МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Об’єктно-орієнтоване програмування»
на тему:

«Сапер»

студента І курсу групи ВТ-21-1
спеціальності 121 «Інженерія
програмного забезпечення»
Маньківський Владислав Вячеславович
(прізвище, ім’я та по-батькові)

Керівник ст.викл.каф. ІПЗ Чижмотря О.В.

Дата захисту: " ___ " _____ 2022 р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ В.Л. Левківський
(підпис) (прізвище та ініціали)
_____ Г.В. Марчук
(підпис) (прізвище та ініціали)
_____ О.В. Чижмотря
(підпис) (прізвище та ініціали)

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Факультет інформаційно-комп'ютерних технологій
Кафедра інженерії програмного забезпечення
Освітній рівень: бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Зав. кафедри
_____ А.В. Морозов
"____"_____2022р.

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТУ

1. Тема роботи: сапер,
керівник курсового проекту: ст. викл. каф. ІПЗ Чижмотря О.В.
2. Строк подання студентом: " 08 " червня 2022р.
3. Вхідні дані до роботи: _____
4. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
1. Аналіз аналогічних розробок;
2. Алгоритми роботи програми;
3. Опис роботи програми.

5. Перелік графічного матеріалу(з точним зазначенням обов'язкових креслень)
_____
_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2	Левківський В.Л., ст. викладач каф. КН	дата	дата
1,2	Марчук Г.В., ст. викладач каф. КН	дата	дата
1,2	Чижмотря О.В., ст. викладач каф. КН	дата	дата

7. Дата видачі завдання " 15 " квітня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проекту	Строк виконання етапів проекту	Примітки
1	Постановка задачі	20.05.2022	виконано
2	Пошук, огляд та аналіз аналогічних розробок	21.05.2022	виконано
3	Формулювання технічного завдання	25.05.2022	виконано
4	Опрацювання літературних джерел	02.06.2022	виконано
5	Проектування структури	07.06.2022	виконано
6	Написання програмного коду	23.06.2022	виконано
7	Налагодження	23.06.2022	виконано
8	Написання пояснювальної записки	01.07.2022	виконано
9	Захист	09.07.2022	виконано

Студент _____ Маньківський В.В.

(підпис) (прізвище та ініціали)

Керівник проекту _____ Чижмотря О.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсової роботи на тему «Сапер» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 36 сторінках друкованого тексту.

Пояснювальна записка має 37 сторінок додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 29 рисунків. Загальний обсяг роботи – 73 сторінок.

КЛЮЧОВІ СЛОВА: САПЕР, МІНИ, РІВНІ, РЕЙТИНГ, ШВИДКІСТЬ.

ЗМІСТ

ВСТУП

Актуальність теми. Мова програмування C# була створена відносно недавно – в кінці 1998 року командою спеціалістів Microsoft. ЇЇ ціллю була можливість створення різноманітних програм для платформ Microsoft.NET. Така прив'язаність платформ Microsoft .NET та мови C# в подальшому була усунена, проте сама платформа .NET у всіх версіях містить компілятор кодів даної мови програмування. Це дозволяє запускати додатки без налаштування додаткового програмного забезпечення.

Сама назва «C#» пов'язана з тим, що її синтаксис дуже нагадує C++. Загалом, C# перейняв багато позитивних рис своїх попередників - Delphi, C++, Java та інших. При цьому із C# були вилучені проблемні алгоритми.

Мова C# актуальна в першу чергу тому, що дозволяє більш раціонально створювати популярні на сьогодні інтернет-додатки. C# тісно інтегрована з мовою XML, різноманітними веб-технологіями. Також популярність C# серед розробників обумовлена тим, що вона інтегрувала в собі переваги мов Java та C++, при чому з C# були виключені деякі сумнівні директиви, макроси, відмінені глобальні перемінні.

Сучасні ігри почали створювати надто казуальними, в яких не потрібно думати головою та бути максимально зосередженим. Тоді чому б не робити ремастери старих ігор, таких як наприклад "Сапер". На перший погляд це проста гра без сенсу, але якшо почати в неї грати, можна зрозуміти що все не так легко.

Метою розробки курсової роботи є створення додатку для гри в сапера, а також удосконалення навичок роботи із мовою "C#", класами та методами. Поставлено завдання, реалізувати додаток з наступними функціями:

-Визначення витраченого часу на перемогу

-Запис рекордів у файл

-Виведення рекордів всіх гравців

-Вибір параметрів для гри

Об’єктом дослідження є технологія роботи з Windows Foundation, класами.

Предметом дослідження є робота с конструкторами класів, їх об’єктами, елементами керування WPF.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз задачі, засобів та методів її вирішення

Тематикою курсової роботи є гра “ Сапер”.

Коли говоримо про ігри, перше що приходить на думку, це щось сучасне, або казуальне, і все більше та більше забуваємо такі класичні ігри, як наприклад “Сапер”.

Головна мета якого, це відкрити всі порожні клітинки на полі, не потрапивши на міну. Клітина набуває значення, відповідне кількості мін навколо неї.

Якщо грати на рейтинг, потрібно швидко приймати рішення, котру клітинку обрати, аби отримати швидший результат за часом, саме це і додає максимальну складність, так як необхідно швидко приймати рішення, це навичка, яка дуже необхідна у нашому житті, так як, іноді люди не спроможні швидко щось обрати, саме ця гра, більш за все, допоможе зруйнувати цей бар’єр.

Функціонал програмного додатку:

-Визначення витраченого часу на перемогу

-Запис рекордів у файл

-Виведення рекордів всіх гравців

-Вибір параметрів для гри

1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи.

Для початку, хотілось би сказати про програму, яка була моїм натхненням під час створення програмного додатку. Minesweeper – гра, яка була скопійована з класичного сапера, перша згадка якого була за часів Windows 95-Windows XP.(рис 1.1)

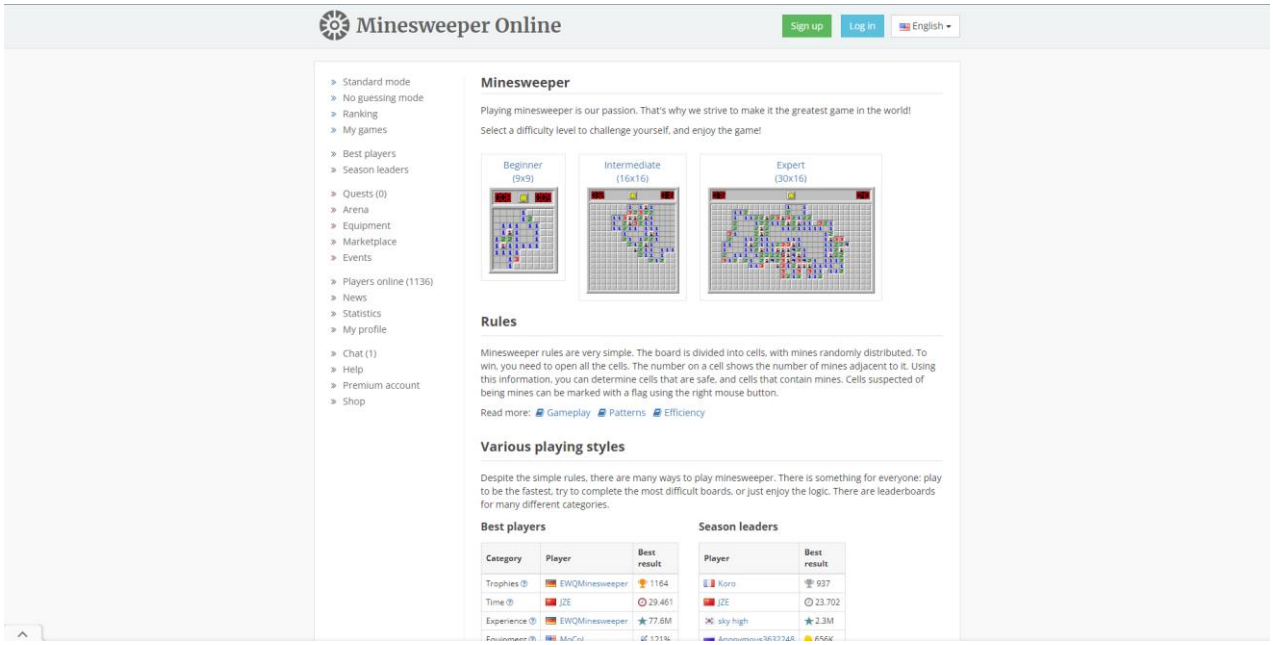

Рис. 1.1

У Minesweeper можна обрати один з трьох рівнів важкості, або створити власний рівень, з власними налаштуваннями. Але, мене зацікавило те, що присутня таблиця рекордів, в якій можна переглянути найкращих гравців за різними параметрами.

Один з гарних сучасних прикладів, це Minesweeper від Google. В ньому теж присутні всі ознаки класичного сапера, але вже з повністю іншим кольоровим дизайном. Але відсутній рейтинг та власна гра. Інших особливостей не було знайдено.(рис. 1.2)

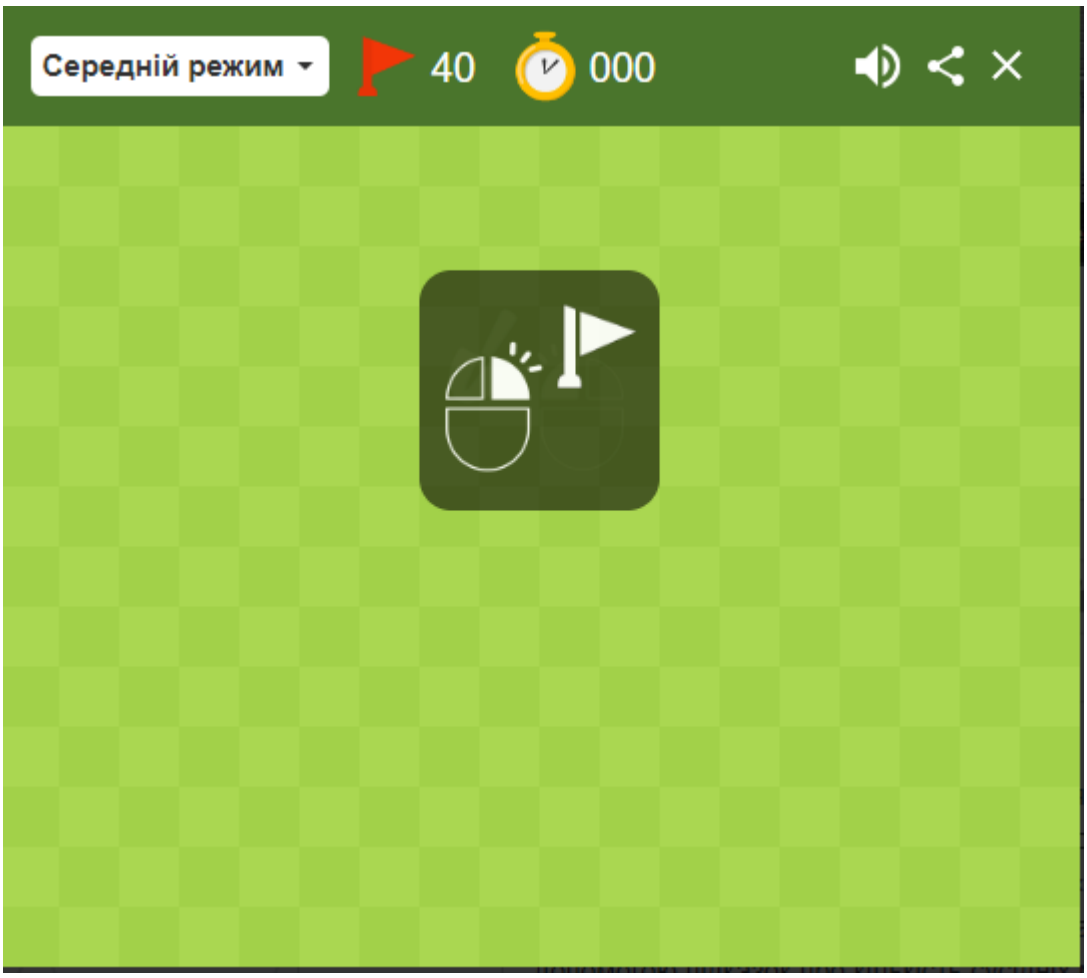

Рис. 1.2

І останій приклад, який демонструє, що більшість варіацій сапера однокові. Це так, тому що правила завжди однокові, і змінювати просто немає що, окрім видалення якихось елементів, або зміни дизайну.(рис. 1.3)

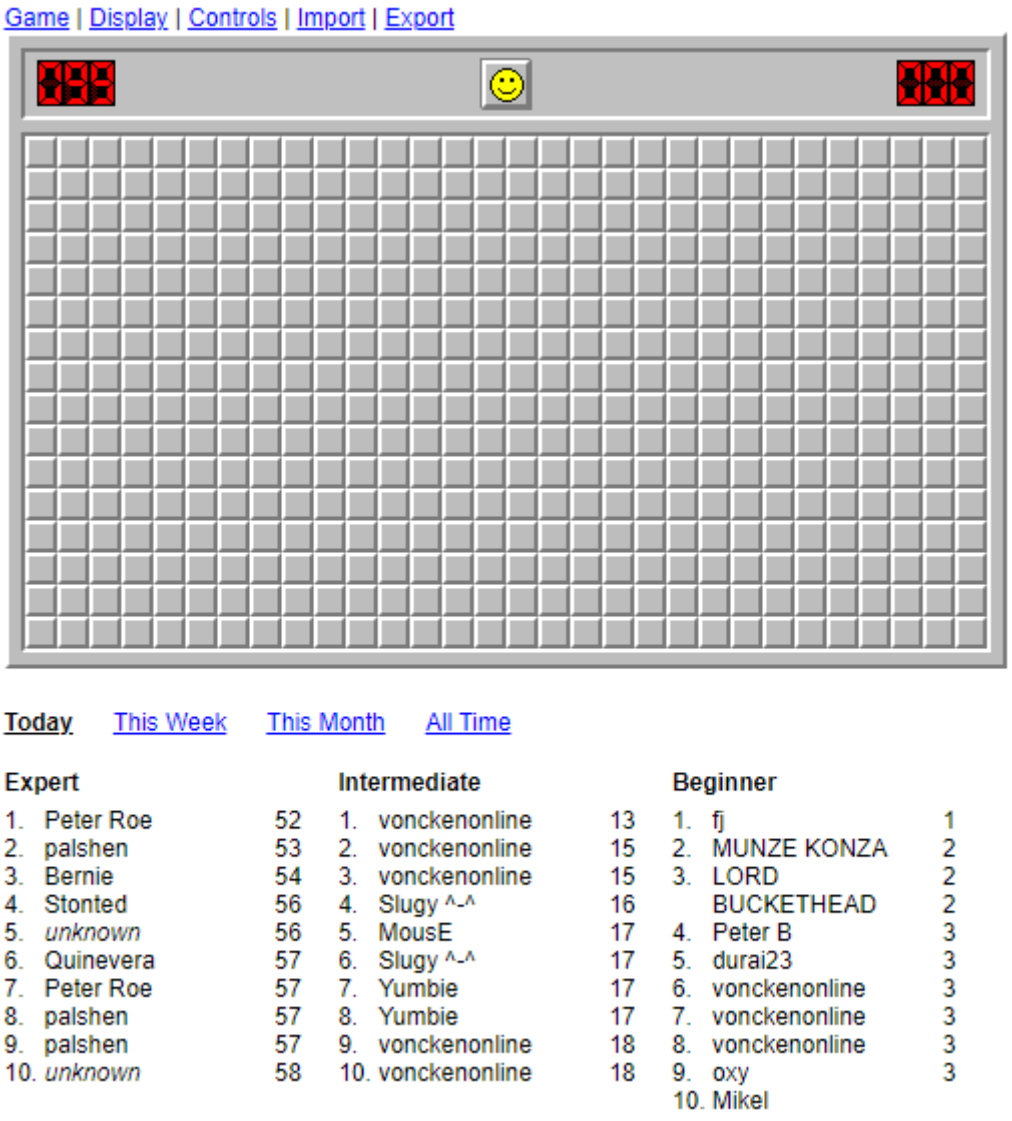

Рис. 1.3

Висновки до першого розділу.

В першому розділі було проаналізовано задачу, та те, що вона має містити, а саме: три рівні гри, специфічний дизайн, який міг би відокремитися від інших, та рейтинг найкращих результатів. Також, було проаналізовано такі сервіси, як minesweeper.online, minesweeper від Google та minesweeperonline.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування загального алгоритму роботи програми

На рисунку 2.1. представлена блок-схема продукту, що розробляється.

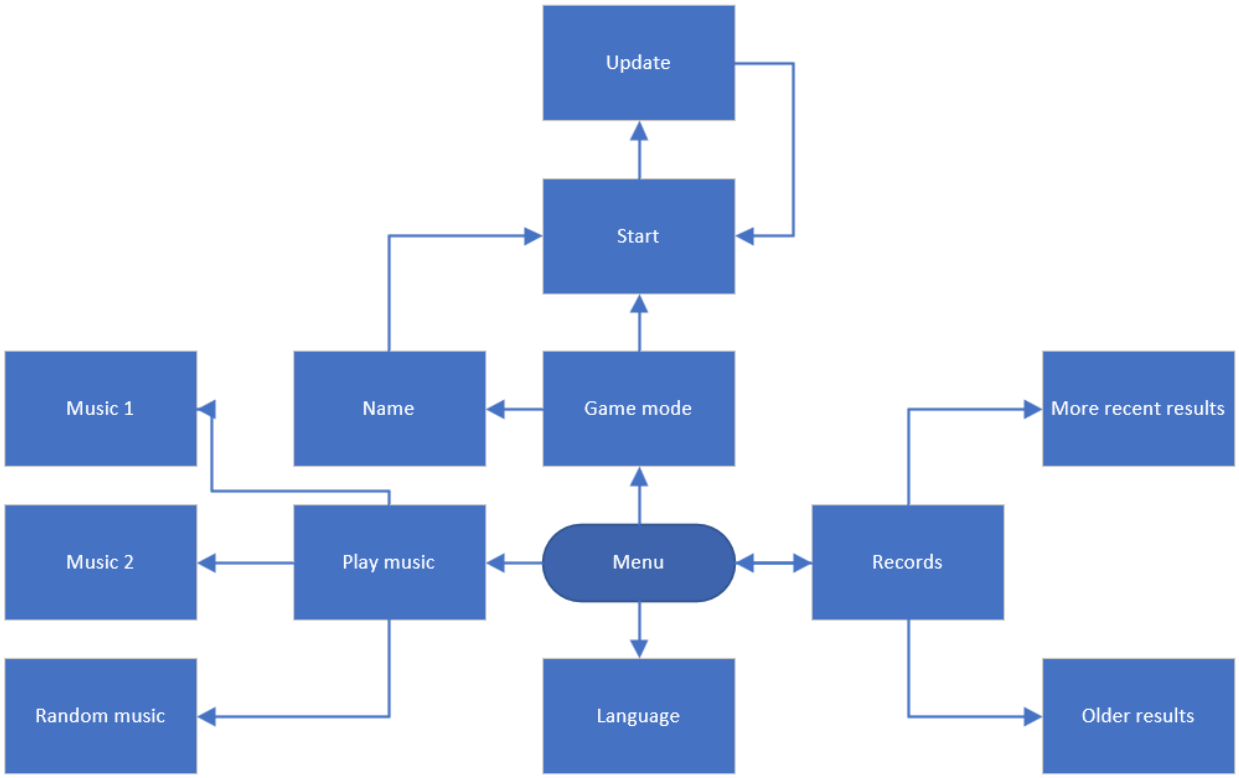

Рис 2.1. Загальна структура

При запуску програми одразу кидається в очі те, що можна обрати 4 режими гри: легкий, середній, важкий та кастомний. В останньому можна обрати будь-які налаштування, а саме висота, ширина та кількість мін.

У полі “Музика” побачити чотири кнопки, зупинити, обрати першу, обрати другу, будь-яку, та поле для введення гучності, яка по замовчуванню буде обрана на 50.

Ще за власним бажанням можна обрати одну з трьох локалізацій, а саме українська, англійська та польська.

До старту гру, гравець може записати своє ім’я, аби воно з’явилося у рейтингу, якщо це не буде виконано, ім’я буде обрано автоматично “-”.

Під час самої гри, гравець може оновити поле. До першого ходу, якщо ввести послідовність up, up, down, down, left, right, left, right, B, A активується пасхалка, яка змінить головну тему додатку на темну.

Під час перегляду рекордів, можна побачити, що час гравця зліва, від його ім’я, так як так зручніше переглядати. В рекордах передбачена можливість перегляду більш гірших результатів.

Отже, при запуску програми користувач може обрати наступну дію:

- Змінити мову (українська, англійська, польська)
- Обрати режим гри
- Переглянути рекорди
- Обрати музику
- Почати гру

2.2 Розробка функціональних алгоритмів роботи програми:

На рисунку 2.2 представлено загальна діаграма класів програмного забезпечення.

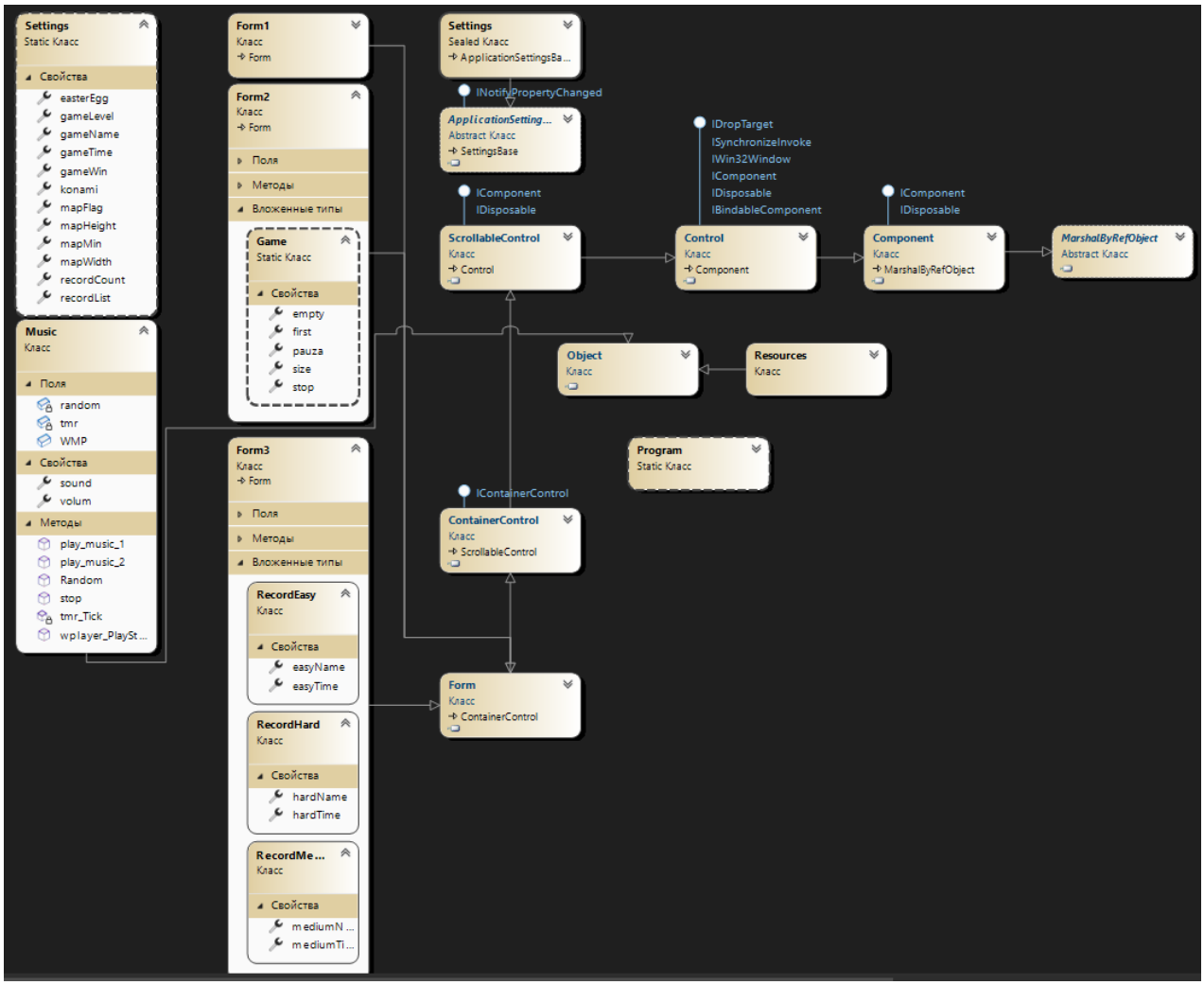

Рис. 2.2. Діаграма класів

На діаграмі класів можна побачити залежність та зв’язок між класами. Головним батьківським класом є клас Object, від якого ідуть інші класи, такі як Resources, Settings, Component тощо. Також від нього буде наслідуватися клас Form, який в свою чергу є батьківським класом для класів Form1, Form2, Form3. Від класів Object, Form2 та Form3 наслідуються такі класи як Game, RecordEasy, RecordMedium, RecordHard, Music ще присутній клас, який не наслідується, Settings. Розберемо кожен клас окремо.

Клас Settings є ключовим класом для програмного додатку, так як його єдина та найголовніша його мета, це збереження даних між формами. Далі

розберемо його поля. easterEgg – необхідне для збереження порядок клавіш, введених користувачем для активації пасхалки. gameLevel – рівень важкості, обраний гравцем. gameName – ім’я гравця, яке буде використане в рейтингу. gameTime – час, витрачений гравцем. gameWin – перевірка на перемогу гравця. konami – перевірка, чи активована пасхалка. mapFlag – кількість прапорців, яких можна згенерувати відповідно кількості мін. mapHeight – кількість клітинок у висоту. mapWidth – кількість клітинок у ширину. mapMin – кількість мін на полі. recordCount – запис того, скільки максимально рекордів у трьох рівнях важкості. recordList – номер сторінки в рекордах, яка зараз відкрита.

Трохи про інші класи. Music – відповідає за музику. Game – дані, необхідні для роботи гри. RecordEasy, RecordMedium, RecordHard – класи, необхідні для збереження рекордів з файлів.

Рис 2.3

Один із основних алгоритмів, це генерація мін, так як окрім них необхідно згенерувати цифри, які допоможуть при проходженні гри. Генерація відбувається під час першого ходу, так як на ньому не може бути

міни, разом з цим була виконана перевірка на те, чи на полі всього одна клітинка, якщо так, то міна згенерується саме на ній. Не менш важлива генерація цифр, вона відбувається додаванням значення 1 до кожної клітини, навколо міни, разом з перевіркою кінця мапи.(рис 2.4)

Рис 2.4. Блок схема генерації мін та цифр

Лістинг алгоритму генерації мін та цифр:

```
private void mapGenerate(int first_i, int first_j)
        {
            Settings.mapFlag = Settings.mapMin;
            Random rnd = new Random();
            for (int i = 0; i < Settings.mapHeight; i++)
            {
                for (int j = 0; j < Settings.mapWidth; j++)
                {
                    map[i, j] = 0;
                    buttons[i, j].Image = imgFind(5, 3);
                }
            }
            textBox2.Text = Settings.mapFlag.ToString();
            map[first_i, first_j] = 100;
            for (int k = 0; k < Settings.mapMin;)
            {
                int i = rnd.Next(0, Settings.mapHeight);
                int j = rnd.Next(0, Settings.mapWidth);
                if (map[i, j] != 10 && map[i, j] != 100)
                {
                    map[i, j] = 10;
                    k++;
                }
                if (Settings.mapHeight + Settings.mapWidth == 2)
                {
                    map[0, 0] = 10;
                    k++;
                }
            }
            map[first_i, first_j] = 0;
            if (Settings.mapHeight + Settings.mapWidth == 2)
            {
                map[0, 0] = 10;
            }
            for (int i = 0; i < Settings.mapHeight; i++)
            {
                for (int j = 0; j < Settings.mapWidth; j++)
                {
                    if (map[i, j] == 10)
                    {
                        if (i + 1 < Settings.mapHeight && map[i + 1, j] != 10)
                            map[i + 1, j]++;//1
                        if (i - 1 >= 0 && map[i - 1, j] != 10)
                            map[i - 1, j]++;//2
                        if (j + 1 < Settings.mapWidth && map[i, j + 1] != 10)
                            map[i, j + 1]++;//3
                        if (j - 1 >= 0 && map[i, j - 1] != 10)
```

```
                            map[i, j - 1]++;//4
                        if (i + 1 < Settings.mapHeight && j + 1 < Settings.mapWidth
&& map[i + 1, j + 1] != 10)
                            map[i + 1, j + 1]++;//5
                        if (i - 1 >= 0 && j + 1 < Settings.mapWidth && map[i - 1, j +
1] != 10)
                            map[i - 1, j + 1]++;//6
                        if (i + 1 < Settings.mapHeight && j - 1 >= 0 && map[i + 1, j
- 1] != 10)
                            map[i + 1, j - 1]++;//7
                        if (i - 1 >= 0 && j - 1 >= 0 && map[i - 1, j - 1] != 10)
                            map[i - 1, j - 1]++;//8
                    }
                }
            }
            Settings.gameTime = 0;
        }
```

Інший не менш важливий алгоритм, це натискання на кнопки. На ліву та праву клавіші мишки був виконаний власний метод. Ліва має розкривати клітинку, окрім випадку коли там прапорець, якщо це порожня клітина, буде здійснена перевірка на те, чи навколо неї немає іншої пустої клітинки, це буде продовжуватися доти, доки вони не закінчаться.(рис. 2.5)

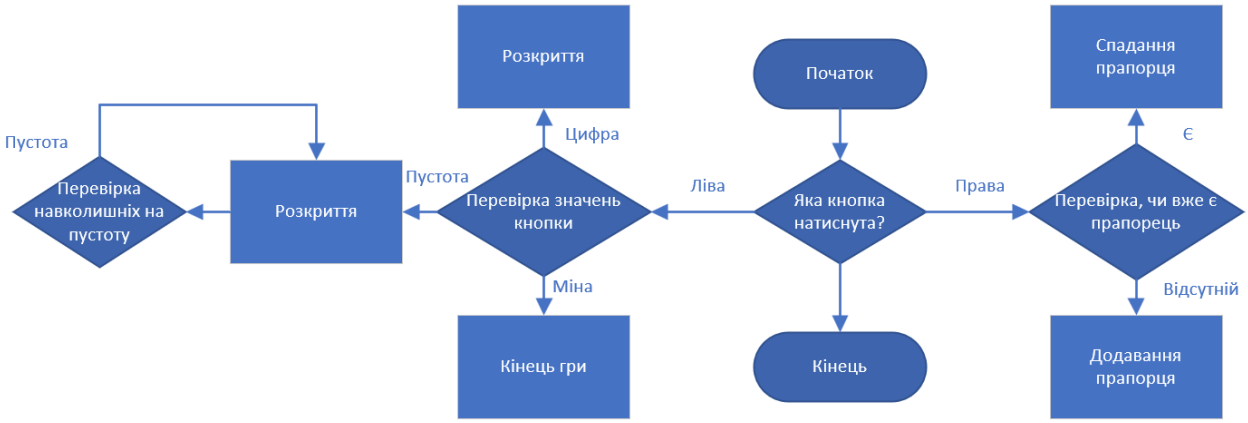

Рис 2.5. Блок схема генерації мін та цифр

Лістинг алгоритму натискання на клавішу:

```
        private void pressed(object sender, MouseEventArgs e)
        {
            Button pressedButton = (Button)sender;
            switch (e.Button.ToString())
            {
                case "Right":
                    pressedRight(pressedButton);
                    break;
                case "Left":
                    pressedLeft(pressedButton);
                    break;
            }
```

```
}
```

2.3 Розробка програмного забезпечення:

Після завершення проектування програмного забезпечення, відразу було розпочато розробку програмного забезпечення. Спочатку було створено форму головного меню, в яку було додано кнопки налаштування рівня, рейтинг, музика та локалізація. Після завершення розробки меню, було розпочато працю над формою гри, там присутнє саме поле для гри, кнопка перезапуску форми, кнопка паузи, витрачений час та кількість прапорців. На завершення було проведено роботу над рейтингом, там присутні шість полів для виведення, три для часу, три для ім'я, кожне під свій рівень важкості. Було додано окремо ще три кнопки, одна аби прогорнути сторінку рекордів назад, інша уперед, та остання для повернення до меню.

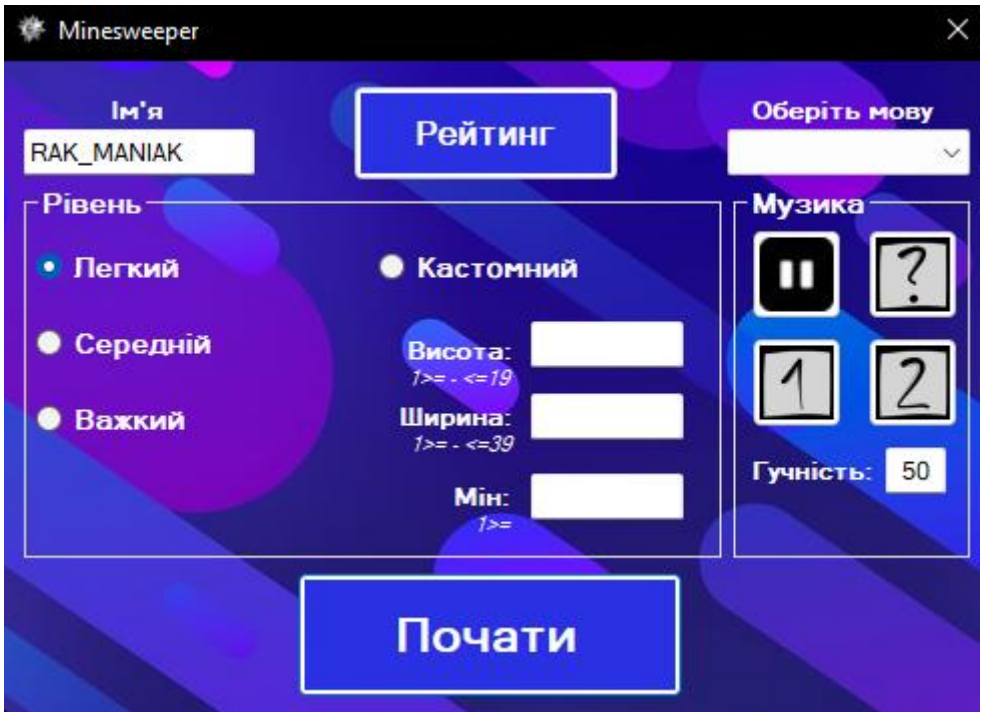

Рис.2.6. Головне меню програмного забезпечення

Висновки до другого розділу:

У даному розділі було розроблено загальний алгоритм роботи програми, реалізовані та описані основні алгоритми. Було розроблено основне програмне

забезпечення, наведено допоміжні блок-схеми, наведено лістинг основних алгоритмів.

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1 Опис роботи з програмним додатком:

Після запуску програми ми бачимо меню, в якому одразу заповнено ім’я останнього користувача.(рис. 3.1 – рис. 3.2)

У меню одразу можна обрати одну із трьох мов, а саме українську(рис. 3.1), англійську(рис. 3.2) та польську.(рис. 3.3)

За бажанням, користувач може обрати одну із двох запропонованих музик, або увімкнути рандом серед них.(рис. 3.1)

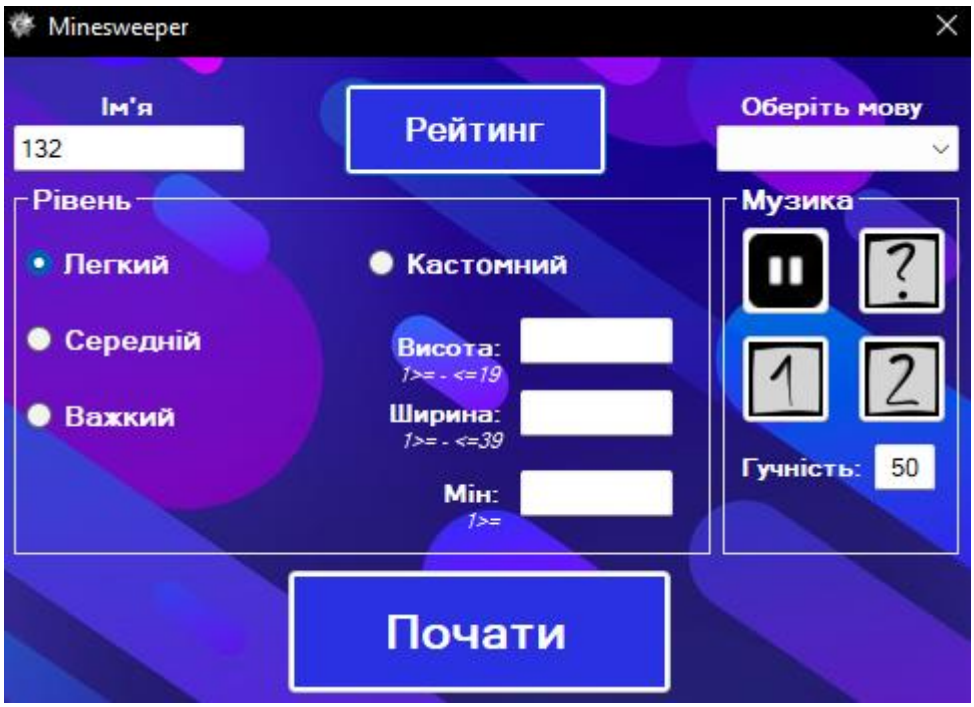

Рис.3.1. Автоматичне введення ім’я та інтерфейс на українській

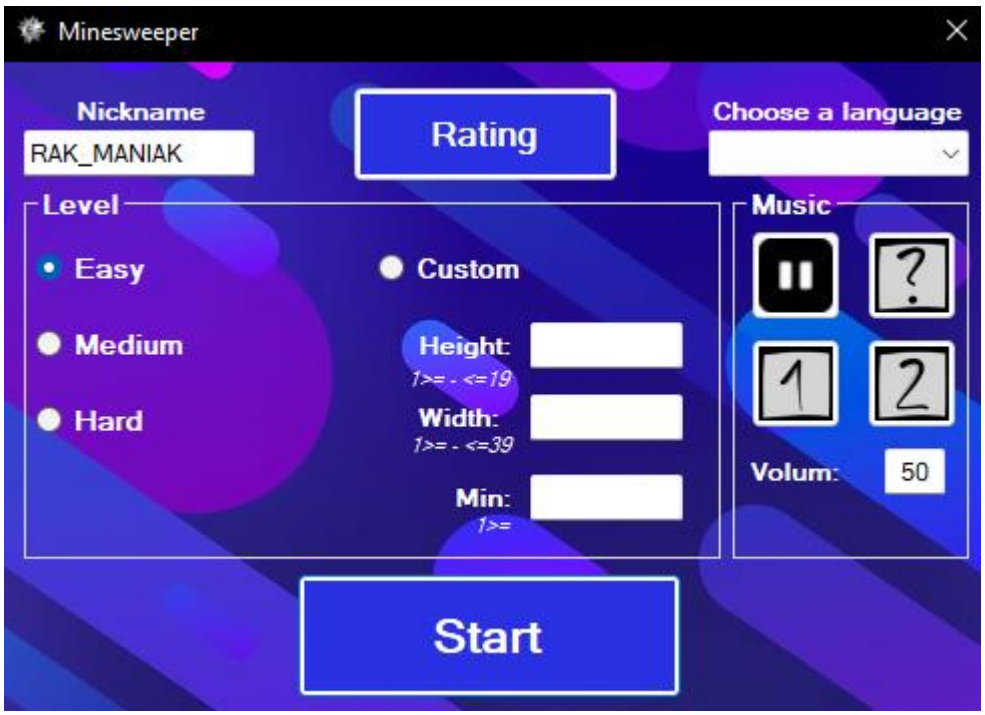

Рис.3.2. Автоматичне введення ім’я та інтерфейс на англійській

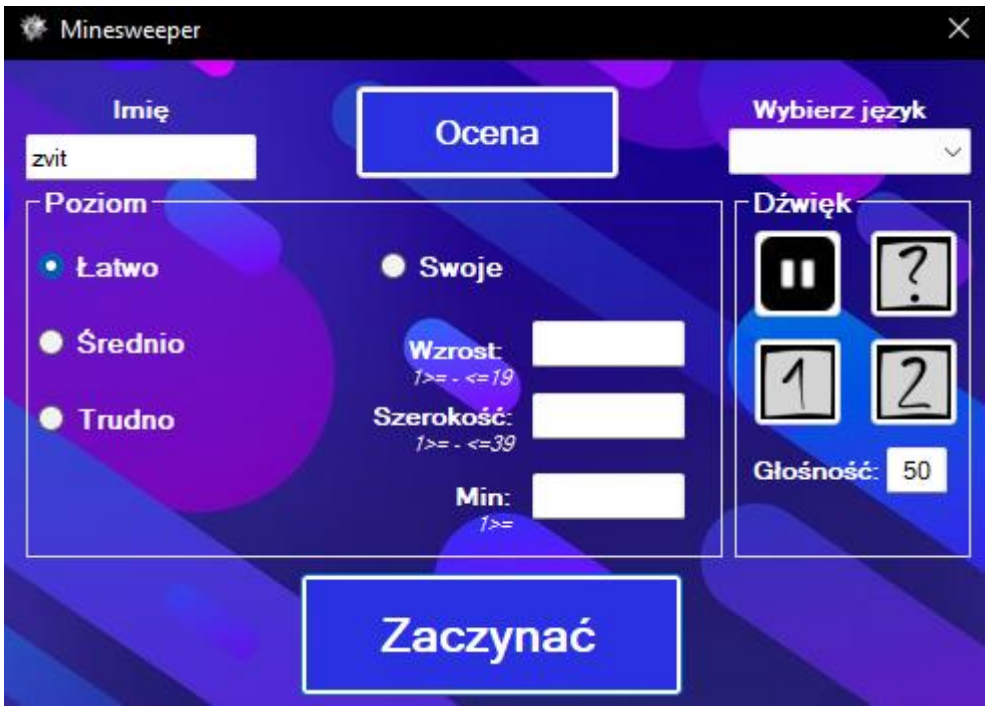

Рис.3.3. Інтерфейс на польській

Є можливість обрати рівень складності, їх різниця у тому, що із його збільшенням, збільшується поле та кількість мін, або кастомний рівень із власними налаштуваннями.(рис. 3.4 – рис. 3.7)

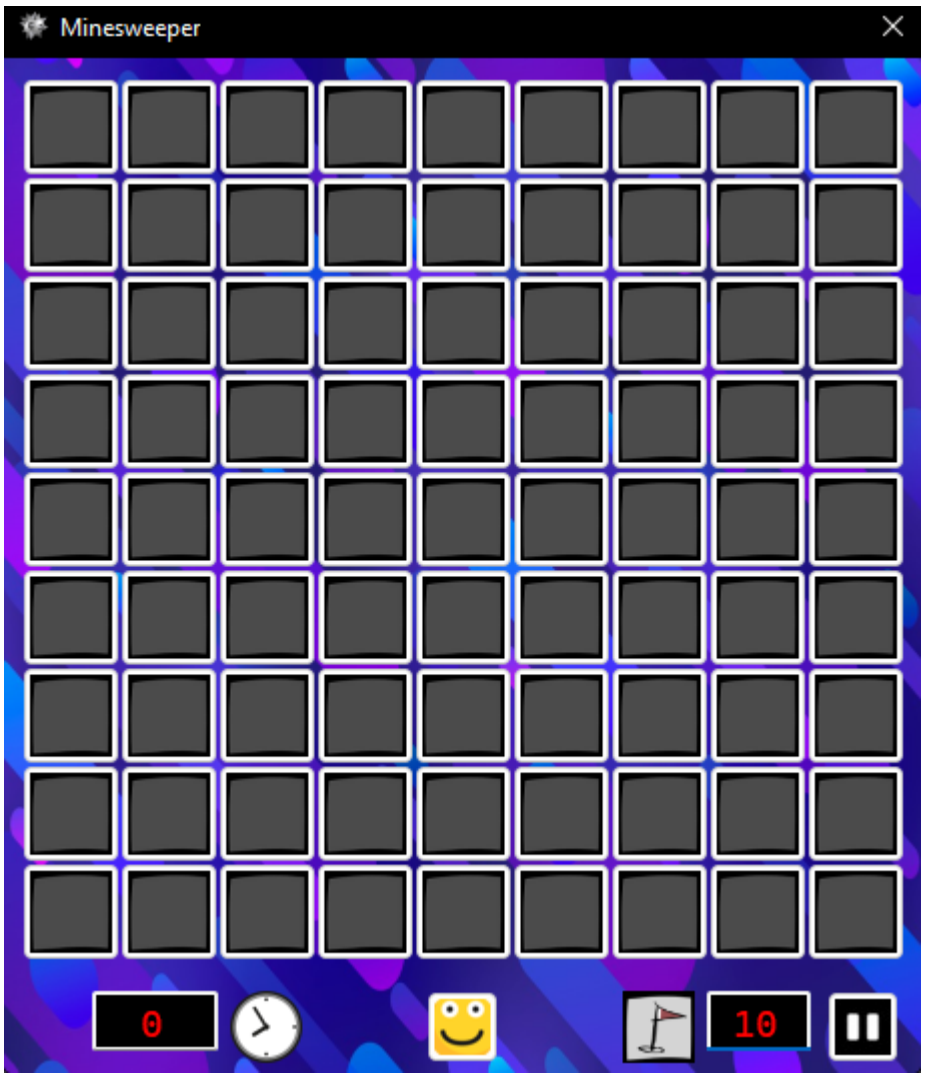

Рис.3.4. Легкий рівень

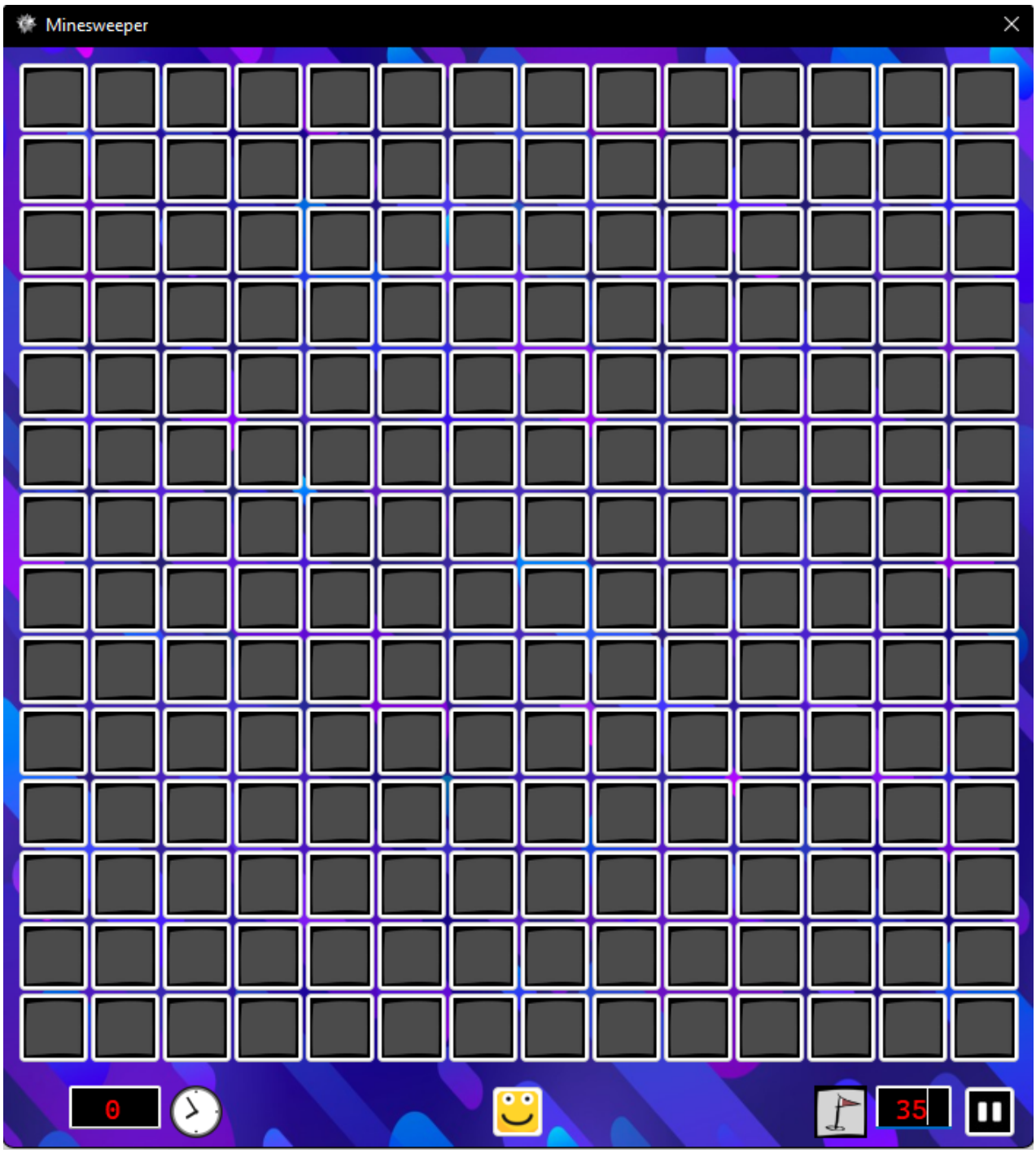

Рис.3.5. Середній рівень

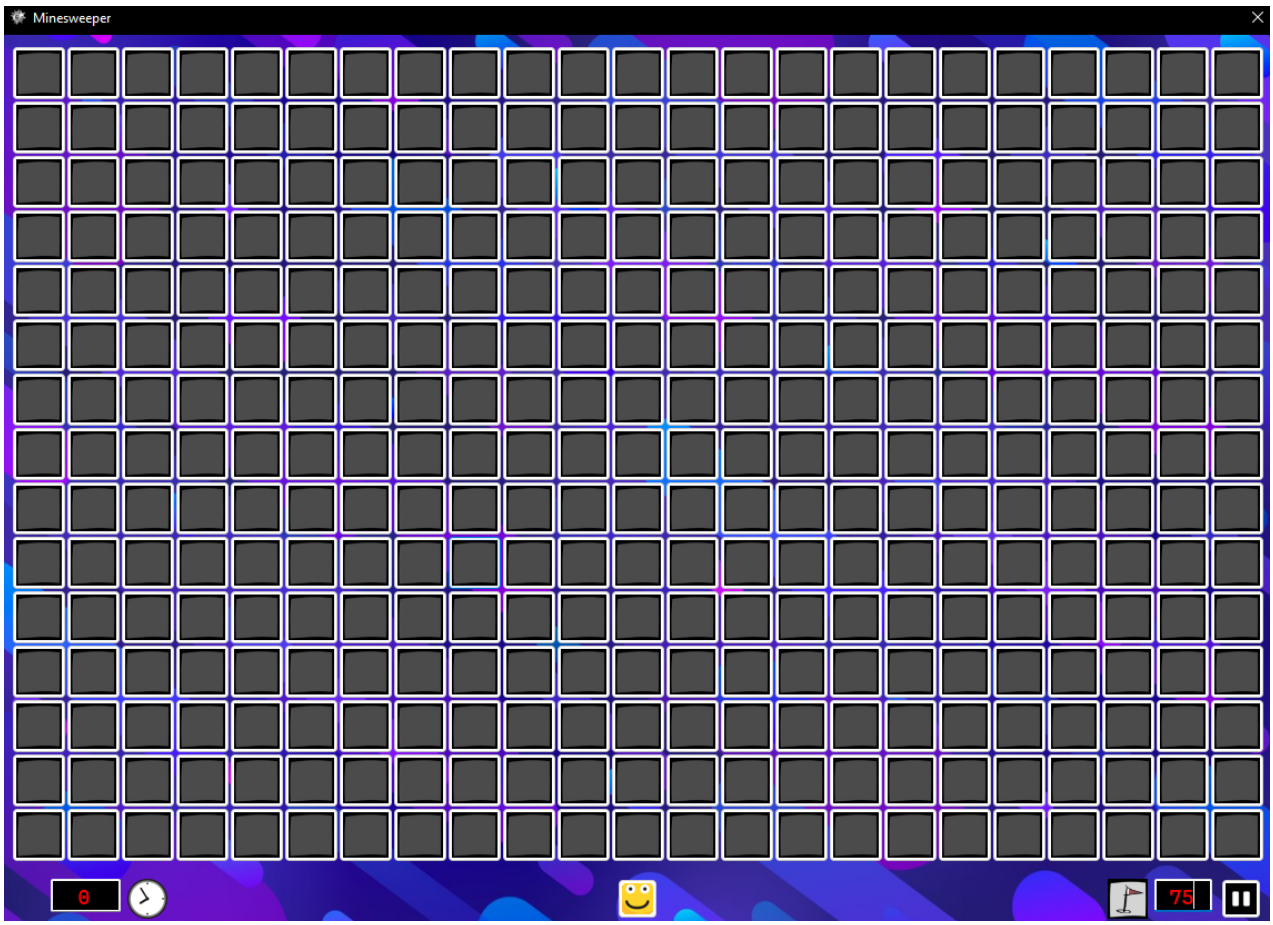

Рис.3.6. Важкий рівень

Рис.3.7. Кастомний рівень

Якщо ввести код конамі активується пасхалка та форма змінить тему на темну, якщо ввести повторно, то з’явиться звичайна тема.(рис. 3.8)

Рис.3.8. Темна тема

Генерація мін відбувається після першого ходу, як і відлік часу, який буде занесений до рейтингу, отже потрапити на цьому ході на міну неможливо, окрім варіанту, коли на полі лише одна клітина.(рис. 3.9 – рис. 3.10)

Рис.3.9. Поле 1x1

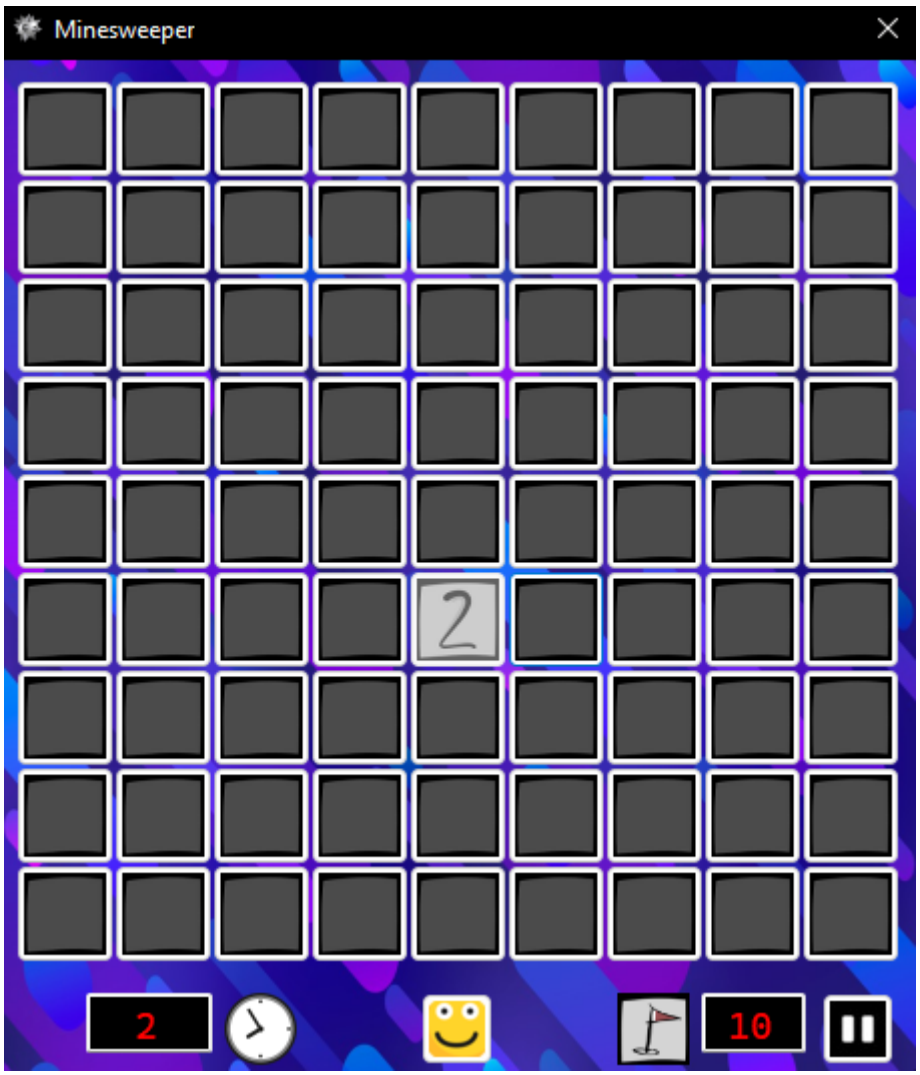

Рис.3.10. Перший хід

У додатку передбачена пауза, яка зупиняє час, та ховає кнопки.(рис. 3.11)

Рис.3.11. Пауза

Можна побачити, що є поле, у якому відображена кількість прапорців, що рівноцінна кількості мін. Поки стоїць прапорець, гравець не зможе розкрити клітинку, що врятує від випадкового натискання. Якщо гравець зніме прапорець, з’явиться знак питання, який буде попереджати, що там знаходився прапорець.(рис. 3.12)

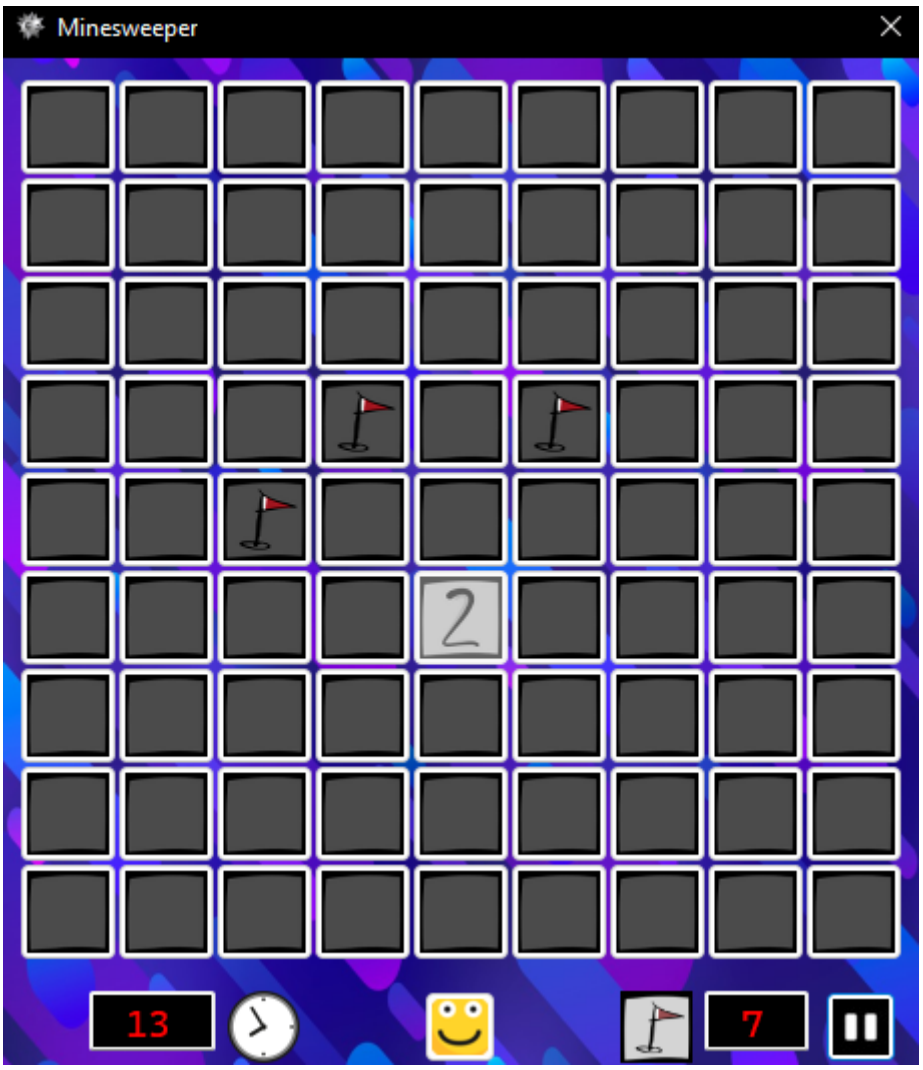

Рис.3.12. Реалізація прапорців

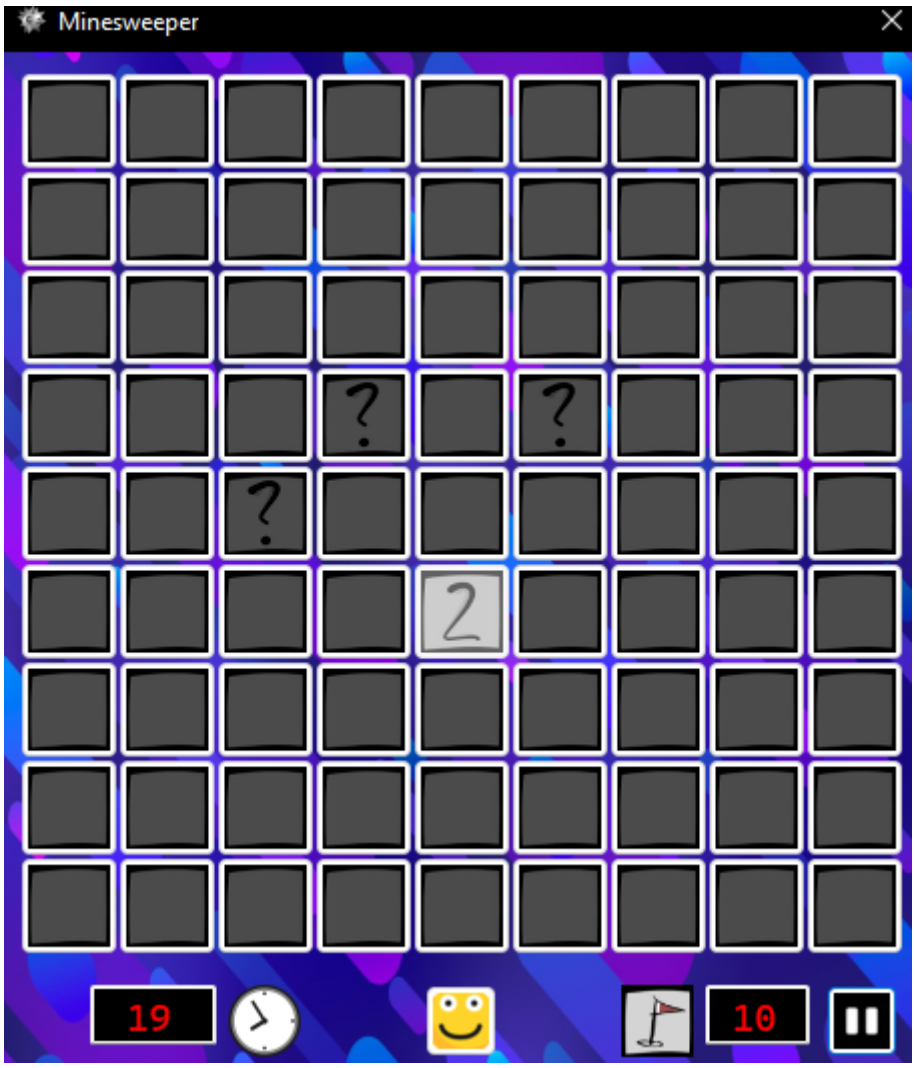

Рис.3.13. Зняття прапорців

Якщо розкрити на порожню клітинку, розкриються і інші порожні клітинки.(рис. 3.14)

Рис.3.14. Зняття прапорців

Якщо натиснути на міну, гравець програє, буде виведено відповідне повідомлення (рис. 3.15), та відкриті всі клітинки.(рис. 3.16)

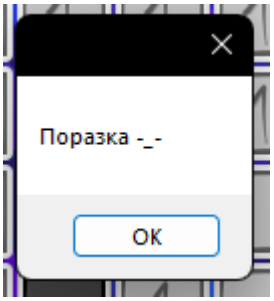

Рис.3.15. Повідомлення про поразку

Якщо гравцю не сподобалася генерація, він може перезавантажити форму, натиснувши на смайл внизу, якщо натиснути на міну, смайл зміниться на інший.(рис. 3.16)

Рис.3.16. Поразка та зміна смайлика

Аби перемогти, необхідно відкрити всі клітинки, які не міни, це можна здійснити за допомогою цифр, які позначають скільки навколо кнопки мін.(рис. 3.17 – рис. 3.18)

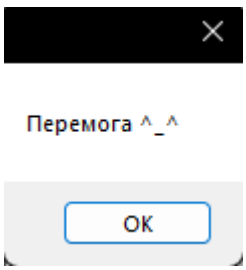

Рис.3.17. Повідомлення про перемогу

Рис.3.18. Перемога

У меню можна перейти до рейтингу, в якому можна переглянути найкращі результати. Ще присутні кнопки, які додають навігацію між старішими-новішими результатами.(рис. 3.19 – рис. 3.20)

Рис.3.19. Рейтинг, сторінка 1

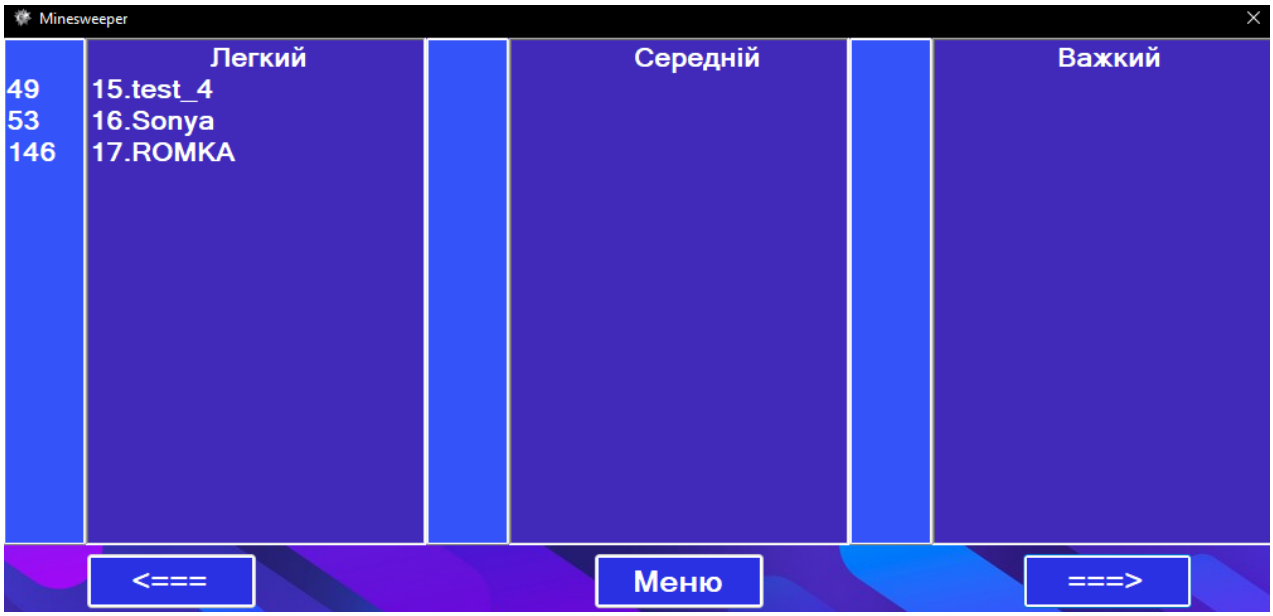

Рис.3.20. Рейтинг, сторінка 2

3.2 Тестування роботи програмного забезпечення

Під час тестування, булу виявлено дві помилки. Одна пов’язана з тим, що якщо в кастомному рівні обрати поле 1x1, то програма видасть помилку, це було швидко вирішено, та продемонстровано його роботу на рисунку 3.9. Інша ж помилка пов’язана з тим, що результати перемоги не заносилися до рекорду, проблема була у тому, що був заданий неправильний порядок перевірок.(рис. 3.19)

Висновки до третього розділу

Отже, після реалізації всіх основних алгоритмів, та проведення тестування додатку, було виявлено дві помилки, які були швидко виправленні.

ВИСНОВКИ

Отже, під час написання курсової роботи було отриманні корисні знання про таку гру, як сапер, реалізацію його алгоритму, музики до нього, та рейтингу гравців. Було вдосконалено навички користування такою мовою програмування, як C#, в якій було покращено знання щодо класів, файлів, та інших структур даних.

Було написано таку гру, як сапер, зрозумілу та цікаву у використанні програму, яка дозволяє позмагатися з іншими гравцями, та весело провести час.

Даний програмний продукт було реалізовано за допомогою середовища розробки програмного забезпечення Microsoft Visual Studio Enterprise 2022 Preview, в додатку Windows Forms (WF). Була обрана платформа .Net Framework і мова програмування C#. При реалізації був використаний об’єктно–орієнтований підхід.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ(мінімум 10)

1) Приклад сапера №1
   https://minesweeper.online/
2) Приклад сапера №2
   https://www.google.com/search?q=minesweeper&oq=&aqs=chrome.1.69i59i450l8.480931622j0j15&sourceid=chrome&ie=UTF-8
3) Приклад сапера №3
   https://minesweeperonline.com/
4) Опис та пояснення що таке клас
   https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes
5) Опис та реалізація такого класу, як Queue
   https://docs.microsoft.com/en-us/dotnet/api/system.collections.queue?view=net-6.0
6) Клас Task, який не повертає значення, виконуючи його асинхронно
   https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task?view=net-6.0
7) Вбудування елемента керування Windows Media Player
   https://docs.microsoft.com/en-us/windows/win32/wmp/embedding-the-windows-media-player-control-in-a-c--solution
8) Подія закриття форми
   https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.form.closing?view=windowsdesktop-6.0
9) Подвійна буферизована властивість
   https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.doublebuffered?view=windowsdesktop-6.0
10) File Класс
    https://docs.microsoft.com/ru-ru/dotnet/api/system.io.file?view=net-6.0

ДОДАТКИ

Додаток А

Лістинг Program:

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

namespace main
{
    public class Music
    {
        Random random = new Random();
        public int sound { get; set; }
        public int volum { get; set; }
        public static WMPLib.WindowsMediaPlayer WMP = new
WMPLib.WindowsMediaPlayer();
        Timer tmr = new Timer();
        public void play_music_1()
        {
            Task.Run(() =>
            {
                try
                {
                    tmr.Interval = 10;
                    tmr.Stop();
                    WMP.URL = @"files\\1.mp3";
                    WMP.settings.volume = volum;
                    WMP.controls.play();
                    tmr.Tick += new EventHandler(tmr_Tick);
                    WMP.PlayStateChange += new
WMPLib._WMPOCXEvents_PlayStateChangeEventHandler(wplayer_PlayStateChange);
                }
                catch (Exception ex) { }
            });
        }
        public void play_music_2()
        {
            Task.Run(() =>
            {
                try
                {
                    tmr.Interval = 10;
                    tmr.Stop();
                    WMP.URL = @"files\\2.mp3";
                    WMP.settings.volume = volum;
                    WMP.controls.play();
                    tmr.Tick += new EventHandler(tmr_Tick);
                    WMP.PlayStateChange += new
WMPLib._WMPOCXEvents_PlayStateChangeEventHandler(wplayer_PlayStateChange);
                }
                catch (Exception ex) { }
            });
        }

        void tmr_Tick(object sender, EventArgs e)
        {
            WMP.controls.stop();
            if (sound == 1)
                play_music_1();
            if (sound == 2)
                play_music_2();
```

```
            if (sound == 3)
                Random();
        }
        public void wplayer_PlayStateChange(int NewState)
        {
            if (NewState == (int)WMPLib.WMPPlayState.wmppsMediaEnded)
            {
                tmr.Start();
            }
        }
        public void Random()
        {
            if (random.Next(1, 3) == 1)
                play_music_1();
            if (random.Next(1, 3) == 2)
                play_music_2();
        }
        public void stop()
        {
            WMP.controls.stop();
            WMP.close();
        }
    }

    static class Settings
    {
        public static int mapHeight { get; set; }
        public static int mapWidth { get; set; }
        public static int mapMin { get; set; }
        public static int mapFlag { get; set; }
        public static int gameWin { get; set; }
        public static int gameTime { get; set; }
        public static int gameLevel { get; set; }
        public static string gameName { get; set; }
        public static int recordList { get; set; }
        public static int recordCount { get; set; }
        public static int easterEgg { get; set; }
        public static bool konami { get; set; }
    }

    internal static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Лістинг Form1:

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Globalization;
using WMPLib;
using System.IO;

namespace main
{
    public partial class Form1 : Form
    {
        Music music = new Music();
        public Form1()
        {
            System.Threading.Thread.CurrentThread.CurrentUICulture =
CultureInfo.GetCultureInfo(Properties.Settings.Default.Language);
            System.Threading.Thread.CurrentThread.CurrentCulture =
CultureInfo.GetCultureInfo(Properties.Settings.Default.Language);
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            FormBorderStyle = FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
            this.FormClosing += Form1_Closing;
            textBox1.MaxLength = 10;
            textBox2.MaxLength = 10;
            textBox3.MaxLength = 10;
            textBox4.MaxLength = 14;
            textBox5.MaxLength = 3;
            nickname(true);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Settings.gameName = textBox4.Text;
            if (Settings.gameName.Length == 0)
```

```
    Settings.gameName = "-";

if (radioButton1.Checked)
{
    nickname(false);
    Settings.gameLevel = 1;
    Settings.mapHeight = 9;
    Settings.mapWidth = 9;
    Settings.mapMin = 10;
    this.Hide();
    Form2 game = new Form2();
    game.Show();
}

if (radioButton2.Checked)
{
    nickname(false);
    Settings.gameLevel = 2;
    Settings.mapHeight = 14;
    Settings.mapWidth = 14;
    Settings.mapMin = 35;
    this.Hide();
    Form2 game = new Form2();
    game.Show();
}

if (radioButton3.Checked)
{
    nickname(false);
    Settings.gameLevel = 3;
    Settings.mapHeight = 15;
    Settings.mapWidth = 23;
    Settings.mapMin = 75;
    this.Hide();
    Form2 game = new Form2();
    game.Show();
}
```

```
if (radioButton4.Checked)
{
    Settings.gameLevel = 0;
    int mh, mw, mm, min = 0;
    if (int.TryParse(textBox1.Text, out mh) && mh > min && mh < 20)
    {
        mh = int.Parse(textBox1.Text);
        Settings.mapHeight = mh;
    }
    else
    {
        if (Properties.Settings.Default.Language == "en-US")
            MessageBox.Show("Error entering height!");
        else if (Properties.Settings.Default.Language == "pl-PL")
            MessageBox.Show("Błąd podczas wprowadzania wysokości!");
        else
            MessageBox.Show("Помилка введення висоти!");
        textBox1.Text = null;
        mh = 0;
    }

    ////

    if (int.TryParse(textBox2.Text, out mw) && mw > min && mw < 40)
    {
        mw = int.Parse(textBox2.Text);
        Settings.mapWidth = mw;
    }
    else
    {
        if (Properties.Settings.Default.Language == "en-US")
            MessageBox.Show("Width input error!");
        else if (Properties.Settings.Default.Language == "pl-PL")
            MessageBox.Show("Błąd podczas wprowadzania wysokości!");
        else
            MessageBox.Show("Błąd wprowadzania szerokości!");
        textBox2.Text = null;
        mw = 0;
```

```
        }

        ////

        if (int.TryParse(textBox3.Text, out mm) && mm > 0)
        {
            if (mm > (mh * mw) * 0.9)
                mm = Convert.ToInt32(Math.Round((mh * mw) * 0.9, 2));
            else if (mh == 1 && mw == 1)
                mm = 1;
            else
                mm = int.Parse(textBox3.Text);
            Settings.mapMin = mm;
        }
        else
        {
            if (Properties.Settings.Default.Language == "en-US")
                MessageBox.Show("Error entering the number of mines!");
            else if (Properties.Settings.Default.Language == "pl-PL")
                MessageBox.Show("Błąd podczas wprowadzania liczby min!");
            else
                MessageBox.Show("Помилка введення кількості мін!");
            textBox3.Text = null;
            mm = 0;
        }

        if (mh > 0 && mw > 0 && mm > 0)
        {
            nickname(false);
            this.Hide();
            Form2 game = new Form2();
            game.Show();
        }
    }
    Settings.mapFlag = Settings.mapMin;

}
private async Task nickname(bool k)
```

```
{
    if (Settings.gameName != "-" && k == false)
    {
        File.Create("files\\nickname.txt").Close();
        StreamWriter write = new StreamWriter("files\\nickname.txt", true);
        write.WriteLineAsync(Settings.gameName);
        write.Close();
    }
    if (k == true)
    {
        StreamReader read = File.OpenText("files\\nickname.txt");
        textBox4.Text = await read.ReadLineAsync();
        read.Close();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
    Form3 record = new Form3();
    record.Show();
}
bool lang = false;
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    lang = true;
    if (languages.SelectedIndex == 0 && lang)
    {
        System.Threading.Thread.CurrentThread.CurrentUICulture = CultureInfo.GetCultureInfo("uk-UA");
        System.Threading.Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("uk-UA");
        Properties.Settings.Default.Language = "uk-UA";
        Properties.Settings.Default.Save();
        Application.Restart();
        languages.Text = "Українська";
    }
    else if (languages.SelectedIndex == 1)
    {
        System.Threading.Thread.CurrentThread.CurrentUICulture = CultureInfo.GetCultureInfo("en-US");
```

```
        System.Threading.Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("en-US");
        Properties.Settings.Default.Language = "en-US";
        Properties.Settings.Default.Save();
        Application.Restart();
        languages.Text = "English";
    }
    else if (languages.SelectedIndex == 2)
    {
        System.Threading.Thread.CurrentThread.CurrentUICulture = CultureInfo.GetCultureInfo("pl-PL");
        System.Threading.Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("pl-PL");
        Properties.Settings.Default.Language = "pl-PL";
        Properties.Settings.Default.Save();
        Application.Restart();
        languages.Text = "Polski";
    }
}

private void textBox5_TextChanged(object sender, EventArgs e)
{
}

private void button_stop_Click(object sender, EventArgs e)
{
    music.sound = 0;
    music.stop();
}

private void button_music_1_Click(object sender, EventArgs e)
{
    volum();
    music.sound = 1;
    music.play_music_1();
}

private void button_music_2_Click(object sender, EventArgs e)
{
    volum();
    music.sound = 2;
```

```
    music.play_music_2();
}

private void button3_Click(object sender, EventArgs e)
{
    volum();
    music.sound = 3;
    music.Random();
}
void volum()
{
    int k;
    if (int.TryParse(textBox5.Text, out k))
    {
        if (k < 1)
        {
            textBox5.Text = "1";
            music.volum = 1;
        }
        else if (k > 100)
        {
            textBox5.Text = "100";
            music.volum = 100;
        }
        else music.volum = k;
    }
    else
    {
        textBox5.Text = null;
    }
}
private void Form1_Closing(Object sender, FormClosingEventArgs e)
{
    Settings.gameName = textBox4.Text;
    if (Settings.gameName.Length == 0)
        Settings.gameName = "-";
    nickname(false);
}
```

```
        private void radioButton4_CheckedChanged(object sender, EventArgs e) { }
        private void radioButton1_CheckedChanged(object sender, EventArgs e) { }
        private void radioButton2_CheckedChanged(object sender, EventArgs e) { }
        private void radioButton3_CheckedChanged(object sender, EventArgs e) { }
        private void label6_Click(object sender, EventArgs e) { }
        private void label5_Click(object sender, EventArgs e) { }
        private void label2_Click(object sender, EventArgs e) { }
        private void label1_Click(object sender, EventArgs e) { }
        private void label3_Click(object sender, EventArgs e) { }
    }
}
```

Лістинг Form2:

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



namespace main
{
    public partial class Form2 : Form
    {
        public int[,] map = new int[Settings.mapHeight, Settings.mapWidth];
        public Button[,] buttons = new Button[Settings.mapHeight, Settings.mapWidth];
        static class Game
        {
            public static bool first { get; set; }
            public static bool stop { get; set; }
            public static int empty { get; set; }
            public static int size { get; set; }
            public static bool pauza { get; set; }
```

```
}

public Form2()
{
    InitializeComponent();
}

private void Form2_Load(object sender, EventArgs e)
{
    this.KeyDown += new KeyEventHandler(Form_KeyDown);
    Game.stop = false;
    Game.empty = 10;
    Game.size = 49;

    DoubleBuffered = true;
    if (Settings.mapWidth < 5)
    {
        textBox1.Visible = false;
        textBox2.Visible = false;
    }
    if (Settings.mapWidth < 1)
        button1.Visible = false;
    if (Settings.mapWidth < 3)
        button2.Visible = false;
    if (Settings.mapWidth < 8)
    {
        pictureBox1.Visible = false;
        pictureBox2.Visible = false;
    }
    button1.Image = Image.FromFile("files\\dobre.png");
    button2.Image = Image.FromFile("files\\stop.jpg");
    Settings.gameTime = 0;
    Settings.gameWin = 0;
    FormBorderStyle = FormBorderStyle.FixedSingle;
    MaximizeBox = false;
    MinimizeBox = false;
    this.FormClosing += Form2_Closing;
```

```
            Game.first = true;
            sizeGame();
            this.Location = new Point((Screen.PrimaryScreen.WorkingArea.Width - this.Width) / 2,
                    (Screen.PrimaryScreen.WorkingArea.Height - this.Height) / 2);
            addButtons();
            if (Settings.konami == true)
                BackgroundImage = Image.FromFile("files\\space.jpg");
            textBox2.Text = Settings.mapMin.ToString();
        }

        private void Form2_Closing(Object sender, FormClosingEventArgs e)
        {
            Form main = Application.OpenForms[0];
            if (Settings.konami == true)
                main.BackgroundImage = Image.FromFile("files\\space.jpg");
            main.Show();
        }

        public Image imgFind(int xPos, int yPos)
        {
            Bitmap png_out = new Bitmap(Game.size, Game.size);
            Graphics gr = Graphics.FromImage(png_out);
            gr.DrawImage(Image.FromFile("files\\saper.png"), new Rectangle(new Point(0, 0), new Size(Game.size, Game.size)),
(xPos - 1) * 100 - 7, (yPos - 1) * 100 - 7, 116, 116, GraphicsUnit.Pixel);
            return png_out;
        }

        private void mapGenerate(int first_i, int first_j)
        {
            Settings.mapFlag = Settings.mapMin;
            Random rnd = new Random();
            for (int i = 0; i < Settings.mapHeight; i++)
            {
                for (int j = 0; j < Settings.mapWidth; j++)
                {
                    map[i, j] = 0;
                    buttons[i, j].Image = imgFind(5, 3);
                }
```

```
}
textBox2.Text = Settings.mapFlag.ToString();
map[first_i, first_j] = 100;
for (int k = 0; k < Settings.mapMin;)
{
   int i = rnd.Next(0, Settings.mapHeight);
   int j = rnd.Next(0, Settings.mapWidth);
   if (map[i, j] != 10 && map[i, j] != 100)
   {
      map[i, j] = 10;
      k++;
   }
   if (Settings.mapHeight + Settings.mapWidth == 2)
   {
      map[0, 0] = 10;
      k++;
   }
}
map[first_i, first_j] = 0;
if (Settings.mapHeight + Settings.mapWidth == 2)
{
   map[0, 0] = 10;
}
for (int i = 0; i < Settings.mapHeight; i++)
{
   for (int j = 0; j < Settings.mapWidth; j++)
   {
      if (map[i, j] == 10)
      {
         if (i + 1 < Settings.mapHeight && map[i + 1, j] != 10)
            map[i + 1, j]++;//1
         if (i - 1 >= 0 && map[i - 1, j] != 10)
            map[i - 1, j]++;//2
         if (j + 1 < Settings.mapWidth && map[i, j + 1] != 10)
            map[i, j + 1]++;//3
         if (j - 1 >= 0 && map[i, j - 1] != 10)
            map[i, j - 1]++;//4
         if (i + 1 < Settings.mapHeight && j + 1 < Settings.mapWidth && map[i + 1, j + 1] != 10)
```

```
                map[i + 1, j + 1]++;//5
            if (i - 1 >= 0 && j + 1 < Settings.mapWidth && map[i - 1, j + 1] != 10)
                map[i - 1, j + 1]++;//6
            if (i + 1 < Settings.mapHeight && j - 1 >= 0 && map[i + 1, j - 1] != 10)
                map[i + 1, j - 1]++;//7
            if (i - 1 >= 0 && j - 1 >= 0 && map[i - 1, j - 1] != 10)
                map[i - 1, j - 1]++;//8
        }
      }
   }
   Settings.gameTime = 0;
}
private void sizeGame()
{
   this.Width = Settings.mapWidth * Game.size + Game.empty * 3 + Game.empty / 2;
   this.Height = Settings.mapHeight * Game.size + Game.empty * 2 + 86;
   pictureBox3.Width = this.Width;
   pictureBox3.Height = this.Height - 86;
}

private void addButtons()
{
   for (int i = 0; i < Settings.mapHeight; i++)
   {
      for (int j = 0; j < Settings.mapWidth; j++)
      {
         Button button = new Button();
         button.Location = new Point(Game.size * j + Game.empty, Game.size * i + Game.empty);
         button.Size = new Size(Game.size, Game.size);
         button.Image = imgFind(5, 3);
         button.MouseUp += new MouseEventHandler(pressed);
         this.Controls.Add(button);
         buttons[i, j] = button;
      }
   }
}
private void pressed(object sender, MouseEventArgs e)
{
```

```
    Button pressedButton = (Button)sender;
    switch (e.Button.ToString())
    {
        case "Right":
            pressedRight(pressedButton);
            break;
        case "Left":
            pressedLeft(pressedButton);
            break;
    }
}

private void pressedRight(Button pressedButton)
{
    int k = 0;
    int i = pressedButton.Location.Y / Game.size;
    int j = pressedButton.Location.X / Game.size;
    if (Settings.mapFlag > 0)
    {
        if (map[i, j] >= 0)
        {
            switch (map[i, j])
            {
                case 0:
                    map[i, j] = -11;
                    buttons[i, j].Image = imgFind(4, 3);
                    Settings.mapFlag--;
                    k++;
                    break;
                case 1:
                    map[i, j] = -1;
                    buttons[i, j].Image = imgFind(4, 3);
                    Settings.mapFlag--;
                    k++;
                    break;
                case 2:
                    map[i, j] = -2;
                    buttons[i, j].Image = imgFind(4, 3);
```

```
            Settings.mapFlag--;
            k++;
            break;
        case 3:
            map[i, j] = -3;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
            break;
        case 4:
            map[i, j] = -4;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
            break;
        case 5:
            map[i, j] = -5;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
            break;
        case 6:
            map[i, j] = -6;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
            break;
        case 7:
            map[i, j] = -7;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
            break;
        case 8:
            map[i, j] = -8;
            buttons[i, j].Image = imgFind(4, 3);
            Settings.mapFlag--;
            k++;
```

```
                break;
            case 10:
                map[i, j] = -10;
                buttons[i, j].Image = imgFind(4, 3);
                Settings.mapFlag--;
                k++;
                break;
        }
    }
}

if (map[i, j] < 0 && map[i, j] >= -11 && k == 0)
{
    switch (map[i, j])
    {
        case -11:
            map[i, j] = 0;
            buttons[i, j].Image = imgFind(2, 3);
            Settings.mapFlag++;
            break;
        case -10:
            map[i, j] = 10;
            buttons[i, j].Image = imgFind(2, 3);
            Settings.mapFlag++;
            break;
        case -8:
            map[i, j] = 8;
            buttons[i, j].Image = imgFind(2, 3);
            Settings.mapFlag++;
            break;
        case -7:
            map[i, j] = 7;
            buttons[i, j].Image = imgFind(2, 3);
            Settings.mapFlag++;
            break;
        case -6:
            map[i, j] = 6;
            buttons[i, j].Image = imgFind(2, 3);
```

```
                Settings.mapFlag++;
                break;
            case -5:
                map[i, j] = 5;
                buttons[i, j].Image = imgFind(2, 3);
                Settings.mapFlag++;
                break;
            case -4:
                map[i, j] = 4;
                buttons[i, j].Image = imgFind(2, 3);
                Settings.mapFlag++;
                break;
            case -3:
                map[i, j] = 3;
                buttons[i, j].Image = imgFind(2, 3);
                Settings.mapFlag++;
                break;
            case -2:
                map[i, j] = 2;
                buttons[i, j].Image = imgFind(2, 3);
                Settings.mapFlag++;
                break;
            case -1:
                map[i, j] = 1;
                buttons[i, j].Image = imgFind(2, 3);
                Settings.mapFlag++;
                break;
        }
    }
    textBox2.Text = Settings.mapFlag.ToString();
}

private void stepStart(int i, int j)
{
    int t = 0;
    Queue coordXY = new Queue();
    coordXY.Enqueue(i);
    coordXY.Enqueue(j);
```

```
map[i, j] = 11;
while (coordXY.Count > 0)
{
   t++;
   if (t % 2 != 0)
   {
      i = (int)coordXY.Dequeue();
   }
   if (t % 2 == 0)
   {
      j = (int)coordXY.Dequeue();
      if (map[i, j] == 11)
      {
         visibility(i, j);
         if (i - 1 >= 0 && (map[i - 1, j] == 0 || map[i - 1, j] == -11))
         {
            if (map[i - 1, j] == -11)
            {
               Settings.mapFlag++;
            }
            coordXY.Enqueue(i - 1);
            coordXY.Enqueue(j);
            map[i - 1, j] = 11;
         }
         if (i + 1 < Settings.mapHeight && (map[i + 1, j] == 0 || map[i + 1, j] == -11))
         {
            if (map[i + 1, j] == -11)
            {
               Settings.mapFlag++;
            }
            coordXY.Enqueue(i + 1);
            coordXY.Enqueue(j);
            map[i + 1, j] = 11;
         }
         if (j - 1 >= 0 && (map[i, j - 1] == 0 || map[i, j - 1] == -11))
         {
            if (map[i, j - 1] == -11)
            {
```

```
            Settings.mapFlag++;
        }
        coordXY.Enqueue(i);
        coordXY.Enqueue(j - 1);
        map[i, j - 1] = 11;
    }
    if (j + 1 < Settings.mapWidth && (map[i, j + 1] == 0 || map[i, j + 1] == -11))
    {
        if (map[i, j + 1] == -11)
        {
            Settings.mapFlag++;
        }
        coordXY.Enqueue(i);
        coordXY.Enqueue(j + 1);
        map[i, j + 1] = 11;
    }
    if (i + 1 < Settings.mapHeight && j + 1 < Settings.mapWidth && (map[i + 1, j + 1] == 0 || map[i + 1, j + 1] == -
11))
    {
        if (map[i + 1, j + 1] == -11)
        {
            Settings.mapFlag++;
        }
        coordXY.Enqueue(i + 1);
        coordXY.Enqueue(j + 1);
        map[i + 1, j + 1] = 11;
    }
    if (j + 1 < Settings.mapWidth && i - 1 >= 0 && (map[i - 1, j + 1] == 0 || map[i - 1, j + 1] == -11))
    {
        if (map[i - 1, j + 1] == -11)
        {
            Settings.mapFlag++;
        }
        coordXY.Enqueue(i - 1);
        coordXY.Enqueue(j + 1);
        map[i - 1, j + 1] = 11;
    }
    if (j - 1 >= 0 && i + 1 < Settings.mapHeight && (map[i + 1, j - 1] == 0 || map[i + 1, j - 1] == -11))
```

```
{
    if (map[i + 1, j - 1] == -11)
    {
        Settings.mapFlag++;
    }
    coordXY.Enqueue(i + 1);
    coordXY.Enqueue(j - 1);
    map[i + 1, j - 1] = 11;
}
if (i - 1 >= 0 && j - 1 >= 0 && (map[i - 1, j - 1] == 0 || map[i - 1, j - 1] == -11))
{
    if (map[i - 1, j - 1] == -11)
    {
        Settings.mapFlag++;
    }
    coordXY.Enqueue(i - 1);
    coordXY.Enqueue(j - 1);
    map[i - 1, j - 1] = 11;
}



if (i - 1 >= 0 && map[i - 1, j] != 0 && map[i - 1, j] != 11)
{
    visibility(i - 1, j);
}
if (i + 1 < Settings.mapHeight && map[i + 1, j] != 0 && map[i + 1, j] != 11)
    visibility(i + 1, j);

if (j - 1 >= 0 && map[i, j - 1] != 0 && map[i, j - 1] != 11)
    visibility(i, j - 1);

if (j + 1 < Settings.mapWidth && map[i, j + 1] != 0 && map[i, j + 1] != 11)
    visibility(i, j + 1);

if (i + 1 < Settings.mapHeight && j + 1 < Settings.mapWidth && map[i + 1, j + 1] != 0 && map[i + 1, j + 1] != 11)
    visibility(i + 1, j + 1);

if (j + 1 < Settings.mapWidth && i - 1 >= 0 && map[i - 1, j + 1] != 0 && map[i - 1, j + 1] != 11)
```

```
                visibility(i - 1, j + 1);

            if (j - 1 >= 0 && i + 1 < Settings.mapHeight && map[i + 1, j - 1] != 0 && map[i + 1, j - 1] != 11)
                visibility(i + 1, j - 1);

            if (i - 1 >= 0 && j - 1 >= 0 && map[i - 1, j - 1] != 0 && map[i - 1, j - 1] != 11)
                visibility(i - 1, j - 1);
        }
    }
  }
  textBox2.Text = Settings.mapFlag.ToString();

  for (i = 0; i < Settings.mapHeight; i++)
  {
    for (j = 0; j < Settings.mapWidth; j++)
    {
      if (map[i, j] == 11)
        map[i, j] = 0;
    }
  }
}

private void pressedLeft(Button pressedButton)
{
  int i = pressedButton.Location.Y / Game.size;
  int j = pressedButton.Location.X / Game.size;
  if (map[i, j] >= 0)
  {
    if (Game.first)
    {
      mapGenerate(i, j);
      Game.first = false;
    }
    if (map[i, j] != 0)
      visibility(i, j);
    else
      stepStart(i, j);
  }
```

```
}
private void visibility(int i, int j)
{
    if (map[i, j] >= 0)
    {
        if (map[i, j] == 0 || map[i, j] == 11)
        {
            if (map[i, j] == -11)
                map[i, j] = 11;
            buttons[i, j].Enabled = false;
            buttons[i, j].Image = imgFind(1, 4);
            Settings.gameWin++;
        }
        if (map[i, j] == 1)
        {
            buttons[i, j].Enabled = false;
            buttons[i, j].Image = imgFind(1, 1);
            Settings.gameWin++;
            map[i, j] = 99;
        }
        if (map[i, j] == 2)
        {
            buttons[i, j].Enabled = false;
            buttons[i, j].Image = imgFind(2, 1);
            Settings.gameWin++;
            map[i, j] = 99;
        }
        if (map[i, j] == 3)
        {
            buttons[i, j].Enabled = false;
            buttons[i, j].Image = imgFind(3, 1);
            Settings.gameWin++;
            map[i, j] = 99;
        }
        if (map[i, j] == 4)
        {
            buttons[i, j].Enabled = false;
            buttons[i, j].Image = imgFind(4, 1);
```

```
    Settings.gameWin++;
    map[i, j] = 99;
}
if (map[i, j] == 5)
{
    buttons[i, j].Enabled = false;
    buttons[i, j].Image = imgFind(5, 1);
    Settings.gameWin++;
    map[i, j] = 99;
}
if (map[i, j] == 6)
{
    buttons[i, j].Enabled = false;
    buttons[i, j].Image = imgFind(1, 2);
    Settings.gameWin++;
    map[i, j] = 99;
}
if (map[i, j] == 7)
{
    buttons[i, j].Enabled = false;
    buttons[i, j].Image = imgFind(2, 2);
    Settings.gameWin++;
    map[i, j] = 99;
}
if (map[i, j] == 8)
{
    buttons[i, j].Enabled = false;
    buttons[i, j].Image = imgFind(3, 2);
    Settings.gameWin++;
    map[i, j] = 99;
}
if (map[i, j] == 10)
{
    buttons[i, j].Enabled = false;
    buttons[i, j].Image = imgFind(5, 2);
    map[i, j] = 100;
    endGame(0);
}
```

```
            if ((Settings.gameWin == (Settings.mapWidth * Settings.mapHeight) - Settings.mapMin) && (Settings.mapHeight + Settings.mapWidth != 2))
                endGame(1);


            if (map[i, j] < 0 || map[i, j] >= -11)
            {
                switch (map[i, j])
                {
                    case -11:
                        buttons[i, j].Enabled = false;
                        map[i, j] = 0;
                        buttons[i, j].Image = imgFind(1, 4);
                        Settings.mapFlag++;
                        break;
                    case -10:
                        buttons[i, j].Enabled = false;
                        map[i, j] = 10;
                        buttons[i, j].Image = imgFind(5, 2);
                        Settings.mapFlag++;
                        break;
                    case -8:
                        buttons[i, j].Enabled = false;
                        map[i, j] = 8;
                        buttons[i, j].Image = imgFind(3, 2);
                        Settings.mapFlag++;
                        break;
                    case -7:
                        buttons[i, j].Enabled = false;
                        map[i, j] = 7;
                        buttons[i, j].Image = imgFind(2, 2);
                        Settings.mapFlag++;
                        break;
                    case -6:
                        buttons[i, j].Enabled = false;
                        map[i, j] = 6;
                        buttons[i, j].Image = imgFind(1, 2);
                        Settings.mapFlag++;
                        break;
```

```
                case -5:
                    buttons[i, j].Enabled = false;
                    map[i, j] = 5;
                    buttons[i, j].Image = imgFind(5, 1);
                    Settings.mapFlag++;
                    break;
                case -4:
                    buttons[i, j].Enabled = false;
                    map[i, j] = 4;
                    buttons[i, j].Image = imgFind(4, 1);
                    Settings.mapFlag++;
                    break;
                case -3:
                    buttons[i, j].Enabled = false;
                    map[i, j] = 3;
                    buttons[i, j].Image = imgFind(3, 1);
                    Settings.mapFlag++;
                    break;
                case -2:
                    buttons[i, j].Enabled = false;
                    map[i, j] = 2;
                    buttons[i, j].Image = imgFind(2, 1);
                    Settings.mapFlag++;
                    break;
                case -1:
                    buttons[i, j].Enabled = false;
                    map[i, j] = 1;
                    buttons[i, j].Image = imgFind(1, 1);
                    Settings.mapFlag++;
                    break;
            }
            textBox2.Text = Settings.mapFlag.ToString();
        }
    }
}
private void endGame(int game)
{
    if (Game.stop == false)
```

```
{
    timer1.Enabled = false;
    switch (game)
    {
        case 0:
            button1.Image = Image.FromFile("files\\zle.png");
            if (Properties.Settings.Default.Language == "en-US")
                MessageBox.Show("Defeat -_-");
            else if (Properties.Settings.Default.Language == "pl-PL")
                MessageBox.Show("Pokonać -_-");
            else
                MessageBox.Show("Поразка -_-");
            for (int i = 0; i < Settings.mapHeight; i++)
            {
                for (int j = 0; j < Settings.mapWidth; j++)
                {
                    buttons[i, j].Enabled = false;
                    if (map[i, j] == 10 || map[i, j] == -10)
                    {
                        buttons[i, j].Image = imgFind(4, 2);
                    }
                }
            }
            break;
        case 1:
            recordWrite();
            if (Properties.Settings.Default.Language == "en-US")
                MessageBox.Show("Victory ^_^");
            else if (Properties.Settings.Default.Language == "pl-PL")
                MessageBox.Show("Zwycięstwo ^_^");
            else
                MessageBox.Show("Перемога ^_^");
            for (int i = 0; i < Settings.mapHeight; i++)
            {
                for (int j = 0; j < Settings.mapWidth; j++)
                {
                    if (map[i, j] == 10 || map[i, j] == -10)
                    {
```

```
                    buttons[i, j].Enabled = false;
                    buttons[i, j].Image = imgFind(4, 3);
                }
            }
        }
        break;
    }
    Game.stop = true;
  }
}

private async Task recordWrite()
{
  int check = 0;
  if (Settings.gameLevel == 1)
  {
    string[] readText = File.ReadAllLines("files\\easy.txt");
    for (int i = 0; i < System.IO.File.ReadAllLines("files\\easy.txt").Length; i++)
    {
      if (readText[i] == Settings.gameName)
      {
        check = 1;
      }
      if (check == 1 && int.Parse(readText[i + 1]) > Settings.gameTime)
      {
        readText[i + 1] = Settings.gameTime.ToString();
        File.WriteAllLines("files\\easy.txt", readText);
        check = 2;
        break;
      }
    }
    if (check == 0)
    {
      StreamWriter writer = new StreamWriter("files\\easy.txt", true);
      writer.WriteLineAsync(Settings.gameName.ToString());
      writer.WriteLineAsync(Settings.gameTime.ToString());
      writer.Close();
    }
```

```
}
if (Settings.gameLevel == 2)
{
    string[] readText = File.ReadAllLines("files\\medium.txt");
    for (int i = 0; i < System.IO.File.ReadAllLines("files\\medium.txt").Length; i++)
    {
        if (readText[i] == Settings.gameName)
        {
            check = 1;
        }
        if (int.Parse(readText[i + 1]) > Settings.gameTime && check == 1)
        {
            readText[i + 1] = Settings.gameTime.ToString();
            File.WriteAllLines("files\\medium.txt", readText);
            check = 2;
            break;
        }
    }
    if (check == 0)
    {
        StreamWriter writer = new StreamWriter("files\\medium.txt", true);
        writer.WriteLineAsync(Settings.gameName.ToString());
        writer.WriteLineAsync(Settings.gameTime.ToString());
        writer.Close();
    }
}
if (Settings.gameLevel == 3)
{
    string[] readText = File.ReadAllLines("files\\hard.txt");
    for (int i = 0; i < System.IO.File.ReadAllLines("files\\hard.txt").Length; i++)
    {
        if (readText[i] == Settings.gameName)
        {
            check = 1;
        }
        if (int.Parse(readText[i + 1]) > Settings.gameTime && check == 1)
        {
            readText[i + 1] = Settings.gameTime.ToString();
```

```
                File.WriteAllLines("files\\hard.txt", readText);
                check = 2;
                break;
            }
        }
        if (check == 0)
        {
            StreamWriter writer = new StreamWriter("files\\hard.txt", true);
            writer.WriteLineAsync(Settings.gameName.ToString());
            writer.WriteLineAsync(Settings.gameTime.ToString());
            writer.Close();
        }
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (Game.first == false && Game.pauza == false)
        Settings.gameTime++;
    textBox1.Text = Settings.gameTime.ToString();
}

private void button1_Click(object sender, EventArgs e)
{
    this.Close();
    Form main = Application.OpenForms[0];
    main.Hide();
    Form2 game = new Form2();
    game.Show();
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
```

```
    if (Game.pauza == false)
    {
        Game.pauza = true;
        pictureBox3.Visible = true;
    }
    else
    {
        Game.pauza = false;
        pictureBox3.Visible = false;
    }
}
void Form_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Up && Settings.easterEgg == 1)
    {
        Settings.easterEgg = 2;
        e.SuppressKeyPress = true;
    }
    else if (e.KeyCode == Keys.Up)
    {
        Settings.easterEgg = 1;
        e.SuppressKeyPress = true;
    }
    else if (e.KeyCode == Keys.Down && Settings.easterEgg == 2)
    {
        Settings.easterEgg = 3;
        e.SuppressKeyPress = true;
    }
    else if (e.KeyCode == Keys.Down && Settings.easterEgg == 3)
    {
        Settings.easterEgg = 4;
        e.SuppressKeyPress = true;
    }
    else if (e.KeyCode == Keys.Left && Settings.easterEgg == 4)
    {
        Settings.easterEgg = 5;
        e.SuppressKeyPress = true;
    }
```

```
else if (e.KeyCode == Keys.Right && Settings.easterEgg == 5)
{
   Settings.easterEgg = 6;
   e.SuppressKeyPress = true;
}
else if (e.KeyCode == Keys.Left && Settings.easterEgg == 6)
{
   Settings.easterEgg = 7;
   e.SuppressKeyPress = true;
}
else if (e.KeyCode == Keys.Right && Settings.easterEgg == 7)
{
   Settings.easterEgg = 8;
   e.SuppressKeyPress = true;
}
else if (e.KeyCode == Keys.B && Settings.easterEgg == 8)
{
   Settings.easterEgg = 9;
   e.SuppressKeyPress = true;
}
else if (e.KeyCode == Keys.A && Settings.easterEgg == 9)
{
   if (Settings.konami == true)
   {
      Settings.easterEgg = 10;
      BackgroundImage = Image.FromFile("files\\background.jpg");
      Settings.konami = false;
   }
   else
   {
      Settings.easterEgg = 10;
      BackgroundImage = Image.FromFile("files\\space.jpg");
      Settings.konami = true;
      e.SuppressKeyPress = true;
   }
}
else
{
```

```
            Settings.easterEgg = 0;
            e.SuppressKeyPress = true;
        }
    }
  }
}
```

Лістинг Form3:

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

namespace main
{
    public partial class Form3 : Form
    {
        public class RecordEasy
        {
            public string easyName { get; set; }
            public int easyTime { get; set; }
        }
        public class RecordMedium
        {
            public string mediumName { get; set; }
            public int mediumTime { get; set; }
        }
        public class RecordHard
        {
            public string hardName { get; set; }
            public int hardTime { get; set; }
        }
        public Form3()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            DoubleBuffered = true;
            if (Settings.konami == true)
            {
                BackgroundImage = Image.FromFile("files\\space.jpg");
                richTextBox1.BackColor = Color.Black;
                richTextBox2.BackColor = Color.Black;
                richTextBox3.BackColor = Color.Black;
                textBox4.BackColor = Color.Black;
                textBox5.BackColor = Color.Black;
                textBox6.BackColor = Color.Black;
            }
            Settings.recordCount = 0;
            this.FormClosing += Form3_Closing;
            FormBorderStyle = FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
```

```
            Settings.recordList = 0;
            read_Easy();
            read_Medium();
            read_Hard();
        }

        private void Form3_Closing(Object sender, FormClosingEventArgs e)
        {
            Form main = Application.OpenForms[0];
            if (Settings.konami == true)
                main.BackgroundImage = Image.FromFile("files\\space.jpg");
            main.Show();
        }

        private async Task read_Hard()
        {
            richTextBox3.Text = null;
            textBox6.Text = null;
            List<RecordHard> recordHard = new List<RecordHard>();
            string line;
            StreamReader readerHard = File.OpenText("files\\hard.txt");
            for (int i = 0; (line = await readerHard.ReadLineAsync()) != null; i++)
            {
                recordHard.Add(new RecordHard() { hardName = line, hardTime =
int.Parse(await readerHard.ReadLineAsync()) });
                if (i > Settings.recordCount)
                    Settings.recordCount = i;
            }
            readerHard.Close();

            if (Properties.Settings.Default.Language == "en-US")
                richTextBox3.Text += "\t\tHard";
            else if (Properties.Settings.Default.Language == "pl-PL")
                richTextBox3.Text += "\t\tTrudny";
            else
                richTextBox3.Text += "\t\tВажкий";

            recordHard.Sort(delegate (RecordHard a, RecordHard b)
            { return a.hardTime.CompareTo(b.hardTime); });
            for (int i = 14 * Settings.recordList; i < 14 * (Settings.recordList + 1)
|| i < recordHard.Count; i++)
            {
                richTextBox3.Text += "\n" + (i + 1) + "." + recordHard[i].hardName;
                textBox6.Text += "\t" + recordHard[i].hardTime;
            }
        }


        private async Task read_Medium()
        {
            richTextBox2.Text = null;
            textBox5.Text = null;
            List<RecordMedium> recordMedium = new List<RecordMedium>();
            string line;
            StreamReader readerMedium = File.OpenText("files\\medium.txt");
            for (int i = 0; (line = await readerMedium.ReadLineAsync()) != null; i++)
            {
                recordMedium.Add(new RecordMedium() { mediumName = line, mediumTime =
int.Parse(await readerMedium.ReadLineAsync()) });
                if (i > Settings.recordCount)
                    Settings.recordCount = i;
            }
            readerMedium.Close();

            if (Properties.Settings.Default.Language == "en-US")
```

```
                richTextBox2.Text += "\t\tMedium";
            else if (Properties.Settings.Default.Language == "pl-PL")
                richTextBox2.Text += "\t\tŚrednio";
            else
                richTextBox2.Text += "\t\tСередній";

            recordMedium.Sort(delegate (RecordMedium a, RecordMedium b)
            { return a.mediumTime.CompareTo(b.mediumTime); });
            for (int i = 14 * Settings.recordList; i < 14 * (Settings.recordList + 1)
|| i < recordMedium.Count; i++)
            {
                richTextBox2.Text += "\n" + (i + 1) + "." +
recordMedium[i].mediumName;
                textBox5.Text += "\t" + recordMedium[i].mediumTime;
            }
        }
        private async Task read_Easy()
        {
            richTextBox1.Text = null;
            textBox4.Text = null;
            List<RecordEasy> recordEasy = new List<RecordEasy>();
            string line;
            StreamReader readerEasy = File.OpenText("files\\easy.txt");
            for (int i = 0; (line = await readerEasy.ReadLineAsync()) != null; i++)
            {
                recordEasy.Add(new RecordEasy() { easyName = line, easyTime =
int.Parse(await readerEasy.ReadLineAsync()) });
                if (i > Settings.recordCount)
                    Settings.recordCount = i;
            }
            readerEasy.Close();

            if (Properties.Settings.Default.Language == "en-US")
                richTextBox1.Text += "\t\tEasy";
            else if (Properties.Settings.Default.Language == "pl-PL")
                richTextBox1.Text += "\t\tŁatwo";
            else
                richTextBox1.Text += "\t\tЛегкий";

            recordEasy.Sort(delegate (RecordEasy a, RecordEasy b)
             { return a.easyTime.CompareTo(b.easyTime); });
            for (int i = 14 * Settings.recordList; i < 14 * (Settings.recordList + 1)
&& i < recordEasy.Count; i++)
            {
                richTextBox1.Text += "\n" + (i + 1) + "." + recordEasy[i].easyName;
                textBox4.Text += "\t" + recordEasy[i].easyTime;
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (Settings.recordList >= 1)
            {
                Settings.recordList--;
                read_Easy();
                read_Medium();
                read_Hard();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (14 * (Settings.recordList + 1) < Settings.recordCount + 1)
            {
                Settings.recordList++;
```

```
                read_Easy();
                read_Medium();
                read_Hard();
            }
        }
        private void button3_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void richTextBox1_TextChanged(object sender, EventArgs e) { }
        private void textBox1_TextChanged(object sender, EventArgs e) { }
    }
}
```