**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 “Шибениця”

Студента \_2\_ курсу КН–22–1 групи

Ступінь вищої освіти «Бакалавр»

(бакалавр, магістр)

Спеціальність 122 – «Комп’ютерні науки»

Освітньо-професійна програма

«Комп’ютерні науки»

Радіонов Д.О.

(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-22-1 семестр \_\_3\_\_

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Радіонову Дмитру Олеговичу \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема роботи: Розробка WPF додатку “Шибениця”
2. Термін здачі студентом роботи 1 грудня 2023 р
3. Вихідні дані до роботи: ColorThemes.json (зберігає кольорові теми для програми), Settings.json (містить значення налаштувань проекту),
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 Радіонов Дмитро Олегович

(підпис)

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

(підпис) (ініціали та прізвище)

«1» жовтня 2023 р.

РЕФЕРАТ

Курсова робота містить 31 сторінок, 2 розділи, 10 рисунків, 1 таблиця, 5 використаних джерел.

Об’єкт розробки – WPF додаток «Шибениця».

Мета: створення віконного Wpf додатку, що реалізує гру «Шибениця».

Під час виконання завдання, поставленого на курсову роботу, було виконано аналіз алгоритму гри «Шибениця». Визначено основні функціональні та не функціональні вимоги до додатку, що створюється. Побудована модель даних, що дозволить вирішити наступні задачі: створення віконного інтерфейсу задля графічного відображення етапів гри 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 проводити гру,  
можна вибрати будь-яку букву і всі аналогічні букви будуть відкриті

ОБ’ЄКТ, ІНКАПСУЛЯЦІЯ, ІНТЕРФЕЙС, КЛАС

**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 4](#_Toc151681433)

[1.2.1 Функціональні вимоги 4](#_Toc151681434)

[1.2.2 Нефункціональні вимоги 4](#_Toc151681435)

[1.3 Опис алгоритму основних задач/підзадач у роботі 5](#_Toc151681436)

[1.3.1 Структура програмного забезпечення 5](#_Toc151681437)

[Висновки до розділу 5](#_Toc151681438)

[2 Опис розробки програмного забезпечення 6](#_Toc151681439)

[2.1 Структура програмного забезпечення 6](#_Toc151681440)

[2.2 Опис роботи програми 8](#_Toc151681441)

[2.3 Функціональна схема програми 9](#_Toc151681442)

[2.4 Опис інтерфейсу програми 10](#_Toc151681443)

[Висновки до розділу 10](#_Toc151681444)

[ВИСНОВКИ 11](#_Toc151681445)

[СПИСОК ЛІТЕРАТУРИ 12](#_Toc151681446)

[Додаток А XAML розмітки вікон 13](#_Toc151681447)

[Додаток Б C# коди класів вікон 16](#_Toc151681448)

[Додаток B C# основні класи 24](#_Toc151681449)

[Додаток Г Тестування додатку 30](#_Toc151681450)

ВСТУП

Технологія WPF (Windows Presentation Foundation) є частиною екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

WPF є дуже поширеною платформою для розробки програмного забезпечення із графічним інтерфейсом на мові програмування C#.

Якщо при створені традиційних додатків на основі WinForms за візуалізацію елементів керування відповідали такі частини ОС Windows, як User32 та GDI+, то додатки WPF засновані на **DirectX**.В цьому і є головна відмінність рендерингу графіки у WPF: велика частина роботи по візуалізації графіки, як і звичайних кнопок, так і складних 3D-моделей, лежить на графічному процесорі на відеокарті, що також дає можливість користуватись апаратним прискоренням графіки.

Ще однією важливою особливістю WPF є використання мови декларативної розмітки XAML, заснованого на XML: ви можете створити гарний графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керуючих мовах, як C#.

Платформа WPF підтримує багато можливостей для створення різних додатків: створення анімацій, binding, стилі, шаблони, велика кількість елементів.

Тому WPF – непоганий варіант для створення ігри «Крапки», де за допомогою анімацій, ресурсів, стилів та біндингу можна створити гарний та зрозумілий інтерфейс. А за допомогою вбудованих класів та інтерфейсів зручно робиться логіка гри.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Делегати. Стандартні делегати. Для чого використовуються.

Делегати є однією з ключових концепцій мови програмування C#. Вони дозволяють передавати метод як параметр в інший метод, що надає багато корисних можливостей.

Основне використання делегатів полягає в реалізації механізму зворотного виклику (callback). Вони дозволяють передавати виклик методу з одного об'єкта в інший, забезпечуючи зв'язок між цими об'єктами. Таким чином, вони дозволяють будь-якій частині програми визвати метод, який визначений в іншій частині програми.

Основні переваги використання делегатів:

1. Розривання купи залежностей: Використання делегатів дозволяє скласти код таким чином, що об'єкт, який виконує виклик методу, не залежить від конкретного об'єкта, на якому має бути викликано цей метод. Це сприяє зменшенню залежностей між об'єктами і полегшує тестування і розподілену роботу.

2. Зростання перевикористовування коду: Делегати дозволяють використовувати методи з об'єктів різних класів, сприяючи перевикористанню коду. Наприклад, ви можете передати делегату вкладений метод з одного класу і викликати його з іншого класу.

3. Підтримка асинхронного програмування: Делегати є ключовими компонентами для реалізації асинхронного програмування в C#. Вони дозволяють визначити методи зворотного виклику, які можна викликати після завершення виконання асинхронної операції.

4. Реалізація подій: Делегати використовуються для реалізації подій в C#. Події дозволяють класу або об'єкту повідомити інші класи або об'єкти про виникнення певної події.

Оглядаючи всі ці використання делегатів, можно сказати, що вони є потужним інструментом для досягнення гнучкості і розширення функціональності коду.

1.2 Аналіз технічного завдання на роботу

Мета курсового проєкту – створення віконного Wpf додатку, що реалізує гру «Шибениця».

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

**1.2.1 Функціональні вимоги**

1. Графічне відображення шибеницію. Гра повинна відображати зображення шибениці, на якій поступово з’являються риси в разі неправильного вгадування букви
2. Показ заданого слова. Гра повинна показувати пусті місця, що відповідають кількості літер в загаданому слові і розташовувати вгадані букви на відповідних місцях
3. Перевірка правильності вгадування. Гра повинна перевіряти, чи є введена буква в загаданому слові і оновлювати відображення загаданого слова та шибениці відповідно до результату.
4. Обмеження на кількість неправильних вгадувань. Гра повинна обмежувати кількість неправильних вгадувань, після чого вважати гру програною.

**1.2.2 Нефункціональні вимоги**

1. Зовнішній інтерфейс користувача має бути реалізованим за допомогою створення вікон в середовищі WPF додатку.
2. Додаток має бути віконним
3. Усі поля введення повинні бути захищені від некоректного введення.
4. Використання патерну MVVM тa прив’язки даних .

1.3 Опис алгоритму основних задач/підзадач у роботі

Розробка алгоритмів функціонування програми в контексті гри “Шибениця” у середовищі C# WPF передбачає створення моделі роботи та функціональності програми як єдиної системи. На даному етапі необхідно передбачити обмеження, які можуть у майбутньому вплинути на коректне розширення функціоналу програми.

Програма розробляється на мові програмування C# з використанням технології WPF. Основна мета програми – відображення гри “Шибениця”.

Основні кроки роботи програми:

* Користувачу відображається головний екран програми в якому є загадане слово та букви на які можна натиснути.
* При введенні правильної літери вона відображається у загаданному слові, якщо неправильна то у вікні з’являється рисочка
* У разі якщо користувач помиляється 10 разів то гра припиняється, та викликається вікно з можливістю повторно розпочати гру.
* Коли користувач вгадує слово, то показується вікно перемоги, в якому можна вийти або почати знову.

Висновки до розділу

У першому розділі розглянуті базові принципи ООП, створена розширена постановка задачі, проаналізована предметна область, сформовано вимоги до функціональних можливостей програми та розглянуто технічне завдання. Складений план розробки програми та функції, що повинні бути реалізовані у програмі. Також, розроблений алгоритм роботи програми.

2 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Структура програмного забезпечення

Структура програмного забезпечення є поєднанням усіх застосованих модулів програми, що утворюють працюючу систему. Кожний модуль повинен реалізовувати функції програми, що вирішують основні завдання проєкту.

Застосунок створений за допомою модулю Windows Presentation Foundation (WPF) на базі мови с#.

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

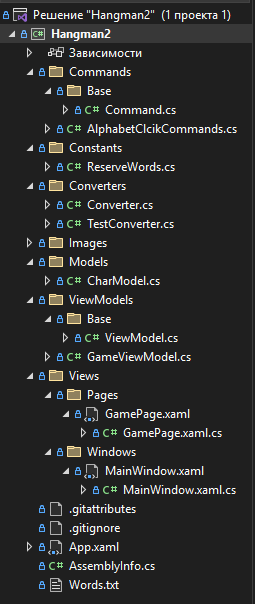


Рисунок 2.1 – Оглядач рішень

Таблиця 2.1 – Призначення модулів програми

|  |  |
| --- | --- |
| **Назва модуля** | **Призначення модуля** |
| Command.cs | Абстрактний клас, який реалізує інтерфейс Icommand, і містить базовий функціонал для створення команд |
| AlphabetClickCommands.cs | Є командою зв’язаною з натисканням на літеру алфавіта у грі |
| ReserveWords.cs | Клас, який містить резерв слів |
| Converter.cs | Клас вікна, що містить методи для коректної роботи вікна AccountMenu.xaml |
| TestConverter.cs |  |
| ViewModel.cs | Клас, який оповіщує прив’язані об’єкти що характеристика була змінена.OnPropertyChanged |
| GameViewModel.cs | Зв’язування даних з елементами керування на сторінці |
| GamePage.xaml |  |
| GamePage.xaml.cs |  |
| MainWindow.xaml |  |
| MainWindow.xaml.cs |  |

2.2 Опис роботи програми

Після запуску програми відкривається головне вікно, в якому відображається алфавіт на який можна натиснути та загадане слово яке обирається випадковим чином із заданого файлу. При натисканні правильної букви на місці загаданого слова відображаються букви. Якщо користувач натиснув неправильну букву то з’являється риска, при 10 помилках викликається вікно програшу, в якому пропонується розпочати заново або закрити програму.

**2.3 Функціональна схема програми**

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

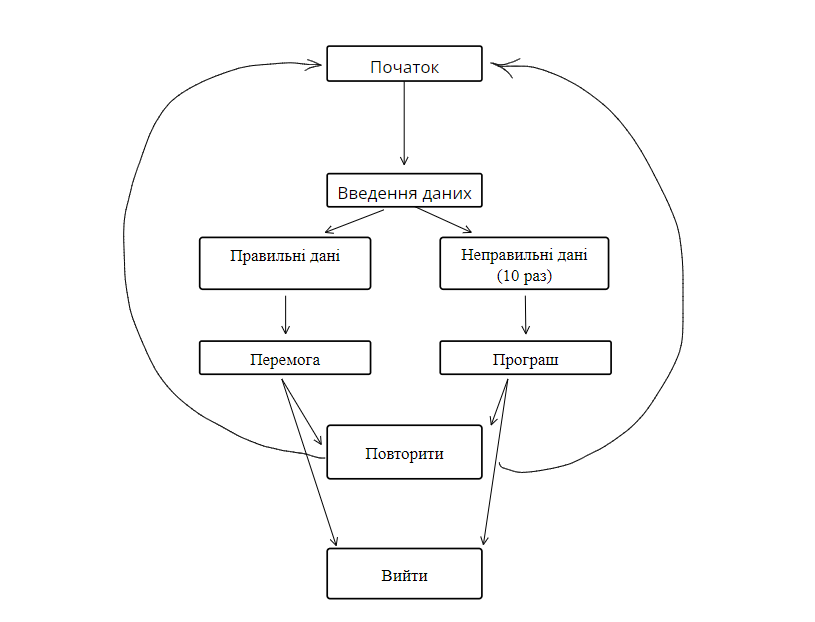


Рисунок 2.4 – Функціональна схема програмного коду

Як видно з рис.2.4 після запуску за стосунку першим відображається вікно з алфавітом і загаданим словом.

2.4 Опис інтерфейсу програми

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

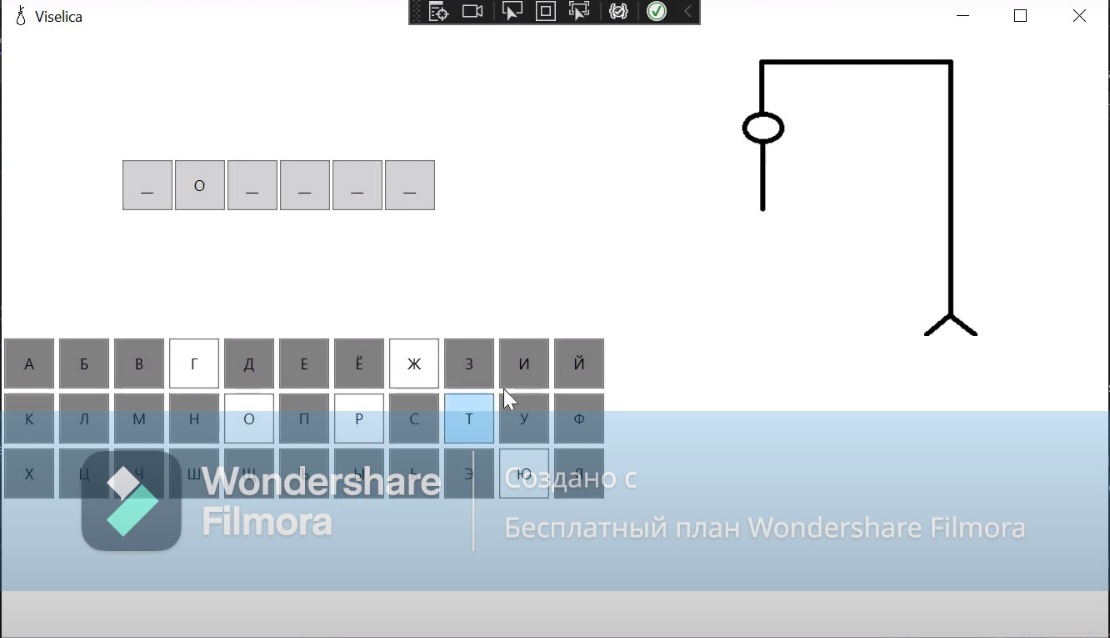


Рисунок 2.5 – Інтерфейс вікна MainWindow

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

Курсовий проєкт спрямований на створення WPF додатку «Шибениця». Розглянуті базові принципи ООП, проаналізована предметна область та розглянуто технічне завдання, реалізовано можливість реєстрації та авторизації, можливість створити платіж за книгу та оплатити його, можливість переглянути свої платежі.

Для виконання даного проєкту була використана мова програмування C#, яка добре підходить для розробки програм з використанням принципів ООП.

Важливим етапом розробки стали планування й реалізація користувальницького інтерфейсу. Основною метою було задоволення ряду потреб: комфортність роботи з інтерфейсом, швидкодія при обміні даних, ефективність управління, реалізація необхідних функцій. Використання приємних для більшості людей кольорів, а також захист від усіляких способів некоректного введення.

Таким чином, у курсового проєкту продемонстровані найважливіші методи роботи з програмним середовищем WPF, його інструментами та функціями.

СПИСОК ЛІТЕРАТУРИ

1. Полное руководство по языку программирования С# 11 и платформе .NET 7. Інформаційний ресурс. <https://metanit.com/sharp/tutorial/>
2. Джейсон Шрейкер. Кровь пот и пиксели. Форс, 2019. 368 с.
3. Тайнан Сильвестр. Геймдизайн. Санкт-Петербург: Питер, 2021. 448 с.
4. Вайсфельд М. Об’єктно-ориентированное мышление. Санкт-Петербург: Питер, 2014. 304 с.
5. Уроки створення додатку Windows Forms C#. Інформаційний ресурс. <https://itproger.com/course/csharp-app>

Додаток А  
XAML розмітки вікон

**Файл GamePage.xaml**

<Page x:Class="Hangman2.Views.Pages.GamePage"

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:Hangman2.Views.Pages;assembly=Hangman2"

xmlns:vm="clr-namespace:Hangman2.ViewModels;assembly=Hangman2"

xmlns:c="clr-namespace.Hangman2.Converters;assembly=Hangman2"

xmlns:ce="clr-namespace.Hangman2.Commands;assembly=Hangman2"

mc:Ignorable="d"

Title="GamePage" Height="450" Width="800">

<Page.Resources>

<vm:GameViewModel x:Key="gvm"/>

<c:TestConverter x:Key="TestConverter"/>

<ce:AlphabetClickCommand x:Key="AlphabetClickCommand"/>

<DataTemplate x:Key="WordTemplate">

<Button Width="40"

Height="40"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Margin="1"

Content="{Binding Text}">

<Button.Background>

<SolidColorBrush Color="LightGray"></SolidColorBrush>

</Button.Background>

</Button>

</DataTemplate>

<DataTemplate x:Key="Alphabet\_level2">

<Button Command="{StaticResource AlphabetClickCommand}"

Width="40"

Height="40"

Background="{Binding Color}"

Margin="2"

Content="{Binding Text}">

<Button.CommandParameter>

<MultiBinding Converter="{StaticResource TestConverter}">

<Binding/>

<Binding Source="{StaticResource gvm}"/>

</MultiBinding>

</Button.CommandParameter>

</Button>

</DataTemplate>

<DataTemplate x:Key="Alphabet\_level1">

<ItemsControl ItemsSource="{Binding}"

ItemTemplate="{DynamicResource Alphabet\_Level2}"

HorizontalAlignment="Left"

VerticalAlignment="Top">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</DataTemplate>

</Page.Resources>

<Grid DataContext="{StaticResource gvm}">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ItemsControl Grid.Column="0"

ItemsSource="{Binding Word}"

ItemTemplate="{DynamicResource WordTemplate}"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

<Image Source="{Binding ImagePath}"

Grid.Column="1"/>

</Grid>

<Grid Grid.Row="1">

<ItemsControl ItemsSource="{Binding Alphabet}"

ItemTemplate="{DynamicResource Alphabet\_Level1}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Vertical"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

</ItemsControl>

</Grid>

</Grid>

</Page>

**Файл MainWindow.xaml**

<Window x:Class="Hangman2.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:Hangman2"

mc:Ignorable="d"

MinWidth="500"

MinHeight="305"

Title="Hangman2" Icon="../../Images/icon.png" Height="520" Width="900">

<Grid>

<Frame Source="../Pages/GamePage.xaml"/>

</Grid>

</Window>

Додаток Б  
C# коди класів вікон

**Клас Command.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace Hangman2.Commands.Base

{

public abstract class Command : ICommand

{

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

public abstract bool CanExecute(object parameter);

public abstract void Execute(object parameter);

}

}

**Клас AlphabetClickCommands.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Hangman2.Commands.Base;

using Hangman2.Models;

using Hangman2.ViewModels;

namespace Hangman2.Commands

{

class AlphabetClickCommands : Command

{

public override bool CanExecute(object parameter)

{

return true;

}

public override void Execute(object parameter)

{

object[] parameters = parameter as object[];

CharModel chr = parameters[0] as CharModel;

GameViewModel gvm = parameters[1] as GameViewModel;

gvm.AlphabetClick(chr);

//chr.Parent.AlphabetClick(chr);

}

}

}

**Клас Converter.cs**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Data;

namespace Hangman2.Converters

{

abstract class Converter : IValueConverter

{

public abstract object Convert(object v, Type t, object p, CultureInfo c);

public virtual object ConvertBack(object v, Type t, object p, CultureInfo c) =>

throw new NotSupportedException("Зворотне перетворення не підтримується");

}

}

**Клас TestConverter.cs**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Data;

using Hangman2.Views.Pages;

namespace Hangman2.Converters

{

public class TestConverter : IMultiValueConverter

{

public object Convert(object[] values, Type targeType, object parameter, CultureInfo culture)

{

return values.Clone();

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Клас CharModel.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media;

using Hangman2.ViewModels.Base;

namespace Hangman2.Models

{

public class CharModel : ViewModel

{

private string text;

public string Text

{

get => text;

set => Set(ref text, value);

}

private bool enable;

public bool Enable

{

get => enable;

set

{

Set(ref enable, value);

if (enable)

{

Color = "gray";

}

else

{

Color = "white";

}

}

}

private string color = "gray";

public string Color

{

get => color;

set => Set(ref color, value);

}

}

}

**КласViewModel.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Hangman2.ViewModels.Base

{

public abstract class ViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string proptyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(proptyName));

}

protected virtual bool Set<T>(ref T field, T value, [CallerMemberName] string proptyName = null)

{

if (Equals(field, value))

{

return false;

}

field = value;

OnPropertyChanged(proptyName);

return true;

}

}

}

**Файл GameViewModel.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media.TextFormatting;

using System.Collections.ObjectModel;

using System.Web;

using System.Threading.Tasks.Dataflow;

using System.Windows;

using Hangman2.Constants;

using System.Windows.Controls;

using System.Xaml;

using System.Windows.Markup.Localizer;

using System.IO;

using System.Windows.Documents;

using Hangman2.Models;

using Hangman2.ViewModels.Base;

using Hangman2.Views.Pages;

namespace Hangman2.ViewModels

{

public class GameViewModel : ViewModel

{

#region Word Text

private string? wordText;

public string WordText

{

get => wordText!; set => Set(ref wordText, value); }

#endregion

private string? imagePath;

public string ImagePath

{

get => imagePath!;

set => Set(ref imagePath, value);

}

private ObservableCollection<CharModel> word = new();

public ObservableCollection<CharModel> Word

{

get => word;

set => Set(ref word, value);

}

private ObservableCollection<ObservableCollection<CharModel>> alphabet = new();

public ObservableCollection<ObservableCollection<CharModel>> Alphabet

{

get => alphabet;

set => Set(ref alphabet, value);

}

private int step;

public int Step

{

get => step;

set => Set(ref step, value);

}

private bool gameEnable;

public bool GameEnable

{

get => gameEnable;

set => Set(ref gameEnable, value);

}

public GameViewModel()

{

StartGame();

}

public void GenerateAlphabet()

{

string alphabet\_set = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

Alphabet.Clear();

for (int i = 0; i < 3; i++)

{

ObservableCollection<CharModel> row = new();

row.Clear();

for (int j = 0; j < 11; j++)

{

row.Add(new CharModel() { Text = alphabet\_set[i \* 11 + j].ToString(), Enable = true, Color = "gray" });

}

Alphabet.Add(row);

}

}

public void ShowWord()

{

for (int i = 0; i < WordText.Length; i++)

{

Word[i].Text = wordText![i].ToString();

}

}

public void UpdateTexturePath()

{

ImagePath = $"../../Images/{step}.png";

}

public void Win()

{

MessageBox.Show("Перемога", "Ура");

RestartGame();

}

public void Lose()

{

MessageBox.Show("Ви програли", "Ви програли");

RestartGame();

}

public void RestartGame()

{

MessageBoxResult res = MessageBox.Show("Грати знову?", "Грати", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (res == MessageBoxResult.Yes)

{

StartGame();

}

else

{

Application.Current.Shutdown();

}

}

public void StartGame()

{

Step = 0;

string[] Words = ReadWordsFromFile("Words.txt");

Random rnd = new();

int index = rnd.Next(0, Words.Length);

WordText = Words[index].ToUpper();

GenerateWord(wordText!);

UpdateTexturePath();

GenerateAlphabet();

GameEnable = true;

}

public static string[] ReadWordsFromFile(string path)

{

try

{

using FileStream fstream = File.OpenRead(path);

byte[] buffer = new byte[fstream.Length];

fstream.Read(buffer, 0, buffer.Length);

string textFromFile = Encoding.Default.GetString(buffer);

string[] array = textFromFile.Split("\n");

for (int i = 0; i < array.Length - 1; i++)

{

array[i] = array[i][..(array[i].Length - 1)];

}

return array;

}

catch (FileNotFoundException)

{

return ReserveWords.reserve;

}

}

public void GenerateWord(string str)

{

Word.Clear();

for (int i = 0; i < str.Length; i++)

{

Word.Add(new CharModel() { Text = "\_\_\_", Enable = true, Color = "gray" });

}

}

public void AlphabetClick(CharModel chr)

{

if (!chr.Enable) return;

chr.Enable = false;

if (OpenChar(chr.Text))

{

int free = 0;

for (int i = 0; i < Word.Count; i++)

{

if (Word[i].Text == "\_\_\_")

{

free++;

}

}

if (free < 1)

{

Win();

}

}

else

{

step++;

UpdateTexturePath();

if (step == 9) ;

{

ShowWord();

Lose();

}

}

}

public bool OpenChar(string text)

{

bool inWord = false;

for (int i = 0; i < WordText.Length; i++)

{

if (WordText[i].ToString() == text)

{

Word[i].Text = text;

inWord = true;

//break;

}

}

return inWord;

}

}

}

Додаток B  
C# основні класи

**Клас Field.cs**

public class Field

{

public static int ROWS = 10;

public static int COLS = 17;

public static int CellSize { get; set; } = 39;

public float[] playersDomination = { 0.0f, 0.0f };

public CellsField cellsField;

public DotsField dotsField;

public ObservableCollection<CellsViewModel> Cells { get { return cellsField.Cells; } }

public ObservableCollection<DotsViewModel> Dots { get { return dotsField.Dots; } }

public Field()

{

Dot.SetCountersZero();

cellsField = new CellsField(this);

dotsField = new DotsField(this);

}

}

**Клас CellsField.cs**

public class CellsField

{

public int ROWS;

public int COLS;

public Field from;

public ObservableCollection<CellsViewModel> Cells { get; set; } = new ObservableCollection<CellsViewModel>();

public CellsField(Field fromField)

{

from = fromField;

ROWS = Field.ROWS - 1;

COLS = Field.COLS - 1;

for (int i = 0; i < ROWS; ++i)

{

for (int j = 0; j < COLS; ++j)

{

Cells.Add(new CellsViewModel(i, j, "0,0", "1,1",

CurrentColors.GetBackgroundColor, CurrentColors.GetBackgroundColor));

}

}

}

private void ChangeCellAt(int i, int j, CellsViewModel model)

{

Cells.RemoveAt(i \* COLS + j);

Cells.Insert(i \* COLS + j, model);

}

public void CheckAllCells()

{

for (int i = 0; i < ROWS; ++i)

{

for (int j = 0; j < COLS; ++j)

{

if (isEmptyCellAt(i, j)) continue;

if (CheckForFullPatterns(i, j)) continue;

CheckForHalfPatterns(i, j);

}

}

}

private bool isEmptyCellAt(int i, int j)

{

string defaultColor = CurrentColors.Theme[GameColors.Default];

int defaults = 0;

foreach (int \_i in new int[]{ i, i + 1 })

foreach (int \_j in new int[]{ j, j + 1 })

defaults += (from.dotsField[\_i, \_j].BackgroundColor == defaultColor) ? 1 : 0;

return defaults > 1;

}

private bool CheckForFullPatterns(int i, int j)

{

int k = -1;

if (CheckPattern(i, j, "0000")) k = 0;

else if (CheckPattern(i, j, "1111")) k = 1;

else return false;

ChangeCellAt(i, j, new CellsViewModel(i, j, "0,0", "1,1", CurrentColors.MainColorOfPlayer(k), CurrentColors.MainColorOfPlayer(k)));

from.playersDomination[k] += 1.0f;

return true;

}

private bool CheckForHalfPatterns(int i, int j)

{

string[][] patterns = { new string[] { "a00\*", "\*00a", "0\*a0", "0a\*0" },

new string[] { "b11\*", "\*11b", "1\*b1", "1b\*1" }};

for (int num = 0; num < 2; ++num)

{

foreach (string pattern in patterns[num])

{

if (CheckPattern(i, j, pattern))

{

int a = pattern.IndexOf('\*');

int b = pattern.IndexOf((num == 0) ? 'a' : 'b');

ChangeCellAt(i, j, new CellsViewModel(i, j, $"{b % 2},{b / 2}", $"{a % 2},{a / 2}",

CurrentColors.MainColorOfPlayer(num),

CurrentColors.GetBackgroundColor));

from.playersDomination[num] += 0.5f;

return true;

}

}

}

return false;

}

// pattern містить чотири символи, для TL TR BL BR відповідно:

// - "0" будь-який колір першого гравця

// - "1" будь-який колір другого гравця

// - "a" світлий колір першого гравця

// - "b" світлий колір другого гравця

// інше - будь-який колір взагалі

private bool CheckPattern(int i, int j, string pattern)

{

int lightCounter = 0;

int[,] IJ = { { i, j }, { i, j + 1 }, { i + 1, j }, { i + 1, j + 1 } };

for (int num = 0; num < IJ.GetLength(0); ++num)

{

string color = from.dotsField.ColorAt(IJ[num, 0], IJ[num, 1]);

switch (pattern[num])

{

case '0':

case '1':

GameColors m = (pattern[num] == '0') ? GameColors.Player1 : GameColors.Player2;

if (!(color == CurrentColors.Theme[m] ||

color == CurrentColors.lightFrom[CurrentColors.Theme[m]]))

return false;

if (color == CurrentColors.lightFrom[CurrentColors.Theme[m]])

lightCounter++;

break;

case 'a':

case 'b':

if (color != CurrentColors.LightColorOfPlayer(pattern[num] - 'a'))

return false;

break;

default:

break;

}

}

if (!pattern.Contains('\*') && lightCounter < 2)

return false;

return true;

}

}

**Клас DotsField.cs**

public class DotsField

{

private Field from;

public int ROWS;

public int COLS;

public ObservableCollection<DotsViewModel> Dots { get; set; } = new ObservableCollection<DotsViewModel>();

public string ColorAt(int i, int j) => Dots[i \* COLS + j].BackgroundColor;

public DotsViewModel this[int i, int j] => Dots[i \* COLS + j];

public bool[,] checkedDots;

private void SetCheckedDotsFalse()

{

for (int i = 0; i < ROWS; ++i)

{

for (int j = 0; j < COLS; ++j)

{

checkedDots[i, j] = false;

}

}

}

public DotsField(Field fromField)

{

from = fromField;

ROWS = Field.ROWS;

COLS = Field.COLS;

checkedDots = new bool[ROWS, COLS];

for (int i = 0; i < ROWS; ++i)

{

for (int j = 0; j < COLS; ++j)

{

Dots.Add(new DotsViewModel(CurrentColors.Theme[GameColors.Default], i, j));

}

}

}

private bool NotOccupiedFrom(int i, int j, string color)

{

checkedDots[i, j] = true;

int[,] IJ = { { i, j + 1 }, { i, j - 1 }, { i + 1, j }, { i - 1, j } };

bool isFree = false;

for (int num = 0; num < IJ.GetLength(0); ++num)

{

if (IJ[num, 0] < 0 || IJ[num, 1] < 0 || IJ[num, 0] >= ROWS || IJ[num, 1] >= COLS) return true;

if (!checkedDots[IJ[num, 0], IJ[num, 1]] &&

(ColorAt(IJ[num, 0], IJ[num, 1]) == color ||

ColorAt(IJ[num, 0], IJ[num, 1]) == CurrentColors.Theme[GameColors.Default] ||

CurrentColors.lightFrom.Any(col => col.Value == ColorAt(IJ[num, 0], IJ[num, 1])))

)

{

isFree = isFree || NotOccupiedFrom(IJ[num, 0], IJ[num, 1], color);

}

}

return isFree;

}

public bool GameFunc(string forColor)

{

SetCheckedDotsFalse();

bool hereIsEmptyDots = false;

for (int i = 0; i < ROWS; ++i)

{

for (int j = 0; j < COLS; ++j)

{

if (ColorAt(i, j) == CurrentColors.Theme[GameColors.Default])

hereIsEmptyDots = true;

if (ColorAt(i, j) != forColor)

continue;

if (!NotOccupiedFrom(i, j, ColorAt(i, j)))

{

string color = CurrentColors.lightFrom[CurrentColors.getOpposite[forColor]];

for (int \_i = 0; \_i < ROWS; ++\_i)

{

for (int \_j = 0; \_j < COLS; ++\_j)

{

if (checkedDots[\_i, \_j])

{

this[\_i, \_j].BackgroundColor = color;

this[\_i, \_j].Occupied = true;

}

}

}

}

else

{

SetCheckedDotsFalse();

}

}

}

return hereIsEmptyDots;

}

}

Додаток Г  
Тестування додатку

