Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЕКТ**

з \_\_Принципи конструювання програмного забезпечення\_\_

(назва дисципліни)

на тему:

**«Гра «Водопровідник»»**

Виконала: студентка \_\_1\_\_\_ курсу, групи\_\_\_143\_\_\_\_

спеціальності \_\_\_\_\_121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Буряк О. І.\_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Дячук Р.Л.\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Нормоконтролер \_\_\_\_\_\_\_Комісарчук В.В.\_\_\_\_\_

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці, 20\_21\_

Чернівецький національний університет імені Юрія Федьковича

Кафедра \_Програмного забезпечення комп’ютерних систем\_\_\_

Спеціальність \_Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсовий проект студенту**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Буряк Ользі Іванівні\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема проекту \_ **Гра «Водопровідник»**\_\_\_\_\_\_\_\_

2. Вихідні дані до проекту:

\_– розробити 5 видів труб;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– розробити функціонал, який дозволяє рандомно згенерувати один з 5 видів труб, повернути цю трубу згідно з потребами користувача та встановити її в доступне місце на гральному полі;

– розробити функціонал для розуміння етапу гри (гру завершено/гра триває);\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– розробити фунціонал, який обмежує користувача в часі складання трубопроводу та пускає воду за 5 секунд до завершення цього часу;

– розробити можливість збереження даних з результатами гри (ім’я гравця та час складання) для формування таблиці результатів.

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

\_– розробити загальні вимоги до програми;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– описати модулі програми та алгоритми роботи у вигляді блок-схем;\_\_\_\_\_\_

\_– описати методи програми;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– опис користувацького інтерфейсу;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– зробити висновки по розробленій програмі;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– навести текст програми у вигляді програмного коду;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

\_– блок-схеми роботи з програмою;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– скріншоти роботи з програмою.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

РЕФЕРАТ

В курсовому проекті розроблено інтерактивну гру «Водопровідник» для ПК, які працюють під керуванням операційної системи Windows.

Гра розрахована на користувачів, які не мають спеціальної комп’ютерної підготовки. Досвід роботи не вимагається.

Область застосування – програмне забезпечення (ігри) для недорогих ПК з обмеженими обчислювальними ресурсами.

Розробка реалізована засобами фреймворку WPF на мові C# та XAML. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на ОС Windows.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

Курсовий проект містить: 78 с., 10 рис., 9 табл., 1 додатки, 4 джерел.

*ОПЕРАЦІЙНА СИСТЕМА, ІНТЕРАКТИВНА ГРА, ПК, WPF, XAML, C#.*

SUMMARY

The interactive game "Plumber" is developed as the course project for PCs that run Windows operating system.

The game is designed for users who do not have special computer training. Work experience is not required.

The scope of application is software (games) for low-cost PCs with limited computing resources.

Development is implemented by means of WPF framework using C# language. This environment is easy to use for fast and high-quality creation of applications on Windows.

This development can be expanded in the future with the addition of new functionality and modified processing logic.

The course project contains: 78 p., 10 fig., 9 table., 1 appendices, 4 sources.

*OPERATING SYSTEM, INTERACTIVE GAME, PC, WPF, XAML, C #.*

**ЗМІСТ**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ……………………………………..…..…6

1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ…………..….……7
   1. Загальні вимоги до програми…………………………………………….7
   2. Призначення та область застосування………………………….…….…9
   3. Функціональні вимоги……………………………………………………9
2. ОПИС ПРОГРАМИ…………………………………………………….……15
   1. Структура програми………………………………………………..……15
   2. Опис методів програми…………………………………………….……18
   3. Програмні засоби………………………………………………...………22
   4. Опис користувацького інтерфейсу………………………………..……23

ВИСНОВКИ……………………………………………………………...………31

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………....….……32

ДОДАТКИ………………………………………………………………..………33

Додаток А. Скролінг (текст) програми………………………………….….33

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

ОС – операційна система.

ПК — персональний комп’ютер, електронна обчислювальна машина, що призначена для зберігання і переробки інформації.

WPF – Windows Presentation Foundation — графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з .NET Framework 3.0 в складі цієї платформи.

XAML – eXtensible Application Markup Language — декларативна мова розмітки.

**1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ**

**1.1 Загальні вимоги до програми**

**1.1.1 Вимоги до графічного інтерфейсу користувача**

1. Робоча мова інтерфейсу – українська.
2. Необхідно сконструювати такі вікна:
   1. вікно головного меню, в якому передбачити кнопки для доступу до головних вікон гри («Грати», «Досягнення», «Правила гри», «Допомога», «Вихід»).
   2. вікно з правилами гри, в якому передбачити кнопку повернення до головного меню.
   3. вікно допомоги, в якому передбачити кнопку повернення до головного меню.
   4. вікно з таблицею результатів проведених ігор, в якому передбачити кнопку повернення до головного меню.
   5. вікно з ігровим полем, в якому передбачити поле кнопок для основної гри, кнопку повороту для рандомно згенерованої труби, кнопку для запуску нової гри та кнопку виклику викидного меню.
   6. викидне меню, в якому передбачити кнопку повернення до гри зі збереженням всіх попередніх дій гравця та таймером, кнопки виходу в головне меню та кнопку початку нової гри.
   7. вікно з формою вводу імені, коли гру виграно, в якому передбачити текстове поле та кнопку «ОК», яка закриває вікно та повертає гравця до попереднього вікна.
3. Додатково необхідно передбачити управління з клавіатури, де клавіша Esc буде закривати поточне меню, клавіша F1 буде викликати вікно з правилами до гри або викидне меню, коли гру почато, клавіша Enter буде підтверджувати вибір користувача або завершувати ввід даних, клавіша Tab буде переходити до наступної кнопки в вікні, а комбінація клавіш Tab+Shift – повертатися до попередньої кнопки, доступної в вікні.
4. Поява відповідного повідомлення при програші гри, виборі кнопки «Вихід» в головному меню, виборі недоступної кнопки для того, щоб зробити хід, в вікні ігрового поля та при вичерпанні 30 секунд гри, щоб попередити учасника про кінець гри через 15 секунд.
5. Кольорова гама (палітра) за замовчуванням:

* фон ؘ– підкладка кольору # 0D7C87 з тематичними зображеннями та ефектом накладеного блюру:

Рисунок 1 – Фон гри

* зовнішній вигляд та типи труб – див. Табл.1;
* кнопки керування чорного кольору з прозорістю 0.8.

1. У програмі необхідно використати шрифти Finger Paint звичайного накреслення для відображення назви гри в головному меню, Icegirl [rus by aLiNcE] звичайного накреслення для пунктів меню та заголовків в інших вікнах, а також Yahfie звичайного накреслення для відображення головного тексту.

**1.1.2 Вимоги до архітектури програми**

1. Використання стандартних контейнерних класів, які реалізують:

* завантаження нової гри;
* зчитування / запис конфігурації гри
* завантаження медіа-файлів;
* робота з медіа-файлами;
* завантаження та запис в текстові файли.

**1.1.3 Вимоги до функціональності додатка**

1. Початок гри – висвітлення головного меню, з якого користувач може перейти безпосередньо до самої гри (кнопка «Грати») або переглянути правила гри натисненням на кнопку «Правила гри», переглянути таблицю рекордів натисненням на кнопку «Досягнення», переглянути контактні дані натисненням на кнопку «Про нас». Також передбачено отримання інформації про копірайтинг програми завдяки натисненню на кнопку «Вихід»;
2. При переході до вікна гри відбувається запуск таймера та рандомно генерується зображення труби та створюється об’єкт цієї труби;
3. Згенерована труба з’являється в верхній правій частині поля;
4. Поряд з трубою існує кнопка, натискаючи на яку, користувач може змінити поворот труби, повертаючи її послідовно на 90 градусів;
5. Традиційно при натисненні на кнопку ігрового поля вперше відбувається генерація об’єктів першої та останньої труби, добавлення їх в список всіх ігрових об’єктів поточної гри та встановлення координат меж ігрового поля відносно координат екрану користувача;
6. При натисненні на кнопку ігрового поля відбувається перевірка доступності кнопки, тобто перевірка, чи має поточна труба стик з попередньою на місці вибраної кнопки;
7. Також натиснення на кнопку передбачає перевірку на статус гри. Перевірка полягає в обчисленні кількість вільних стиків останньої поставленої труби та чи не впираються вони в межі ігрового поля та в уже поставлені труби;
8. Окрім перевірки завершення гри, метод встановлює ще виграш/програш користувача: якщо координати вільного стику поточної труби стикається з координатою входу в останню трубу, то гра завершена і гравець отримує виграш;
9. Якщо кнопка недоступна – користувач отримує відповідне повідомлення;
10. Якщо перевірка пройшла успішно і додати трубу на кнопку логічно можливо, тобто існує стик з попередньою трубою, то поточна труба добавляється в список ігрових об’єктів поточної гри, відбувається генерація нової труби;
11. Коли таймер досягає 30 секунд, гравець отримує повідомлення про те, що через 15 секунд гру буде завершено і піде вода;
12. За 5 секунд до вичерпання часу, перша труба починає заповнюватися водою, а після кожна труба заповнюється водою і гравець не має можливості ставити нові труби, гру завершено;
13. Генерація наступної фігури, яка з’являється у верхній частині ігрового поля, після нарахування балів гравцю.
14. Якщо кінця гри досягнуто і гру виграно, тобто остання покладена користувачем труба має стик з останньою трубою ланцюга, то дані з таймеру зберігаються в окремий файл, після відкривається вікно з повідомленням про виграш та поле, куди потрібно ввести ім’я гравця та натиснути клавішу «ОК», після цього дані будуть дозаписані в файл;
15. При виборі вікна з Таблицею результатів, з файлу, куди попередньо були записані результати ігор, зчитується інформація, котра в вигляді об’єкту гравця додається в масив; масив сортується та готова інформація відображається в вікні в вигляді «Гравець [ім’я] – [результат гри] секунд»;

**1.2 Призначення та область застосування**

**Мета роботи** полягає у розробці інтерактивної гри «Водопровідник» для ПК, які працюють під керуванням ОС Windows.

# Реалізована версія гри «Водопровідник» призначена для розважання користувачів ПК, що працюють під управлінням ОС Windows, при проведенні дозвілля, тренуванні просторової уяви, реакції та прогнозування своїх дій.

# Область застосування – програмне забезпечення для недорогих ПК з обмеженими обчислювальними ресурсами.

**1.3 Функціональні вимоги**

До програмного забезпечення висуваються такі функціональні вимоги:

1. Види труб, які повинна забезпечувати гра:

Таблиця 1 – Параметри фігур гри

|  |  |  |
| --- | --- | --- |
| **Вид труби** | **Вид труби під час заповнення водою** | **Характеристики** |
|  |  | Пряма труба з двома стиками, генерується як перша труба на ігровому полі. |
|  |  | Перехресна труба з 4 стиками. Під час гри гравець може вибрати тільки один з режимів використання: працює лівий та правий стики або верхній та нижній. |
|  |  | Потрійна труба, має 3 стики, котрі можуть бути задіяні бідь-яким обраним користувачем чином. |
|  |  | Г-подібна труба, має 2 стики. |
|  |  | Фінальна труба, генерується на початку гри, має тільки один вхід. |

* 1. Програма повинна забезпечувати зміну положення згенерованої труби в просторі, повертати зображення на 90 градусів при кожному натисненні на кнопку.
  2. Поле гри – прямокутник висотою 5 клітинок, шириною 5 клітинок, кожна клітинка розміром 70 на 70 пікселів.
  3. Якщо з початку гри минуло 30 секунд, то гравець отримує відповідне повідомлення, що йому потрібно зібрати водопровід за 15 секунд, через 10 секунд перша труба починає заповнюватися водою, ще через 5 частина поля, де генерується труба, змінює своє зображення, по трубах починає текти вода і всі подальші ходи блокуються, гравець отримує повідомлення про невдале закінчення гри.
  4. Мета гри – якнайшвидше скласти водопровід від першої труби, що розташована в верхньому лівому краю ігрового поля, до останньої труби, розташованої в діагонально протилежному місці на полі.
  5. Гра закінчується, коли координати вільного стику покладеної гравцем труби збігаються з координатами входу в останню трубу. Гра повинна забезпечувати повідомлення про закінчення гри та генерацію текстового поля для запису даних гравця.
  6. Гра повинна забезпечувати можливість почати гри заново або її повне завершення з виходом до операційної системи.

1. Функціональні вимоги до режиму гравця:
   1. Гра повинна надавати можливість обертати згенеровану трубу за допомогою кнопки в вікні гри, розташованої поряд з місцем генерації.
   2. Напрям обертання – вправо.
   3. Кут обертання – 90 градусів за одне натискання на кнопку.
   4. Труба повинна встановлюватися на місце натиснутої кнопки, якщо та доступна.
   5. Гра повинна забезпечувати підрахунок часу одного ігрового акту, запам’ятовування його та інших даних користувача при виграші та ведення відсортованої за кращими показниками часу таблиці гравців.
   6. Таблиця рекордів повинна зберігатися у постійній пам’яті доти, поки її не буде очищено.
   7. Гра повинна забезпечувати можливість призупинення процесу гри з подальшим продовженням її з того самого місця (виклик викидного меню).
   8. Гра повинна забезпечувати можливість показу наступної труби, коли поточну поставлено на місце на ігровому полі.

**2. ОПИС ПРОГРАМИ**

**2.1 Структура програми**

**2.1.1 Модулі програми**

Робота розробленого програмного забезпечення реалізується наступними модулями:

1. App – стандартний клас WPF, в якому реалізовано вхід в програму і відкриття робочого вікна.
2. MainWindow – клас, в якому реалізовано початкове меню гри.
3. Scores – клас, в якому реалізовано опис рекордів.
4. HelpBox – клас, в якому реалізовано вікно з інформацією.
5. HowToPlay – клас, в якому описано правила гри.
6. WinWindow – клас, в якому реалізовано логіку збереження даних гравця при успішному закінченні гри.
7. GameField – клас, в якому реалізовано ігрове поле та описано логіку гри.
8. GameObject – абстрактний клас, в якому представлено загальну логіку моделей гри.
9. GameObjectPlace – клас, в якому описано положення моделі на ігровому полі.
10. GameObjectType – енумератор, в якому зберігаються всі ігрові моделі програми.
11. Tube1, Tube2, Tube3, Tube4, Tube5 – класи-наслідники, в яких описано конкретний вид труби та його поведінку.

**2.1.2 Алгоритми роботи програми**

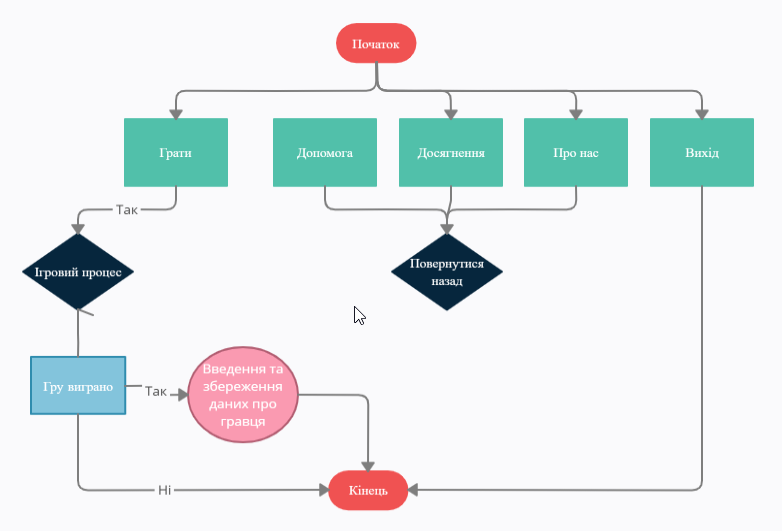


Рисунок 2 – Узагальнена схема роботи гри

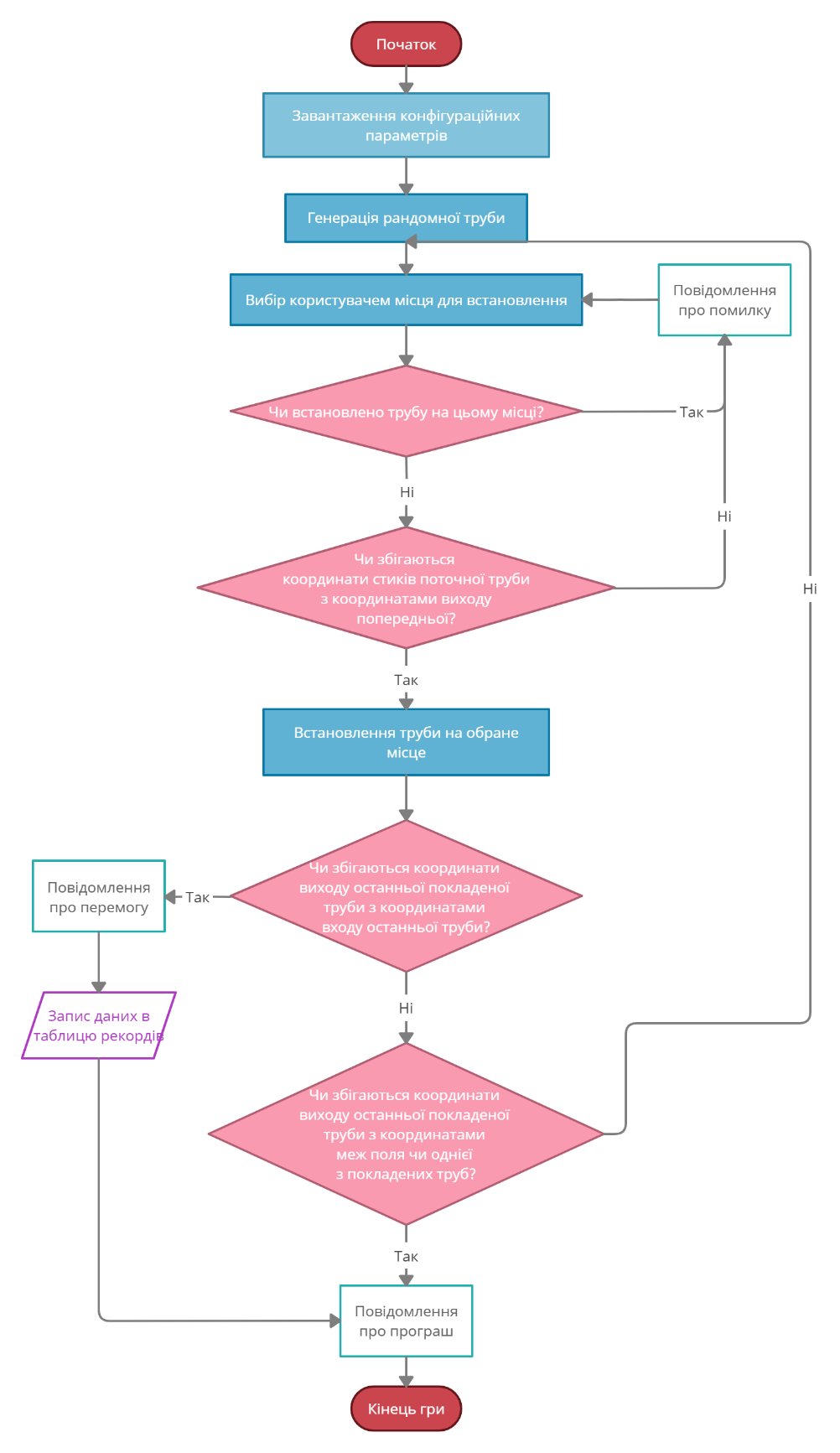


Рисунок 3 – Блок-схема ігрового процесу

**2.2 Опис методів програми**

Список методів класу MainWindow та їх опис наведено в табл.2.

Таблиця 2 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *MainWindow()* | Ініціалізує вікно головного меню. |
| 2 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |
| 3 | *Play\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно ігрового поля, закриває вікно меню. |
| 4 | *HowToPlay\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно з правилами, закриває вікно меню. |
| 5 | *HelpBox\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно з контактною інформацією, закриває вікно меню. |
| 6 | *Scores\_Click(object sender, RoutedEventArgs e)* | За допомогою метода ReadFile() струкури Gamer формує відсортований список гравців, створює вікно таблиці результатів, куди передає цей список та закриває поточне вікно. |
| 7 | *Exit\_Click(object sender, RoutedEventArgs e)* | Закриває поточне вікно. |

Список методів класу Scores та їх опис наведено в табл.3.

Таблиця 3 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *Scores(List<Gamer> gamers)* | Ініціалізує вікно досягнень, зчитує список гравців та почергово добавляє кожного гравця в текстове поле вікна. |
| 2 | *ComeBackButton\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно головного меню, закриває поточне меню. |
| 3 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |

Список методів класу HelpBox та їх опис наведено в табл.4.

Таблиця 4 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *HelpBox()* | Ініціалізує вікно контактної інформації. |
| 2 | *ComeBackButton\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно головного меню, закриває поточне меню. |
| 3 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |

Список методів класу HowToPlay та їх опис наведено в табл.5.

Таблиця 5 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *HowToPlay()* | Ініціалізує вікно з правилами. |
| 2 | *ComeBackButton\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває вікно головного меню, закриває поточне меню. |
| 3 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |

Список методів класу WinWindow та їх опис наведено в табл.6.

Таблиця 6 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *WinWindow()* | Ініціалізує вікно виграшу гри. |
| 2 | *GamerName\_TextChanged(object sender, TextChangedEventArgs e)* | Зчитує текст з віконця для введення та зберігає його в поле класу. |
| 3 | *OK\_Click(object sender, RoutedEventArgs e)* | Зберігає введені дані в файл, який служить імітованою базою даних для формування таблиці результатів. |
| 4 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |
| 5 | *GamerName\_KeyDown(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш, коли текстове поле знаходиться в фокусі, коли натиснено Enter – результат введення зберігається в файл та вікно закривається. |
| 6 | *GamerName\_GotFocus(object sender, RoutedEventArgs e)* | Коли поле вводу знаходиться в фокусі, початковий текст повідомлення-підказки очищується. |

Список методів класу GameField та їх опис наведено в табл.7.

Таблиця 7 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *GameField()* | Ініціалізує вікно ігрового поля, додатково очищує масив ігрових об’єктів, установлює таймер та рандомно генерує першу трубу. |
| 2 | *SetStartTubes()* | Ініціалізує першу та останню трубу, котрі знаходяться на ігровому полі за замовчуванням, встановлює координати та кількість їхніх доступних входів, додає їх в масив ігрових об’єктів, додає зображення першої труби в масив з зображеннями кнопок, встановлює останньою покладеною трубою першу трубу та формує масив з координатами меж поля. |
| 3 | *Replay\_Click(object sender, RoutedEventArgs e)* | Створює та відкриває нове ігрове вікно, закриває поточне. |
| 4 | *myTestKey(object sender, KeyEventArgs e)* | Зчитує натиснення клавіш на клавіатурі, реалізує управління клавішами. |
| 5 | *MenuBarButton\_Click(object sender, RoutedEventArgs e)* | Зупиняє таймер, створює та відкриває нове діалогове вікно викидного меню, куди передає стан поточного вікна |
| 6 | *SetTimer()* | Ініціалізує та запускає таймер. |
| 7 | *timer\_Tick(object sender, EventArgs e)* | Рахує зміну таймеру, передає результати в лейбл вікна, щоб таймер змінювався, здійснює перевірку таймера за допомогою функції timerCheck(). |
| 8 | *timerCheck()* | Здійснює перевірки часу та згідно з результатами виводить повідомлення та змінює колір лейблу з часом в ігровому вікні, коли таймер досягнув 30 секунд, запускає гіф-файл з витіканням води в першу трубу, коли таймер досягнув 41 секунди та зупиняє гру, коли таймер досягнув 45 секунд. |
| 9 | *SetNewTube()* | Перевіряє, чи завершено гру, і якщо гра триває, рандомно генерує число і згідно з ним встановлює зображення труби в спеціально відведене на полі місце та запам’ятовує цю трубу, якщо гру завершено – викликає функцію завершення гри. Також встановлює неможливість повороту кнопкою, коли випадає перехресна труба. |
| 10 | *ShowWater(GameObject tube, Image image)* | Викликає функцію генерації шляху до зображення по завершенню гри для отриманої труби та встановлює це зображення для отриманого зображення. |
| 11 | *Button\_Click(object sender, RoutedEventArgs e)* | Тільки при першому натисненні на кнопку, викликає функцію *SetStartTubes().* Встановлює зображення згенерованої труби як зображення натиснутої кнопки, зберігає локацію поточної труби, додає поточну трубу в масив ігрових об’єктів та встановлює її як останню трубу ланцюга. |
| 12 | *IsAvailableButton(object sender, RoutedEventArgs e)* | Перевіряє логічну доступність кнопки, звіряючи координати вільних стиків поточної та попередньої труб. |
| 13 | *Rotation\_Click(object sender, RoutedEventArgs e)* | Викликає функцію встановлення нових координат стиків для поточної труби та повертає зображення поточної труби. |
| 14 | *SetJointCoordinates(Point point)* | Встановлює нові координати стиків для поточної труби. |
| 15 | *FinishTheGame()* | Заповнює труби водою та виводить відповідне повідомлення(якщо гру програно) або викликає вікно для заповнення даних гравця(якщо гру виграно). |
| 16 | *IsGameOver()* | Перевіряє, чи закінчено гру, звіряючи координати вільного стику поточної труби з координатами стику останньої труби, координатами меж поля та координатами труб, збережених в масиві ігрових об’єктів, та встановлює чи завершено та чи виграно гру. |
| 17 | *SaveInformation()* | Записує в файл з даними дані з таймера та створює нове діалогове вікно з полем для вводу даних. |

Список методів класу GameObject та їх опис наведено в табл.8.

Таблиця 8 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *SetNewJoint()* | Віртуальний метод, який послідовно зсуває координати стиків. |
| 2 | *SetWinPath()* | Встановлює шлях до зображення об’єкту, котре встановлюється, коли гру завершено. |

Список методів інтерфейсу IGamer та їх опис наведено в табл.9.

Таблиця 9 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1 | *ReadFile();* | Зчитує інформацію для таблиці результатів та зберігає в масив геймерів. |

Класи Tube1, Tube2, Tube3, Tube4, Tube5 реалізують методи абстрактного класу GameObject, а структура Gamer реалізує метод інтерфейсу IGamer.

**2.3 Програмні засоби**

Розробка реалізована засобами середовища Visual Studio на мові C#. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на ОС Windows.

Windows забезпечує повний набір програмного забезпечення для ПК: операційну систему, сполучні програми і ключові сервіси операційної системи.

Windows – група сімейств комерційних операційних систем (OC) корпорації Microsoft, орієнтованих на управління за допомогою графічного інтерфейсу. Кожне сімейство обслуговує певний сектор комп'ютерної індустрії. Активні сімейства Microsoft Windows включають Windows NT і Windows IoT; вони можуть включати підродини (наприклад, Windows Server або Windows Embedded Compact) (Windows CE). Неактивні сімейства Microsoft Windows включають Windows 9x, Windows Mobile і Windows Phone. Спочатку Windows була всього лише графічною програмою-надбудовою для поширеної в 1980-х і 1990-х роках операційної системи MS-DOS.

**2.4 Опис користувацького інтерфейсу**

При запуску гри гравець має можливість вибрати запустити гру, переглянути правила гри, дізнатися контактну інформацію розробника, вийти з програми або переглянути рекордну таблицю з результатами попередніх ігор за допомогою кнопок «Грати», «Правила гри», «Про нас», «Вихід», «Досягнення».

Після натискання, з’являється нове вікно на екрані, де розташовані такі компоненти як кнопки виклику викидного меню, перезапуску гри, що знаходяться з лівого боку, кнопка повороту зображення, що розташована з правого боку поряд із компонентом, що відповідає за генерацію труби, поле з зображеннями всіх видів труб та поле 5 на 5, складене з окремих кнопок, де за замовчанням встановлені перша (з лівого верхнього краю) та остання (з правого нижнього краю) труби.

# Після закінчення гри з’являється вікно з сповіщенням про вдало складений водопровід та текстовим полем, куди користувача просять ввести ім’я та натиснути кнопку «ОК», розташовану під цим полем.

Розглянемо дії користувача при роботі з програмою. Користувач запускає гру. На екрані ПК з’являється головне вікно програми (рис.4).



Рисунок 4 – Головне вікно програми

Тут ми бачимо п’ять основних кнопок, одна для початку гри, інші три для перегляду різноманітної інформації про гру:

* Грати – кнопка переходу до вікна гри.
* Досягнення – кнопка переходу до вікна з таблицею результатів різних учасників гри.
* Правила гри – кнопка переходу до вікна з описаними правилами гри.
* Про нас – кнопка переходу до вікна з контактною інформацією.
* Вихід – кнопка виходу з гри.

# При натисненні на кнопку «Досягнення», ми переходимо до вікна з таблицею результатів (рис.5).

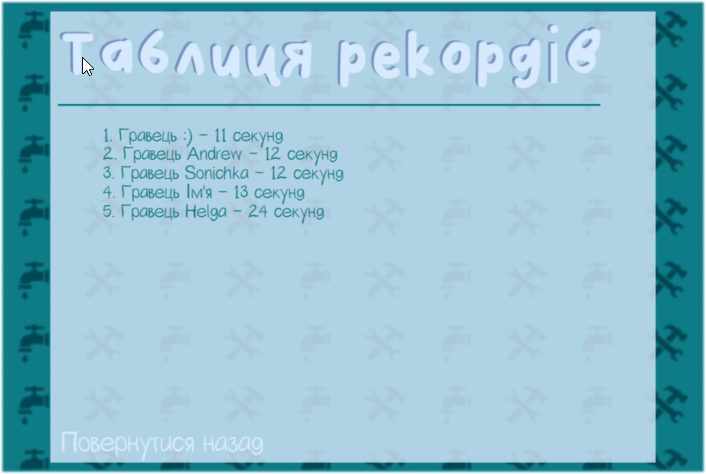


Рисунок 5 – Таблиця рекордів

В даному вікні користувач може переглянути результати проведених ігор, у порядку від найкращого часу завершення гри до найгіршого.

У випадку великої кількості даних, з’являється прокрутка, і гравець може прогорнути результати.

Передбачена кнопка «Повернутися назад», при натисненні якої гравець потрапляє знову в Головне меню програми.

# При натисненні на кнопку «Правила гри», відбувається перехід до вікна, де описані головні положення гри (рис.6).

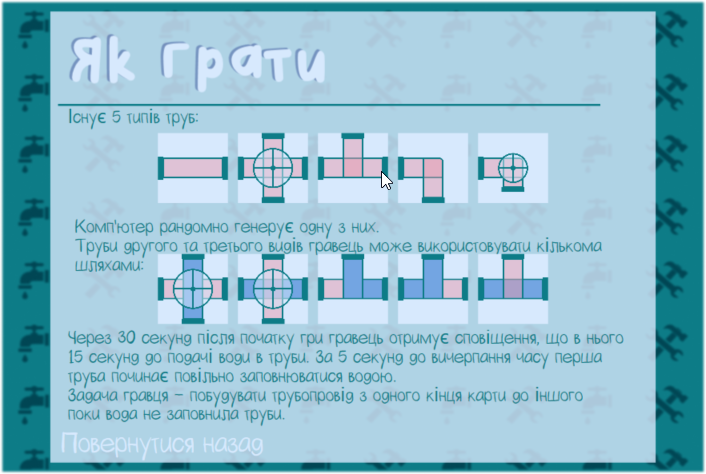


Рисунок 6 – Правила гри

В даному вікні користувачу доступні правила гри та показані основні види труб, а також яким чином вони можуть бути застосовані протягом гри.

Передбачена кнопка «Повернутися назад», при натисненні якої гравець потрапляє знову в Головне меню програми.

# При натисненні на кнопку «Про нас», користувач здійснює перехід до вікна з контактною інформацією про розробника (рис.7).

Передбачена кнопка «Повернутися назад», при натисненні якої гравець потрапляє знову в Головне меню програми.

При натисненні на кнопку «Вихід» відбувається вихід з програми.



Рисунок 7 – Контактна інформація

# При натисненні на кнопку «Грати», користувач здійснює перехід до вікна з ігровим полем (рис.8).

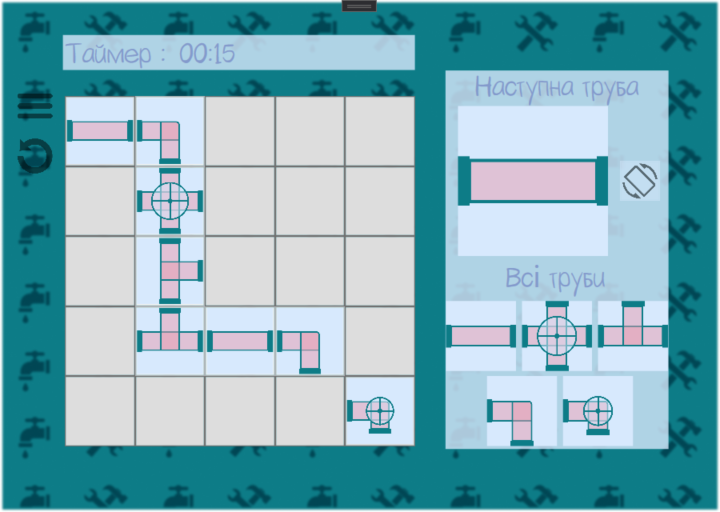


Рисунок 8 – Графічне відображення гри

## Переходячи в дане вікно, гравець починає гру, вмикається таймер та генерується нова труба.

## При натисненні на кнопку в лівому верхньому кутку ігрового поля викликається викидне меню та гра ставиться на паузу (рис.9).



Рисунок 9 – Викидне меню

## В викидному меню представлені кнопки початку нової гри, продовження попередньої, виходу в головне меню та загалом виходу з програми.

## При вдалому закінченні гри вводимо ім’я для зберігання результату (рис.10). При натисненні на кнопку «ОК», зберігається дані користувача.

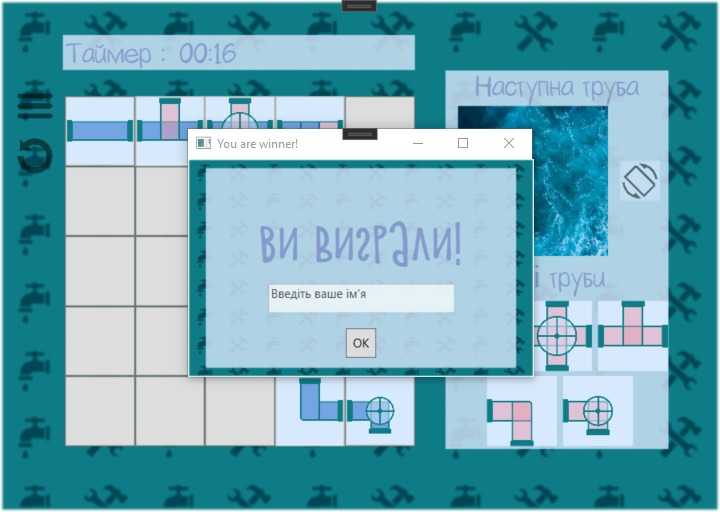


Рисунок 10 – Графічне відображення вікна для введення даних гравця при перемозі

**ВИСНОВКИ**

В результаті виконання курсового проекту розроблена гра «Водопровідник» під OC Windows.

Розроблене програмне забезпечення має наступні можливості:

1. Ставити трубу на відповідну кнопку.
2. Здійснити оберт труби праворуч на 90 градусів довільну кількість разів.
3. Введення ім’я гравця.
4. Зберігати та переглядати рекорди.

В даній роботі використовуються Windows SDK Tools – це пакет інструментів для розробників, призначення якого полягає в тестуванні додатків з можливістю їх налагодження в реальному часі, що значно спрощує розробку.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

## Metanit [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – metanit.com © 2012-2021. – Режим доступу: <https://metanit.com/sharp/wpf/> (дата звернення 27.05.2021) – Руководство по WPF.

## Microsoft [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – © Microsoft 2021. – Режим доступу: <https://docs.microsoft.com/> (дата звернення 01.06.2021) – What is WPF?

## WPF Tutorial [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – © wpf-tutorial.com 2007-2021. – Режим доступу: <https://wpf-tutorial.com/> (дата звернення 23.05.2021) – The complete WPF tutorial.

## StackOverflow [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – © 2021 Stack Exchange Inc. – Режим доступу: <https://stackoverflow.com/> (дата звернення 24.05.2021).

**ДОДАТКИ**

**ДОДАТОК А. Скролінг (текст) програми**

\*\*\*App.xaml\*\*\*

<Application x:Class="Plumber.App"

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

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

xmlns:local="clr-namespace:Plumber"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Style x:Key="Finger\_Paint">

<Setter Property="TextElement.FontFamily" Value="/Resources/fonts/#Finger Paint" />

</Style>

<Style x:Key="Barriecito">

<Setter Property="TextElement.FontFamily" Value="/Resources/fonts/#Barriecito" />

</Style>

<Style x:Key="Architects\_Daughter">

<Setter Property="TextElement.FontFamily" Value="/Resources/fonts/#Architects Daughter" />

</Style>

</Application.Resources>

</Application>

\*\*\*App.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace Plumber

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

}

}

\*\*\*MainWindow.xaml\*\*\*

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

mc:Ignorable="d"

KeyDown="myTestKey"

WindowStartupLocation="CenterScreen"

Title="Plumber" Height="512" Width="720">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<TextBlock x:Name="Plumber" Margin="0,0,0,280" Width="374" Height="111"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Finger Paint" FontStyle="Normal"

FontWeight="Normal" FontSize="75" LineHeight="110.5"

TextAlignment="Center" HorizontalAlignment="Center" Foreground="#EBE8E8">

<TextBlock.Effect>

<DropShadowEffect

ShadowDepth="4"

Direction="135"

Color="#0D7C87"

Opacity="0.5"

BlurRadius="0.5" />

</TextBlock.Effect>

PLUMBER

</TextBlock>

<Rectangle x:Name="MenuRectangle" Fill="#D7E9FD" Width="260"

Height="270" Margin="0,144,0,60" RadiusX="50px" RadiusY="50px">

</Rectangle>

<Line HorizontalAlignment="Center" VerticalAlignment="Center" X1="150" Y1="-10" Y2="-10" Stroke="#7084AF" StrokeThickness="1"/>

<Line HorizontalAlignment="Center" VerticalAlignment="Center" X1="150" Y1="-70" Y2="-70" Stroke="#7084AF" StrokeThickness="1"/>

<Button Name="Play" Margin="0, -80, 0, 0" Width="150"

Height="70" Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="Play\_Click">

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]" FontSize="65px"

FontWeight="Normal" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock.Effect>

<DropShadowEffect

ShadowDepth="2"

Direction="200"

Color="#0D7C87"

Opacity="0.3"

BlurRadius="0.5" />

</TextBlock.Effect>

Грати</TextBlock>

</Button>

<Button Name="Scores" Width="190" Height="50" Margin="0, 50, 0, 0"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Click="Scores\_Click">

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]" FontSize="40px"

FontWeight="Normal" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Top" >

Досягнення

</TextBlock>

</Button>

<Button Name="HowToPlay" Width="190" Height="50" Margin="0,140,0,0"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="HowToPlay\_Click">

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]" FontSize="35px"

FontWeight="Normal" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Top" >

Правила гри

</TextBlock>

</Button>

<Button Name="HelpBox" Width="190px" Height="50" Margin="0,230,0,0"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="HelpBox\_Click">

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]" FontSize="35px"

FontWeight="Normal" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Top" >

Про нас

</TextBlock>

</Button>

<Button Name="About" Width="190" Height="30" Margin="0,0,0,-330"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Click="Exit\_Click">

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]" FontSize="20px"

FontWeight="Normal" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Top">

В и х і д

</TextBlock>

</Button>

</Grid>

</Window>

\*\*\*MainWindow.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.IO;

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.Navigation;

using System.Windows.Shapes;

namespace Plumber

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.F1)

{

HowToPlay howToPlay = new HowToPlay();

howToPlay.Show();

this.Close();

}

else if (e.Key == Key.Escape)

{

this.Close();

}

}

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

{

GameField gameField = new GameField();

gameField.Show();

this.Close();

}

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

{

HowToPlay howToPlay = new HowToPlay();

howToPlay.Show();

this.Close();

}

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

{

HelpBox helpBox = new HelpBox();

helpBox.Show();

this.Close();

}

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

{

List<Gamer> gamers = new Gamer().ReadFile();

Scores scores = new Scores(gamers);

scores.Show();

this.Close();

}

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

{

this.Close();

}

}

}

\*\*\*Scores.xaml\*\*\*

<Window x:Class="Plumber.Scores"

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:Plumber"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

KeyDown="myTestKey"

Title="Plumber" Height="512" Width="720">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<Rectangle Width="605" Height="451.5" Margin="0,0,0,0" Fill="#D7E9FD" Opacity="0.8"/>

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]"

Width="600" Height="65" FontSize="80px"

FontWeight="Normal" Foreground="#D7E9FD"

Margin="15,-15,5,346">

<TextBlock.Effect>

<DropShadowEffect

ShadowDepth="3"

Direction="150"

Color="#839ACC"

Opacity="0.5"

BlurRadius="0.5" />

</TextBlock.Effect>

Таблиця рекордів

</TextBlock>

<Line X1="58" X2="600" Y1="105" Y2="105" StrokeThickness="2" Stroke="#0D7C87"/>

<Grid Name="Results" Height="280" Width="500" Margin="0,60,0,0">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<TextBlock Name="TextResult"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="20px" Foreground="#0D7C87"></TextBlock>

</ScrollViewer>

</Grid>

<Button Name="ComeBackButton" Width="231" Height="45" Margin="-35,390,324,-32"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="ComeBackButton\_Click">

<TextBlock Name="Exit" Width="231" Height="35" FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#D7E9FD">Повернутися назад</TextBlock>

</Button>

</Grid>

</Window>

\*\*\*Scores.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

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;

namespace Plumber

{

/// <summary>

/// Interaction logic for Scores.xaml

/// </summary>

public partial class Scores : Window

{

private List<Gamer> Players = new List<Gamer>();

public List<Gamer> players

{

get

{

return Players;

}

set

{

players = value;

}

}

public Scores(List<Gamer> gamers)

{

InitializeComponent();

Players = gamers;

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

{

TextResult.Text += $"{i + 1}. " + players[i].ToString() + "\n";

}

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

}

\*\*\*HelpBox.xaml\*\*\*

<Window x:Class="Plumber.HelpBox"

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:Plumber"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

KeyDown="myTestKey"

Title="Plumber" Height="512" Width="720">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<Rectangle Width="605" Height="451.5" Margin="0,0,0,0" Fill="#D7E9FD" Opacity="0.8"/>

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]o"

Width="300" Height="65" FontSize="80px"

FontWeight="Normal" Foreground="#D7E9FD"

Margin="8,-9,312,340">

<TextBlock.Effect>

<DropShadowEffect

ShadowDepth="3"

Direction="150"

Color="#839ACC"

Opacity="0.5"

BlurRadius="0.5" />

</TextBlock.Effect><Run Language="uk-ua" Text=" "/><Run Text="Про нас"/></TextBlock>

<Line X1="58" X2="600" Y1="105" Y2="105" StrokeThickness="2" Stroke="#0D7C87"/>

<TextBlock Width="325" Height="28" Margin="25,76,270,292"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="25px" Foreground="#0D7C87">

Маєш запитання? Повідом нас!

</TextBlock>

<Button Width="50" Height="50" Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Opacity="0.5" Margin="10,0,425,100">

<Image Source="D:\helga\university\programming\university\year\images\icons\email.png" Width="34" />

</Button>

<TextBlock Width="269.5" Height="28"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Architects Daughter"

FontSize="20px" Foreground="#0D7C87" Margin="153,133,198,235">

olgaburiak1207@gmail.com

</TextBlock>

<Button Width="50" Height="50" Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Opacity="0.5" Margin="10,100,425,60">

<Image Source="D:\helga\university\programming\university\year\images\icons\insta.png" Width="35" />

</Button>

<TextBlock Width="269.5" Height="28"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Architects Daughter"

FontSize="20px" Foreground="#0D7C87" Margin="153,204,198,164">

@bhelga\_\_

</TextBlock>

<Button Width="50" Height="50" Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Opacity="0.5" Margin="10,170,425,0">

<Image Source="D:\helga\university\programming\university\year\images\icons\telegram.png" Width="35" />

</Button>

<TextBlock Width="269.5" Height="28"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Architects Daughter"

FontSize="20px" Foreground="#0D7C87" Margin="153,269,198,99">

@k\_andela

</TextBlock>

<Button Name="ComeBackButton" Width="231" Height="45" Margin="5,430,364,2"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="ComeBackButton\_Click">

<TextBlock Name="Exit" Width="231" Height="55" FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#D7E9FD">Повернутися назад</TextBlock>

</Button>

</Grid>

</Window>

\*\*\*HelpBox.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

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;

namespace Plumber

{

/// <summary>

/// Interaction logic for HelpBox.xaml

/// </summary>

public partial class HelpBox : Window

{

public HelpBox()

{

InitializeComponent();

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

}

\*\*\*HowToPlay.xaml\*\*\*

<Window x:Class="Plumber.HowToPlay"

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:Plumber"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

KeyDown="myTestKey"

Title="Plumber" Height="512" Width="720">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<Rectangle Width="605" Height="451.5" Margin="0,0,0,0" Fill="#D7E9FD" Opacity="0.8"/>

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Icegirl [rus by aLiNcE]"

Width="300" Height="65" FontSize="80px"

FontWeight="Normal" Foreground="#D7E9FD"

Margin="25,-10,295,341">

<TextBlock.Effect>

<DropShadowEffect

ShadowDepth="3"

Direction="150"

Color="#839ACC"

Opacity="0.5"

BlurRadius="0.5" />

</TextBlock.Effect>

Як грати

</TextBlock>

<Line Name="Line1" X1="58" X2="600" Y1="105" Y2="105" StrokeThickness="2" Stroke="#0D7C87"/>

<TextBlock Width="269.5" Height="28" Margin="25,70,326,298"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="20px" Foreground="#0D7C87">

Існує 5 типів труб:

</TextBlock>

<UniformGrid Width="400" Height="70" Rows="1" Columns="5" Margin="0,100,0,238" HorizontalAlignment="Center">

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube1\_1.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube4\_4.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube5\_5.png"/>

</UniformGrid>

<TextBlock Width="561" Height="140" Margin="75,219,84,137"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="20px" Foreground="#0D7C87">

Комп'ютер рандомно генерує одну з них.<LineBreak/>

Труби другого та третього видів гравець може використовувати кількома <LineBreak/>

шляхами:

</TextBlock>

<UniformGrid Width="400" Height="70" Rows="1" Columns="5" Margin="-50,170,-50,66" HorizontalAlignment="Center">

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2\_filled1.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2\_filled2.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled1.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled2.png"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled3.png"/>

</UniformGrid>

<TextBlock Width="561" Height="140" Margin="25,290,34,-36"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="20px" Foreground="#0D7C87">

Через 30 секунд після початку гри гравець отримує сповіщення, що в нього залишилося <LineBreak/>

15 секунд до подачі води в труби. За 5 секунд до вичерпання часу перша <LineBreak/>

труба починає повільно заповнюватися водою. <LineBreak/>

Задача гравця – побудувати трубопровід з одного кінця карти до іншого<LineBreak/>

поки вода не заповнила труби.<LineBreak/>

</TextBlock>

<Button Name="ComeBackButton" Width="231" Height="35" Margin="-35,390,324,-28"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Click="ComeBackButton\_Click">

<TextBlock Name="Exit" Width="231" Height="35" FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#D7E9FD">Повернутися назад</TextBlock>

</Button>

</Grid>

</Window>

\*\*\*HowToPlay.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

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;

namespace Plumber

{

/// <summary>

/// Interaction logic for HowToPlay.xaml

/// </summary>

public partial class HowToPlay : Window

{

public HowToPlay()

{

InitializeComponent();

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

}

\*\*\*WinWindow.xaml\*\*\*

<Window x:Class="Plumber.WinWindow"

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:Plumber"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

Title="You are winner!" Height="256" Width="360">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<Rectangle Width="310" Height="200" Margin="0,0,0,0" Fill="#D7E9FD" Opacity="0.8"/>

<TextBlock FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Shnobel" FontSize="50px"

FontWeight="Regular" TextAlignment="Center" Foreground="#839ACC"

HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,60,0,0">Ви виграли!</TextBlock>

<TextBox Name="GamerName" Width="200" Height="50" Margin="80,125,80,65" MaxLength="100"

TextChanged="GamerName\_TextChanged"

GotFocus="GamerName\_GotFocus" KeyDown="GamerName\_KeyDown" Opacity="0.7">Введіть ваше ім'я</TextBox>

<Button Name="OK" Width="30" Height="30" Margin="115,130,115,-20"

Click="OK\_Click">OK</Button>

</Grid>

</Window>

\*\*\*WinWindow.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

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;

namespace Plumber

{

/// <summary>

/// Interaction logic for WinWindow.xaml

/// </summary>

public partial class WinWindow : Window

{

private TextBox textBox;

public WinWindow()

{

InitializeComponent();

}

private void GamerName\_TextChanged(object sender, TextChangedEventArgs e)

{

textBox = (TextBox)sender;

}

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

{

try

{

using (StreamWriter sw = new StreamWriter(@"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\scoreTable.txt", true, Encoding.Default))

{

sw.WriteLine(textBox.Text);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

this.DialogResult = true;

}

private void GamerName\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

try

{

using (StreamWriter sw = new StreamWriter(@"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\scoreTable.txt", true, Encoding.Default))

{

sw.WriteLine(textBox.Text);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

this.DialogResult = true;

}

}

private void GamerName\_GotFocus(object sender, RoutedEventArgs e)

{

TextBox tb = (TextBox)sender;

tb.Text = string.Empty;

tb.GotFocus -= GamerName\_GotFocus;

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{

try

{

using (StreamWriter sw = new StreamWriter(@"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\scoreTable.txt", true, Encoding.Default))

{

sw.WriteLine("Random Guy");

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

this.DialogResult = true;

}

}

}

}

\*\*\*GameField.xaml\*\*\*

<Window x:Class="Plumber.GameField"

Name="ThisGameField"

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:Plumber"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

KeyDown="myTestKey"

Title="Plumber" Height="512" Width="720"

ResizeMode="NoResize" WindowStyle="None">

<UserControl>

<UserControl.Resources>

<Image x:Key="resourceImage" Source="/whiteScreen.png"/>

</UserControl.Resources>

<Grid>

<Grid.Background>

<ImageBrush ImageSource="D:\helga\university\programming\university\year\images\background\Background2.png" Stretch="Fill"/>

</Grid.Background>

<Rectangle Width="352" Height="35" Margin="40.5,10.5,282.75,417.5" Fill="#D7E9FD" Opacity="0.8"/>

<TextBlock Name="TextTimer" Width="164.5" Height="40" Margin="65,42,490,430"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#839ACC">Таймер :</TextBlock>

<Label Name="LTimer" Content="00:00" FontSize="30px" Foreground="#839ACC"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\\#Yahfie"

Width="100.5" Height="40" Margin="174,37,445,435" RenderTransformOrigin="0.607,0.438"></Label>

<Rectangle Width="223" Height="378.5" Margin="405.5,12.5,11.5,5.04" Fill="#D7E9FD" Opacity="0.8"/>

<Button Name="MenuBarButton" Width="50" Height="50" Margin="-650,-300,0,0"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}"

Opacity="0.5" Click="MenuBarButton\_Click">

<Image Source="D:\helga\university\programming\university\year\images\icons\menu.png" Width="40"/>

</Button>

<Button Width="50" Height="50" Margin="-650,-200,0,0"

Style="{StaticResource {x:Static ToolBar.ButtonStyleKey}}" Opacity="0.5"

Click="Replay\_Click">

<Image Source="D:\helga\university\programming\university\year\images\icons\replay.png" Width="35" />

</Button>

<UniformGrid x:Name="GamingField" Width="350" Height="350" Rows="5" Columns="5" Margin="-240,30,0,0">

<Button Grid.Row="0" Grid.Column="0" x:Name="FirstButton">

<Grid>

<Image Name="FirstTubeImage" Source="/Tube1\_1.png" Visibility="Visible"/>

<MediaElement Name="FirstTubeVideo" Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\filing1.gif" Visibility="Hidden" LoadedBehavior="Manual" UnloadedBehavior="Stop"/>

</Grid>

</Button>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button Click="Button\_Click"/>

<Button x:Name="LastButton">

<Image Name="LastTubeImage" Source="/Tube5\_5.png"/>

</Button>

</UniformGrid>

<Grid Name="NextTube" Width="150" Height="150"

Margin="408,48,62,198">

<Image x:Name="NewTubeImage" RenderTransformOrigin="0.5,0.5" Source="/water.jpg"/>

</Grid>

<Button Name="Rotation" Width="40" Height="40" Margin="550,80,-10,230" Background="#D7E9FD"

BorderBrush="#D7E9FD" Opacity="0.5" Click="Rotation\_Click">

<Image Source="D:\helga\university\programming\university\year\images\icons\rotation.png" Width="35"/>

</Button>

<TextBlock Name="NewTubeText" Width="164.5" Height="35" Margin="475,75,81,402"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#839ACC">Наступна труба</TextBlock>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube1\_1.png"

Width="70" Height="70" Margin="396,243,154,83"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2.png"

Width="70" Height="70" Margin="472,243,78,83"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3.png"

Width="70" Height="70" Margin="548,243,2,83"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube4\_4.png"

Width="70" Height="70" Margin="437,318,113,8"/>

<Image Source="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube5\_5.png"

Width="70" Height="70" Margin="513,318,37,8"/>

<TextBlock Name="AllTubes" Height="35" Margin="505,266,111,211"

FontFamily="D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\fonts\#Yahfie"

FontSize="30px" Foreground="#839ACC">Всі труби</TextBlock>

</Grid>

</UserControl>

</Window>

\*\*\*GameField.xaml.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Text;

using System.Threading;

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 System.Windows.Threading;

namespace Plumber

{

/// <summary>

/// Interaction logic for GameField.xaml

/// </summary>

public partial class GameField : Window

{

private double[] Fields = new double[] { -2585.0000000000005, -391.6, -2865.0000000000005 - 70, -671.6 - 70 };

private string FinalPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\water.jpg";

private string FailPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\FailGame.png";

private string ScoreTablePath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\scoreTable.txt";

private bool StartGame = true;

private bool GameOver = false;

private bool Win = false;

private int \_tube2Counter = 0;

private int \_tube3Counter = 0;

private int Timer = 0;

DispatcherTimer timer;

private List<Image> images = new List<Image>();

private int Rotate = 0;

private GameObject currentTube;

private RotateTransform rotateTransform;

private List<GameObject> Tubes = new List<GameObject>();

GameObject FirstTube;

GameObject LastTube;

GameObject lastChainTube;

public Window MyMainWindow { get; set; }

public GameField()

{

InitializeComponent();

Tubes.Clear();

SetTimer();

SetNewTube();

}

private void SetStartTubes()

{

LastTube = new Tube5() { GameObjectJointCoordinate = new List<double>() { LastButton.PointFromScreen(new Point(0,0)).X },

GameObjectPlace = new GameObjectPlace(LastButton.PointFromScreen(new Point(0, 0)).X, LastButton.PointFromScreen(new Point(0, 0)).Y) };

Tubes.Add(LastTube);

FirstTube = new Tube1() { GameObjectJointCoordinate = new List<double>() { FirstButton.PointFromScreen(new Point(0, 0)).X - 70},

GameObjectPlace = new GameObjectPlace(FirstButton.PointFromScreen(new Point(0, 0)).X, FirstButton.PointFromScreen(new Point(0, 0)).Y) };

FirstTube.AmountOfJoint--;

Tubes.Add(FirstTube);

images.Add(FirstTubeImage);

lastChainTube = FirstTube;

Fields = new double[] { FirstButton.PointFromScreen(new Point(0, 0)).X, FirstButton.PointFromScreen(new Point(0, 0)).Y,

LastButton.PointFromScreen(new Point(0, 0)).X - 70, LastButton.PointFromScreen(new Point(0, 0)).Y - 70 };

}

private void myTestKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.F1)

{

timer.Stop();

MenuBar menuBar = new MenuBar();

menuBar.MyMainWindow = this;

menuBar.ShowDialog();

if (menuBar.DialogResult == true)

{

timer.Start();

}

}

else if (e.Key == Key.Escape)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

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

{

timer.Stop();

GameField gameField = new GameField();

gameField.Show();

this.Close();

}

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

{

timer.Stop();

MenuBar menuBar = new MenuBar();

menuBar.MyMainWindow = this;

menuBar.ShowDialog();

if (menuBar.DialogResult == true)

{

timer.Start();

}

}

public void SetTimer()

{

timer = new DispatcherTimer();

timer.Tick += new EventHandler(timer\_Tick);

timer.Interval = TimeSpan.FromSeconds(1);

timer.Start();

}

private void timer\_Tick(object sender, EventArgs e)

{

Timer++;

timerCheck();

TimeSpan time = TimeSpan.FromSeconds(Timer);

DateTime dateTime = DateTime.Today.Add(time);

string displayTime = dateTime.ToString("mm:ss");

LTimer.Content = displayTime;

}

private void timerCheck()

{

if (Timer == 30)

{

TextTimer.Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#0D7C87"));

LTimer.Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#0D7C87"));

MessageBox.Show("Зараз ти маєш 15 секунд, щоб скласти водопровід!");

}

else if (Timer == 41)

{

FirstTubeVideo.Visibility = Visibility.Visible;

FirstTubeVideo.Play();

}

else if (Timer == 45)

{

if (StartGame == true)

{

SetStartTubes();

StartGame = false;

}

FirstTubeImage.Source = new BitmapImage(new Uri(FirstTube.WinPath));

FirstTubeVideo.Visibility = Visibility.Hidden;

FirstTubeVideo.Pause();

GameOver = true;

Win = false;

FinishTheGame();

}

}

private void SetNewTube()

{

if (GameOver == false && StartGame == false) IsGameOver();

Random random = new Random();

int randomValue = random.Next(1, 5);

Uri uri;

BitmapImage bitmap = new BitmapImage();

if (GameOver == false)

{

switch (randomValue)

{

case 1:

currentTube = new Tube1();

break;

case 2:

if (\_tube2Counter < 1)

{

currentTube = new Tube2();

\_tube2Counter++;

}

else goto case 1;

break;

case 3:

if (\_tube3Counter < 2)

{

currentTube = new Tube3();

\_tube3Counter++;

}

else goto case 4;

break;

case 4:

currentTube = new Tube4();

break;

default:

MessageBox.Show("Error!");

break;

}

uri = new Uri(currentTube.Path);

bitmap = new BitmapImage(uri);

NewTubeImage.Source = bitmap;

}

else

{

FinishTheGame();

}

if (currentTube.gameObjectType == GameObjectType.Tube2)

{

Rotation.IsEnabled = false;

}

else

{

Rotation.IsEnabled = true;

}

}

private void ShowWater(GameObject tube, Image image)

{

tube.SetWinPath();

Uri uri = new Uri(tube.WinPath);

BitmapImage bitmap = new BitmapImage(uri);

image.Source = bitmap;

}

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

{

if (StartGame == true)

{

SetStartTubes();

StartGame = false;

}

if (((Button)sender).IsEnabled == true && IsAvailableButton(sender, e) == true)

{

Grid grid = new Grid();

var uri = new Uri(currentTube.Path);

var bitmap = new BitmapImage(uri);

Image image = new Image();

image.Source = bitmap;

image.RenderTransformOrigin = new Point(0.5, 0.5);

image.RenderTransform = rotateTransform;

grid.Children.Add(image);

((Button)sender).Content = grid;

((Button)sender).IsEnabled = false;

Point location = ((Button)sender).PointFromScreen(new Point(0, 0));

currentTube.GameObjectPlace = new GameObjectPlace(location.X, location.Y);

SetJointCoordinates(location);

currentTube.ObjRotation = Rotate;

Tubes.Add(currentTube);

rotateTransform = new RotateTransform(0);

NewTubeImage.RenderTransform = rotateTransform;

Rotate = 0;

lastChainTube = currentTube;

images.Add(image);

SetNewTube();

}

else

{

MessageBox.Show("Ця клітинка недоступна!");

}

}

private bool IsAvailableButton(object sender, RoutedEventArgs e)

{

if (lastChainTube.gameObjectType == GameObjectType.Tube2)

{

if (lastChainTube.GameObjectJoint[0] == 2)

{

if ((((Button)sender).PointFromScreen(new Point(0, 0)).X - lastChainTube.GameObjectPlace.XCoordinate == -70 && ((Button)sender).PointFromScreen(new Point(0, 0)).Y == lastChainTube.GameObjectPlace.YCoordinate))

{

if (currentTube.GameObjectJoint[0] == 1)

{

currentTube.GameObjectJoint[0]++;

lastChainTube.GameObjectJoint[2]++;

return true;

}

}

else

{

return false;

}

}

if (lastChainTube.GameObjectJoint[1] == 2)

{

if ((((Button)sender).PointFromScreen(new Point(0, 0)).Y - lastChainTube.GameObjectPlace.YCoordinate == -70 && ((Button)sender).PointFromScreen(new Point(0, 0)).X == lastChainTube.GameObjectPlace.XCoordinate))

{

if (currentTube.GameObjectJoint[1] == 1)

{

currentTube.GameObjectJoint[1]++;

lastChainTube.GameObjectJoint[3]++;

return true;

}

}

else

{

return false;

}

}

if (lastChainTube.GameObjectJoint[2] == 2)

{

if ((((Button)sender).PointFromScreen(new Point(0, 0)).X - lastChainTube.GameObjectPlace.XCoordinate == 70 && ((Button)sender).PointFromScreen(new Point(0, 0)).Y == lastChainTube.GameObjectPlace.YCoordinate))

{

if (currentTube.GameObjectJoint[2] == 1)

{

currentTube.GameObjectJoint[2]++;

lastChainTube.GameObjectJoint[0]++;

return true;

}

}

else

{

return false;

}

}

if (lastChainTube.GameObjectJoint[3] == 2)

{

if ((((Button)sender).PointFromScreen(new Point(0, 0)).Y - lastChainTube.GameObjectPlace.YCoordinate == 70 && ((Button)sender).PointFromScreen(new Point(0, 0)).X == lastChainTube.GameObjectPlace.XCoordinate))

{

if (currentTube.GameObjectJoint[3] == 1)

{

currentTube.GameObjectJoint[3]++;

lastChainTube.GameObjectJoint[1]++;

return true;

}

}

else

{

return false;

}

}

}

else if ((((Button)sender).PointFromScreen(new Point(0, 0)).X - lastChainTube.GameObjectPlace.XCoordinate == 70 && ((Button)sender).PointFromScreen(new Point(0, 0)).Y == lastChainTube.GameObjectPlace.YCoordinate))

{

if (currentTube.GameObjectJoint[2] == 1 && lastChainTube.GameObjectJoint[0] == 1)

{

if (currentTube.gameObjectType == GameObjectType.Tube2)

{

currentTube.GameObjectJoint[1]--;

currentTube.GameObjectJoint[3]--;

}

currentTube.GameObjectJoint[2]++;

lastChainTube.GameObjectJoint[0]++;

return true;

}

}

else if (((Button)sender).PointFromScreen(new Point(0, 0)).X - lastChainTube.GameObjectPlace.XCoordinate == -70 && (((Button)sender).PointFromScreen(new Point(0, 0)).Y == lastChainTube.GameObjectPlace.YCoordinate))

{

if (currentTube.GameObjectJoint[0] == 1 && lastChainTube.GameObjectJoint[2] == 1)

{

if (currentTube.gameObjectType == GameObjectType.Tube2)

{

currentTube.GameObjectJoint[1]--;

currentTube.GameObjectJoint[3]--;

}

currentTube.GameObjectJoint[0]++;

lastChainTube.GameObjectJoint[2]++;

return true;

}

}

else if (((Button)sender).PointFromScreen(new Point(0, 0)).Y - lastChainTube.GameObjectPlace.YCoordinate == 70 && ((Button)sender).PointFromScreen(new Point(0, 0)).X == lastChainTube.GameObjectPlace.XCoordinate)

{

if (currentTube.GameObjectJoint[3] == 1 && lastChainTube.GameObjectJoint[1] == 1)

{

if (currentTube.gameObjectType == GameObjectType.Tube2)

{

currentTube.GameObjectJoint[0]--;

currentTube.GameObjectJoint[2]--;

}

currentTube.GameObjectJoint[3]++;

lastChainTube.GameObjectJoint[1]++;

return true;

}

}

else if (((Button)sender).PointFromScreen(new Point(0, 0)).Y - lastChainTube.GameObjectPlace.YCoordinate == -70 && (((Button)sender).PointFromScreen(new Point(0, 0)).X == lastChainTube.GameObjectPlace.XCoordinate))

{

if (currentTube.GameObjectJoint[1] == 1 && lastChainTube.GameObjectJoint[3] == 1)

{

if (currentTube.gameObjectType == GameObjectType.Tube2)

{

currentTube.GameObjectJoint[0]--;

currentTube.GameObjectJoint[2]--;

}

currentTube.GameObjectJoint[1]++;

lastChainTube.GameObjectJoint[3]++;

return true;

}

}

return false;

}

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

{

currentTube.SetNewJoint();

Rotate++;

switch (Rotate)

{

case 0:

rotateTransform = new RotateTransform(0);

break;

case 1:

rotateTransform = new RotateTransform(90);

break;

case 2:

rotateTransform = new RotateTransform(180);

break;

case 3:

rotateTransform = new RotateTransform(270);

break;

default:

Rotate = 0;

goto case 0;

}

NewTubeImage.RenderTransform = rotateTransform;

}

private void SetJointCoordinates(Point point)

{

if (currentTube.GameObjectJoint[0] == 1)

{

currentTube.GameObjectJointCoordinate.Add(point.X);

}

if (currentTube.GameObjectJoint[1] == 1)

{

currentTube.GameObjectJointCoordinate.Add(point.Y);

}

if (currentTube.GameObjectJoint[2] == 1)

{

currentTube.GameObjectJointCoordinate.Add(point.X - 70);

}

if (currentTube.GameObjectJoint[3] == 1)

{

currentTube.GameObjectJointCoordinate.Add(point.Y - 70);

}

}

private async void FinishTheGame()

{

Tubes.RemoveAt(0);

var uri = new Uri(FinalPath);

var bitmap = new BitmapImage(uri);

NewTubeImage.Source = bitmap;

if (Win == true)

{

Tubes.Add(LastTube);

images.Add(LastTubeImage);

lastChainTube.GameObjectJoint[2]++;

int i = 0;

foreach (var tube in Tubes)

{

await Task.Delay(1000);

if (i == images.Count - 1)

{

ShowWater(tube, images[i]);

await Task.Delay(2000);

SaveInformation();

}

else

{

ShowWater(tube, images[i]);

}

i++;

}

}

else

{

timer.Stop();

int i = 0;

foreach (var tube in Tubes)

{

await Task.Delay(1000);

if (i == images.Count - 1)

{

ShowWater(tube, images[i]);

await Task.Delay(1000);

uri = new Uri(FailPath);

bitmap = new BitmapImage(uri);

images[i].Source = bitmap;

await Task.Delay(2000);

MessageBox.Show("Гру програно!!!");

}

else

{

ShowWater(tube, images[i]);

}

i++;

}

}

}

private void IsGameOver()

{

foreach (var coord in lastChainTube.GameObjectJointCoordinate)

{

if (coord == LastTube.GameObjectJointCoordinate[0] && lastChainTube.GameObjectPlace.YCoordinate == LastTube.GameObjectPlace.YCoordinate)

{

GameOver = true;

Win = true;

timer.Stop();

}

else

{

int counter = 0;

int j = 0;

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

{

if (lastChainTube.GameObjectJoint[i] == 1)

{

counter++;

j = i;

}

}

if (counter == 1 || lastChainTube.gameObjectType == GameObjectType.Tube3 || lastChainTube.gameObjectType == GameObjectType.Tube2)

{

counter = 0;

foreach (var jcoord in lastChainTube.GameObjectJointCoordinate)

{

foreach (var field in Fields)

{

if (jcoord == field)

{

if (lastChainTube.gameObjectType == GameObjectType.Tube3)

{

if ((lastChainTube.GameObjectPlace.XCoordinate == field && lastChainTube.GameObjectJoint[0] == 1)||

(lastChainTube.GameObjectPlace.XCoordinate - 70 == field && lastChainTube.GameObjectJoint[2] == 1) ||

(lastChainTube.GameObjectPlace.YCoordinate == field && lastChainTube.GameObjectJoint[1] == 1) ||

(lastChainTube.GameObjectPlace.YCoordinate - 70 == field && lastChainTube.GameObjectJoint[3] == 1))

{

counter++;

}

}

else if (lastChainTube.gameObjectType == GameObjectType.Tube2)

{

if ((lastChainTube.GameObjectPlace.XCoordinate == field && lastChainTube.GameObjectJoint[0] == 1) ||

(lastChainTube.GameObjectPlace.XCoordinate - 70 == field && lastChainTube.GameObjectJoint[2] == 1) ||

(lastChainTube.GameObjectPlace.YCoordinate == field && lastChainTube.GameObjectJoint[1] == 1) ||

(lastChainTube.GameObjectPlace.YCoordinate - 70 == field && lastChainTube.GameObjectJoint[3] == 1) ||

(lastChainTube.GameObjectPlace.XCoordinate == field && lastChainTube.GameObjectJoint[0] == 0) ||

(lastChainTube.GameObjectPlace.XCoordinate - 70 == field && lastChainTube.GameObjectJoint[2] == 0) ||

(lastChainTube.GameObjectPlace.YCoordinate == field && lastChainTube.GameObjectJoint[1] == 0) ||

(lastChainTube.GameObjectPlace.YCoordinate - 70 == field && lastChainTube.GameObjectJoint[3] == 0))

{

counter++;

}

}

else if (lastChainTube.gameObjectType == GameObjectType.Tube2)

{

if (lastChainTube.GameObjectJoint[0] == 2)

lastChainTube.GameObjectJoint[2] = 2;

else if (lastChainTube.GameObjectJoint[1] == 2)

lastChainTube.GameObjectJoint[3] = 2;

else if (lastChainTube.GameObjectJoint[2] == 2)

lastChainTube.GameObjectJoint[0] = 2;

else if (lastChainTube.GameObjectJoint[3] == 2)

lastChainTube.GameObjectJoint[1] = 2;

}

else

{

GameOver = true;

break;

}

}

}

if (counter == 2 || (counter == 1 && lastChainTube.gameObjectType == GameObjectType.Tube2))

{

GameOver = true;

break;

}

}

foreach (var tube in Tubes)

{

if ((counter == 0 && lastChainTube.gameObjectType != GameObjectType.Tube3) || counter == 1)

{

if (j == 0)

{

if (lastChainTube.GameObjectPlace.XCoordinate == tube.GameObjectPlace.XCoordinate - 70 && lastChainTube.GameObjectPlace.YCoordinate == tube.GameObjectPlace.YCoordinate)

{

GameOver = true;

break;

}

}

else if (j == 1)

{

if (lastChainTube.GameObjectPlace.YCoordinate == tube.GameObjectPlace.YCoordinate - 70 && lastChainTube.GameObjectPlace.XCoordinate == tube.GameObjectPlace.XCoordinate)

{

GameOver = true;

break;

}

}

else if (j == 2)

{

if (lastChainTube.GameObjectPlace.XCoordinate - 70 == tube.GameObjectPlace.XCoordinate && lastChainTube.GameObjectPlace.YCoordinate == tube.GameObjectPlace.YCoordinate)

{

GameOver = true;

break;

}

}

else if (j == 3)

{

if (lastChainTube.GameObjectPlace.YCoordinate - 70 == tube.GameObjectPlace.YCoordinate && lastChainTube.GameObjectPlace.XCoordinate == tube.GameObjectPlace.XCoordinate)

{

GameOver = true;

break;

}

}

}

}

}

}

}

}

private void SaveInformation()

{

try

{

using (StreamWriter sw = new StreamWriter(ScoreTablePath, true, Encoding.Default))

{

sw.WriteLine(Timer);

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

WinWindow winWindow = new WinWindow();

winWindow.ShowDialog();

}

\*\*\*GameObject.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Windows;

using System.Text;

namespace Plumber

{

abstract class GameObject

{

private GameObjectPlace gameObjectPlace;

public GameObjectPlace GameObjectPlace

{

get

{

return gameObjectPlace;

}

set

{

gameObjectPlace = value;

}

}

private int[] gameObjectJoint = new int[] { 0, 0, 0, 0 };

public int[] GameObjectJoint

{

get

{

return gameObjectJoint;

}

set

{

gameObjectJoint = value;

}

}

private int objRotatin;

public int ObjRotation

{

get

{

return objRotatin;

}

set

{

objRotatin = value;

}

}

private List<double> gameObjectJointCoordinate = new List<double>();

public List<double> GameObjectJointCoordinate

{

get

{

return gameObjectJointCoordinate;

}

set

{

gameObjectJointCoordinate = value;

}

}

private int amountOfJoint;

public int AmountOfJoint

{

get

{

return amountOfJoint;

}

set

{

amountOfJoint = value;

}

}

private string path;

public string Path

{

get

{

return path;

}

set

{

path = value;

}

}

private string winPath;

public string WinPath

{

get

{

return winPath;

}

set

{

winPath = value;

}

}

private GameObjectType GameObjectType;

public GameObjectType gameObjectType

{

get

{

return GameObjectType;

}

set

{

GameObjectType = value;

}

}

public virtual void SetNewJoint()

{

int[] temp = new int[4];

for (int i = 1; i < GameObjectJoint.Length; i++)

{

temp[i] = GameObjectJoint[i - 1];

if (i == 1)

{

temp[0] = GameObjectJoint[GameObjectJoint.Length - 1];

}

}

GameObjectJoint = temp;

}

public abstract void SetWinPath();

}

}

\*\*\*GameObjectPlace.cs \*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class GameObjectPlace

{

private double xCoordinate;

public double XCoordinate

{

get

{

return xCoordinate;

}

set

{

xCoordinate = value;

}

}

private double yCoordinate;

public double YCoordinate

{

get

{

return yCoordinate;

}

set

{

yCoordinate = value;

}

}

public GameObjectPlace(double xCoordinate, double yCoordinate)

{

XCoordinate = xCoordinate;

YCoordinate = yCoordinate;

}

}

}

\*\*\*GameObjectType.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

enum GameObjectType

{

Tube1 = 1,

Tube2,

Tube3,

Tube4,

Tube5

}

}

\*\*\*Tube1.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class Tube1 : GameObject

{

public Tube1()

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube1\_1\_filled.png";

Path = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube1\_1.png";

gameObjectType = GameObjectType.Tube1;

GameObjectJoint = new int[] { 1, 0, 1, 0 };

AmountOfJoint = 2;

}

public override void SetNewJoint()

{

base.SetNewJoint();

}

public override void SetWinPath()

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube1\_1\_filled.png";

}

}

}

\*\*\*Tube2.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class Tube2 : GameObject

{

public Tube2()

{

Path = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2.png";

gameObjectType = GameObjectType.Tube2;

GameObjectJoint = new int[] { 1, 1, 1, 1 };

AmountOfJoint = 4;

}

public override void SetNewJoint()

{

base.SetNewJoint();

}

public override void SetWinPath()

{

if (GameObjectJoint[0] == 2 && GameObjectJoint[2] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2\_filled2.png";

}

else if (GameObjectJoint[1] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2\_filled1.png";

}

else

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube2\_2\_filled.png";

}

}

}

}

\*\*\*Tube3.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class Tube3 : GameObject

{

public Tube3()

{

Path = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3.png";

gameObjectType = GameObjectType.Tube3;

GameObjectJoint = new int[] { 1, 1, 1, 0 };

AmountOfJoint = 3;

}

public override void SetNewJoint()

{

base.SetNewJoint();

}

public override void SetWinPath()

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled.png";

if (ObjRotation == 0)

{

if (GameObjectJoint[1] == 2 && GameObjectJoint[2] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled1.png";

}

else if (GameObjectJoint[1] == 2 && GameObjectJoint[0] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled2.png";

}

else if (GameObjectJoint[0] == 2 && GameObjectJoint[2] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled3.png";

}

}

else if (ObjRotation == 1)

{

if (GameObjectJoint[0] == 2 && GameObjectJoint[1] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled1.png";

}

else if (GameObjectJoint[1] == 2 && GameObjectJoint[2] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled2.png";

}

else if (GameObjectJoint[1] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled3.png";

}

}

else if (ObjRotation == 2)

{

if (GameObjectJoint[0] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled1.png";

}

else if (GameObjectJoint[2] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled2.png";

}

else if (GameObjectJoint[0] == 2 && GameObjectJoint[2] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled3.png";

}

}

else if (ObjRotation == 3)

{

if (GameObjectJoint[0] == 2 && GameObjectJoint[1] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled1.png";

}

else if (GameObjectJoint[0] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled2.png";

}

else if (GameObjectJoint[1] == 2 && GameObjectJoint[3] == 2)

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube3\_3\_filled3.png";

}

}

}

}

}

\*\*\*Tube4.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class Tube4 : GameObject

{

public Tube4()

{

Path = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube4\_4.png";

gameObjectType = GameObjectType.Tube4;

GameObjectJoint = new int[] { 1, 0, 0, 1 };

AmountOfJoint = 2;

}

public override void SetNewJoint()

{

base.SetNewJoint();

}

public override void SetWinPath()

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube4\_4\_filled.png";

}

}

}

\*\*\*Tube5.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

class Tube5 : GameObject

{

public Tube5()

{

Path = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube5\_5.png";

gameObjectType = GameObjectType.Tube5;

GameObjectJoint = new int[] { 1, 0, 0, 0 };

AmountOfJoint = 1;

}

public override void SetNewJoint()

{

base.SetNewJoint();

}

public override void SetWinPath()

{

WinPath = @"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\images\tubes\Tube5\_5\_filled.png";

}

}

}

\*\*\*IGamer.cs\*\*\*

using System;

using System.Collections.Generic;

using System.Text;

namespace Plumber

{

interface IGamer

{

List<Gamer> ReadFile();

}

}

\*\*\*Gamer\*\*\*

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Plumber

{

public struct Gamer : IComparable, IGamer

{

private string name;

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

private int amountOfSeconds;

public int AmountOfSeconds

{

get

{

return amountOfSeconds;

}

set

{

amountOfSeconds = value;

}

}

public int CompareTo(object obj)

{

if (obj == null) return 1;

Gamer otherGamer = (Gamer)obj;

if (otherGamer.Equals(null) != true)

return this.AmountOfSeconds.CompareTo(otherGamer.AmountOfSeconds);

else

throw new ArgumentException("Object is not a Temperature");

}

public List<Gamer> ReadFile()

{

List<Gamer> gamers = new List<Gamer>();

Gamer temp = new Gamer();

FileStream fs = new FileStream(@"D:\helga\university\programming\university\year\PlumberWPF\Plumber\Resources\scoreTable.txt", FileMode.OpenOrCreate);

using (StreamReader sr = new StreamReader(fs))

{

string line;

int i = 3;

while ((line = sr.ReadLine()) != null)

{

if (i % 2 == 0)

{

temp.Name = line;

gamers.Add(temp);

}

else

{

temp.AmountOfSeconds = Convert.ToInt32(line);

}

i++;

}

}

gamers.Sort();

return gamers;

}

public override string ToString()

{

return $"Гравець {Name} – {AmountOfSeconds} секунд";

}

}

}

\*\*\*AssemblyInfo.cs\*\*\*

using System.Windows;

[assembly: ThemeInfo(

ResourceDictionaryLocation.None, //where theme specific resource dictionaries are located

//(used if a resource is not found in the page,

// or application resource dictionaries)

ResourceDictionaryLocation.SourceAssembly //where the generic resource dictionary is located

//(used if a resource is not found in the page,

// app, or any theme specific resource dictionaries)

)]