НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський Політехнічний Інститут імені Ігоря Сікорського»

Кафедра Технічної кібернетики

Ігрова програма «Сапер»

Курсова робота

3 дисципліни «Основи програмування»

Керівник Виконавець

ас. Тимофєєва Ю.С. ст. Журба М.А.

зал. книжка № ІТ-7402

«Допущений до захисту» гр. ІТ-74

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(особистий підпис керівника) (особистий підпис виконавця)

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018р. «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018р.

Захищений з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка)

Київ – 2018

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський Політехнічний Інститут імені Ігоря Сікорського»

Кафедра технічної кібернетики

Дисципліна «Основи програмування»

Курс\_\_\_\_ Група\_\_\_\_\_\_\_ Семестр\_\_\_\_

ЗАВДАННЯ

на курсову роботу студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1.Тема роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.Строк здачі студентом закінченої роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.Вихідні дані до роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.Зміст пояснювальної записки (перелік питань, які підлягають розробці)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів виконання курсової роботи | Строк виконання етапів роботи | Підписи або примітки |
| 1. | Отримання та узгодження теми курсової роботи |  |  |
| 2. | Огляд існуючих рішень з тематики роботи |  |  |
| 3. | Розробка моделі та структури програми |  |  |
| 4. | Вибір алгоритмів розв'язання задачі |  |  |
| 5. | Розробка діалогового інтерфейсу програми |  |  |
| 6. | Кодування програми |  |  |
| 7. | Налагодження та перевірка програми |  |  |
| 8. | Оформлення пояснювальної записки |  |  |
| 9. | Захист курсової роботи |  |  |
| 10. |  |  |  |
| 11. |  |  |  |
| 12. |  |  |  |
| 13. |  |  |  |
| 14. |  |  |  |
| 15. |  |  |  |
| 16. |  |  |  |
| 17. |  |  |  |
| 18. |  |  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(особистий підпис керівника) (особистий підпис виконавця)

АНОТАЦІЯ

У даному документі міститься вся інформація щодо програми, розробленої відповідно до теми курсової роботи, гри Сапер. Тут присутні короткі відомості про гру, опис математичної моделі та структури програми, режимів програми, взаємодію програми з користувачем. Детально розказано про використані функції та алгоритми, порядок їх визову, їхню взаємодію. Описані вхідні та вихідні дані. Додана інструкція по інсталяції та видаленню програми з комп'ютера, а також присутні умови проектування програми та її застосування. В кінці є висновок, який містить інформацію про результати, а в додатку наводиться текст програми та скріншоти з прикладами роботи програми.

ЗМІСТ

ВСТУП6

1. МАТЕМАТИЧНА МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ7
2. ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ8
3. СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ11
4. ОПИС ПРОГРАМИ14
5. ІНСТРУКЦІЯ ПРОГРАМІСТА20
6. КЕРІВНИЦТВО КОРИСТУВАЧА22

ВИСНОВКИ25

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ26

ДОДАТОК А27

ДОДАТОК Б71

ВСТУП

Minesweeper або Сапер – це однокористувацька гра-головоломка. Була створена у 60х роках, а у 70х почала здобувати значну популярність . Згодом було розроблено багато варіацій, наприклад просторовий Сапер, Сапер в режимі шестикутників та інші, проте головна ідея гри залишалась і залишається однаковою: ігрове поле розділено на комірки, частину яких заміновано. Гравцю необхідно відкрити всі комірки без мін.

Програма була розроблена з метою вдосконалення старих версій гри, а також демонстрації здобутих у курсі «[Основи програмування - 2. Програмування структурованих даних](http://wiki.kpi.ua/index.php/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20-%202.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%85%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_(20002020))» навичок роботи з технологією Windows Forms.

Використані програмні засоби: Visual Studio 2017, Advanced Installer 14.8.

1. МАТЕМАТИЧНА МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ

За основу було взято варіант Саперу від Microsoft, до якого були додані певні вдосконалення.

По-перше, після першого натискання відкривається одразу певна частина ігрового поля. Це досягається тим, що в натиснутій комірці та навколо неї міни відсутні.

По-друге, після відкриття всіх незамінованих комірок, або після правильного позначення всіх замінованих комірок прапорцями, гра завершується перемогою.

По-третє, коли значення у комірці дорівнює кількості невідкритих комірок навколо неї, при натисканні правої кнопки миші відбувається автоматичне позначення цих комірок прапорцями.

По-четверте, коли значення у комірці дорівнює кількості позначених прапорцями комірок навколо неї, при натисканні лівої кнопки миші відбувається автоматичне відкриття закритих комірок навколо, якщо такі є.

Перше покращення зроблене з метою дати гравцю можливість нормально розпочати рівень, а не програвати після першого ж натискання, друге — для уникнення програшу у ситуаціях, коли неможливо стовідсотково відкрити комірку. Останні два покращення допомагають гравцю проходити рівні набагато швидше, уникаючи зайвих натискань.

Як математичну модель було обрано двовимірний масив. Тип цього масиву - самостійно створений тип класу­­, який дає можливість зберігати та обробляти повну інформацію про комірку на ігровому полі.

1. ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ
   1. Режими роботи програми
      1. Перелік режимів роботи програми.

Гра має чотири рівні складності:

* «Can I play, daddy?». Найлегший рівень. Розмір ігрового поля - 10Х10. Кількість мін - дев'ять (9%)
* «Don't hurt me». Середній рівень. Розмір ігрового поля - 16Х16. Кількість мін - сорок (15,6%)
* «Bring 'em on!». Складний рівень. Розмір ігрового поля - 16Х30. Кількість мін - дев'яносто дев'ять (20,6%)
* «I am death incarnate!» Надскладний рівень. Розмір ігрового поля - 20Х35. Кількість мін - сто сімдесят п'ять (25%)

Жартівливе найменування рівнів мотивує гравця на проходження складніших рівнів. Так як на останньому рівні міни складають чверть всього поля, він є дуже складний для проходження, тому легко отримати велику аудиторію гравців, які довго намагатимуться пройти гру повністю.

2.1.2 Опис режимів роботи програми.

Для реалізації рівнів була написана функція Game, що приймає цілочисельні значення n,m,k - розміри ігрового поля та кількість мін на ньому. Саме в залежності від обраного рівня й будуть передаватися різні значення. У функції створюється двовимірний масив і заповнюється комірками. Кожна комірка створюється програмно. Ігрове поле відображається на екрані, потім очікується перше натискання. Після нього відбувається генерація самого поля. k разів вибираються випадково згенеровані координати для міни. Міна буде поставлена у відповідну позицію, якщо виконуються наступні умови:

* у цій позиції ще нема міни
* згенерована позиція не потрапляє в окіл комірки першого натискання.
  1. Структура діалогу програми

При запуску програми користувач потрапляє в головне меню. Там він побачить кнопки для переходу до чотирьох вищезазначених рівнів. Під кожною з кнопок буде відображатися статистична інформація по проходженню рівня, а саме: кількість перемог, відсоток перемог, найкращий час, середній час.

Під час гри користувач може повернутися до головного меню, вибрати інший рівень, відкрити Help-файл або закрити програму.

Все це вибирається у меню-шапці.

* 1. Склад та структура меню користувача
     1. Перелік елементів меню форми головного меню програми
* «Can I play, daddy?» - кнопка для початку легкого рівня
* «Don't hurt me» - кнопка для початку середнього рівня
* «Bring 'em on!» - кнопка для початку складного рівня
* «I am death incarnate!» - кнопка для надскладного рівня

2.3.2 Перелік елементів меню ігрової форми програми.

* «New game» у меню-шапці пропонує вибрати один з пунктів випадаючого списку, а саме перехід до головного меню або до іншого рівня
* «Help». Відкриває HTML - файл з описом правил гри з прикладами
* «Exit». Кнопка для закриття програми. Перед закриттям відбувається збереження даних про статистику рівня.

2.3.3 Формат виведення на екран повідомлень та засоби виклику допомоги та довідкової інфорації.

У головному меню на екрані відображається статистична інформація по проходженню кожному з рівнів у вигляду тексту.

Також після закінчення рівня на екрані спливе повідомлення про перемогу або поразку гравця, в залежності від результату гри.

У шапці меню присутня кнопка «Help», після натискання якої у браузері буде відкрито HTML - сторінку з правилами гри, порадами та прикладами.

1. СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ

3.1 Розроблені класи

Було створено клас Field. Так як взаємодія з коміркою ігрового поля нагадує взаємодію зі звичайним Button (натискання лівою чи правою кнопкою миші), то створений клас Field наслідує Button та має деякі додаткові властивості. Він потрібен для зберігання інформації про певну комірку, та зв'язку її з двовимірним масивом.

Перелік полів класу Field:

* int x,y - поля цілочисельного типу, координати положення комірки на ігровому полі
* bool Checked - поле логічного типу, дає відповідь на те, чи комірка відкрита
* bool Flagged - поле логічного типу, дає відповідь на те, чи комірку було помічено прапорцем
* char VALUE - поле символьного типу, зберігає інформацію про тип комірки: 'm' - якщо комірка з міною, '0' - '8' - якщо комірка без міни.

У класі відсутній конструктор, прототип класу Field створюється порожнім, присутні гетери та сетери для керованого доступу до стану об'єктів.

3.2 Опис глобальних змінних

Наступні змінні використовуються для забезпечення правильного функціонування програми:

* field - двовимірний масив типу Field, допоміжна зміна, призначена для збереження всіх комірок та проводження операцій з ними у зручній формі
* date, stopWatch - змінні типу DateTime, допоміжні змінні, необхідні для роботи секундоміра
* timer - змінна типу Timer, тимчасова змінна, необхідна для опису секундоміру для відображення поточного часу гри
* fieldHeight, fieldWidth - змінні цілочисельного типу, допоміжні змінні, зберігають інформацію про розміри поля
* minesQuantity - змінна цілочисельного типу, допоміжна змінна, зберігає інформацію про кількість мін на ігровому полі
* levelDifficulty - змінна цілочисельного типу, допоміжна змінна, зберігає інформацію про складність рівня
* nodeCounter - змінна цілочисельного типу, допоміжна змінна, зберігає інформацію про кількість вже відкритих комірок
* mineCounter - змінна цілочисельного типу, допоміжна змінна, зберігає інформацію про кількість правильно позначених прапорцями мін
* firstClick - змінна логічного типу, допоміжна змінна, дає відповідь за запитання, чи було зроблене перше натискання на кормірку.

Наступні змінні використовуються для створення елементів інтерфейсу:

* btnEasyLvl, btnLowmediumLvl, btnHighmediumLvl, btnHardLvl - змінні типу Button, допоміжні змінні, необхідні для створення кнопок для переходу до рівнів
* lStopwatch, lMines, easyLvlLabelStats, easyLvlLabelTime, lowmediumLvlLabelStats, lowmediumLvlLabelTime, highmediumLvlLabelStats, highmediumLvlLabelTime, hardLvlLabelStats, hardLvlLabelTime - змінні типу Label, допоміжні змінні, поля для виводу тексту з інформацією про статистику рівнів
* minesLogo, easyLvlCrown, easyLvlStopwatch, lowmediumLvlCrown, lowmediumLvlStopwatch, highmediumLvlCrown, highmediumLvlStopwatch, hardLvlCrown, hardLvlStopwatch - змінні типу PictureBox, допоміжні змінні, необхідні для відображення картинок з позначками у головному меню
* easyLvlInfo, lowmediumLvlInfo, highmediumLvlInfo, hardLvlInfo - масиви типу long, допоміжні змінні, необхідні для збереження інформації про статистику рівнів. Кожний масив містить чотири елементи: кількість перемог, кількість зіграних разів, найкращий час у мілісекундах, середній час у мілісекундах
* info - масив типу string, допоміжна змінна, сюди зчитається вся інформація про статистику рівнів з відповідного файлу, містить 4 елемента. Якщо користувач пошкодить або видалить файл, це не призведе до зупинки програми, всі дані просто обнулятися, файл буде створено програмно
* text - змінна строкового типу, допоміжна змінна, необхідна для формування тексту, який буде записано у вищезгадані Label-и або у файл зі збереженою інформацією про статистику рівнів

3.3 Файли використані у програмі

Перелік використаних файлів:

* times.txt - текстовий файл зі збереженою інформацією про статистику рівнів. Він містить чотири рядка , у кожному з яких через пробіл міститься чотири числа: кількість перемог, кількість зіграних разів, найкращий час у мілісекундах, середній час у мілісекундах. Використовується як на зчитування, так і на запис.
* rules.html - HTML - сторінка з правилами гри. Відкривається при натисканні кнопки Help під час користування програмою.

1. ОПИС ПРОГРАМИ

4.1 Опис функцій програми

Далі наведено перелік усіх функцій, що використовувались у програмі з їх детальним поясненням.

* msForm\_Load(object sender, EventArgs e) - подія, що відбувається при завантаженні головної форми, тобто при запуску програми. Тут створюються елементи головного меню: Buttons, Labels, PuctureBoxes (про них було сказано у розділі 3.2), завантажується збережена інформація
* readInfo(long[] arr, int index) - функція зчитування даних про статистику рівня. Приймає масив arr - куди вписати інформацію, index - звідки зчитується ця інформація у масиві info.

Як було зазначено у підрозділі 3.2, масив info містить 4елемента, кожен з яких - рядок з чотирьох чисел.

Функція розділяє рядок по пробілу, конвертує кожен з отриманих елементів з рядку у число і записує у переданий масив arr

* Clearing() - функція, що обнуляє змінні nodeCounter, mineCounter, firstClick та видаляє всі об'єкти, які не є потрібними у конкретний момент користування програмою (комірки, секундомір, елементи для відображення кількості мін, що залишились).

Вона визивається, коли користувач переходить до головного меню або змінює рівень під час гри.

* \_Update() - функція для оновлення інформації про статистику всіх рівнів, занесення цієї інформації у відповідні масиви та виведення її у елементах головного меню
* hideAll() - функція, яка скриває всі елементи головного меню, роблячи значення Visible кожного з них рівним false
* showAll() - функція, яка відображає всі елементи головного меню, роблячи значення Visible кожного з них рівним true
* Finish(bool passed, int level) - функція, що виконуються при завершенні рівня.

Якщо рівень було пройдено, всі міни позначається прапорцями, а інші комірки відкриваються. Інформація про пройдений рівень оновлюється. Значення нульового та першого елементів масиву \_LvlInfo збільшується на 1, потім порівнюється час, за який було пройдено рівень, зі значенням третього елементу масиву. Якщо він менший, то значенню елементу буде присвоєно час проходження рівня. Потім вираховується середній час по формулі (LvlInfo[3]\*(LvlInfo[0]-1) + ЧАС\_ПРОХОЖЕННЯ) / LvlInfo[0]. Тобто попередній середній час множиться на кількість зіграних разів з перемогою до цієї перемоги, додається ще одне значення часу, а потім ділиться на оновлене значення перемог.

Якщо рівень не було пройдено, всі непомічені міни позначаються мінами, всі інші комірки залишаються без змін. Інформація про рівень оновлюється. У цьому випадку значення першого елементу масиву збільшується на 1

* MainMenuToolStripMenuItem\_Click(object sender, EventArgs e) - подія, що відбувається при натисканні кнопки Main Menu. У ній викликаються функції Clearing, \_Update, showAll. Тобто всі непотрібні елементи знищаться, змінні обнуляться, інформація буде оновлена, а всі елементи головного меню з'являться на екрані.
* Game(int n, int m, int k) - функція, яка відбувається при натисканні кнопки якогось з рівнів. Тут створюється секундомір, який очікує першого натискання і потім починає працювати, створюється ігрове поле розміру n\*m, елементи для відображення кількості непозначених мін.

При створенні комірок для ігрового поля їм задається їхнє положення на ігровому полі, розміри, колір, положення на формі, тип та розмір шрифту, значення Checked і Flagged присвоюються false

* btnEasyLvl\_Click(object sender, EventArgs e), btnLowmediumLvl\_Click(object sender, EventArgs e), btnHighmediumLvl\_Click(object sender, EventArgs e), btnHardLvl\_Click(object sender, EventArgs e) - події, що відбуваються при натисканні на кнопку одного з рівнів у головному меню. У них змінні fieldHeight, fieldWidth, levelDifficulty приймають відповідних значень, а потім викликається функція Game
* tickTimer(object sender, EventArgs e) - подія, яка відбувається при запуску секундоміра ( він вмикається після першого натискання комірки ). У ній постійно вираховується різниця між поточним часом та часом першого натискання. Потім час форматується у строку та виводиться на екран. Це дає можливість користувачу відслідковувати тривалість гри
* countFlags(Field node) - функція, яка приймає комірку як аргумент, вираховує кількість прапорців навколо неї та повертає це значення. Ця функція потрібна для автоматичного відкриття комірок.
* countClosed(Field node) - функція, яка приймає комірку як аргумент, вираховує кількість відкритих комірок навколо неї та повертає це значення. Ця функція потрібна для автоматичного позначення прапорців.
* openField(Field node) - функція, яка виконується, коли комірку було натиснуто лівою кнопкою миші.

У ній спочатку перевіряється, чи було вже зроблене перше натискання. Якщо ні, то перш за все по ігровому полю розставляються міни таким чином, щоб вони не потрапили в окіл комірки першого натискання. Це зроблене для того, щоб на початку рівня одразу відкрилась частина поля. Заводиться таймер з прив'язкою до події tickTimer.

Якщо перше натискання вже було зроблено, можливі наступні випадки:

1) У натиснутій комірці знаходиться міна, ця комірка помічена прапорцем - тоді прапорець знімається, гра продовжується;

2) У натиснутій комірці знаходиться міна, ця комірка не помічена прапорцем - тоді гра закінчується поразкою, користувачу виводиться відповідне повідомлення, викликається функція Finish, у якій аргумент passed приймає значення false;

3) У комірці та навколо неї мін немає, тобто значення комірки - 0 - тоді викликаються функції openField від всіх навколишніх комірок.

Варто зазначити, що у коді передбачені перевірки на те, де знаходиться натиснута комірка, аби не вийти за межі двовимірного масиву;

4) Комірка порожня і невідкрита, навколо неї є міна(міни), тобто значення комірки 1-8 - тоді комірка відкривається;

5) Комірка порожня і відкрита, навколо неї є міна(міни), тобто значення комірки 1-8 - тоді викликається функція countFlags і по можливості виконується виклик функції openField від всіх навколишніх комірок. Якщо кількість відкритих комірок дорівнює розміру поля мінус кількість мін, то рівень завершується з перемогою. Викликаються функція Finish, у якій аргумент passed приймає значення true

* field\_MouseDown(object sender, MouseEventArgs e) - подія, що відбувається при натисканні на комірку. Подія onMouseDown була обрана через те, що на відміну від події onClick, перша дозволяє відрізнити лівий та правий клік миші.

Отже, якщо користувач натиснув на комірку лівою кнопкою миші, то, як вже було зазначено, викликається функція openField від натиснутої комірки.

Якщо користувач натиснув на комірку правою кнопкою миші, то можливі наступні випадки:

1) Комірка закрита і непомічена - тоді комірка помічається прапорцем, якщо комірка позначена правильно (тобто у ній знаходиться міна), то змінна mineCounter збільшується на 1, якщо mineCounter дорівнює кількості мін, то це означає, що всі міни позначено правильно і рівень завершується перемогою. Викликаються функція Finish, у якій аргумент passed приймає значення true;

2) Комірка закрита і помічена - тоді прапорець зникає, якщо у комірці знаходиться міна, то змінна mineCounter зменшується на 1;

3) Комірка відкрита - тоді викликається функція countClosed від натиснутої комірки і по можливості виконується автоматичне позначення навколишніх комірок прапорцями

* helpToolStripMenuItem\_Click(object sender, EventArgs e) - подія, що викликається при натисканні кнопки Help у меню-шапці. У браузері відкривається HTML - сторінка з правилами, прикладами, порадами.
* exitToolStripMenuItem\_Click(object sender, EventArgs e) - подія, що викликається при натисканні кнопки Exit. Вся інформація про статистику рівнів заноситься до файлу. Програма закривається.

Зауваження 1: При позначенні прапорців або відображення мін при поразці, завантажуються відповідні картинки з ресурсів. Якщо користувач пошкодить або видалить ці файли, це не призведе до зупинки програми. Інформація буде відображатися текстом: для позначення прапорця буде використовуватися знак 'f ', для позначення міни - 'M'.

Зауваження 2: Під час розробки програми багато уваги приділялось інтерфейсу гри. На сьогоднішній день інтерфейс займає важливе місце, часто саме гарно оформлений додаток буде завантажений та використаний, хоча може будим менш функціональним.

У підрозділі 6.2 присутні скріншоти програми, де можна побачити, з якою увагою обирались кольори ігрового поля та комірок. Закрите поле має темно-червоний колір. Він не дратує очі та пробуджує уваги й кмітливість. Відкриті комірки мають рожевий колір - він символізує безпеку. Тобто якщо комірка відкрита, то на неї вже не слід звертати увагу. [3]

4.2 Вхідні дані та вихідні дані

Вхідними даними є файл зі збереженою інформацією про статистику рівнів. Він обробляється одразу ж при запуску програми.

При закінченні користування програмою, оновлена статистика заноситься назад до текстового файлу.

* 1. Опис усіх повідомлень

Під час користування програмою користувач може отримати повідомлення

про поразку або перемогу під час проходження рівня, а також повідомлення про відсутність певних файлів, якщо їх було видалено або пошкоджено.

1. ІНСТРУКЦІЯ ПРОГРАМІСТА

5.1 Умови проектування та загальні відомості

Назва програми – Minesweeper (з англійської мови - Сапер).

Додаткове програмне забезпечення під час проектування програми не використовувалося.

Середовище програмування Visual Studio 2017. Мова програмування C#.

5.1.1 Характеристики платформи.

Програма проектувалася на ПК наступними характеристиками:

* процесор Intel Core i5-6200U 2.40GHZ
* кількість оперативної пам’яті 16 ГБ.
* операційна система Windows 10 Pro x64

Для проектування програми не потрібні особливі периферійні пристрої. Достатньо комп’ютерної миші/тачпаду та монітору.

5.2 Керівництво розробника програми

5.2.1 Опис етапів проектування програми.

Етапи розробки програми:

* Завантаження Visual Studio Community 2017 з офіційного сайту Microsoft (https://www.visualstudio.com/ru/downloads/?rr=https%3A%2F%2Fwww.google.com.ua%2F). Встановлення програми, дотримуючись всіх вказівок, які висвічуються при процесі встановлення
* Створення макету проекту: форму головного меню з усіма їх елементами, створюються файли для збереження прогресу гри та результатів користувача, додання файлів - картинок до папки з ресурсами, додання файлу з правилами гри
* Створення необхідного для цієї програми класу Field (дивись підрозділ 3.1) і реалізування усіх необхідніх функцій та обробників подій для елементів раніше створеного інтерфейсу (дивись підрозділ 4.1). Під час цього етапу корисно використовувати офіційну документацію Visual Studio від Microsoft [1]
* Необхідно скомпілювати програму та зібрати виконавчий файл за допомогою стандартного інтерфейсу Visual Studio 2017.

5.2.2 Оформлення інсталяції.

Інсталяційний пакет оформлюється за допомогою програми Advanced Installer 14.8. На цьому етапі слід обрати назву програми, вказати серед необхідного програмного забезпечення .Net Framework версії не менше ніж 4.5. Далі треба додати усі файли, які мають міститися в установчому пакеті, а саме:

* виконавчий файл
* файли результатів користувача
* картинки
* html-файл (help-файл).

1. КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Загальні відомості та умови застосування програми

Назва програми – «Minesweeper» (з англ. Сапер).

Ігрова програма має розважальний характер.

Єдиним необхідним програмним забезпеченням є .Net Framework версії не менше 4.5.

Програма вимагає платформи на ОС Windows.

Для роботи з програмою необхідна комп’ютерна миша/тачпад та монітор.

* 1. Характеристики програми

Дана версія гри Сапер має 4 рівня складності, можливість збереження інформації про статистику рівня, можливість авто позначення прапорців, авто відкриття комірок, гарантію відкриття частини ігрового поля після першого натискання, додаткові умови успішного закінчення рівня.

* 1. Інсталяція та видалення

Інсталяція та видалення є стандартною для всіх програм Windows.

Для інсталяції програми треба запустити інсталяційний файл та слідувати простій інструкції інсталяції.

Для видалення програми достатньо скористатися «Панеллю керування», що є вбудованою у ОС Windows.

6.5 Послідовність дій користування програмою

Після запуску програми користувач повинен обрати режим гри з кнопок у головному меню або у шапці-меню (рисунок 6.1).

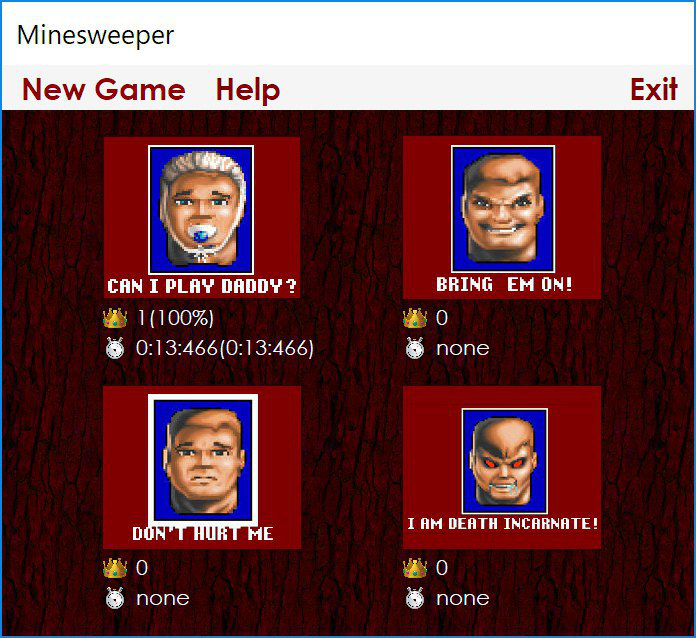


Рисунок 6.1 – Головне меню програми.

Далі відкриється ігрова форма і користувач може розпочинати гру. У випадку, якщо користувач не знайомий з правилами гри можна скористатися кнопкою «Help».



Рисунок 6.2 – Зразок ігрового поля (легкий рівень).

Щоб відкрити поле достатньо натиснути по ньому лівою кнопкою миші. Щоб поставити прапорець слід натиснути на комірці правою кнопкою миші.

Щоб перейти до головного меню необхідно натиснути на кнопку New Game у шапці-меню та вибрати відповідний пункт.

При необхідності у допомозі, користувачу слід натиснути кнопку Help.

Для завершення гри треба натиснути кнопку Exit.

ВИСНОВКИ

Під час виконання курсової роботи було розроблено удосконалений варіант класичної версії гри Сапер за допомогою знань і навичок, здобутих протягом освоєння курси [Основи програмування - 2. Програмування структурованих даних](http://wiki.kpi.ua/index.php/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20-%202.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%85%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_(20002020)). Для цього був створений клас Field (див підрозділ 3.1), що дав можливість зберігати повну інформацію про комірку ігрового поля. Всі комірки зберігаються у двовимірному масиві, саме його було використану як математичну модель. Було проведена велика робота по створенню приємного та зручного інтерфейсу. Були передбачені всі виняткові ситуації. Таким чином програма завжди запуститься і буде працювати, незважаючи на необачність користувача. Був створений HTML - файл з правилами гри, з якими можна ознайомитися натиснувши відповідну кнопку у шапці-меню. Програма була оформлена як установчий пакет з інсталяційним інтерфейсом а також можливістю видалити її через стандартну панель керування Windows у разі необхідності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Visual Studio від Microsoft.

URL: https://msdn.microsoft.com

2. Сайт відповідей та запитань.

URL: https://stackoverflow.com/

3. Значення кольорів

URL: http://procikave.com/znachennya-koloriv.html

ДОДАТОК А

Текст програми

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;

using System.Diagnostics;

namespace minesweeper

{

public partial class msForm : Form

{

/\*Ініціалізація глобальних змінних \*/

Field[,] field; // class Field - покращенний class Button

DateTime date,stopWatch; // Ці змінні використовуються для відображення

Timer timer; // поточного часу при проходженні рівня

int fieldHeight, fieldWidth, minesQuantity, levelDifficulty, nodeCounter=0, mineCounter=0; // Розміри поля, к-сть мін, відкритих комірок та поставлених флагів

bool firstClick = false; // Чи було зроблено перше натискання на комірку

/\* Наступні елементи потрібні для інтерфейсу головногом меню \*/

Button btnEasyLvl, btnLowmediumLvl, btnHighmediumLvl, btnHardLvl;

Label lStopwatch, lMines, easyLvlLabelStats, easyLvlLabelTime, lowmediumLvlLabelStats, lowmediumLvlLabelTime, highmediumLvlLabelStats, highmediumLvlLabelTime, hardLvlLabelStats, hardLvlLabelTime;

PictureBox minesLogo, easyLvlCrown, easyLvlStopwatch, lowmediumLvlCrown, lowmediumLvlStopwatch, highmediumLvlCrown, highmediumLvlStopwatch, hardLvlCrown, hardLvlStopwatch;

/\* Наступні елементи потрібні для збереження інформації про рекорди на кожному рівні \*/

long[] easyLvlInfo = new long[4], lowmediumLvlInfo = new long[4], highmediumLvlInfo = new long[4], hardLvlInfo = new long[4];

string text;

string[] info;

public msForm()

{

InitializeComponent();

}

/\*

\* Далі йде опис вже загаданого class-y Field. Він наслідає властивості Button, але також має

\* Value - чим є комірка: міною чи числом

\* Cheked - чи комірка перевірена (натиснута, відкрита)

\* Flagged - чи стоїть у комірці флаг

\* X,Y - положення комірки

\* \*/

public class Field : Button

{

private bool isChecked, isFlagged;

private int x, y;

private char VALUE;

public bool Checked { get { return isChecked; } set { isChecked = value; } }

public bool Flagged { get { return isFlagged; } set { isFlagged = value; } }

public int X { get { return x; } set { x = value; } }

public int Y { get { return y; } set { y = value; } }

public char Value { get { return VALUE; } set { VALUE = value; } }

}

//Подія при запуску форми. Тут створюються елемети головного меню

private void msForm\_Load(object sender, EventArgs e)

{

this.ControlBox = false;

try

{ this.BackgroundImage = Image.FromFile(@"images\bg.png"); }

catch { this.BackColor = Color.Black; }

this.Size = new Size(720, 650);

/\* Buttom for easy level \*/

btnEasyLvl = new Button();

this.Controls.Add(btnEasyLvl);

btnEasyLvl.BackColor = Color.FromName("Maroon");

try { btnEasyLvl.Image = Image.FromFile(@"images\btnEasyLevel.png"); }

catch

{

btnEasyLvl.Text = "CAN I PLAY, DADDY?";

btnEasyLvl.ForeColor = Color.White;

}

btnEasyLvl.Size = new Size(200, 165);

btnEasyLvl.FlatAppearance.BorderSize = 0;

btnEasyLvl.FlatStyle = FlatStyle.Popup;

btnEasyLvl.Click += new EventHandler(btnEasyLvl\_Click);

btnEasyLvl.Left = 100;

btnEasyLvl.Top = 70;

/\* Buttom for low-medium level \*/

btnLowmediumLvl = new Button();

this.Controls.Add(btnLowmediumLvl);

btnLowmediumLvl.BackColor = Color.FromName("Maroon");

try { btnLowmediumLvl.Image = Image.FromFile(@"images\btnLowmediumLevel.png"); }

catch

{

btnLowmediumLvl.Text = "DONT HURT ME!";

btnLowmediumLvl.ForeColor = Color.White;

}

btnLowmediumLvl.Size = new Size(200, 165);

btnLowmediumLvl.FlatAppearance.BorderSize = 0;

btnLowmediumLvl.FlatStyle = FlatStyle.Popup;

btnLowmediumLvl.Click += new EventHandler(btnLowmediumLvl\_Click);

btnLowmediumLvl.Left = 100;

btnLowmediumLvl.Top = 320;

/\* Buttom for high-medium level \*/

btnHighmediumLvl = new Button();

this.Controls.Add(btnHighmediumLvl);

btnHighmediumLvl.BackColor = Color.FromName("Maroon");

try { btnHighmediumLvl.Image = Image.FromFile(@"images\btnHighmediumLevel.png"); }

catch

{

btnHighmediumLvl.Text = "BRING 'EM ON!";

btnHighmediumLvl.ForeColor = Color.White;

}

btnHighmediumLvl.Size = new Size(200, 165);

btnHighmediumLvl.FlatAppearance.BorderSize = 0;

btnHighmediumLvl.FlatStyle = FlatStyle.Popup;

btnHighmediumLvl.Click += new EventHandler(btnHighmediumLvl\_Click);

btnHighmediumLvl.Top = 70;

btnHighmediumLvl.Left = 400;

/\* Buttom for hard level \*/

btnHardLvl = new Button();

this.Controls.Add(btnHardLvl);

btnHardLvl.BackColor = Color.FromName("Maroon");

try { btnHardLvl.Image = Image.FromFile(@"images\btnHardLevel.png"); }

catch

{

btnHardLvl.Text = "I AM DEATH INCARNATE!";

btnHardLvl.ForeColor = Color.White;

}

btnHardLvl.Size = new Size(200, 165);

btnHardLvl.FlatAppearance.BorderSize = 0;

btnHardLvl.FlatStyle = FlatStyle.Popup;

btnHardLvl.Click += new EventHandler(btnHardLvl\_Click);

btnHardLvl.Top = 320;

btnHardLvl.Left = 400;

/\* Reading "leaderboards" info \*/

try

{

info = File.ReadAllLines(@"saves\times.txt");

}

catch

{

info = new string[4];

for (int i = 0; i < 4; i++)

info[i] = "0 0 0 0\r\n";

}

readInfo(easyLvlInfo, 0);

readInfo(lowmediumLvlInfo, 1);

readInfo(highmediumLvlInfo, 2);

readInfo(hardLvlInfo, 3);

/\* Crating additional Labels and PictureBoxes for displaying leaderboards info \*/

/\* For easy level \*/

easyLvlCrown = new PictureBox();

this.Controls.Add(easyLvlCrown);

easyLvlCrown.BackgroundImageLayout = ImageLayout.Stretch;

easyLvlCrown.Size = new Size(25, 25);

easyLvlCrown.Top = btnEasyLvl.Top + btnEasyLvl.Height + 5;

easyLvlCrown.Left = btnEasyLvl.Left;

easyLvlCrown.BackColor = Color.Transparent;

easyLvlStopwatch = new PictureBox();

this.Controls.Add(easyLvlStopwatch);

easyLvlStopwatch.BackgroundImageLayout = ImageLayout.Stretch;

easyLvlStopwatch.Size = new Size(25, 25);

easyLvlStopwatch.Top = btnEasyLvl.Top + btnEasyLvl.Height + 35;

easyLvlStopwatch.Left = btnEasyLvl.Left;

easyLvlStopwatch.BackColor = Color.Transparent;

easyLvlLabelStats = new Label();

this.Controls.Add(easyLvlLabelStats);

easyLvlLabelStats.Font = new Font("Century Gothic", 7, FontStyle.Regular);

easyLvlLabelStats.Top = easyLvlCrown.Top;

easyLvlLabelStats.Left = easyLvlCrown.Left + 30;

easyLvlLabelStats.BackColor = Color.Transparent;

easyLvlLabelStats.ForeColor = Color.White;

easyLvlLabelTime = new Label();

this.Controls.Add(easyLvlLabelTime);

easyLvlLabelTime.Font = new Font("Century Gothic", 7, FontStyle.Regular);

easyLvlLabelTime.Top = easyLvlStopwatch.Top;

easyLvlLabelTime.Left = easyLvlCrown.Left + 30;

easyLvlLabelTime.BackColor = Color.Transparent;

easyLvlLabelTime.ForeColor = Color.White;

easyLvlLabelTime.AutoSize = true;

/\* For lowmedium level \*/

lowmediumLvlCrown = new PictureBox();

this.Controls.Add(lowmediumLvlCrown);

lowmediumLvlCrown.BackgroundImageLayout = ImageLayout.Stretch;

lowmediumLvlCrown.Size = new Size(25, 25);

lowmediumLvlCrown.Top = btnLowmediumLvl.Top + btnLowmediumLvl.Height + 5;

lowmediumLvlCrown.Left = btnLowmediumLvl.Left;

lowmediumLvlCrown.BackColor = Color.Transparent;

lowmediumLvlStopwatch = new PictureBox();

this.Controls.Add(lowmediumLvlStopwatch);

lowmediumLvlStopwatch.BackgroundImageLayout = ImageLayout.Stretch;

lowmediumLvlStopwatch.Size = new Size(25, 25);

lowmediumLvlStopwatch.Top = btnLowmediumLvl.Top + btnLowmediumLvl.Height + 35;

lowmediumLvlStopwatch.Left = btnLowmediumLvl.Left;

lowmediumLvlStopwatch.BackColor = Color.Transparent;

lowmediumLvlLabelStats = new Label();

this.Controls.Add(lowmediumLvlLabelStats);

lowmediumLvlLabelStats.Font = new Font("Century Gothic", 7, FontStyle.Regular);

lowmediumLvlLabelStats.Top = lowmediumLvlCrown.Top;

lowmediumLvlLabelStats.Left = lowmediumLvlCrown.Left + 30;

lowmediumLvlLabelStats.BackColor = Color.Transparent;

lowmediumLvlLabelStats.ForeColor = Color.White;

lowmediumLvlLabelTime = new Label();

this.Controls.Add(lowmediumLvlLabelTime);

lowmediumLvlLabelTime.Font = new Font("Century Gothic", 7, FontStyle.Regular);

lowmediumLvlLabelTime.Top = lowmediumLvlStopwatch.Top;

lowmediumLvlLabelTime.Left = lowmediumLvlCrown.Left + 30;

lowmediumLvlLabelTime.BackColor = Color.Transparent;

lowmediumLvlLabelTime.ForeColor = Color.White;

lowmediumLvlLabelTime.AutoSize = true;

/\* For highmedium level \*/

highmediumLvlCrown = new PictureBox();

this.Controls.Add(highmediumLvlCrown);

highmediumLvlCrown.BackgroundImageLayout = ImageLayout.Stretch;

highmediumLvlCrown.Size = new Size(25, 25);

highmediumLvlCrown.Top = btnHighmediumLvl.Top + btnHighmediumLvl.Height + 5;

highmediumLvlCrown.Left = btnHighmediumLvl.Left;

highmediumLvlCrown.BackColor = Color.Transparent;

highmediumLvlStopwatch = new PictureBox();

this.Controls.Add(highmediumLvlStopwatch);

highmediumLvlStopwatch.BackgroundImageLayout = ImageLayout.Stretch;

highmediumLvlStopwatch.Size = new Size(25, 25);

highmediumLvlStopwatch.Top = btnHighmediumLvl.Top + btnHighmediumLvl.Height + 35;

highmediumLvlStopwatch.Left = btnHighmediumLvl.Left;

highmediumLvlStopwatch.BackColor = Color.Transparent;

highmediumLvlLabelStats = new Label();

this.Controls.Add(highmediumLvlLabelStats);

highmediumLvlLabelStats.Font = new Font("Century Gothic", 7, FontStyle.Regular);

highmediumLvlLabelStats.Top = highmediumLvlCrown.Top;

highmediumLvlLabelStats.Left = highmediumLvlCrown.Left + 30;

highmediumLvlLabelStats.BackColor = Color.Transparent;

highmediumLvlLabelStats.ForeColor = Color.White;

highmediumLvlLabelTime = new Label();

this.Controls.Add(highmediumLvlLabelTime);

highmediumLvlLabelTime.Font = new Font("Century Gothic", 7, FontStyle.Regular);

highmediumLvlLabelTime.Top = highmediumLvlStopwatch.Top;

highmediumLvlLabelTime.Left = highmediumLvlCrown.Left + 30;

highmediumLvlLabelTime.BackColor = Color.Transparent;

highmediumLvlLabelTime.ForeColor = Color.White;

highmediumLvlLabelTime.AutoSize = true;

/\* For hard level \*/

hardLvlCrown = new PictureBox();

this.Controls.Add(hardLvlCrown);

hardLvlCrown.BackgroundImageLayout = ImageLayout.Stretch;

hardLvlCrown.Size = new Size(25, 25);

hardLvlCrown.Top = btnHardLvl.Top + btnHardLvl.Height + 5;

hardLvlCrown.Left = btnHardLvl.Left;

hardLvlCrown.BackColor = Color.Transparent;

hardLvlStopwatch = new PictureBox();

this.Controls.Add(hardLvlStopwatch);

hardLvlStopwatch.BackgroundImageLayout = ImageLayout.Stretch;

hardLvlStopwatch.Size = new Size(25, 25);

hardLvlStopwatch.Top = btnHardLvl.Top + btnHardLvl.Height + 35;

hardLvlStopwatch.Left = btnHardLvl.Left;

hardLvlStopwatch.BackColor = Color.Transparent;

hardLvlLabelStats = new Label();

this.Controls.Add(hardLvlLabelStats);

hardLvlLabelStats.Font = new Font("Century Gothic", 7, FontStyle.Regular);

hardLvlLabelStats.Top = hardLvlCrown.Top;

hardLvlLabelStats.Left = hardLvlCrown.Left + 30;

hardLvlLabelStats.BackColor = Color.Transparent;

hardLvlLabelStats.ForeColor = Color.White;

hardLvlLabelTime = new Label();

this.Controls.Add(hardLvlLabelTime);

hardLvlLabelTime.Font = new Font("Century Gothic", 7, FontStyle.Regular);

hardLvlLabelTime.Top = hardLvlStopwatch.Top;

hardLvlLabelTime.Left = hardLvlCrown.Left + 30;

hardLvlLabelTime.BackColor = Color.Transparent;

hardLvlLabelTime.ForeColor = Color.White;

hardLvlLabelTime.AutoSize = true;

try

{

easyLvlCrown.BackgroundImage = Image.FromFile(@"images\crown.png");

lowmediumLvlCrown.BackgroundImage = Image.FromFile(@"images\crown.png");

highmediumLvlCrown.BackgroundImage = Image.FromFile(@"images\crown.png");

hardLvlCrown.BackgroundImage = Image.FromFile(@"images\crown.png");

}

catch { }

try

{

easyLvlStopwatch.BackgroundImage = Image.FromFile(@"images\stopwatch.png");

lowmediumLvlStopwatch.BackgroundImage = Image.FromFile(@"images\stopwatch.png");

highmediumLvlStopwatch.BackgroundImage = Image.FromFile(@"images\stopwatch.png");

hardLvlStopwatch.BackgroundImage = Image.FromFile(@"images\stopwatch.png");

}

catch { }

\_Update();

}

/\* Функція зчитування даних про рекорди відповідного рівня. Приймає

\* масив arr - куди записати інформацію

\* index - звідки зчитувати цю інформацію

\*/

protected void readInfo(long[] arr, int index)

{

for (int i = 0; i < 4; i++)

{

try

{

arr[i] = Convert.ToInt32(info[index].Split(' ')[i]);

}

catch

{

arr[i] = 0;

}

}

}

/\* Функція, що видаляє всі елементи, які не є необХідними

\* у даний момент гри (всі комірки, секундомір, елементи

\* для відображення к-сті мін, що залишились), та обнуляє

\* змінні \*/

protected void Clearing()

{

nodeCounter = 0;

mineCounter = 0;

firstClick = false;

if (field!=null)

foreach (Field f in field)

f.Dispose();

this.Size = new Size(720, 650);

if (lStopwatch!=null)

lStopwatch.Dispose();

if (timer != null)

timer.Dispose();

if (lMines != null)

lMines.Dispose();

if (minesLogo != null)

minesLogo.Dispose();

}

/\* Функція для оновлення інформації про рекорди всіХ рівнів, занесення цієї інформації

\* до відповідних масивів та виведення її у головному меню

\*/

protected void \_Update()

{

text = easyLvlInfo[0].ToString();

if (easyLvlInfo[1] != 0)

text += "(" + Math.Round(Convert.ToDouble(easyLvlInfo[0]) / Convert.ToDouble(easyLvlInfo[1]) \* 100).ToString() + "%)";

easyLvlLabelStats.Text = text;

if (easyLvlInfo[2] == 0)

text = "none";

else

text = (easyLvlInfo[2] / 60000).ToString() + ":" + (((easyLvlInfo[2]) - (easyLvlInfo[2] / 60000) \* 60000) / 1000).ToString() + ":" + (easyLvlInfo[2] % 1000).ToString() + "("+ (easyLvlInfo[3] / 60000).ToString() + ":" + (((easyLvlInfo[3]) - (easyLvlInfo[3] / 60000) \* 60000) / 1000).ToString() + ":" + (easyLvlInfo[3] % 1000).ToString()+")";

easyLvlLabelTime.Text = text;

text = lowmediumLvlInfo[0].ToString();

if (lowmediumLvlInfo[1] != 0)

text += "(" + Math.Round(Convert.ToDouble(lowmediumLvlInfo[0]) / Convert.ToDouble(lowmediumLvlInfo[1]) \* 100).ToString() + "%)";

lowmediumLvlLabelStats.Text = text;

if (lowmediumLvlInfo[2] == 0)

text = "none";

else

text = (lowmediumLvlInfo[2] / 60000).ToString() + ":" + (((lowmediumLvlInfo[2]) - (lowmediumLvlInfo[2] / 60000) \* 60000) / 1000).ToString() + ":" + (lowmediumLvlInfo[2] % 1000).ToString() + "(" + (lowmediumLvlInfo[3] / 60000).ToString() + ":" + (((lowmediumLvlInfo[3]) - (lowmediumLvlInfo[3] / 60000) \* 60000) / 1000).ToString() + ":" + (lowmediumLvlInfo[3] % 1000).ToString() + ")";

lowmediumLvlLabelTime.Text = text;

text = highmediumLvlInfo[0].ToString();

if (highmediumLvlInfo[1] != 0)

text += "(" + Math.Round(Convert.ToDouble(highmediumLvlInfo[0]) / Convert.ToDouble(highmediumLvlInfo[1]) \* 100).ToString() + "%)";

highmediumLvlLabelStats.Text = text;

if (highmediumLvlInfo[2] == 0)

text = "none";

else

text = (highmediumLvlInfo[2] / 60000).ToString() + ":" + (((highmediumLvlInfo[2]) - (highmediumLvlInfo[2] / 60000) \* 60000) / 1000).ToString() + ":" + (highmediumLvlInfo[2] % 1000).ToString() + "(" + (highmediumLvlInfo[3] / 60000).ToString() + ":" + (((highmediumLvlInfo[3]) - (highmediumLvlInfo[3] / 60000) \* 60000) / 1000).ToString() + ":" + (highmediumLvlInfo[3] % 1000).ToString() + ")";

highmediumLvlLabelTime.Text = text;

text = hardLvlInfo[0].ToString();

if (hardLvlInfo[1] != 0)

text += "(" + Math.Round(Convert.ToDouble(hardLvlInfo[0]) / Convert.ToDouble(hardLvlInfo[1]) \* 100).ToString() + "%)";

hardLvlLabelStats.Text = text;

if (hardLvlInfo[2] == 0)

text = "none";

else

text = (hardLvlInfo[2] / 60000).ToString() + ":" + (((hardLvlInfo[2]) - (hardLvlInfo[2] / 60000) \* 60000) / 1000).ToString() + ":" + (hardLvlInfo[2] % 1000).ToString() + "(" + (hardLvlInfo[3] / 60000).ToString() + ":" + (((hardLvlInfo[3]) - (hardLvlInfo[3] / 60000) \* 60000) / 1000).ToString() + ":" + (hardLvlInfo[3] % 1000).ToString() + ")";

hardLvlLabelTime.Text = text;

}

/\* Функція, яка виконується при завершенні рівня. Неважливо, перемогою чи поразкою

\* Всі комірки становляться неативними. Відоражаються позиції мін (при поразці).

\* Також при перемозі формується нова статистика для рекордів рівня

\*/

public void Finish(bool passed, int level)

{

nodeCounter = 0;

mineCounter = 0;

firstClick = false;

foreach (Field f in field)

{

if (passed)

{

if (f.Value == 'm')

{

f.BackColor = Color.Gold;

try

{

f.BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

f.Text = "f";

}

f.BackgroundImageLayout = ImageLayout.Stretch;

}

else

{

f.BackColor = Color.Salmon;

if (f.Value != '0')

f.Text = f.Value.ToString();

}

}

else

{

if (f.Value == 'm' && !f.Flagged)

{

f.Text = "";

try

{

f.BackgroundImage = Image.FromFile(@"images/mine.png");

}

catch

{

f.Text = "M";

}

f.BackgroundImageLayout = ImageLayout.Stretch;

f.BackColor = Color.Maroon;

}

}

f.Enabled = false;

}

switch (level)

{

case 1:

if (passed)

{

long time = stopWatch.Millisecond + stopWatch.Second \* 1000 + stopWatch.Minute \* 60000;

if (time < easyLvlInfo[2] || easyLvlInfo[2]==0)

easyLvlInfo[2] = time;

easyLvlInfo[3] = Convert.ToInt32(Math.Round(Convert.ToDouble(easyLvlInfo[3]\*easyLvlInfo[0] + time))/(++easyLvlInfo[0]));

easyLvlInfo[1]++;

}

else

{

easyLvlInfo[1]++;

}

break;

case 2:

if (passed)

{

long time = stopWatch.Millisecond + stopWatch.Second \* 1000 + stopWatch.Minute \* 60000;

if (time < lowmediumLvlInfo[2] || lowmediumLvlInfo[2] == 0)

lowmediumLvlInfo[2] = time;

lowmediumLvlInfo[3] = Convert.ToInt32(Math.Round(Convert.ToDouble(lowmediumLvlInfo[3] \* lowmediumLvlInfo[0] + time)) / (++lowmediumLvlInfo[0]));

lowmediumLvlInfo[1]++;

}

else

{

lowmediumLvlInfo[1]++;

}

break;

case 3:

if (passed)

{

long time = stopWatch.Millisecond + stopWatch.Second \* 1000 + stopWatch.Minute \* 60000;

if (time < highmediumLvlInfo[2] || highmediumLvlInfo[2] == 0)

highmediumLvlInfo[2] = time;

highmediumLvlInfo[3] = Convert.ToInt32(Math.Round(Convert.ToDouble(highmediumLvlInfo[3] \* highmediumLvlInfo[0] + time)) / (++highmediumLvlInfo[0]));

highmediumLvlInfo[1]++;

}

else

{

highmediumLvlInfo[1]++;

}

break;

case 4:

if (passed)

{

long time = stopWatch.Millisecond + stopWatch.Second \* 1000 + stopWatch.Minute \* 60000;

if (time < hardLvlInfo[2] || hardLvlInfo[2] == 0)

hardLvlInfo[2] = time;

hardLvlInfo[3] = Convert.ToInt32(Math.Round(Convert.ToDouble(hardLvlInfo[3] \* hardLvlInfo[0] + time)) / (++hardLvlInfo[0]));

hardLvlInfo[1]++;

}

else

{

hardLvlInfo[1]++;

}

break;

}

}

// Подія при натисканні на кнопку "Main menu"

private void MainMenuToolStripMenuItem\_Click(object sender, EventArgs e)

{

Clearing();

showAll();

\_Update();

}

/\* Функція, яка виконується при натисканні копнки якого з рівнів.

\* Тут генерується ігрове поле, заводиться секундомір, стовурюються

\* елементи графічного інтерфейсу \*/

public void Game(int n, int m, int k) //n,m - size, k - mines

{

Clearing();

hideAll();

lStopwatch = new Label();

this.Controls.Add(lStopwatch);

lStopwatch.Text = "00:00:000";

lStopwatch.Font = new Font("Century Gothic", 20,FontStyle.Regular);

lStopwatch.Top = 80;

lStopwatch.BackColor = Color.Transparent;

lStopwatch.ForeColor = Color.White;

lStopwatch.AutoSize = true;

int size = 50, width = size\*m, height = size\*n;

field = new Field[n, m];

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

{

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

{

field[i, j] = new Field();

this.Controls.Add(field[i, j]);

field[i, j].X = i;

field[i, j].Y = j;

field[i, j].BackgroundImageLayout = ImageLayout.Stretch;

field[i, j].Size = new Size(size, size);

field[i, j].Left = ((width+300) - size \* m - 5 \* (m - 1)) / 2 + ((size + 5) \* j) - 5;

field[i, j].Top = 200 + (size + 5) \* i;

field[i, j].Value = '0';

field[i, j].MouseDown += new MouseEventHandler(field\_MouseDown);

field[i, j].BackColor = Color.FromName("Maroon");

field[i, j].Checked = false;

field[i, j].Flagged = false;

field[i,j].Font = new Font("Century Gothic", 10, FontStyle.Bold);

}

}

lMines = new Label();

this.Controls.Add(lMines);

lMines.Text = k.ToString();

lMines.Font = new Font("Century Gothic", 20, FontStyle.Regular);

lMines.Top = 80;

lMines.BackColor = Color.Transparent;

lMines.ForeColor = Color.White;

lMines.AutoSize = true;

lMines.Left = field[0, m - 1].Left + field[0, m - 1].Width - lMines.Width;

minesLogo = new PictureBox();

this.Controls.Add(minesLogo);

minesLogo.Size = new Size(lStopwatch.Height, lStopwatch.Height);

try

{

minesLogo.BackgroundImage = Image.FromFile(@"images\mine.png");

}

catch

{

}

minesLogo.BackgroundImageLayout = ImageLayout.Stretch;

minesLogo.Left = lMines.Left - minesLogo.Width-10;

minesLogo.Top = lMines.Top;

minesLogo.BackColor = Color.Transparent;

lStopwatch.Left = field[0, 0].Left-15;

this.Size = new Size(width+300, height+400);

}

// Фунція, яка скриває всі елменти головного меню

protected void hideAll()

{

btnEasyLvl.Visible = false;

btnLowmediumLvl.Visible = false;

btnHighmediumLvl.Visible = false;

btnHardLvl.Visible = false;

easyLvlCrown.Visible = false;

easyLvlStopwatch.Visible = false;

easyLvlLabelStats.Visible = false;

easyLvlLabelTime.Visible = false;

lowmediumLvlCrown.Visible = false;

lowmediumLvlStopwatch.Visible = false;

lowmediumLvlLabelStats.Visible = false;

lowmediumLvlLabelTime.Visible = false;

highmediumLvlCrown.Visible = false;

highmediumLvlStopwatch.Visible = false;

highmediumLvlLabelStats.Visible = false;

highmediumLvlLabelTime.Visible = false;

hardLvlCrown.Visible = false;

hardLvlStopwatch.Visible = false;

hardLvlLabelStats.Visible = false;

hardLvlLabelTime.Visible = false;

}

// Функція, яка відображає всі елементи головного меню

protected void showAll()

{

btnEasyLvl.Visible = true;

btnLowmediumLvl.Visible = true;

btnHighmediumLvl.Visible = true;

btnHardLvl.Visible = true;

easyLvlCrown.Visible = true;

easyLvlStopwatch.Visible = true;

easyLvlLabelStats.Visible = true;

easyLvlLabelTime.Visible = true;

lowmediumLvlCrown.Visible = true;

lowmediumLvlStopwatch.Visible = true;

lowmediumLvlLabelStats.Visible = true;

lowmediumLvlLabelTime.Visible = true;

highmediumLvlCrown.Visible = true;

highmediumLvlStopwatch.Visible = true;

highmediumLvlLabelStats.Visible = true;

highmediumLvlLabelTime.Visible = true;

hardLvlCrown.Visible = true;

hardLvlStopwatch.Visible = true;

hardLvlLabelStats.Visible = true;

hardLvlLabelTime.Visible = true;

}

// Подія, що відбувається при натисканні на кнопку легкого рівня

protected void btnEasyLvl\_Click(object sender, EventArgs e) //n,m - size, k - mimes

{

fieldHeight = 10;

fieldWidth = 10;

minesQuantity = 9;

levelDifficulty = 1;

Game(10, 10, 9);

}

// Подія, що відбувається при натисканні на кнопку середнього рівня

protected void btnLowmediumLvl\_Click(object sender, EventArgs e) //n,m - size, k - mimes

{

fieldHeight = 16;

fieldWidth = 16;

minesQuantity = 40;

levelDifficulty = 2;

Game(16, 16, 40);

}

// Подія, що відбувається при натисканні на кнопку складного рівня

protected void btnHighmediumLvl\_Click(object sender, EventArgs e) //n,m - size, k - mimes

{

fieldHeight = 16;

fieldWidth = 30;

minesQuantity = 99;

levelDifficulty = 3;

Game(16, 30, 99);

}

// Подія, що відбувається при натисканні на кнопку надскладного рівня

protected void btnHardLvl\_Click(object sender, EventArgs e) //n,m - size, k - mimes

{

fieldHeight = 20;

fieldWidth = 35;

minesQuantity = 175;

levelDifficulty = 4;

Game(20, 35, 175);

}

/\* Функція для відображення часу секундоміра. Порівнюється різниця в часі

\* між тим,коли секундомір був заведений та поточним часом

\*/

private void tickTimer(object sender, EventArgs e)

{

if (timer.Enabled)

{

long tick = DateTime.Now.Ticks - date.Ticks;

stopWatch = new DateTime();

if (lStopwatch.Text != "59:59:999")

{

stopWatch = stopWatch.AddTicks(tick);

lStopwatch.Text = String.Format("{0:mm:ss:fff}", stopWatch);

}

else

timer.Enabled = false;

}

}

// Функція, яка виконується при натисканні на комірку лівої кнопкої миші

public void openField(Field node)

{

/\* Якщо ще не було зроблено кліку, то міни розставляються таким чином, аби після першого

\* кліку відкрилась якась частина ігрового поля \*/

if (!firstClick)

{

firstClick = true;

date = DateTime.Now;

timer = new Timer();

timer.Interval = 10;

timer.Tick += new EventHandler(tickTimer);

timer.Enabled = true;

for (int z = 0; z < minesQuantity; z++)

{

Random rnd = new Random();

int x = rnd.Next(0, fieldHeight);

int y = rnd.Next(0, fieldWidth);

if (field[x, y].Value == 'm' || (Math.Abs(x-node.X)<=1 && Math.Abs(y-node.Y)<=1))

while (field[x, y].Value == 'm' || (Math.Abs(x - node.X) <= 1 && Math.Abs(y - node.Y) <= 1))

{

x = rnd.Next(0, fieldHeight);

y = rnd.Next(0, fieldWidth);

}

field[x, y].Value = 'm';

if (x - 1 >= 0 && y - 1 >= 0)

if (field[x - 1, y - 1].Value != 'm')

field[x - 1, y - 1].Value = Convert.ToChar(Convert.ToInt32(field[x - 1, y - 1].Value) + 1);

if (x - 1 >= 0 && y >= 0)

if (field[x - 1, y].Value != 'm')

field[x - 1, y].Value = Convert.ToChar(Convert.ToInt32(field[x - 1, y].Value) + 1);

if (x - 1 >= 0 && y + 1 <= fieldWidth - 1)

if (field[x - 1, y + 1].Value != 'm')

field[x - 1, y + 1].Value = Convert.ToChar(Convert.ToInt32(field[x - 1, y + 1].Value) + 1);

if (x >= 0 && y - 1 >= 0)

if (field[x, y - 1].Value != 'm')

field[x, y - 1].Value = Convert.ToChar(Convert.ToInt32(field[x, y - 1].Value) + 1);

if (x >= 0 && y + 1 <= fieldWidth - 1)

if (field[x, y + 1].Value != 'm')

field[x, y + 1].Value = Convert.ToChar(Convert.ToInt32(field[x, y + 1].Value) + 1);

if (x + 1 <= fieldHeight - 1 && y - 1 >= 0)

if (field[x + 1, y - 1].Value != 'm')

field[x + 1, y - 1].Value = Convert.ToChar(Convert.ToInt32(field[x + 1, y - 1].Value) + 1);

if (x + 1 <= fieldHeight - 1 && y >= 0)

if (field[x + 1, y].Value != 'm')

field[x + 1, y].Value = Convert.ToChar(Convert.ToInt32(field[x + 1, y].Value) + 1);

if (x + 1 <= fieldHeight - 1 && y + 1 <= fieldWidth - 1)

if (field[x + 1, y + 1].Value != 'm')

field[x + 1, y + 1].Value = Convert.ToChar(Convert.ToInt32(field[x + 1, y + 1].Value) + 1);

//for (int i = 0; i < n; i++)

//{

// for (int j = 0; j < m; j++)

// {

// field[i, j].Text = Convert.ToString(field[i, j].Value);

// }

//}

}

}

// Якщо в комірці міна і вона не помічена, гравець програє

if (node.Value == 'm')

{

if (!node.Flagged)

{

timer.Enabled = false;

MessageBox.Show("You lose!");

Finish(false,levelDifficulty);

}

// Якщо міна, але помічена, мітка знімається

else

{

node.Flagged = false;

node.Text = "";

node.BackColor = Color.FromName("Maroon");

node.BackgroundImage = null;

mineCounter--;

}

}

// Якщо комірка порожня, то автоматично відкриваються комірки навколо неї

// Це робиться за допомогою визову функції openField від навколишніх комірок

else if (node.Value == '0')

{

if (!node.Checked)

{

node.Text = "";

node.Flagged = false;

node.BackgroundImage = null;

node.BackColor = Color.FromName("Salmon");

node.Checked = true;

nodeCounter++;

if (node.X - 1 >= 0 && node.Y - 1 >= 0 && !field[node.X - 1, node.Y - 1].Checked)

openField(field[node.X - 1, node.Y - 1]);

if (node.X - 1 >= 0 && !field[node.X - 1, node.Y].Checked)

openField(field[node.X - 1, node.Y]);

if (node.X - 1 >= 0 && node.Y + 1 <= fieldWidth - 1 && !field[node.X - 1, node.Y + 1].Checked)

openField(field[node.X - 1, node.Y + 1]);

if (node.Y - 1 >= 0 && !field[node.X, node.Y - 1].Checked)

openField(field[node.X, node.Y - 1]);

if (node.Y + 1 <= fieldWidth - 1 && !field[node.X, node.Y + 1].Checked)

openField(field[node.X, node.Y + 1]);

if (node.X + 1 <= fieldHeight - 1 && node.Y - 1 >= 0 && !field[node.X + 1, node.Y - 1].Checked)

openField(field[node.X + 1, node.Y - 1]);

if (node.X + 1 <= fieldHeight - 1 && !field[node.X + 1, node.Y].Checked)

openField(field[node.X + 1, node.Y]);

if (node.X + 1 <= fieldHeight - 1 && node.Y + 1 <= fieldWidth - 1 && !field[node.X + 1, node.Y + 1].Checked)

openField(field[node.X + 1, node.Y + 1]);

}

}

// Інакше комірка відкривається. Якщо всі можливі комірки відкриті, гра рівень завершується з перемогою

else

{

if (!node.Checked)

{

node.Text = Convert.ToString(node.Value);

node.BackgroundImage = null;

node.BackColor = Color.FromName("Salmon");

node.Checked = true;

node.Flagged = false;

nodeCounter++;

if (nodeCounter == fieldHeight \* fieldWidth - minesQuantity)

{

timer.Enabled = false;

MessageBox.Show("You win!");

Finish(true,levelDifficulty);

}

}

else

{

if (countFlags(node) == Convert.ToInt32(node.Value.ToString()))

{

if (node.X - 1 >= 0 && node.Y - 1 >= 0 && !field[node.X - 1, node.Y - 1].Checked && !field[node.X - 1, node.Y - 1].Flagged)

openField(field[node.X - 1, node.Y - 1]);

if (node.X - 1 >= 0 && !field[node.X - 1, node.Y].Checked && !field[node.X - 1, node.Y].Flagged)

openField(field[node.X - 1, node.Y]);

if (node.X - 1 >= 0 && node.Y + 1 <= fieldWidth - 1 && !field[node.X - 1, node.Y + 1].Checked && !field[node.X - 1, node.Y + 1].Flagged)

openField(field[node.X - 1, node.Y + 1]);

if (node.Y - 1 >= 0 && !field[node.X, node.Y - 1].Checked && !field[node.X, node.Y - 1].Flagged)

openField(field[node.X, node.Y - 1]);

if (node.Y + 1 <= fieldWidth - 1 && !field[node.X, node.Y + 1].Checked && !field[node.X, node.Y + 1].Flagged)

openField(field[node.X, node.Y + 1]);

if (node.X + 1 <= fieldHeight - 1 && node.Y - 1 >= 0 && !field[node.X + 1, node.Y - 1].Checked && !field[node.X + 1, node.Y - 1].Flagged)

openField(field[node.X + 1, node.Y - 1]);

if (node.X + 1 <= fieldHeight - 1 && !field[node.X + 1, node.Y].Checked && !field[node.X + 1, node.Y].Flagged)

openField(field[node.X + 1, node.Y]);

if (node.X + 1 <= fieldHeight - 1 && node.Y + 1 <= fieldWidth - 1 && !field[node.X + 1, node.Y + 1].Checked && !field[node.X + 1, node.Y + 1].Flagged)

openField(field[node.X + 1, node.Y + 1]);

}

}

}

}

// Функція для підрахнку поставлених флагів навколо комірки

public int countFlags(Field node)

{

int flagCounter = 0;

if (node.X - 1 >= 0 && node.Y - 1 >= 0 && field[node.X - 1, node.Y - 1].Flagged)

flagCounter++;

if (node.X - 1 >= 0 && field[node.X - 1, node.Y].Flagged)

flagCounter++;

if (node.X - 1 >= 0 && node.Y + 1 <= fieldWidth - 1 && field[node.X - 1, node.Y + 1].Flagged)

flagCounter++;

if (node.Y - 1 >= 0 && field[node.X, node.Y - 1].Flagged)

flagCounter++;

if (node.Y + 1 <= fieldWidth - 1 && field[node.X, node.Y + 1].Flagged)

flagCounter++;

if (node.X + 1 <= fieldHeight - 1 && node.Y - 1 >= 0 && field[node.X + 1, node.Y - 1].Flagged)

flagCounter++;

if (node.X + 1 <= fieldHeight - 1 && field[node.X + 1, node.Y].Flagged)

flagCounter++;

if (node.X + 1 <= fieldHeight - 1 && node.Y + 1 <= fieldWidth - 1 && field[node.X + 1, node.Y + 1].Flagged)

flagCounter++;

return flagCounter;

}

// Функція для підрахнку закритих комірок навколо комірки

public int countClosed(Field node)

{

int closedCounter = 0;

if (node.X - 1 >= 0 && node.Y - 1 >= 0 && !field[node.X - 1, node.Y - 1].Checked)

closedCounter++;

if (node.X - 1 >= 0 && !field[node.X - 1, node.Y].Checked)

closedCounter++;

if (node.X - 1 >= 0 && node.Y + 1 <= fieldWidth - 1 && !field[node.X - 1, node.Y + 1].Checked)

closedCounter++;

if (node.Y - 1 >= 0 && !field[node.X, node.Y - 1].Checked)

closedCounter++;

if (node.Y + 1 <= fieldWidth - 1 && !field[node.X, node.Y + 1].Checked)

closedCounter++;

if (node.X + 1 <= fieldHeight - 1 && node.Y - 1 >= 0 && !field[node.X + 1, node.Y - 1].Checked)

closedCounter++;

if (node.X + 1 <= fieldHeight - 1 && node.Y >= 0 && !field[node.X + 1, node.Y].Checked)

closedCounter++;

if (node.X + 1 <= fieldHeight - 1 && node.Y + 1 <= fieldWidth - 1 && !field[node.X + 1, node.Y + 1].Checked)

closedCounter++;

return closedCounter;

}

// Подія при натисканні на кормірку. В залежності від того, яка кнопка була натиснута, виконуються різні дії

protected void field\_MouseDown(object sender, MouseEventArgs e)

{

Field node = (Field)sender;

if (e.Button == MouseButtons.Left)

openField(node);

if (e.Button == MouseButtons.Right && firstClick)

{

if (!node.Checked)

{

if (!node.Flagged)

{

try

{

node.BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

node.Text = "f";

}

node.Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

node.BackColor = Color.FromName("Gold");

if (node.Value == 'm')

mineCounter++;

if (mineCounter == minesQuantity)

{

timer.Enabled = false;

MessageBox.Show("You win!");

Finish(true, levelDifficulty);

}

}

else

{

node.BackgroundImage = null;

node.Text = "";

node.Flagged = false;

lMines.Text = (Convert.ToInt32(lMines.Text) + 1).ToString();

node.BackColor = Color.FromName("Maroon");

if (node.Value == 'm')

mineCounter--;

}

}

else

{

if (countClosed(node) == Convert.ToUInt32(node.Value.ToString()))

{

if (node.X - 1 >= 0 && node.Y - 1 >= 0 && !field[node.X - 1, node.Y - 1].Checked && !field[node.X - 1, node.Y - 1].Flagged)

{

try

{

field[node.X - 1, node.Y - 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X - 1, node.Y - 1].Text = "f";

}

field[node.X - 1, node.Y - 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X - 1, node.Y - 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X - 1 >= 0 && node.Y >= 0 && !field[node.X - 1, node.Y].Checked && !field[node.X - 1, node.Y].Flagged)

{

try

{

field[node.X - 1, node.Y].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X - 1, node.Y].Text = "f";

}

field[node.X - 1, node.Y].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X - 1, node.Y].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X - 1 >= 0 && node.Y + 1 <= fieldWidth - 1 && !field[node.X - 1, node.Y + 1].Checked && !field[node.X - 1, node.Y + 1].Flagged)

{

try

{

field[node.X - 1, node.Y + 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X - 1, node.Y + 1].Text = "f";

}

field[node.X - 1, node.Y + 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X - 1, node.Y + 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X >= 0 && node.Y - 1 >= 0 && !field[node.X, node.Y - 1].Checked && !field[node.X, node.Y - 1].Flagged)

{

try

{

field[node.X, node.Y - 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X, node.Y - 1].Text = "f";

}

field[node.X , node.Y - 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X , node.Y - 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X >= 0 && node.Y + 1 <= fieldWidth - 1 && !field[node.X, node.Y + 1].Checked && !field[node.X, node.Y + 1].Flagged)

{

try

{

field[node.X, node.Y + 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X, node.Y + 1].Text = "f";

}

field[node.X, node.Y + 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X , node.Y + 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X + 1 <= fieldHeight - 1 && node.Y - 1 >= 0 && !field[node.X + 1, node.Y - 1].Checked && !field[node.X + 1, node.Y - 1].Flagged)

{

try

{

field[node.X + 1, node.Y - 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X + 1, node.Y - 1].Text = "f";

}

field[node.X + 1, node.Y - 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X + 1, node.Y - 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X + 1 <= fieldHeight - 1 && node.Y >= 0 && !field[node.X + 1, node.Y].Checked && !field[node.X + 1, node.Y].Flagged)

{

try

{

field[node.X + 1, node.Y].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X + 1, node.Y].Text = "f";

}

field[node.X + 1, node.Y].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X + 1, node.Y].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (node.X + 1 <= fieldHeight - 1 && node.Y + 1 <= fieldWidth - 1 && !field[node.X + 1, node.Y + 1].Checked && !field[node.X + 1, node.Y + 1].Flagged)

{

try

{

field[node.X + 1, node.Y + 1].BackgroundImage = Image.FromFile(@"images\flag.png");

}

catch

{

field[node.X + 1, node.Y + 1].Text = "f";

}

field[node.X + 1, node.Y + 1].Flagged = true;

lMines.Text = (Convert.ToInt32(lMines.Text) - 1).ToString();

field[node.X + 1, node.Y + 1].BackColor = Color.FromName("Gold");

mineCounter++;

}

if (mineCounter == minesQuantity)

{

timer.Enabled = false;

MessageBox.Show("You win!");

Finish(true, levelDifficulty);

}

}

}

}

}

// Подія при натискання Help. Відкривається HTML файл з правилами гри.

private void helpToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

Process.Start(@"saves\rules.html");

}

catch

{

MessageBox.Show("Sorry. File is corrupted or deleted!");

}

}

// Подія при натисканні кнопки Exit. Всі необхідні дані записуються у файл і програма закривається

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

text = easyLvlInfo[0].ToString() + ' ' + easyLvlInfo[1].ToString() + ' ' + easyLvlInfo[2].ToString() + ' ' + easyLvlInfo[3].ToString() + "\r\n" + lowmediumLvlInfo[0].ToString() + ' ' + lowmediumLvlInfo[1].ToString() + ' ' + lowmediumLvlInfo[2].ToString() + ' ' + lowmediumLvlInfo[3].ToString() + "\r\n" + highmediumLvlInfo[0].ToString() + ' ' + highmediumLvlInfo[1].ToString() + ' ' + highmediumLvlInfo[2].ToString() + ' ' + highmediumLvlInfo[3].ToString() + "\r\n" + hardLvlInfo[0].ToString() + ' ' + hardLvlInfo[1].ToString() + ' ' + hardLvlInfo[2].ToString() + ' ' + hardLvlInfo[3].ToString();

if (Directory.Exists(@"saves"))

File.WriteAllText(@"saves\times.txt",text);

else

{

Directory.CreateDirectory(@"saves");

File.WriteAllText(@"saves\times.txt", text);

}

this.Close();

}

}

}

ДОДАТОК Б

Результати роботи програми

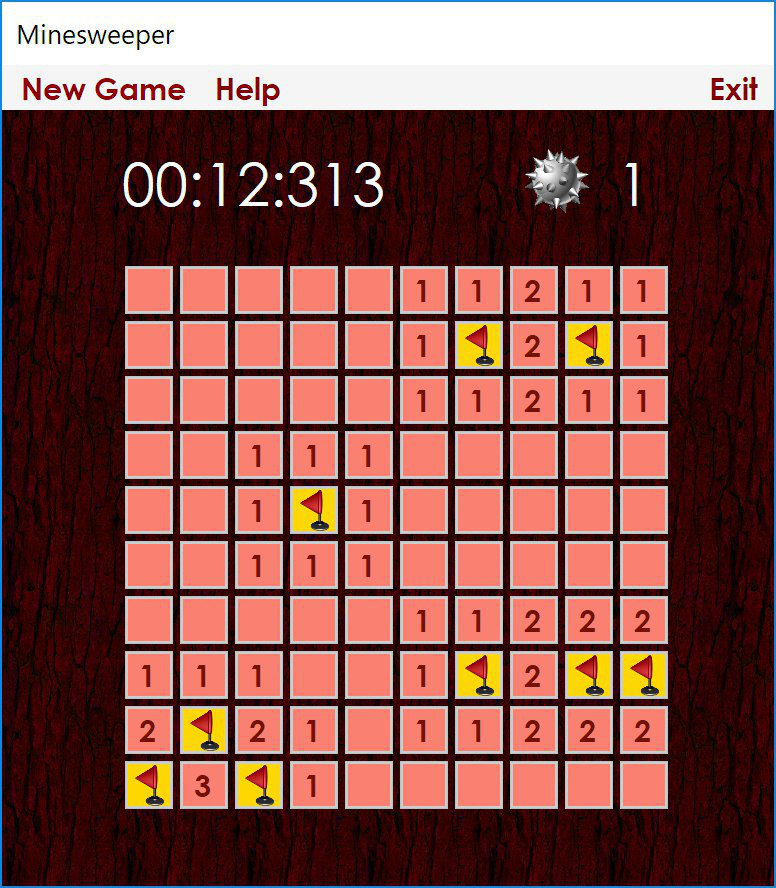


Рисунок Б.1 – Приклад успішного завершення гри.

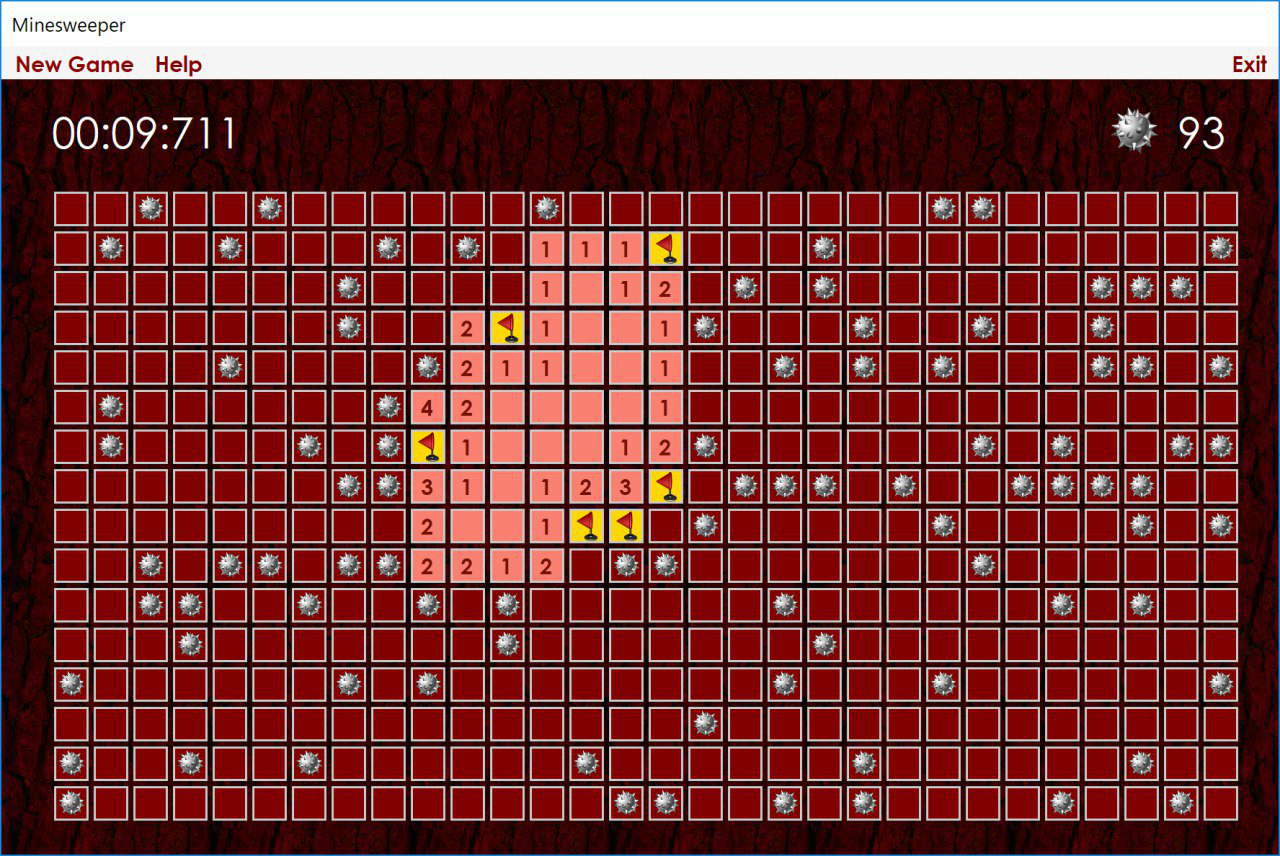


Рисунок Б.2 – Приклад неуспішного завершення гри

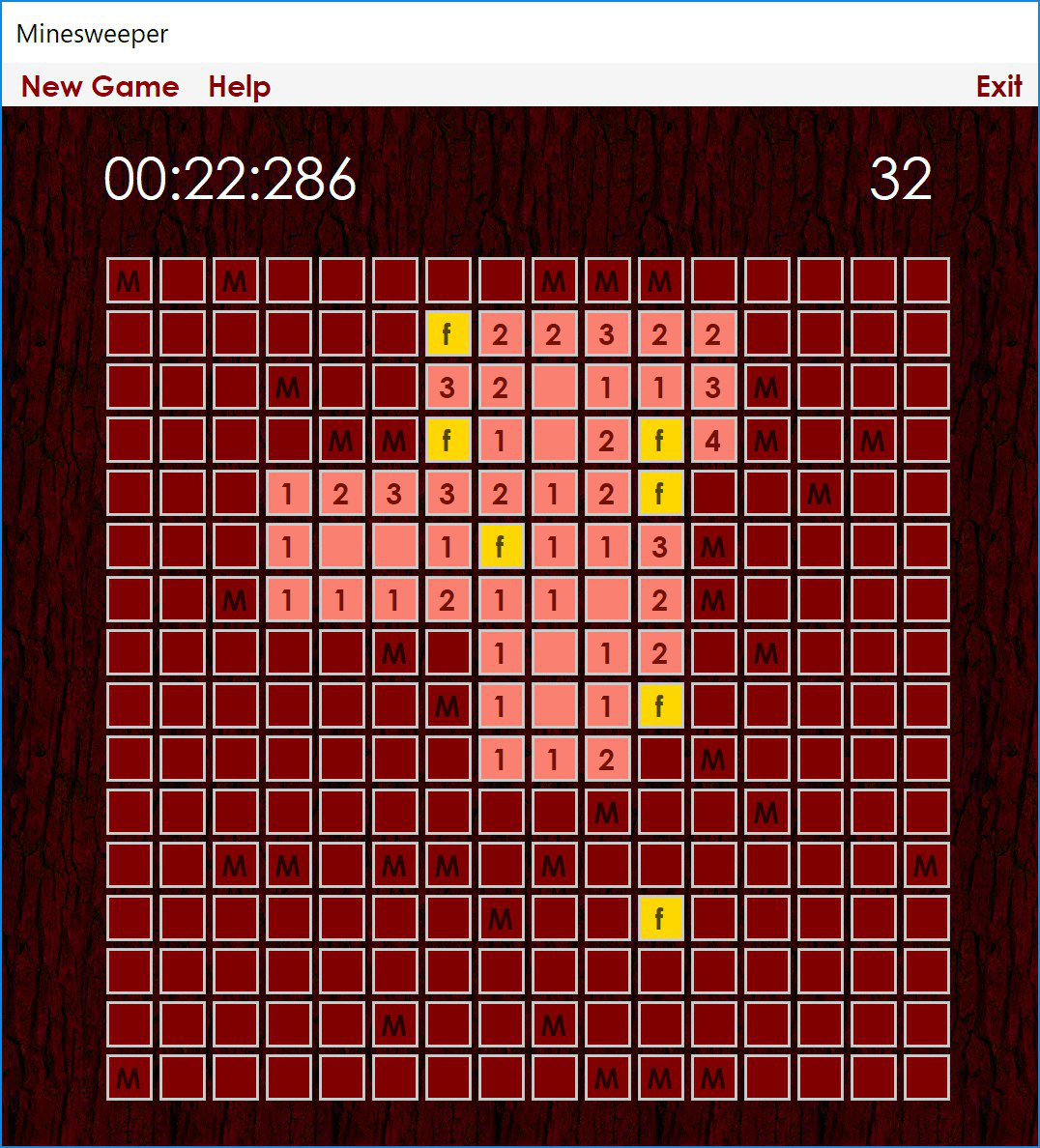


Рисунок Б.3 – Приклад відображення при видалених/пошкоджених файлах



Рисунок Б.4 – Фрагмент Help - файлу