinding Key="Down" Command="{Binding DownArrowCommand}"/>

<!-- developer commands -->

<KeyBinding Key="Tab" Command="{Binding ShowArrayStateCommand}"/>

</Window.InputBindings>

<Window.Resources>

<Style x:Key="TextBlockRectangleClassicStyle">

<Setter Property="TextBlock.FontSize" Value="30"/>

<Setter Property="TextBlock.Padding" Value="0,33,0,0"/>

<Setter Property="TextBlock.Margin" Value="2,2,2,2"/>

<Setter Property="TextBlock.Background" Value="#696969"/>

<Setter Property="TextBlock.Foreground" Value="#FFFAFA"/>

<Setter Property="TextBlock.FontFamily" Value="Code Pro Bold"/>

</Style>

<Style x:Key="ButtonStyle" TargetType="Button">

<Setter Property="Background" Value="#FF2B2B2B"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Padding" Value="10"/>

<Setter Property="Margin" Value="-3, 20, 6, 20"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="1" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="TextBoxStyle" TargetType="TextBox">

<Setter Property="Background" Value="#FF2B2B2B"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Padding" Value="10"/>

<Setter Property="Margin" Value="10"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="BorderBrush" Value="#FF2B2B2B"/>

</Style>

</Window.Resources>

<Grid Margin="0,0,0,102">

<Grid.RowDefinitions>

<RowDefinition Height="80"/>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

<RowDefinition Height="100"/>

<RowDefinition Height="20"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="20"/>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="20"/>

</Grid.ColumnDefinitions>

<Button Content="Почати знову" Command="{Binding StartOverCommand}" Style="{StaticResource ResourceKey=ButtonStyle}" Grid.Column="1" Grid.ColumnSpan="2"/>

<StackPanel Grid.Column="3">

<TextBlock Text="Рахунок гри " Margin="10, 20, 10, 0" TextAlignment="Center" FontSize="14" Foreground="White"/>

<TextBlock Text="{Binding Path=Score, UpdateSourceTrigger=PropertyChanged}" Margin="10, 0, 10, 10" TextAlignment="Center" FontSize="14" Foreground="White"/>

</StackPanel>

<StackPanel Grid.Column="4">

<TextBlock Text="Кращий" Margin="10, 20, 10, 0" TextAlignment="Center" FontSize="14" Foreground="White"/>

<TextBlock Text="{Binding Path=TheBestScore, UpdateSourceTrigger=PropertyChanged}" Margin="10, 0, 10, 10" TextAlignment="Center" FontSize="14" Foreground="White"/>

</StackPanel>

<TextBlock Text="{Binding Path=GameData.MainArray00, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="1" Grid.Column="1" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}" />

<TextBlock Text="{Binding Path=GameData.MainArray01, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="1" Grid.Column="2" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray02, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="1" Grid.Column="3" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray03, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="1" Grid.Column="4" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray10, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="2" Grid.Column="1" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray11, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="2" Grid.Column="2" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray12, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="2" Grid.Column="3" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray13, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="2" Grid.Column="4" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray20, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="3" Grid.Column="1" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray21, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="3" Grid.Column="2" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray22, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="3" Grid.Column="3" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray23, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="3" Grid.Column="4" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray30, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="4" Grid.Column="1" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray31, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="4" Grid.Column="2" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray32, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="4" Grid.Column="3" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<TextBlock Text="{Binding Path=GameData.MainArray33, UpdateSourceTrigger=PropertyChanged}" TextAlignment="Center" Grid.Row="4" Grid.Column="4" Style="{StaticResource ResourceKey=TextBlockRectangleClassicStyle}"/>

<StackPanel Grid.Row="5" Grid.ColumnSpan="6" Orientation="Horizontal" Margin="10,10,10,-95">

<Button Content="Зберегти статистику" Command="{Binding SaveStatisticsCommand}" Margin="10" Height="66"/>

<TextBox Text="{Binding PlayerNickname}" Style="{StaticResource TextBoxStyle}" Width="245" Height="43"/>

</StackPanel>

</Grid>

</Window>

**Файл MainWindow.xaml**

<Window x:Class="\_2048\_game.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_2048\_game.ViewModel"

mc:Ignorable="d"

Title="2048" SizeToContent="Manual" ResizeMode="CanResize" Icon="/2048-game;component/Images/icon.png" Background="#333333" Height="200" Width="200">

<Window.DataContext>

<local:MainWindowViewModel/>

</Window.DataContext>

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Background" Value="#FF2B2B2B"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Padding" Value="40, 8, 40, 8"/>

<Setter Property="Margin" Value="4"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="1" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="15"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="20"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="20"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="1" Text="2048" VerticalAlignment="Center" HorizontalAlignment="Center" FontFamily="Code Pro Bold" FontSize="36" Foreground="White"/>

<StackPanel Grid.Row="1" Grid.Column="1">

<Button Content="Нова игра" Command="{Binding NewGameCommand}"/>

<Button Content="Продовжити..." Command="{Binding ContinueGameCommand}"/>

<Button Content="Статистика" Click="StatisticsButton\_Click"/>

<Button Content="Вихід" Click="Button\_Click"/>

</StackPanel>

</Grid>

</Window>

**Файл StatisticsWindow.xaml**

<Window x:Class="\_2048\_game.StatisticsWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Статистика" Height="300" Width="400"

DataContext="{Binding RelativeSource={RelativeSource Self}}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal" Margin="10">

</StackPanel>

<ListView Grid.Row="1" Name="StatisticsListView" Margin="10" ItemsSource="{Binding Statistics}">

<ListView.View>

<GridView>

<GridViewColumn Header="Нік" DisplayMemberBinding="{Binding Nickname}" Width="120"/>

<GridViewColumn Header="Кількість очок" DisplayMemberBinding="{Binding Score}" Width="120"/>

</GridView>

</ListView.View>

</ListView>

</Grid>

</Window>

Додаток Б  
C# коди класів вікон

**Клас MainWindow.cs**

using \_2048\_game.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Windows;

using System.Windows.Documents;

namespace \_2048\_game

{

public partial class MainWindow : Window

{

List<StatisticItem> statistics = new List<StatisticItem>();

public MainWindow()

{

InitializeComponent();

BinaryFormatter formatter = new BinaryFormatter();

if (File.Exists("statisticsUsers.dat"))

{

using (FileStream stream = new FileStream("statisticsUsers.dat", FileMode.Open))

{

statistics = (List<StatisticItem>)formatter.Deserialize(stream);

}

}

DataContext = new MainWindowViewModel();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void StatisticsButton\_Click(object sender, RoutedEventArgs e)

{

StatisticsWindow statisticsWindow = new StatisticsWindow();

if (StatisticsViewModel.\_statistics != null)

{

statistics.Add(StatisticsViewModel.\_statistics[0]);

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream stream = new FileStream("statisticsUsers.dat", FileMode.Create))

{

formatter.Serialize(stream, statistics);

}

}

statisticsWindow.StatisticsListView.ItemsSource = statistics;

statisticsWindow.Show();

}

}

}

**Клас GameWindow.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using \_2048\_game.ViewModel;

namespace \_2048\_game.View

{

/// <summary>

/// Логика взаимодействия для GameWindow.xaml

/// </summary>

public partial class GameWindow : Window

{

public GameWindow()

{

InitializeComponent();

Closing += GameWindowViewModel.GameWindow\_Closing;

}

public GameWindow(string title)

: this()

{

Title = title;

}

public GameWindow(string title, bool isContinue = false)

{

GameWindowViewModel.isContinue = true;

InitializeComponent();

Closing += GameWindowViewModel.GameWindow\_Closing;

Title = title;

}

}

}

**Клас** **StatisticsWindow.cs**

using System.Windows;

namespace \_2048\_game

{

public partial class StatisticsWindow : Window

{

public StatisticsWindow()

{

InitializeComponent();

}

}

}

Додаток B  
C# основні класи

**Клас GameData .cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2048\_game.Model

{

[Serializable]

class GameData

{

private int[,] mainArray = new int[4,4];

public int[,] MainArray { get => mainArray; set => mainArray = value; }

private int score = 0;

public int Score { get => score; set => score = value; }

#region Свойства для всіх ячейок масиву

public string MainArray00 { get => (MainArray[0, 0] == 0) ? (" ") : (MainArray[0, 0].ToString()); }

public string MainArray01 { get => (MainArray[0, 1] == 0) ? (" ") : (MainArray[0, 1].ToString()); }

public string MainArray02 { get => (MainArray[0, 2] == 0) ? (" ") : (MainArray[0, 2].ToString()); }

public string MainArray03 { get => (MainArray[0, 3] == 0) ? (" ") : (MainArray[0, 3].ToString()); }

public string MainArray10 { get => (MainArray[1, 0] == 0) ? (" ") : (MainArray[1, 0].ToString()); }

public string MainArray11 { get => (MainArray[1, 1] == 0) ? (" ") : (MainArray[1, 1].ToString()); }

public string MainArray12 { get => (MainArray[1, 2] == 0) ? (" ") : (MainArray[1, 2].ToString()); }

public string MainArray13 { get => (MainArray[1, 3] == 0) ? (" ") : (MainArray[1, 3].ToString()); }

public string MainArray20 { get => (MainArray[2, 0] == 0) ? (" ") : (MainArray[2, 0].ToString()); }

public string MainArray21 { get => (MainArray[2, 1] == 0) ? (" ") : (MainArray[2, 1].ToString()); }

public string MainArray22 { get => (MainArray[2, 2] == 0) ? (" ") : (MainArray[2, 2].ToString()); }

public string MainArray23 { get => (MainArray[2, 3] == 0) ? (" ") : (MainArray[2, 3].ToString()); }

public string MainArray30 { get => (MainArray[3, 0] == 0) ? (" ") : (MainArray[3, 0].ToString()); }

public string MainArray31 { get => (MainArray[3, 1] == 0) ? (" ") : (MainArray[3, 1].ToString()); }

public string MainArray32 { get => (MainArray[3, 2] == 0) ? (" ") : (MainArray[3, 2].ToString()); }

public string MainArray33 { get => (MainArray[3, 3] == 0) ? (" ") : (MainArray[3, 3].ToString()); }

#endregion

public GameData(int[,] mainArray)

{

this.mainArray = mainArray;

}

}

}

**Клас StatisticsData.cs**

using System;

[Serializable] // Цей атрибут дозволяє сериализувати объект цього класса

public class StatisticsData

{

public string PlayerNickname { get; set; }

public int Score { get; set; }

public int TheBestScore { get; set; }

}

**Клас RelayCommand.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace \_2048\_game.Infrastructure

{

class RelayCommand : ICommand

{

readonly Action<object> \_execute;

readonly Predicate<object> \_canExecute;

public RelayCommand(Action<object> execute)

: this(execute, null)

{

}

public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)

{

\_execute = execute ?? throw new ArgumentNullException("execute");

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null ? true : \_canExecute.Invoke(parameter);

}

public event EventHandler CanExecuteChanged

{

add

{

CommandManager.RequerySuggested += value;

}

remove

{

CommandManager.RequerySuggested -= value;

}

}

public void Execute(object parameter)

{

\_execute.Invoke(parameter);

}

}

}

Додаток Г  
Тестування додатку

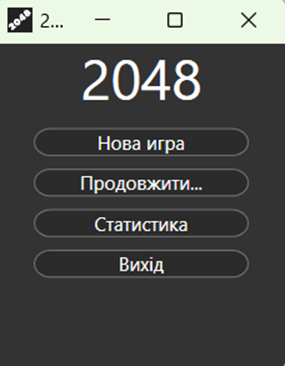
****

Рисунок 1 – Головне меню гри

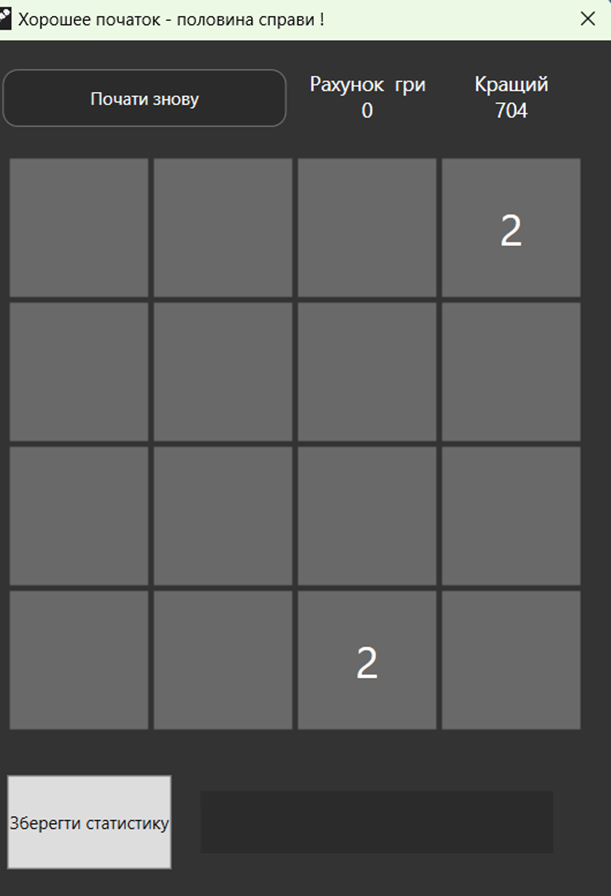


Рисунок 2 – Відображення вікна гри при натисканні на кнопку «Нова гра»

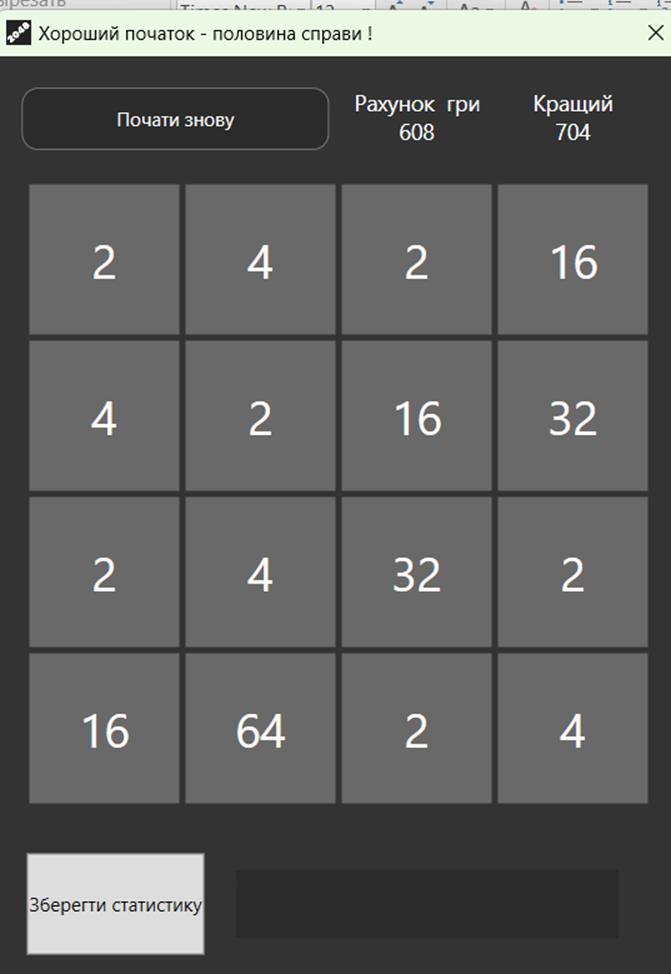


Рисунок 3 – Відображення вікна гри при натисканні кнопки «Продовжити»



Рисунок 4 – Зображена процедура додавання гравця до таблиці статистики

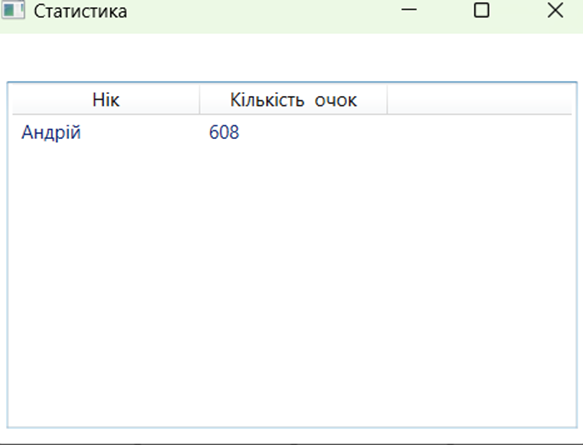


Рисунок 5 – Занесення кравця в таблицю статистики після натискання на кнопку «Зберегти статистику»

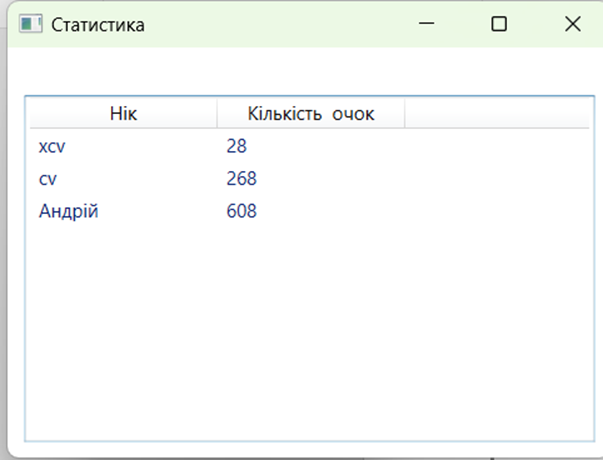


Рисунок 6 – Вікно статистики після додавання нового гравця

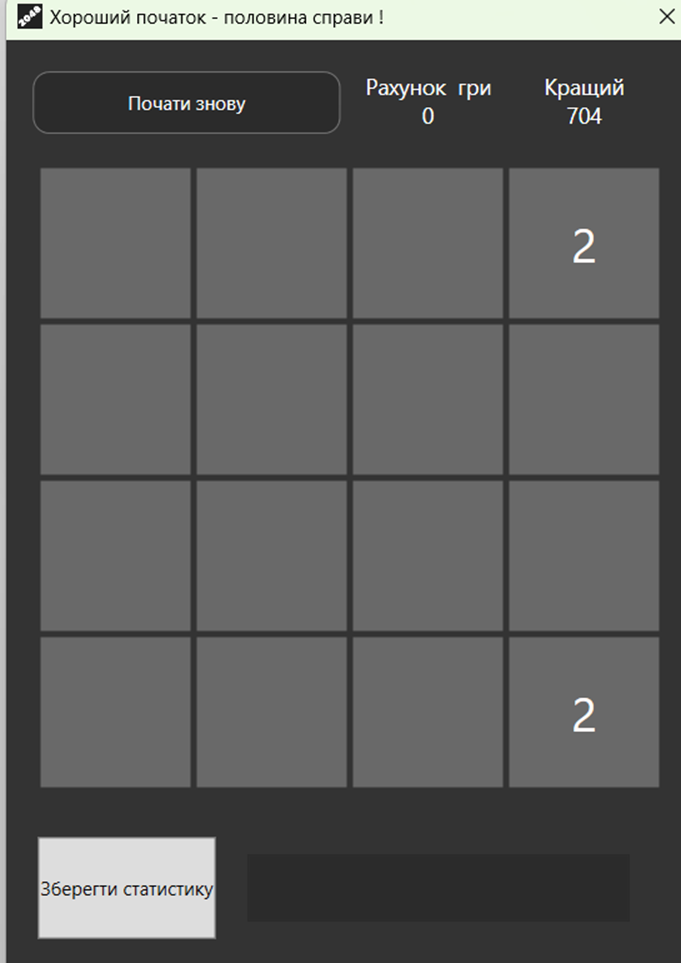


Рисунок 7 – Після натискання кнопки «Почати знову»