

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

(повна назва кафедри)

ДОМАШНЯ КОНТРОЛЬНА РОБОТА

з дисципліни (кредитного **Основи програмування**
модуля)

спеціальність **122 Комп'ютерні науки**

спеціалізація **Комп'ютерні технології в біології та медицині**

На тему **Розробка гри «Сапер»**

(тема індивідуального завдання)

Варіант №

**Виконав студент 1-го курсу гр. БС-02
ДІЛОНГ СЕРГІЙ МИХАЙЛОВИЧ**

*Засвідчую, що у роботі немає запозичень з праць
інших авторів без відповідних посилань.*

Студент (-ка) _____

Перевірів **ст. викл. Ольга ВДОВИЧЕНКО**
ст. викл. Галина КОРНІЄНКО

Бали за роботу студента відповідно до РСО за

Функціональність, Алгоритм, Розрахунок (6- 9 балів)	
Оформлення (2-3 балів)	
Захист (2-3 балів)	
ВСЬОГО (6-15балів)	

(підпис викладачів)

Київ – 2020р.

Київ – 2020 р.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
 Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

ЗАВДАННЯ на домашню контрольну роботу студенту

ДІЛОНГ СЕРГІЙ МИХАЙЛОВИЧ

(прізвище, ім'я, по батькові)

1. Тема роботи (варіант) *Розробка гри «Сапер»*

2. Термін подання студентом роботи *14-21 грудня 2020 року*

3. Вихідні дані до роботи *Гра «Сапер»*

4. Зміст роботи *Розробка гри сапер з застосуванням Tkinter,
Вивчення простих методів запобігання втручання користувача*

5. Дата видачі завдання *19-23 жовтня 2020 р.*

Календарний план

№ з/п	Назва етапів виконання курсової роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання на КР	19-23 жовтня 2020	
2	Оформлення розділу з	До 10 листопада 2020	
3	Оформлення розділу з	До 25 листопада 2020	
4	Оформлення розділу з	До 20 грудня 2020	
5	Оформлення ДКР	10-11 грудня	
6	Подання в електронному вигляді ДКР на перевірку	До 14 грудня 2020	
7	Подання пакету документів по ДКР до захисту	До 15 грудня 2020	
8	Захист ДКР	28-30 грудня 2020	

Студент

(підпис)

Сергій ДІЛОНГ

(ініціали, прізвище)

Керівник роботи

(підпис)

Ольга ВДОВИЧЕНКО

(ініціали, прізвище)

ЗМІСТ

Вступ.....	5
РОЗДІЛ 1 Теоретична частина.....	7
1.1. Про гру	7
1.2. Архітектура.....	8
1.3. Складання користувацького інтерфейсу	9
ВИСНОВОК.....	10
РОЗДІЛ 2 Аналітична частина.....	11
2.1.1. Актуальність.....	11
2.2. Процедурний стиль чи ООП	11
2.2.1. Бізнес логіка	11
2.2.2. Інтерфейс	11
2.3. Ядро Toraz	12
2.4. I/O: User Space	12
2.5. Користувачі та файли збереження.....	12
2.6. Захист від вторгнення	12
2.6.1. Місце збереження	12
2.6.2. Обфускація або кодування.....	13
2.6.3. Збереження списку користувачів	13
2.6.4. Шифрування та збереження.....	14
2.6.5. Алгоритмізація файлів збереження.....	14
ВИСНОВОК.....	14
РОЗДІЛ 3 Практична частина.....	15
3.1. Генерація та збереження ігрового поля	15
3.1.1. Дії користувача.....	15

	4
3.1.2. Генерація мін	15
3.1.3. Генерація чисел на клітинках	15
3.2. Рівні складності	16
3.3. Взаємодія з ігровим полем	16
3.4. Статус гри	17
3.5. Відлік часу	18
3.6. Вхід у систему	18
3.6.1. Реєстрація.....	19
3.7. Збереження.....	20
3.8. Завантаження	21
3.9. Шифрування	21
3.10. Рейтинг гравців.....	22
3.11. Рахунок бомб	23
3.12. Обрання рівню	24
3.13. UIController – Поточний стан програми	24
3.14. Вихід з програми	25
3.15. Застосування піктограми	25
ВИСНОВОК.....	25
Загальні висновки.....	27
Додаток А.....	28
Додаток В	48
Список використаних джерел	52

ВСТУП

Задача. Розробити власну версію класичної комп'ютерної гри «Сапер», використовуючи мову програмування Python, розширити стандартну функціональність гри, додавши можливість зберігати гру у персональному акаунті, та додавши таблицю рекордів. Запобігти втручанню користувача у файли зберігання.

Метою роботи є випробовування можливостей стандартних засобів Python стосовно розробки елементарних комп'ютерних ігор.

Отримані результати: розроблено програмний засіб на мові Python, що являє собою гру «Сапер» відповідно до умов задачі. Систематизовано процес розробки, описано використані архітектурні методи та алгоритми.

Проект також має **практичну користь**: у нових редакціях Windows сапер не встановлено за вмовчуванням (Cobbett, 2009), до того ж, навіть у старих редакціях його можливості були порівняно обмежені, тоді як сучасна версія, як може здатися, перенавантажена хмарними сервісами та рекламою Microsoft. Такий крок можна вважати на краще, адже це дало деякий поштовх ігровій індустрії (Genetski, 2014):

Онлайн-ігрові мережі, які щодня об'єднують мільйони гравців по всьому світу та надають гравцям доступ до хмарного контенту, що покращує ігровий досвід. На додаток до фізичних ігрових консолей та пов'язаних з ними мереж, споживачі також можуть отримувати доступ, купляти та грати у відеоігри через онлайн-ігрові платформи, такі як EA Origin, Microsoft Games for Windows Live та Steam, або через сервіси потокової передачі ігор, такі як OnLive.

Отже можна зробити висновок, що навіть для найпростіших ігор впровадження зручних користувацьких функцій може бути корисним для кінцевого продукту.

Тим не менш, вбачається корисним використання більш класичного дизайну, яким і асоціюється сапер у більшості гравців. Таку можливість чудово надає бібліотека користувацького інтерфейсу Tkinter, що вбудована у

пакет бібліотек Python за вмовчуванням. Використовуючи стандартні елементи можна легко відтворити описаний ефект об'ємних кнопок, що присутній у оригінальній грі.

Отже, хоча користувачі й звикли до класичного зовнішнього вигляду та функціонального мінімалізму саперу, привнесення нових можливостей без руйнування звичного UI може стати корисним: навіть старі ігри мають право на еволюцію. Бібліотека Tkinter є швидким способом інтеграції Python з користувацьким інтерфейсом, у першу чергу через те, що Tkinter постачається з Python за умовчанням.

РОЗДІЛ 1

ТЕОРЕТИЧНА ЧАСТИНА

З поставленого завдання можна виділити наступні задачі:

1. Розробка основного ядра гри
2. Розробка системи для роботи з файлами
3. Розробка ядра зберігання та завантаження
4. Розробка рейтингу користувачів
5. Розробка UI (користувацького інтерфейсу)

1.1. Про гру

Сапер є однією з найстаріших комп'ютерних ігор, що було створено близько у 60-х роках минулого століття. Її популярність розвилася після включення однієї з варіацій гри до *Microsoft Windows Entertainment Pack*, а згодом й до увійшла до Windows (Maher, 2018):

Тому той момент, коли Microsoft додала в 1992 році в Windows 3.1 нову (чи то пак стару) гру, став знаменним. Насправді гра «Сапер» дебютувала як гра одного з перших Entertainment Pack, де вона стала улюбленою для досить великої кількості гравців. Серед них був і не хто інший, як сам Білл Гейтс, який настільки до неї звик, що в результаті видалив її зі свого комп'ютера ... тільки для того, щоб отримувати нову дозу гри на машинах своїх колег.

Але на відміну від Solitaire, в Minesweeper є більше пасток звичних нам відеоігор, в тому числі таймер, мотивуючий грати швидко, щоб набрати максимум очок.

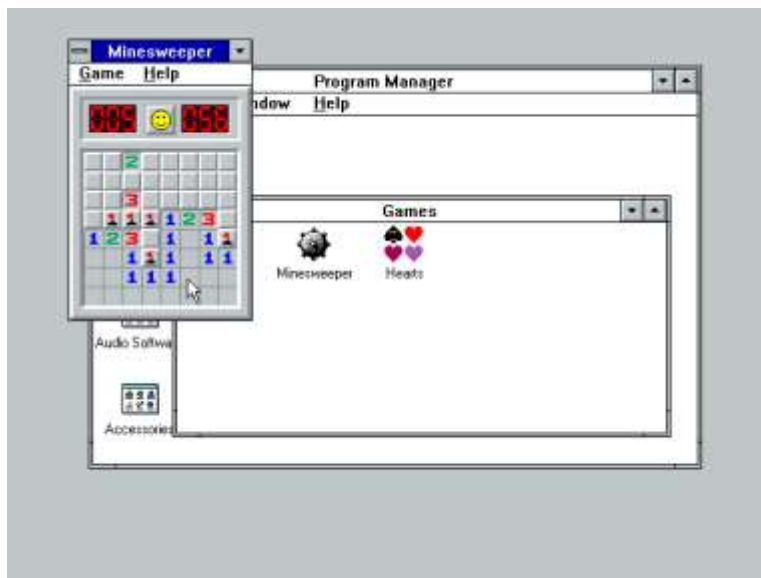


Рисунок 1.1 Оригінальний вигляд гри у складі з Windows 3.1

Та крім залежності, дана гра розвиває логіку та когнітивні здібності (Daphne Bavelier, 2006).

1.2. Архітектура

Одними з головних архітектурних рішень застосунку є рішення строгого розділення останнього на структурні частини, це допомагає спростити зневадження та суттєво підвищує масштабованість застосунку, що суттєво у даному випадку через те, що до основної гри буде додано комплекси зберігання та рейтингу.

Модель, застосована у даній роботі, називається **MVC** (Model – View – Controller), що перекладається як Модель – Вигляд – Контролер. Окрім гнучкості, дана модель спрощує розуміння коду та дає змогу повторного використання компонентів.

Модель – це ядро програми, яке надає та приймає дані, змінює свій стан, для усього іншого середовища працює як «чорний ящик».

Вигляд – комплекс, що відповідає за відображення стану моделі до користувача, найчастіше являє собою використовуваний програмний комплекс для малювання користувацького інтерфейсу.

Контролер – передавач дій користувача до моделі, сповіщувач про необхідність змін, є посередником між користувачем та моделлю.

Уперше модель була описана у 1978 році Р. Тьюге, MVC стала висновком, до якого прийшли розробники під час створення мови програмування Smalltalk (Н., 1978).

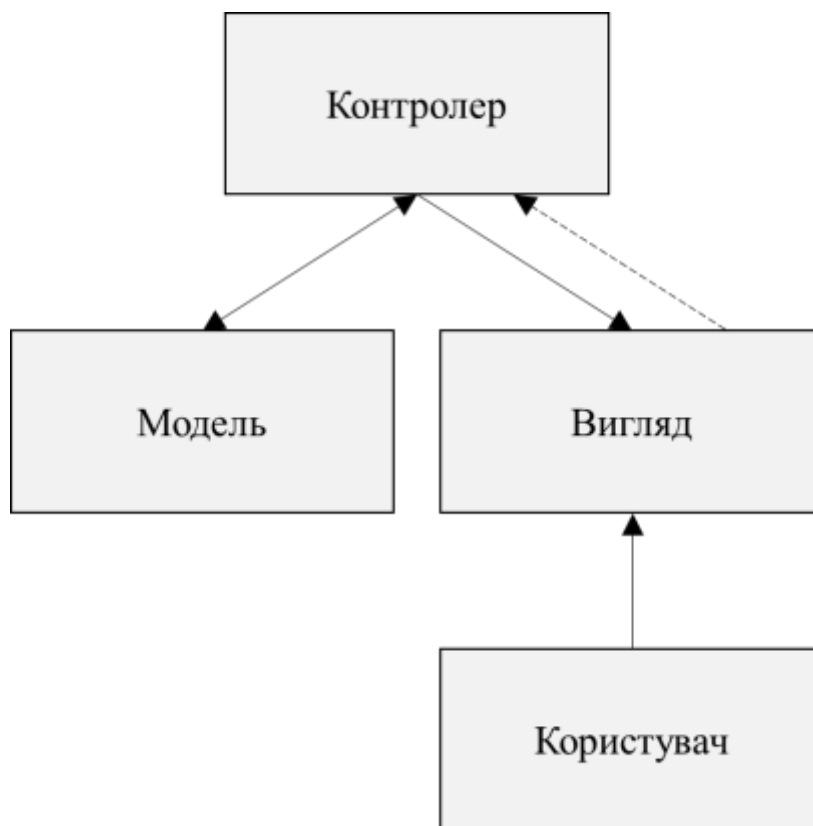


Рисунок 1.2 Схема роботи MVC

Саме основні якості, притаманні моделі MVC, й роблять її найпривабливішим рішенням для даного проєкту.

1.3. Складання користувацького інтерфейсу

Додаткові функції не мають відволікати від власне гри, тому вони не мають привертати до себе уваги. Завдяки тому, що застосунок готується саме під РС, з'являється можливість використовувати ті елементи інтерфейсу, які допустимі лише на ПК, але які у той же час є неймовірно ефективними.

Серед пристроїв вводу електронного пристрою є тип «Вказівників», що контролюють курсор миші. Дані пристрої поділяються за точністю вказівника та можливістю повідомляти про поточний стан вказівника (статус «hover»).

	Точний	Не точний
Підтримує hover	Миша / Активне перо	Джойстик / Контролер
Без hover	Стилус	Сенсорний дисплей

Таблиця 1.1 Пристрої вводу за точністю та наявністю hover

Сьогодні застосунки часто роблять кросплатформеними, тобто такими, щоб код, що був написаний лише один раз, працював усюди. Нажаль, розробка таких інтерфейсів потребувала б додаткових зусиль як з боку програмістів, так і з боку дизайнерів. Користувачі ПК у абсолютній більшості випадків використовують мишу, що дає можливість вмістити усі функціональні елементи інтерфейсу у такий лаконічний елемент, як «Каскадне меню», що знаходиться під заголовком програми:

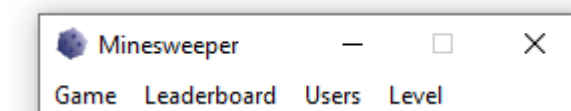


Рисунок 1.3 Приклад реалізації каскадного меню у програмі

Щоб не ускладнювати сприйняття програми, усі елементи, що не причетні до ігрового процесу, можна виділити у спливаючі та діалогові вікна.

За умовою задачі, необхідно реалізувати вибір рівня складності. Для цього каскадне меню підтримує тип *radio*¹.

ВИСНОВОК

Було виконано аналіз використовуваних технологій та методів їх застосування. Описано принцип розробки під назвою MVC. Розглянуто можливі елементи інтерфейсу, що можуть бути задіяні у проєкті, та чому.

За результатами аналізу зроблено висновок, що MVC є гарним методом розробки, якщо за мету стоять вивчення алгоритмізації та вивчення розробки інтерфейсів як дві окремі задачі.

Зроблено висновок, що, зважаючи на пристрої-вказівники, використовувані для ПК, не слід нехтувати унікальними елементами інтерфейсу, що надає ця платформа, такими як каскадні меню та спливаючі вікна.

¹ Radio це такий тип перемикача, коли може бути обрано лише один з запропонованих варіантів, зазвичай позначається колом або пустотою, якщо варіант не обрано, і колом з крапкою/заповненим кружечком, якщо варіант обрано. Через те, що у настановах щодо дизайну Windows та MacOS не існує radio-перемикача для каскадного меню, Tkinter використовує форму, що виглядає як checkbox (Roseman, 2007-2020)

РОЗДІЛ 2

АНАЛІТИЧНА ЧАСТИНА

2.1.1. Актуальність

На даний момент існує безліч реалізацій гри, адже гра стала класикою (Cobbett, 2009), гра навіть отримала розвиток у сучасних ігрових сервісах (Genetski, 2014), тим не менш, те, що це класика, дає нам можливість стверджувати, що величезний попит має дуже диверсифіковані забаганки: від специфічної ОС до особливих примх щодо функціональних можливостей, як-от збереження гри.

Перед початком роботи розглянемо доступні рішення та доберемо найкращі варіанти, щоб оминати архітектурних помилок на етапі проектування.

2.2. Процедурний стиль чи ООП

2.2.1. Бізнес логіка

У даному проєкті для бізнес логіки використовується модель MVC. Це зроблено з метою полегшити майбутню інтеграцію додаткових функцій у гру. Модель MVC прийнято використовувати там, де кожна подія вимагає оновлення інтерфейсу (Попов, 2013), що можна застосувати й до саперу для спрощення розробки.

2.2.2. Інтерфейс

Tkinter є рекомендованою бібліотекою для розробки інтерфейсів, що постачається з Python за умовчанням. Дана бібліотека є дуже старою, через що її API є не зовсім консистентними. Тим не менш, хоча й у Tkinter використовується принцип віджетів-об'єктів, останні є мало сумісними з об'єктно-орієнтованим стилем через велику кількість рішень, що використовує процедурну парадигму. Через це саме контролер використовуватиме процедурний стиль.

Дана частина програми називається «Controller» у MVC, у той час як Tkinter є нашим «View».

2.3. Ядро Toraz

Основну частину програми складає логіка роботи «Саперу». Для спрощення, ця частина програми отримала кодове ім'я *toraz*. Гру може бути заключено у єдиний клас Python, що надає методи та поля різних категорій, що буде розглянуто у практичній частині:

1. Взаємодія з ігровим полем
2. Інформаційні (статистика і т.п.)
3. Службові (для інтеграції з іншими частинами застосунку)

Дана частина програми й називається «Model» у MVC. Вона лише приймає на вхід дії користувача та повертає на вихід поточний стан поля.

2.4. I/O: User Space

Набір функцій та класів, відповідальних за ввід/вивід, було названо *User Space* та виділено у відповідний модуль. Сюди входять єдині для усієї програми функції для запису/читання файлів, які на ходу також виконують їх обфускацію/деобфускацію, тобто ускладнення розуміння інформації людиною, що було виконано згідно з умовою задачі.

2.5. Користувачі та файли збереження

Кожна операційна система має свої стандарти збереження власних файлів користувача, тому саме це місце збереження і було обрано. Перевага цього метода у тому, що файли зберігатимуться на ПК незалежно від того, чи видалено програму, чи ні, тому з'являється можливість безпечно оновлювати версію програми. Даний метод використовує абсолютна більшість програм.

Гра створює каталог у відповідній користувацькій папці для налаштувань програм. Структура з файлів та папок файлової системи використовується для розрізнення загальних файлів (список користувачів, рейтинг) та персональних (файли збереження).

2.6. Захист від вторгнення

Програма активно використовує різноманітні тактики для ускладнення маніпуляцій над файлами гри.

2.6.1. Місце збереження

Шлях файлів збереження програми встановлено згідно з регламентом поточної ОС, тим не менш, у всіх ОС шлях для файлів зберігання програм є

прихованим за умовчанням. Одночасно, спеціалізовані програми для очистки диску чудово розуміють що це за файли, та не будуть їх чіпати.

2.6.2. Обфускація або кодування

Кодування це такий спосіб видозмінення інформації, при якому дану інформацію можна видозмінити до початкового стану, використовуючи обернену функцію, при умові, що алгоритм є відомим усім потенційним сторонам та не потребує додаткових, таємних даних для роботи.

Одним з найпростіших способів спантеличення некваліфікованих користувачів є саме кодування, наприклад, *base64*. Тим не менш, *base64* є доволі відомим методом, що також легко розпізнається людським оком та потребує середніх знань для роботи з інформацією.

Було обрано наступний метод кодування інформації:

1. Текст переводиться у кодування UTF-8
2. До кожного байту UTF-8 застосовується операція XOR на число 0xFF

Даний метод отримує у результаті випадковий на вигляд набір байтів, що нагадує зашифрований файл, яким воно насправді не є. Сенсу у шифруванні загальнодоступних файлів не має, адже тоді ключ доведеться зберігати у програмі, звідки його можна й отримати.

2.6.3. Збереження списку користувачів

Дані зберігатимуться у корені папки гри у єдиному для цього файлі, у форматі JSON та з застосуванням кодування.

Файл буде містити лише імена користувачів та хеші паролів з випадковою «сіллю» для запобігання брутфорсу. Збереження хешів паролів необхідно для того, щоб один користувач не міг використовувати кілька рідних паролів.

Хешування це необоротна операція перетворення інформації, яка завжди дає такий результат, який можна порівняти з пізнішим використанням цієї ж функції з цими ж параметрами. Використовується для перевірки ідентичності інформації без збереження власне інформації.

2.6.4. Шифрування та збереження

У програмі використовується алгоритм шифрування, чимось схожий до шифру Цезаря, але кожна літера робить відступ на n кількість бітів ключа. Ключем є md5 хеш користувацького паролю.

Шифрування унеможливорює доступ до персональних файлів існуючих користувачів.

Файли зберігання шифруються та кодуються описаними методами, зберігаються у окремій папці каталогу програми.

2.6.5. Алгоритмізація файлів збереження

Ігрове поле повністю залежить від чотирьох параметрів:

- Солі для об'єкту Random
- X та Y першого кроку гравця
- Рівню гри

Саме ці дані та інформація про відкриті/промарковані клітинки й потрапляє у файли зберігання. Це фактично унеможливорює втручання у файли зберігання з метою дізнатись про розташування мін, адже міни генеруються алгоритмічно з наданих параметрів.

ВИСНОВОК

Було описано використовувані структури та алгоритми. Описано стиль програмування для різних частин програми и виділено їх у парадигмі MVC. Описано складові ядра Тораз.

Описано методи ускладнення отримання інформації про розташування мін з файлів зберігання шляхом їх шифрування, кодування та алгоритмізації. Методи, описані у даній роботі, намагається мінімально покладатися на сторонні бібліотеки. До того ж, реалізація гри за описаними алгоритмами взагалі не потребує бібліотек, окрім стандартних.

РОЗДІЛ 3

ПРАКТИЧНА ЧАСТИНА

3.1. Генерація та збереження ігрового поля

Розділимо поле сапера на такі шари:

3.1.1. Дії користувача

«Візуальна» частина – Усі зміни до поля, що були внесені користувачем після його початкової генерації: розблоковані клітинки, прапорці й активована міна, якщо така є.

Зберігається у двовимірному масиві й змінюється відповідно до дій користувача.

3.1.2. Генерація мін

Міни – Двовимірний масив, що зберігає «1» у всіх комірках, де наявна міна.

Як описано у аналітичній частині, міни генеруються функцією *random_int* (по два числа для координат) за заданим *seed*. За умовчанням сіллю випадкової функції є поточний час. Якщо міна у даній локації вже є, або дана точка є точкою початку гри, генерується зайва пара чисел, через що змінюється хід розташування кожної наступної міни.

3.1.3. Генерація чисел на клітинках

Підкладка з числами – Наперед обчислена підкладка з числами-підказками, що й дають змогу користувачеві грати.

Для кожної клітинки, обчислюється кількість суміжних мін й записується у двовимірний масив за відповідними координатами.

Дані числа не потрібно зберігати, достатньо генерувати один раз під час завантаження гри. Ці числа повністю залежать від розташування гри та не змінюють свого місцеположення під час гри, тому їх можна використовувати під час обчислювальних операцій, наприклад, алгоритм для очистки поля використовує число у комірці щоб зрозуміти, чи немає міни поряд, згідно з чим вже й робляться відповідні рішення.

Від того, чи є біля поля міна, залежить очистка поля. Якщо очистка починається з такої комірки, поряд з якою є міна, сусідні комірки з числами

також буде відкрито. Якщо ж дана клітинка пуста, комірки буде відкрито до найближчих «числових кордонів» включно.

Гра завжди має облямовувати заблоковані клітинки числами, щоб зменшити кількість ситуацій, коли зробити однозначний хід у грі неможливо.

Недотримання даної умови є частою помилкою під час реалізації «Саперу», хоча вона й є не критичною і гра залишається можливою на нижчих рівнях складності.

3.2. Рівні складності

У даній програмі рівень складності це ціле число від 0 до 2, що й визначає складність гри.

Для цього було створено клас `LevelDescriptor`, який у статичному полі зберігає масив характеристик рівнів складності.

При запиті рівню складності, клас отримує доступ до відповідного індексу у масиві та присвоює собі ці значення.

3.3. Взаємодія з ігровим полем

Взаємодія складається з чотирьох видів дій: клік по пустій комірці, встановлення прапорця, встановлення знаку питання, очищення комірки.

Окрім того, на кожну з цих дій гра має генерувати відповідний стан ігрового поля, повертаючи його у відповідній функції за запитом.

Найпростішими діями у даному випадку є прапорець, знак питання та очищення, бо ці дії є пасивними та не впливають на ігрове поле.

За натисканням на комірку, деякий алгоритм має очистити певні пусті комірки, доки не напнеться на міну. Даний алгоритм не є складним у реалізації та працює рекурсивно:

1. Якщо поточна клітинка не знаходиться поряд з міною або є тією клітинкою, на яку натиснув користувач – очистити й продовжити, якщо є поточна клітина має поряд із собою міну – очистити й припинити виконання функції.
2. Рекурсивне виконання поточної функції на усіх суміжних клітинках


```

def __auto_clean(self, x, y, c, allow_nonzero=False):
    if c is None:
        c = []
    if [x, y] in c:
        return
    c.append([x, y])
    if (self.cache[y][x] == 0 and self.mines[y][x] != 1) or (self.cache[y][x] > 0 and allow_nonzero):
        self.visuals[y][x] = 1
    elif self.cache[y][x] > 0 and self.mines[y][x] != 1:
        self.visuals[y][x] = 1
        return
    else:
        return

    if y > 0:
        self.__auto_clean(x, y - 1, c)
    if y < self.level.level_sizes()[1] - 1:
        self.__auto_clean(x, y + 1, c)
    if x > 0:
        self.__auto_clean(x - 1, y, c)
    if x < self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y, c)
    if y > 0 and x > 0:
        self.__auto_clean(x - 1, y - 1, c)
    if y > 0 and x < self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y - 1, c)
    if y < self.level.level_sizes()[1] - 1 and x > 0:
        self.__auto_clean(x - 1, y + 1, c)
    if y < self.level.level_sizes()[1] - 1 and x < self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y + 1, c)

```

Рисунок 3.1 Реалізація описаної функції на Python

3.4. Статус гри

Гра має три стани – діє, зупинено, призупинено.

Гру, що призупинено, можна перевести назад у стан «діє», тоді як ту, що зупинено – ні.

Зупиненою гра є тоді, коли вона повністю завершена – учасник натиснув на міну та програв, або відкрив усі комірки та виграв. Коли гра стартує, програма зберігає поточний час та починає його відлік.

Коли гра зупиняється – програма рахує скільки часу минуло під час сесії та додає його до часу минулих сесій, це дає змогу зберігати час навіть якщо гру було збережено кілька разів.

Крім цього, ядро Toraz має метод, яким може повідомляти інші частини програми про завершення гри. Це використовується у рейтингу користувачів. При завершенні гри рейтинг сам отримує метрики поточної гри, обраховує бали та, якщо користувач встановив новий рекорд, пропонує його записати.

3.5. Відлік часу

Відлік часу ведеться у секундах, дані про проведений час зберігаються у файлі збереження. Для отримання пройденого часу існує єдина функція, яка застосовується як у інтерфейсі, так й у файлах зберігання.

Коли гра запускається, контролер дає задачу Tkinter рекурсивно оновлювати таймер гри щосекунди.

За це відповідає також функція форматування часу, яка ділить секунди на 60, що є годинами, та прираховує остачу до секунд. Щоб виглядати відповідно до цифрового формату, якщо число містить лише один розряд, до нього спереду дописується нуль. Програма не веде відліку у годинах.

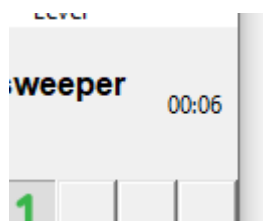


Рисунок 3.2 Активний таймер

3.6. Вхід у систему

Взагалі користувачу необов'язково власноруч користуватися меню «Користувачі»: система сама запропонує увійти, якщо це буде доречно. Функція для виклику форми входу є універсальною та підтримує так званий *callback*, функцію, що буде викликано, як тільки користувач завершить авторизацію або реєстрацію.

Для паролю застосовано спеціальний тип поля вводу з маскою у вигляді зірочок:

```
tk.Entry(w, textvariable=password, show="*").pack()
```

Рисунок 3.3 Застосування маскування

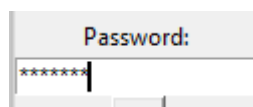


Рисунок 3.4 Результат

Вхід у систему відбувається так:

- Зчитування та розшифровка файлу користувачів
- Виявлення уведеного користувача
- Хешування паролю та звірка з наявним хешем

- Збереження уведених даних у тимчасову сесію у оперативній пам'яті

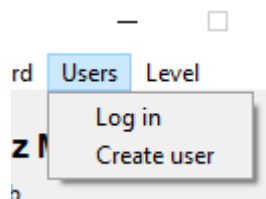


Рисунок 3.5 Меню входу

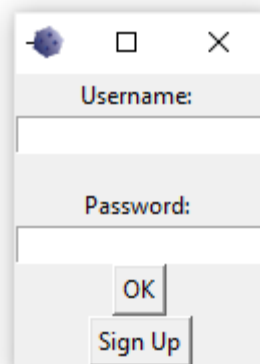


Рисунок 3.6 Форма входу

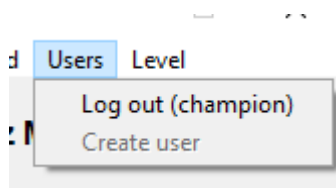


Рисунок 3.7 Користувача авторизовано

3.6.1. Реєстрація

Для реєстрації було створено окреме вікно, яке потребує увести пароль двічі, для впевненості у тому, що пароль було уведено правильно:

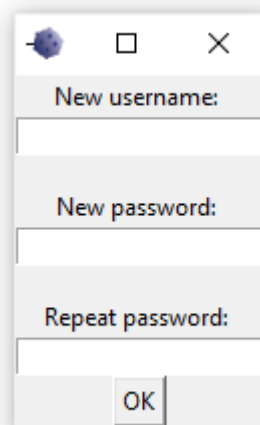


Рисунок 3.8 Форма реєстрації

Якщо користувач вже існує або пароль уведено невірно, буде виведено помилку:

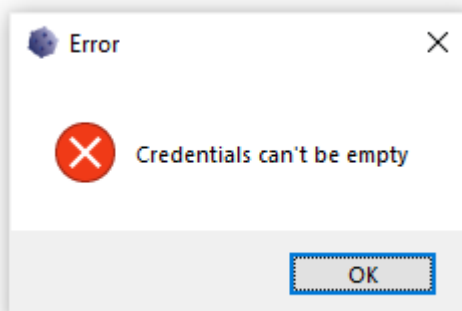


Рисунок 3.9 Дані для реєстрації не уведено

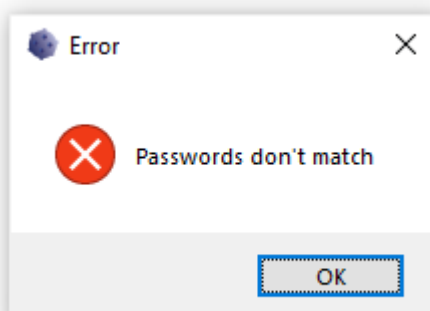


Рисунок 3.10 Паролі не зівпадають

3.7. Збереження

Збереження пропонується лише коли користувач намагається закрити програму, або якщо гру завершено.

Якщо ж користувач власноруч обирає пункт «Зберегти», гру буде збережено та завершено.

Та, перед цим, якщо користувача не авторизовано, йому буде запропоновано увійти або створити нового користувача.

Після шифрування та кодування даних користувачу пропонується почати нову гру, під час збереження поточна гра негайно завершується, щоб попередити нехтування функцією. Якщо виявлено більше десяти збережень, найстаріші видаляються.

```
def gen_name(state: GameState):
    return 'game-' + str(state.salt) + '-' + str(state.first_move_x) + '-' + str(state.first_move_y) + '-' + str(
        time.time() // 1) + '.dat'
```

Рисунок 3.11 Генерація назви файлу

3.8. Завантаження

Завантаження це окреме меню, доступне лише тоді, коли немає активної гри. Список файлів отримується з каталогу користувача та сортується за датою змінення. Якщо виявлено більше десяти збережень, найстаріші видаляються:

```
lst = sorted(lst, key=lambda i: i['date'], reverse=True)
rm = lst[10:]
lst = lst[:10]

for fi in rm:
    fi['file'].unlink()
```

Рисунок 3.12 Реалізація видалення старих збережень

За зберігання відповідає клас зберігача `DataFile`. За умовчанням він не присутній у оперативній пам'яті і створюється лише коли користувач бажає зберегти файл. Якщо ж гру було завантажено зі збереження, даний клас присутній та відповідає за негайне збереження у вже існуючий файл при запиті користувача.

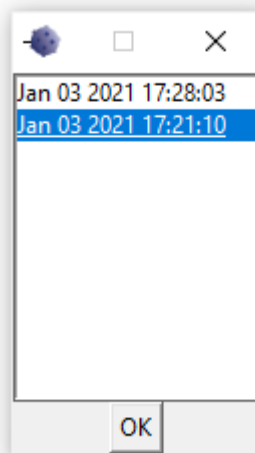


Рисунок 3.13 Форма завантаження

3.9. Шифрування

Шифрування даних відбувається з використанням паролю у сесії за описаним у аналітичній частині алгоритмом:

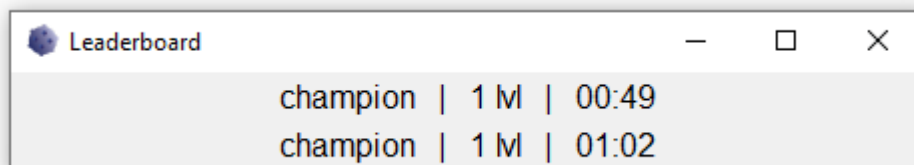
```
def pass_shift(p, string, decrypt=False):
    c = []
    for i in range(len(string)):
        k = p[i % len(p)]
        if decrypt:
            e = chr((ord(string[i]) - ord(k) + 256) % 256)
        else:
            e = chr(ord(string[i]) + ord(k) % 256)
        c.append(e)
    r = ''.join(c)
    return r
```

Рисунок 3.14 Реалізація шифрування за зміщенням байтів по ключу

3.10. Рейтинг гравців

Рейтинг (Leaderboard) використовує ті ж методи захисту, що й файл користувачів, адже рейтинг є загальнодоступним й зашифрувати його неможливо.

Список кращих отримується з цього файлу й сортується на льоту за балами, які буде описано нижче, зберігаються лише перші 20 кращих.



champion		1 M		00:49
champion		1 M		01:02

Рисунок 3.15 Таблиця кращих

Додавання до рейтингу пропонується лише тоді, коли гру виграно і встановлено новий рекорд у поточному рівні складності у балах за формулою:

```
def rate(level, time):
    return (time / (level.level_sizes()[0] * level.level_sizes()[1]) / (level.level + 1)) // 1
```

Рисунок 3.16 Формула рейтингу

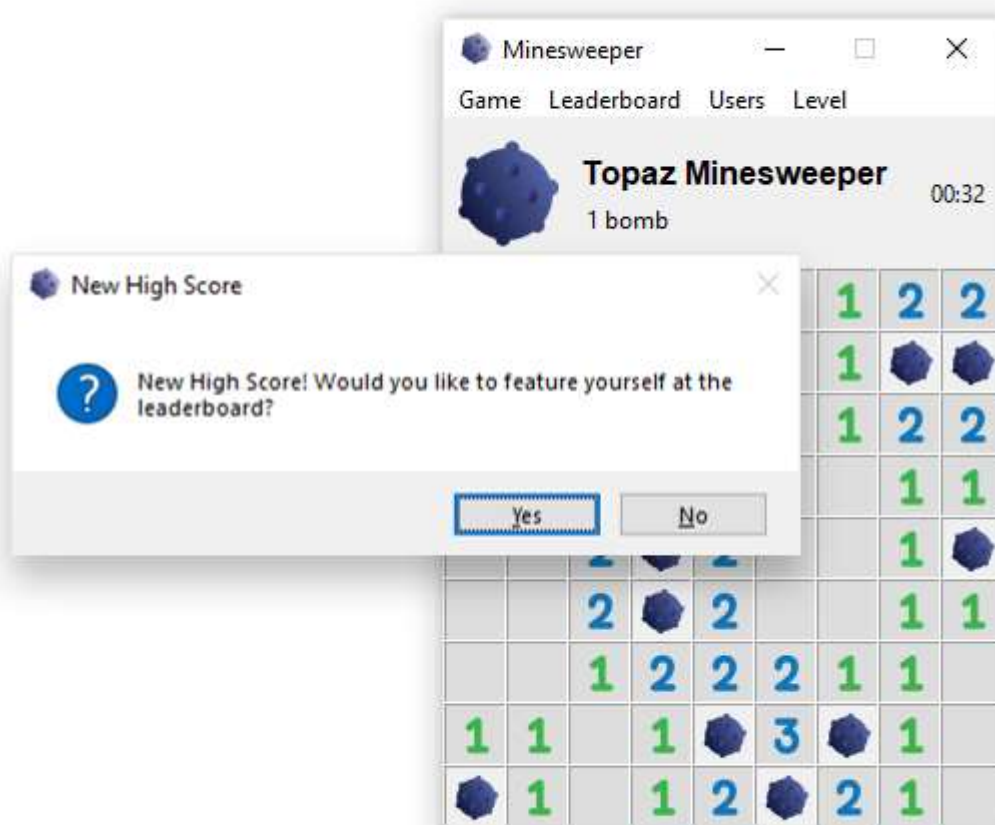


Рисунок 3.17 Пропозиція додати результат до таблиці

3.11. Рахунок бомб

Програма відображає кількість виявлених бомб. Під час гри відображається кількість, що відповідає кількості встановлених прапорців незважаючи на правильність їх розстановки. Коли гру завершено – програма звіряє розстановку прапорців з мінами та враховує лише правильно розставлені, щоб показати фактичний результат.

Дана функція є методом ядра Toraz та використовується по усій програмі так само, як й метод для врахування часу.

```
def bomb_stats(self):
    placed_guesses = 0
    for row in range(self.level.level_sizes()[1]):
        for col in range(self.level.level_sizes()[0]):
            if self.visuals[row][col] == 4 and (self.mines[row][col] == 1 or self.state != 2):
                placed_guesses += 1
    return self.level.mines_amount() - placed_guesses
```

Рисунок 3.18 Реалізація обрахунку бомб

3.12. Обрання рівню

Як було описано, обирання рівня працює у каскадному меню. Вибір доступний лише коли немає поточної гри:

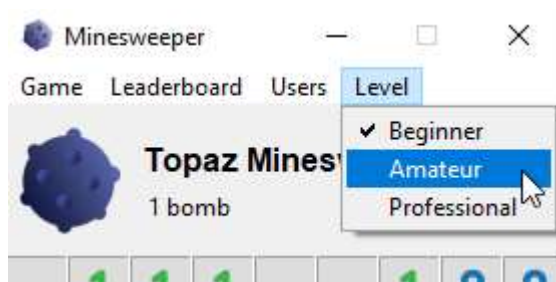


Рисунок 3.19 Зміна рівню після завершення гри

Рівень прив'язаний до «розумної» змінної Tkinter, яка вміє відслідковувати свій стан, змінювати інтерфейс відповідно до свого значення та повідомляти про зміни задану функцію.

При заданні рівню поточний стан ігрового поля скидається й генерується нове заданого розміру.

Це використовується й для кнопки скиду гри: для цього змінна присвоює сама себе.

Дана змінна належить до UIController й встановлюється на етапі ініціалізації гри:

```
class UIController:
    level = tk.IntVar()
```

Рисунок 3.20 Рівень гри у UIController

```
UIController.level.trace("w", handler)
```

Рисунок 3.21 Відслідковування зміни значення

3.13. UIController – Поточний стан програми

UIController це клас статичних змінних що зберігає різноманітні змінні, що необхідні у різних частинах програми:

- Об'єкт вікна Tkinter
- Піктограми. Python вимагає збереження ресурсів у постійній змінній, а не у змиканні, інакше збирач сміття видалить їх з пам'яті й Tkinter не вдасться відобразити картинку (Шуравин, 2020)
- Об'єкт ігрового стану, який також містить й саму гру

- Об'єкт зберігача стану гри (відсутній, якщо файл не збережено)
- Об'єкт рейтингу користувачів
- Об'єкт поточного рівня, оновлення якого викликає генерацію ігрового поля й скид гри

UIController значно спрощує розробку, бо не вимагає постійної передачі параметрів між функціями.

3.14. Вихід з програми

Tkinter дає змогу перехоплювати спробу закрити програму.

Це використовується для того, щоб переконатись, що користувач зберіг гру перед закриттям. Якщо програма має поточну ігрову сесію, а зберігач повідомляє про те, що її не збережено – користувачу ставиться відповідне запитання:

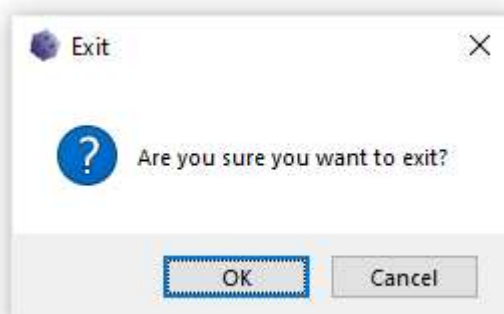


Рисунок 3.22 Підтвердження виходу, якщо сесію не збережено

3.15. Застосування піктограми

Tkinter підтримує присвоєння піктограми для усієї програми, що робиться викликом відповідного методу на головному вікні з вказанням шляху до ресурсу у форматі `.ico`.

При застосуванні піктограми до одного вікна, вона перекочує до усіх інших, що є зручним при розробці.

```
window.iconbitmap(default=resource_path('./ui_bridge/assets/mine_icon.ico'))
```

Рисунок 3.23 Приклад застосування піктограми

ВИСНОВОК

Було реалізовано описані алгоритми та об'єднано у єдиний програмний комплекс використовуючи такі структури, як UIController.

Було описано та показано роботу створеного застосунку, проведено роботу над забезпеченням файлів збереження. Проведено роботу з декорування та покращення користувацького досвіду під час використання створеної гри. Описано які способи інтеграції з Tkinter було обрано та яким чином вони працюють на практиці.

ЗАГАЛЬНІ ВИСНОВКИ

Було розроблено додаток, слідуючи трьом крокам: дослідження й аналіз, побудова архітектури та розробка принципів роботи, алгоритмування, програмування й поєднання різних компонентів у єдиний програмний комплекс.

Було досліджено історію гри «Сапер» та причини її популяризації, занепаду, способи розробки гри-головоломки «Сапер» з застосуванням мови програмування Python та середовища GUI Tkinter.

Отримана гра має можливість створювати користувачів, зберігати прогрес обмежену кількість разів, рахувати витрачений час, додавати результат на загальний список рекордів, застосовує різні практики для передбачення модифікації та переглядання файлів збереження. Було проаналізовано принцип розробки MVC для розділення програми на компоненти, переваги ООП стилю програмування для розробки ігрової логіки та функціонального стилю для роботи з Tkinter, було пояснено, як різні компоненти програми називаються у рамках MVC.

ДОДАТОК А

```

# main.py

from ui_bridge.init import launch_ui

if __name__ == '__main__':
    launch_ui()

# generate_new.py

from state_keeper.state import GameState
from topaz.core import TopazGame
from topaz.level import LevelDescriptor
from topaz.hints import hints
import time
import random

from user_space.leaderboard import process_leaderboard

def generate_new(level: LevelDescriptor, first_move_x, first_move_y,
seed=None, visuals=None, datafile=None):
    if seed is None:
        seed = time.time_ns()
        random.seed(seed)

    mines = [[0 for row in range(level.level_sizes()[0])] for column in
range(level.level_sizes()[1])]
    if visuals is None:
        visuals = [[0 for row in range(level.level_sizes()[0])] for column in
range(level.level_sizes()[1])]

    have_mines = level.mines_amount()
    used_mines = 0

    while used_mines < have_mines:
        row = random.randint(0, level.level_sizes()[1] - 1)
        col = random.randint(0, level.level_sizes()[0] - 1)
        if mines[row][col] != 1 and row != first_move_y and col !=
first_move_x:
            used_mines += 1
            mines[row][col] = 1

    cache = hints(mines)
    game = TopazGame(level, mines, visuals, cache, 1)
    if not (datafile is None):
        game.on_change = lambda state_switch: datafile.changed()
    game.start()
    game.game_done = process_leaderboard
    game.move(first_move_x, first_move_y)

    return GameState(seed, game, first_move_x, first_move_y)

def gen_name(state: GameState):
    return 'game-' + str(state.salt) + '-' + str(state.first_move_x) + '-' +
str(state.first_move_y) + '-' + str(
    time.time() // 1) + '.dat'

# state.py

from topaz.core import TopazGame

```

```

class GameState:
    salt = 0
    game: TopazGame = None
    first_move_x = None
    first_move_y = None

    def __init__(self, salt, game: TopazGame, first_move_x, first_move_y):
        self.salt = salt
        self.game = game
        self.first_move_x = first_move_x
        self.first_move_y = first_move_y

# core.py

from time import time

from topaz.entities.tiles import BombTile, LockedTile, UnlockedTile
from topaz.level import LevelDescriptor

class TopazGame:
    def __init__(self, level, mines, visuals, cache, state=0, duration=0,
on_change=lambda state_switch=None: None):
        self.level = level # type: LevelDescriptor
        self.mines = mines # type: list[list[bool]]
        self.visuals = visuals # type: list[list[int]] # 0 - locked, 1 -
unlocked, 3 - question, 4 - flag, 5 - bomb
        self.cache = cache # list[list[int]]
        self.duration = duration # int
        self.state = state # int
        self.started = 0 # int
        self.on_change = on_change
        self.game_done = lambda: None

    def bomb_stats(self):
        placed_guesses = 0
        for row in range(self.level.level_sizes()[1]):
            for col in range(self.level.level_sizes()[0]):
                if self.visuals[row][col] == 4 and (self.mines[row][col] == 1
or self.state != 2):
                    placed_guesses += 1

        return self.level.mines_amount() - placed_guesses

    def start(self):
        if self.state == 2:
            return 0
        self.state = 1
        self.started = time() // 1
        self.on_change()
        return 1

    def stop(self):
        if self.state == 2:
            return 0
        self.duration += self.timer()
        self.started = 0
        self.state = 0
        self.on_change()
        return 1

    def timer(self):
        if self.state != 1:

```

```

        return self.duration
    return self.duration + (time() // 1 - self.started)

def move(self, x, y):
    if self.state != 1:
        return 0
    if self.mines[y][x] == 1:
        self.visuals[y][x] = 5
        self.stop()
        self.state = 2
        self.game_done()
        self.on_change()
        return 1

    self.__auto_clean(x, y, [], allow_nonzero=True)

    cleared_fields = 0
    clear_fields = 0
    for row in range(self.level.level_sizes()[1]):
        for col in range(self.level.level_sizes()[0]):
            if self.mines[row][col] != 1:
                if self.visuals[row][col] == 1:
                    cleared_fields += 1
                clear_fields += 1

    if cleared_fields == clear_fields:
        self.stop()
        self.state = 2
        self.game_done()

    self.on_change()
    return 1

def put_question(self, x, y):
    if self.state != 1:
        return 0
    if self.visuals[y][x] != 0 and self.visuals[y][x] != 4:
        return 0
    self.visuals[y][x] = 3
    self.on_change()
    return 1

def put_flag(self, x, y):
    if self.state != 1:
        return 0
    if self.visuals[y][x] != 0 and self.visuals[y][x] != 3:
        return 0
    self.visuals[y][x] = 4
    self.on_change()
    return 1

def clear_cell(self, x, y):
    if self.state != 1:
        return 0
    if self.visuals[y][x] != 3 and self.visuals[y][x] != 4:
        return 0
    self.visuals[y][x] = 0
    self.on_change()
    return 1

def __auto_clean(self, x, y, c, allow_nonzero=False):
    if c is None:
        c = []
    if [x, y] in c:
        return

```

```

        c.append([x, y])
        if (self.cache[y][x] == 0 and self.mines[y][x] != 1) or
        (self.cache[y][x] > 0 and allow_nonzero):
            self.visuals[y][x] = 1
        elif self.cache[y][x] > 0 and self.mines[y][x] != 1:
            self.visuals[y][x] = 1
        return
    else:
        return

    if y > 0:
        self.__auto_clean(x, y - 1, c)
    if y < self.level.level_sizes()[1] - 1:
        self.__auto_clean(x, y + 1, c)
    if x > 0:
        self.__auto_clean(x - 1, y, c)
    if x < self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y, c)
    if y > 0 and x > 0:
        self.__auto_clean(x - 1, y - 1, c)
    if y > 0 and x < self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y - 1, c)
    if y < self.level.level_sizes()[1] - 1 and x > 0:
        self.__auto_clean(x - 1, y + 1, c)
    if y < self.level.level_sizes()[1] - 1 and x <
self.level.level_sizes()[0] - 1:
        self.__auto_clean(x + 1, y + 1, c)

def playboard(self):
    data = []

    for y in range(self.level.level_sizes()[1]):
        data.append([])
        for x in range(self.level.level_sizes()[0]):
            visual = self.visuals[y][x]
            mine = self.mines[y][x]
            if visual == 5:
                data[y].append(BombTile(boomed=True))
                continue
            if self.state == 2 and mine == 1:
                data[y].append(BombTile())
                continue
            if visual == 0:
                data[y].append(LockedTile())
                continue
            if visual == 3:
                data[y].append(LockedTile(state=2))
                continue
            if visual == 4:
                data[y].append(LockedTile(state=1))
                continue
            if visual == 1:
                data[y].append(UnlockedTile(num=self.cache[y][x]))
                continue
            data[y].append(LockedTile(state=1))
    return data

# hints.py

def hints(mines):
    cache = []

    for row in range(len(mines)):
        cache.append([])

```

```

for col in range(len(mines[row])):
    cache[row].append(0)

    if row > 0 and mines[row - 1][col] == 1:
        cache[row][col] += 1

    if row < len(mines) - 1 and mines[row + 1][col] == 1:
        cache[row][col] += 1

    if col > 0 and mines[row][col - 1] == 1:
        cache[row][col] += 1

    if col < len(mines[row]) - 1 and mines[row][col + 1] == 1:
        cache[row][col] += 1

    if row > 0 and col > 0 and mines[row - 1][col - 1] == 1:
        cache[row][col] += 1

    if row > 0 and col < len(mines[row]) - 1 and mines[row - 1][col +
1] == 1:
        cache[row][col] += 1

    if row < len(mines) - 1 and col > 0 and mines[row + 1][col - 1]
== 1:
        cache[row][col] += 1

    if row < len(mines) - 1 and col < len(mines[row]) - 1 and
mines[row + 1][col + 1] == 1:
        cache[row][col] += 1

return cache

```

level.py

```

class LevelDescriptor:
    level_map = {0: [9, 9, 10], 1: [16, 16, 40], 2: [16, 30, 90]}

    def level_sizes(self):
        return LevelDescriptor.level_map.get(self.level)[0],
LevelDescriptor.level_map.get(self.level)[1]

    def mines_amount(self):
        return LevelDescriptor.level_map.get(self.level)[2]

    def __init__(self, n: int):
        self.level = n

```

game_tile.py

```

class GameTile:
    name = 'Tile'

```

tiles.py

```

from topaz.entities.game_tile import GameTile

```

```

class UnlockedTile(GameTile):
    num = 0
    name = 'Unlocked Tile'

    def __init__(self, num: int = 0):

```



```

        self.num = num

class LockedTile(GameTile):
    state = 0 # 0 - none, 1 - flag, 2 - question
    name = 'Locked Tile'

    def __init__(self, state: int = 0):
        self.state = state

class BombTile(GameTile):
    boomed = False
    name = 'Bomb Tile'

    def __init__(self, boomed: bool = False):
        self.boomed = boomed

# draw_field.py

from tkinter import *
from tkinter import ttk

from topaz.entities.tiles import UnlockedTile, LockedTile, BombTile
from topaz.level import LevelDescriptor

def draw_field(content, level: LevelDescriptor, field, handler):
    from ui_bridge.ui_controller import UIController

    for row in range(level.level_sizes()[1]):
        for col in range(level.level_sizes()[0]):
            el = field[row][col]
            fr = None
            if isinstance(el, LockedTile):
                fr = ttk.Frame(content, relief='raised', width=30, height=30,
borderwidth=0)

                img = None
                if el.state == 1:
                    img = UIController.assets[10]
                if el.state == 2:
                    img = UIController.assets[11]
                if el.state != 0:
                    l = Label(fr, image=img, height=22, width=22)
                    l.grid(column=0, row=0)
                    l.bind("<Button-3>", lambda *a, row=row, col=col:
handler(col, row, interaction=2))
            else:
                fr = ttk.Frame(content, width=30, height=30, relief='groove',
borderwidth=1)
                if isinstance(el, BombTile):
                    img = UIController.assets[9]
                    Label(fr, image=img, height=25, width=25, bg=('red' if
el.boomed else None)).grid(column=0, row=0)
                if isinstance(el, UnlockedTile):
                    img = UIController.assets[el.num]
                    Label(fr, text=str(el.num), image=img, bg='#ddd',
height=25, width=25).grid(column=1, row=1)

                if not isinstance(el, LockedTile) or (el.state != 1 and el.state
!= 2):
                    fr.bind("<Button-1>", lambda *a, row=row, col=col:
handler(col, row, interaction=1))

```

Додаток А

```

fr.bind("<Button-3>", lambda *a, row=row, col=col: handler(col,
row, interaction=2))

fr.grid(column=col, row=row)

# exit.py

from tkinter import messagebox

def wanna_exit(window):
    from ui_bridge.ui_controller import UIController

    if UIController.state is None or messagebox.askokcancel('Exit', 'Are you
sure you want to exit?'):
        window.destroy()

# init.py

import pathlib
import time
import tkinter as tk

from state_keeper.generate_new import generate_new
from ui_bridge.exit import wanna_exit
from topaz.entities.tiles import LockedTile
from topaz.level import LevelDescriptor
from ui_bridge.draw_field import draw_field
from ui_bridge.login import sign_up_ui
from ui_bridge.resource_path import resource_path
from ui_bridge.save import save
from ui_bridge.login import login_ui
from user_space.data_file import DataFile
from user_space.leaderboard import process_leaderboard, leaders_window
from user_space.session import Session
from user_space.user_dir import prepare_personal_folder

def launch_ui():
    window = tk.Tk()
    window.title('Minesweeper')
    window.resizable(False, False)

    window.iconbitmap(default=resource_path('./ui_bridge/assets/mine_icon.ico'))
    window.option_add('*tearOff', False)
    window.protocol("WM_DELETE_WINDOW", lambda *args: wanna_exit(window))

    def handler(*args):
        UIController.state = None
        UIController.saver = None
        UIController.menu['game'].entryconfigure(0, state='disabled')
        UIController.menu['game'].entryconfigure(1, state='disabled')
        UIController.menu['game'].entryconfigure(2, state='normal')
        UIController.menu['users'].entryconfigure(0, state='normal')
        first_paint(
            window, LevelDescriptor(UIController.level.get()),
            lambda x, y, interaction: move_handler(x, y, interaction=1,
new_game=True)
        )
        UIController.menu['level'].entryconfigure(0, state='normal')
        UIController.menu['level'].entryconfigure(1, state='normal')
        UIController.menu['level'].entryconfigure(2, state='normal')

```

```

menu = tk.Menu(window)
window.config(menu=menu)

from ui_bridge.ui_controller import UIController

UIController.window = window

UIController.menu['game'] = tk.Menu(menu)
menu.add_cascade(label="Game", menu=UIController.menu['game'])

UIController.menu['leader'] = tk.Menu(menu)
menu.add_cascade(label="Leaderboard", menu=UIController.menu['leader'])

UIController.menu['users'] = tk.Menu(menu)
menu.add_cascade(label="Users", menu=UIController.menu['users'])

UIController.menu['level'] = tk.Menu(menu)
menu.add_cascade(label="Level", menu=UIController.menu['level'])

UIController.menu['game'].add_command(label="Reset", state='disabled',
command=handler)
UIController.menu['game'].add_command(label="Save", state='disabled',
command=save)
UIController.menu['game'].add_command(label="Load...", command=load_ui)
UIController.menu['game'].add_separator()
UIController.menu['game'].add_command(label="Exit", command=lambda *args:
wanna_exit(window))

UIController.menu['leader'].add_command(label="See leaderboard",
command=leaders_window)

UIController.menu['level'].add_radiobutton(label="Beginner",
variable=UIController.level, value=0)
UIController.menu['level'].add_radiobutton(label="Amateur",
variable=UIController.level, value=1)
UIController.menu['level'].add_radiobutton(label="Professional",
variable=UIController.level, value=2)

UIController.menu['users'].add_command(label="Log in")
UIController.menu['users'].add_command(label="Create user",
command=lambda: sign_up_ui(lambda: None))

def update_login_menu():
    usr = Session.get_user()
    if usr is None:
        UIController.menu['users'].entryconfigure(0, label="Log in",
command=lambda: login_ui(lambda: None))
        UIController.menu['users'].entryconfigure(1, state='normal')
    else:
        UIController.menu['users'].entryconfigure(0, label='Log out (' +
usr.username + ')',
command=lambda:
Session.apply_user(None))
        UIController.menu['users'].entryconfigure(1, state='disabled')

update_login_menu()
Session.handlers.append(update_login_menu)

logo =
tk.PhotoImage(file=resource_path('./ui_bridge/assets/mine_logo.png'))
tk.Label(window, image=logo, height=70).grid(column=1, row=1, rowspan=2)

name_label = tk.Label(window, text='Topaz Minesweeper')
name_label.config(font=(None, 12, 'bold'))
name_label.grid(column=2, row=1, sticky='sw')

```

```

sign = tk.Label(window, text=' Sergey Dilong BS-02')
sign.grid(column=2, row=2, sticky='nw')
UIController.sign = sign

time = tk.Label(window, text='00:00')
time.grid(column=3, row=1, rowspan=2)

def tick():
    if UIController.state is None:
        time.config(text='00:00')
        time.after(1000, tick)
    return
    timer = UIController.state.game.timer()
    time.config(text=str(int(timer // 60)).zfill(2) + ':' + str(int(timer
% 60)).zfill(2))
    time.after(1000, tick)

UIController.field = tk.Frame(window)
UIController.field.grid(column=1, row=3, columnspan=3)

UIController.level.trace("w", handler)

handler()
tick()
window.mainloop()

def first_paint(window, level, handler):
    from ui_bridge.ui_controller import UIController
    visuals = [[LockedTile() for row in range(level.level_sizes()[0])] for
column in range(level.level_sizes()[1])]
    if not (UIController.field is None):
        UIController.field.destroy()
    UIController.field = tk.Frame(window)
    UIController.field.grid(column=1, row=3, columnspan=3)
    draw_field(UIController.field, level, visuals, handler)
    UIController.sign.config(text=' Sergey Dilong BS-02')

def move_handler(x, y, interaction: int, new_game=False):
    from ui_bridge.ui_controller import UIController
    exam_cell = None

    if new_game:
        UIController.state =
generate_new(LevelDescriptor(UIController.level.get()), x, y)
    else:
        if UIController.state.game.state == 0:
            UIController.state.game.start()
        if interaction == 1:
            UIController.state.game.move(x, y)
        elif interaction == 2 and isinstance(exam_cell :=
UIController.state.game.playboard()[y][x], LockedTile):
            if exam_cell.state == 0:
                UIController.state.game.put_flag(x, y)
            elif exam_cell.state == 1:
                UIController.state.game.put_question(x, y)
            elif exam_cell.state == 2:
                UIController.state.game.clear_cell(x, y)
        refresh()

def refresh():
    from ui_bridge.ui_controller import UIController

```

```

playboard = UIController.state.game.playboard()
if not (UIController.field is None):
    UIController.field.destroy()
    UIController.field = tk.Frame(UIController.window)
    UIController.field.grid(column=1, row=3, columnspan=3)
    draw_field(UIController.field, UIController.state.game.level, playboard,
               move_handler)
    bombs = UIController.state.game.bomb_stats()
    UIController.sign.config(text=' ' + str(bombs) + ' ' + ('bomb' if bombs
== 1 else 'bombs' + ' left'))
    menu_state(1 if UIController.state.game.state == 1 else 0)

def menu_state(state):
    from ui_bridge.ui_controller import UIController

    if state == 1:
        UIController.menu['level'].entryconfigure(0, state='disabled')
        UIController.menu['level'].entryconfigure(1, state='disabled')
        UIController.menu['level'].entryconfigure(2, state='disabled')

        UIController.menu['game'].entryconfigure(0, state='normal')
        UIController.menu['game'].entryconfigure(1, state='normal')
        UIController.menu['game'].entryconfigure(2, state='disabled')

        UIController.menu['users'].entryconfigure(0, state='disabled')
    else:
        UIController.menu['level'].entryconfigure(0, state='normal')
        UIController.menu['level'].entryconfigure(1, state='normal')
        UIController.menu['level'].entryconfigure(2, state='normal')

        UIController.menu['game'].entryconfigure(0, state='normal')
        UIController.menu['game'].entryconfigure(1, state='normal')
        UIController.menu['game'].entryconfigure(2, state='normal')

        UIController.menu['users'].entryconfigure(0, state='normal')

def load_flow():
    from ui_bridge.ui_controller import UIController

    f = prepare_personal_folder()
    lst = []

    for fi in f.iterdir():
        lst.append({'file': fi, 'date': fi.stat().st_mtime})

    lst = sorted(lst, key=lambda i: i['date'], reverse=True)
    rm = lst[10:]
    lst = lst[:10]

    for fi in rm:
        fi['file'].unlink()

    saves = []

    for fi in lst:
        saves.append(time.strftime("%b %d %Y %H:%M:%S",
time.localtime(fi['date'])))

    choices_var = tk.StringVar(value=saves)

    def restart():
        if UIController.state is not None and UIController.state.game.state
!= 2:

```

```

        UIController.state.game.start()
        refresh()
    w.destroy()

w = tk.Toplevel(UIController.window)
w.resizable(False, False)
w.protocol("WM_DELETE_WINDOW", restart)
ls = tk.Listbox(w, height=10, listvariable=choices_var)
ls.pack()

tk.Button(w, text='OK', command=lambda:
load_file(lst[ls.curselection()[0]]['file'])).pack()

def load_file(path: pathlib.Path):
    from ui_bridge.ui_controller import UIController
    UIController.saver = DataFile(path)
    UIController.state = UIController.saver.load()
    UIController.state.game.game_done = process_leaderboard
    refresh()

def load_ui():
    if Session.get_user() is None:
        return login_ui(load_flow)
    load_flow()

# login.py

import tkinter as tk
from tkinter import messagebox
from user_space.login import process_login, sign_up

def login_ui(handler):
    from ui_bridge.ui_controller import UIController

    w = tk.Toplevel(UIController.window)
    w.title('Login')

    tk.Label(w, text='Username: ').pack()
    username = tk.StringVar()
    tk.Entry(w, textvariable=username).pack()

    tk.Label(w, text='\nPassword: ').pack()
    password = tk.StringVar()
    tk.Entry(w, textvariable=password, show="*").pack()

    def success():
        handler()
        w.destroy()

    def error():
        messagebox.showerror(title='Error', message='Incorrect credentials')
        w.lift()

    def go_on():
        process_login(username.get(), password.get(), success, error)

    tk.Button(w, text='OK', command=go_on).pack()
    tk.Button(w, text='Sign Up', command=lambda: sign_up_ui(lambda:
None)).pack()

```

```

def sign_up_ui(handler):
    from ui_bridge.ui_controller import UIController

    w = tk.Toplevel(UIController.window)
    w.title('Sign Up')

    tk.Label(w, text='New username: ').pack()
    username = tk.StringVar()
    tk.Entry(w, textvariable=username).pack()

    tk.Label(w, text='\nNew password: ').pack()
    password = tk.StringVar()
    tk.Entry(w, textvariable=password, show="*").pack()

    tk.Label(w, text='\nRepeat password: ').pack()
    second_password = tk.StringVar()
    tk.Entry(w, textvariable=second_password, show="*").pack()

    def success():
        handler()
        w.destroy()

    def error():
        messagebox.showerror(title='Error', message='Incorrect credentials')
        w.lift()

    def go_on():
        if len(username.get()) == 0 or len(password.get()) == 0:
            messagebox.showerror(title='Error', message='Credentials can\'t
be empty')
            w.lift()
            return
        if second_password.get() != password.get():
            messagebox.showerror(title='Error', message='Passwords don\'t
match')
            w.lift()
            return
        sign_up(username.get(), password.get(), success, error)

    tk.Button(w, text='OK', command=go_on).pack()

# resource_path.py

import sys
import os

def resource_path(relative_path):
    return relative_path

# save.py

from state_keeper.generate_new import gen_name
from ui_bridge.login import login_ui
from user_space.data_file import UserDataFile
from user_space.session import Session
from user_space.user_dir import prepare_personal_folder

def save_flow():
    from ui_bridge.ui_controller import UIController

    if UIController.saver is None:

```

Додаток А

```

        UIController.saver = UserDataFile(gen_name(UIController.state))
        UIController.state.game.on_change = UIController.saver.changed

    if UIController.state.game.state == 1:
        UIController.state.game.stop()

    UIController.saver.save(UIController.state)
    UIController.level.set(UIController.level.get())
    UIController.saver = None

    f = prepare_personal_folder()
    lst = []

    for fi in f.iterdir():
        lst.append({'file': fi, 'date': fi.stat().st_mtime})

    lst = sorted(lst, key=lambda i: i['date'], reverse=True)[10:]

    for fi in lst:
        fi['file'].unlink()

def save():
    if Session.get_user() is None:
        login_ui(save_flow)
        return
    save_flow()

# ui_controller.py

import tkinter as tk

from state_keeper.state import GameState
from ui_bridge.resource_path import resource_path
from user_space.data_file import DataFile

class UIController:
    level = tk.IntVar()
    state: GameState
    field: tk.Frame
    sign: tk.Label
    window: tk.Tk
    saver: DataFile = None
    menu = {}
    assets = [
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/0.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/1.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/2.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/3.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/4.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/5.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/6.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/7.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/8.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/mine.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/flag.png')),
        tk.PhotoImage(file=resource_path('./ui_bridge/assets/question.png')),
    ]

# data_file.py

import base64

```



```

import hashlib
import json
import pathlib

from state_keeper.state import GameState
from topaz.level import LevelDescriptor
from user_space.encrypt import pass_shift
from user_space.file_proxy import encode_file, decode_file
from user_space.session import Session
from user_space.user_dir import prepare_personal_folder

class DataFile:
    def __init__(self, file: pathlib.Path):
        self.file = file
        self.__integrity = file.exists()

    def is_saved(self):
        return self.__integrity

    def save(self, data):
        e = DataFile.__encode(data)
        s = json.dumps(e)
        p = hashlib.md5(Session.get_user().password.encode(encoding='utf-8')).digest().hex()

        r = pass_shift(p, s)

        encode_file(r, self.file)

        self.__integrity = True
        return

    def load(self):
        f = decode_file(self.file)

        p = hashlib.md5(Session.get_user().password.encode(encoding='utf-8')).digest().hex()

        r = pass_shift(p, f, decrypt=True)

        self.__integrity = True
        return DataFile.__decode(json.loads(r))

    def changed(self, state_switch):
        self.__integrity = False

    @staticmethod
    def __encode(state: GameState):
        if state.game.state == 1:
            raise PermissionError("You can't save a running game")

        return {
            's': state.salt,
            'x': state.first_move_x,
            'y': state.first_move_y,
            'l': state.game.level.level,
            't': state.game.timer(),
            'p': state.game.state,
            'v': state.game.visuals,
        }

    @staticmethod
    def __decode(o):
        from state_keeper.generate_new import generate_new

```

```

state = generate_new(
    level=LevelDescriptor(o['l']),
    first_move_x=o['x'],
    first_move_y=o['y'],
    seed=o['s'],
    visuals=o['v']
)

state.game.duration = o['t']
state.game.state = o['p']

return state

class UserDataFile(DataFile):
    def __init__(self, name: str, user: str = None):
        f = prepare_personal_folder(user)
        super().__init__(f / name)

# encrypt.py

def pass_shift(p, string, decrypt=False):
    c = []
    for i in range(len(string)):
        k = p[i % len(p)]
        if decrypt:
            e = chr((ord(string[i]) - ord(k) + 256) % 256)
        else:
            e = chr(ord(string[i]) + ord(k) % 256)
        c.append(e)
    r = ''.join(c)
    return r

# file_proxy.py

def encode_file(data, file, encode=True):
    if encode:
        b = bytearray(data.encode(encoding="utf-8"))
    else:
        b = data

    for i in range(len(b)):
        b[i] ^= 0xff
    open(file, 'wb').write(b)

def decode_file(file, decode=True, fallback='{}'):
    try:
        b = bytearray(open(file, 'rb').read())
    except FileNotFoundError:
        return fallback
    for i in range(len(b)):
        b[i] ^= 0xff
    if decode:
        return b.decode(encoding="utf-8")
    return b

# leaderboard.py

from tkinter import messagebox
import tkinter as tk

```

```

from topaz.entities.tiles import BombTile
from ui_bridge.login import login_ui
from user_space.local_leaderboard import LocalLeaderboard
from user_space.session import Session

def leaderboard_flow():
    from ui_bridge.ui_controller import UIController
    LocalLeaderboard.update_high_score(
        UIController.state.game.level,
        UIController.state.game.timer()
    )
    messagebox.showinfo('Leaderboard', 'Now your result is featured at the
leaderboard')

def process_leaderboard():
    from ui_bridge.ui_controller import UIController

    if UIController.state is None:
        return

    pb = UIController.state.game.playboard()
    for row in range(UIController.state.game.level.level_sizes()[1]):
        for col in range(UIController.state.game.level.level_sizes()[0]):
            if isinstance(pb[row][col], BombTile) and pb[row][col].boomed:
                return

    ch = LocalLeaderboard.check_high_score(UIController.state.game.level)[0]
    if not (ch is True) and LocalLeaderboard.rate(
        UIController.state.game.level,
        UIController.state.game.timer()
    ) <= ch:
        return

    if not messagebox.askyesno(
        'New High Score',
        'New High Score! Would you like to feature yourself at the
leaderboard?'
    ):
        return

    if Session.get_user() is None:
        login_ui(leaderboard_flow)
    else:
        leaderboard_flow()

def leaders_window():
    from ui_bridge.ui_controller import UIController

    w = tk.Toplevel(UIController.window)
    w.title('Leaderboard')

    h = LocalLeaderboard.get_high_score()

    for e in h:
        tk.Label(
            w, text=e['user'] + ' | ' + str(e['level'] + 1) + ' lvl |
' + str(int(e['time'] // 60)).zfill(
                2) + ':' + str(int(e['time'] % 60)).zfill(2), width=50,
font=16
        ).pack()

    if len(h) == 0:

```

```

tk.Label(w, text='No records yet', width=50).pack()

# local_leaderboard.py

from state_keeper.state import GameState
from topaz.level import LevelDescriptor
from user_space.session import Session
from user_space.user_file import load_leaderboard_file, save_leaderboard_file

class LocalLeaderboard:
    @staticmethod
    def get_high_score():
        f = load_leaderboard_file() # type: dict
        r = []

        for element in f['top']:
            lvl = LevelDescriptor(element['level'])
            r.append({
                'user': element['user'],
                'level': element['level'],
                'time': element['time'],
                'score': LocalLeaderboard.rate(lvl, element['time'])
            })
        r = sorted(r, key=lambda i: i['score'])[0:20]
        return r

    @staticmethod
    def check_high_score(level):
        c = LocalLeaderboard.get_high_score()
        return True, c
        cr = list(filter(lambda i: i['level'] == level, c))
        if len(cr) == 0:
            return True, c
        return cr[-1], c

    @staticmethod
    def rate(level, time):
        return (time / (level.level_sizes()[0] * level.level_sizes()[1]) /
                (level.level + 1)) // 1

    @staticmethod
    def update_high_score(level, time):
        c, m = LocalLeaderboard.check_high_score(level)
        rate = LocalLeaderboard.rate(level, time)
        if not isinstance(c, bool) and c >= rate:
            return

        m.insert(0, {
            'user': Session.get_user().username,
            'level': level.level,
            'time': time,
            'score': rate,
        })

        save_leaderboard_file({'top': m})

# login.py

import hashlib
import uuid

from user_space.session import Session

```

```

from user_space.user import User
from user_space.user_file import load_user_file, save_user_file

def process_login(username, password, handler, error):
    j = load_user_file()
    if not (username in j):
        error()
        return
    data = j[username]
    if not check_hash(data['pass'], password):
        error()
        return
    Session.apply_user(User(username, password))
    handler()

def sign_up(username, password, handler, error):
    j = load_user_file()
    if username in j:
        error()
        return
    j[username] = {'pass': hash_pass(password)}
    save_user_file(j)
    process_login(username, password, handler, error)

def hash_pass(password, salt=None):
    if salt is None:
        salt = uuid.uuid4().hex
    return hashlib.sha512((password + salt).encode(encoding='utf-8')).hexdigest() + '.' + salt

def check_hash(hash_str, password):
    hash_arr = hash_str.split('.')
    password = hash_pass(password, hash_arr[1])
    return hash_str == password

# session.py

from user_space.user import User

class Session:
    __user: User = None
    handlers = []

    @staticmethod
    def apply_user(user):
        Session.__user = user
        for handler in Session.handlers:
            handler()

    @staticmethod
    def log_out():
        return

    @staticmethod
    def get_user():
        return Session.__user

# user.py

```

```

class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password

# user_dir.py

import os
import sys
import pathlib

from user_space.session import Session

def get_user_data_dir() -> pathlib.Path:
    if sys.platform == "win32":
        return pathlib.Path(os.path.expandvars("%LOCALAPPDATA%"))
    elif sys.platform == "linux":
        return pathlib.Path(os.path.expandvars("$XDG_DATA_HOME"))
    elif sys.platform == "darwin":
        return pathlib.Path.home() / "Library/Application Support"

def prepare_user_data_dir() -> pathlib.Path:
    d = get_user_data_dir() / "topaz_minesweeper"
    try:
        d.mkdir(parents=True)
    except FileExistsError:
        pass

    return d

def prepare_personal_folder(user: str = None):
    if user is None:
        user = Session.get_user().username

    d = prepare_user_data_dir() / user
    try:
        d.mkdir(parents=True)
    except FileExistsError:
        pass

    return d

# user_file.py

import json

from user_space.file_proxy import decode_file, encode_file
from user_space.user_dir import prepare_user_data_dir

def load_user_file() -> dict:
    d = prepare_user_data_dir()
    return json.loads(decode_file(d / 'users.dat'))

def save_user_file(data):
    d = prepare_user_data_dir() / 'users.dat'
    encode_file(json.dumps(data), d)

```

```
def load_leaderboard_file() -> dict:
    d = prepare_user_data_dir()
    return json.loads(decode_file(d / 'leaders.dat', fallback='{"top": []}'))

def save_leaderboard_file(data):
    d = prepare_user_data_dir() / 'leaders.dat'
    encode_file(json.dumps(data), d)
```

ДОДАТОК В



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО

ЗАХИСТ

домашньої контрольної роботи

З навчальної дисципліни «Основи програмування»
на тему:

РОЗРОБКА ГРИ «САПЕР»

Доповідач:
Студент 1-го курсу,
групи БС-02
Ділонг Сергій Михайлович
Email: me@somnemo.com

Київ – 2020

ЗМІСТ



ВСТУП

Задача. Розробити власну версію класичної комп'ютерної гри «Сапер», використовуючи мову програмування Python, розширити стандартну функціональність гри, додавши можливість зберігати гру у персональному акаунті, та додавши таблицю рекордів. Запобігти втручанню користувача у файли зберігання.

Сапер є однією з найстаріших комп'ютерних ігор, що було створено близько у 60-х роках минулого століття. Її популярність розвилася після включення однієї з варіацій гри до *Microsoft Windows Entertainment Pack*, а згодом й до увійшла до *Windows*.



3

АНАЛІЗ МЕТОДІВ РОЗРОБКИ



Модель, застосована у даній роботі, називається MVC (Model – View – Controller), що перекладається як Модель – Вигляд – Контролер. Окрім гнучкості, дана модель спрощує розуміння коду та дає змогу повторного використання компонентів.

Модель – це ядро програми, яке надає та приймає дані, змінює свій стан, для усього іншого середовища працює як «чорний ящик».

Вигляд – комплекс, що відповідає за відображення стану моделі до користувача, найчастіше являє собою бібліотеку для GUI

Контролер – передавач дій користувача до моделі, сповіщувач про необхідність змін, є посередником між користувачем та моделлю.

4

КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

Види вказівних пристроїв

	Точний	Не точний
Підтримує hover	Миша / Активне перо	Джойстик / Контролер
Без hover	Стилус	Сенсорний дисплей



Radio-перемикач

Унікальні для ПК елементи інтерфейсу

- ✓ Зручні
- ✓ Інтуїтивні



Каскадне меню

Спливаюче вікно

ЗАХИСТ ВІД ВТРУЧАННЯ

Зберігання даних програми у %APPDATA%

Кодування використовуючи операцію XOR

Хешування

Алгоритмізація генерації мін зберігаючи лише seed

Шифрування

Потребує певного рівню знань для втручання

ВИСНОВКИ

Було досліджено історію гри «Сапер» та причини її популяризації, занепаду, способи розробки гри-головоломки «Сапер» з застосуванням мови програмування Python та середовища GUI Tkinter.

Отримана гра має можливість створювати користувачів, зберігати прогрес обмежену кількість разів, рахувати витрачений час, додавати результат на загальний список рекордів, застосовує різні практики для передбачення модифікації та переглядання файлів збереження.

Було проаналізовано принцип розробки MVC для розділення програми на компоненти, переваги ООП стилю програмування для розробки ігрової логіки та функціонального стилю для роботи з Tkinter, було пояснено, як різні компоненти програми називаються у рамках MVC.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Copyright Industry Perspectives: The Pivotal Role of TPMs in the Evolution of the Video Game Industry [Стаття] / авт. Genetski Christian, and Christian Troncoso // JL & Arts, 38, 359. - 2014 p..

Menus [Онлайновий] / авт. Roseman Mark // TkDocs. - 2007-2020 p.. - <https://tkdocs.com/tutorial/menus.html>.

Python. Работа с изображениями в tkinter. [Онлайновий] / авт. Шуравин Александр // База знаний Programming Store. - Programming Store, 25 сепень 2020 p.. - <https://wiki.programstore.ru/python-rabota-s-izobrazheniyami-v-tkinter/>.

Reenskaug/MVC [Онлайновий] / авт. H. Trygve M.. - XEROX PARC , 1978 p.. - <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>.

The Games of Windows [Онлайновий] / авт. Maher Jimmy // The Digital Antiquarian. - 2018 p.. - <https://www.filfre.net/2018/08/the-games-of-windows/>.

The most successful game ever: a history of Minesweeper [Онлайновий] / авт. Cobbett Richard // techradar. - 2009 p.. - <https://www.techradar.com/news/gaming/the-most-successful-game-ever-a-history-of-minesweeper-596504>.

Video Games as a Tool to Train Cognitive Skills [Звіт] / авт. Daphne Bavelier Ph.D.. - Rochester, NY : University of Rochester, 2006.

В чём разница между использованием MVC и MVP [Онлайновий] / авт. Попов Алексей // Habr. - 2013 p.. - <https://habr.com/ru/post/171925/>.