Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.75;004.042

*На правах рукопису*

# ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Розробка ігрового додатку (японський пазл) мовою Ruby ”

Спеціальність – 121 “Інженерія програмного забезпечення”

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

БР.ІПЗ – 92.00.00.000

Студент
ІПЗ-41 _____/Андрій ОРАНСЬКИЙ/

Науковий керівник
д.т.н., проф. _____ /Олексій БИЧКОВ/

Консультант
з питань нормоконтролю
фахівець _____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри
д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

Рішенням Екзаменаційної комісії
випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:
Завідувач кафедри програмних систем і технологій
_____ (О.С.Бичков )
„___”_____ 20___р.

**ЗАВДАННЯ**
**НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ**
**СТУДЕНТУ**

Оранському Андрію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи ”Розробка ігрового додатку (японський пазл) мовою Ruby” затверджена наказом вищого навчального закладу від „__”_____20__р. №_____

2. Строк здачі студентом закінченої роботи _____31.05.2021_____

3. Вихідні дані до роботи
_____За основу була взята аналогічна гра, розроблена протягом проходження подвійного дипломування в університеті міста Ле Ман. Основним напрямком розробки був власне додаток, комфортний графічний інтерфейс користувача, а також наведення справки та допомоги в процесі гри._____
_____

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)
Актуальність даного типу відео-ігор, опис концепції, архітектура програмного забезпечення, принципи та інструменти розробки, опис життєвого циклу та інтегральне тестування продукту_____
_____

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)
Пояснювальна записка містить увесь необхідний графічний матеріал, як-от діаграми, графічний інтерфейс аналогів, представлення фінальної версії продукту_____
_____

**Форма завдання на випускну кваліфікаційну бакалаврську роботу** (на звороті першого аркуша)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 03.01.2021

Керівник _____ _____
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ _____
(підпис) (розшифровка підпису)

# КАЛЕНДАРНИЙ ПЛАН

| Номер і назва етапів бакалаврської роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|
| 1. Аналіз технік, отриманих у Франції | 03.01.2021 – 20.01.2021 | |
| 2. Проектування ПЗ | 21.01.2021 – 01.03.2021 | |
| 3. Аналіз можливих помилок, корективи | 02.03.2021 – 15.03.2021 | |
| 4. Розробка ПЗ | 16.03.2021 – 01.05.2021 | |
| 5. Комплексне тестування на усіх платформах | 05.05.2021 – 21.05.2021 | |
| 6. Виправлення багів | 21.05.2021 – 31.05.2021 | |
| | | |
| | | |

Студент – бакалавр _____
(підпис) (розшифровка підпису)

Керівник роботи _____
(підпис) (розшифровка підпису)

# АНОТАЦІЯ

**Структура та обсяг роботи.** Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 15 рисунків, 3 таблиці, 15 джерел.

Дипломний проект присвячений розробці гри - японської головоломки із повноцінним ігровим функціоналом та графічним інтерфейсом.

У загальних положеннях наведено опис предметного середовища, огляд наявних аналогів, постановка задачі, а також прототип майбутнього додатку.

У розділі проектування розглянено архітектуру ПЗ, середовище програмування Ruby та його бібліотеку GTK+, а також модель і засоби розробки та детальний опис класів. Наведені діаграми послідовностей, класів та прецедентів.

У розділі з функціоналу задокументовані усі вимоги, яким відповідає готовий продукт ПЗ. Детально описаний графічний інтерфейс . Наведена діаграма активностей.

У розділі тестування наведено кілька тестів, реалізованих користувачами. Проведено інтегральне тестування на всіх платформах ПК.

Мова додатку: французька. Пілотний проект розроблювався в рамках проходження стажування в Університеті міста Ле Ман, Франція. Цільова аудиторія продукту – франкомовні діті та підлітки.

RUBY, ДОДАТОК, КОМП’ЮТЕРНА ГРА, GTK, СЕРЕДОВИЩЕ РОЗРОБКИ, КРОСПЛАТФОРМНІСТЬ, КОМПІЛЯЦІЯ, АРКАДА, SQLITE, АРХІТЕКТУРА, ПРОГРАМНИЙ ПРОДУКТ, ЛІЦЕНЗІЯ

# ABSTRACT

**Structure and scope of work.** The explanatory note of the diploma project consists of four sections, contains 15 figures, 3 tables, 15 sources.

The diploma project is dedicated to the development of the game - a Japanese puzzle with full game functionality and graphical interface.

The general provisions provide a description of the subject environment, an overview of existing analogues, problem statement, as well as a prototype of a future application.

The design section discusses the software architecture, the Ruby programming environment and its GTK + library, as well as the model and development tools and a detailed description of the classes. Diagrams of sequences, classes and precedents are given.

The functional section documents all the requirements that the finished software product meets. The graphical interface is described in detail. The diagram of activities is given.

The testing section lists several user-implemented tests. Integrated testing was performed on all PC platforms.

Application language: French. The pilot project was developed as part of an internship at the University of Le Mans, France. The target audience of the product is French-speaking children and teenagers.

RUBY, APPLICATION, COMPUTER VIDEO GAME, GTK, DEVELOPMENT ENVIRONMENT, CROSS-PLATFORM, COMPILATION, ARCADE, SQLITE, ARCHITECTURE, SOFTWARE PRODUCT, LICENCE

# RÉSUMÉ

**Structure et étendue des travaux**. La notice explicative du projet de diplôme se compose de quatre sections, contient 15 figures, 3 tableaux, 15 sources.

Le projet de diplôme est dédié au développement du jeu vidéo - un puzzle japonais avec des fonctionnalités de jeu complètes et une interface graphique utilisateur.

Les dispositions générales fournissent une description de l'environnement du sujet, un aperçu des analogues existants, un énoncé du problème, ainsi qu'un prototype d'une future application.

La section de conception aborde l'architecture logicielle, l'environnement de programmation Ruby et sa bibliothèque GTK+, ainsi que le modèle et les outils de développement et une description détaillée des classes. Des diagrammes de séquences, de classes et de précédents sont représentées.

La section fonctionnelle documente toutes les exigences auxquelles le produit logiciel fini répond. L'interface graphique est décrite en détail. Le schéma des activités est donné.

La section de test répertorie plusieurs tests implémentés par l'utilisateur. Des tests intégrés ont été effectués sur toutes les plates-formes PC.

Langue d'application: français. Le pilote a été développé dans le cadre d'un stage à l'Université du Mans, France. Le cible-marché du produit est les enfants et les adolescents francophones.

RUBY, APPLICATION, JEU VIDÉO, GTK, ENVIRONNEMENT DE DEVELOPPEMENT, CROSSPLATFORM, COMPILATION, ARCADE, SQLITE, ARCHITECTURE, PRODUIT LOGICIEL, LICENCE

# ЗМІСТ

**ВСТУП** ..... **9**

**РОЗДІЛ 1** ..... **13**

**1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ** ..... **13**

1.1 Принципи ігрового процесу ..... 13

1.2 Огляд аналогів ..... 14

1.3 Пілотний проект ..... 16

1.4 Діаграма прецедентів ..... 17

1.5 Зовнішні специфікації ..... 18

**РОЗДІЛ 2** ..... **21**

**2 ПРОЕКТУВАННЯ** ..... **21**

2.1 Архітектура програмного забезпечення ..... 21

*2.1.1 Мова програмування Ruby* ..... *21*

*2.1.2 Модель розробки ПЗ* ..... *24*

*2.1.3 Діаграма класів (Class diagram) та її опис* ..... *26*

*2.1.4 Діаграма послідовностей (Sequence diagram)* ..... *31*

**РОЗДІЛ 3** ..... **33**

**3 ФУНКЦІОНАЛЬНІСТЬ ПЗ** ..... **33**

3.1 Діаграма активностей (Activity diagram) ..... 33

3.2 Огляд функціоналу ПЗ ..... 33

*3.2.1 Документація SAIF BF* ..... *33*

*3.2.2 Графічний інтерфейс користувача* ..... *39*

**РОЗДІЛ 4** ..... **43**

**4 ТЕСТУВАННЯ** ..... **43**

4.1 Функціональне тестування ..... 43

*4.1.1 Тестування на Linux* ..... *43*

*4.1.2 Тестування на Windows та Mac OS* ..... *44*

4.2 Юзабілити-тестування ..... 45

**5 ВИСНОВКИ** ..... **46**

**6 ВИКОРИСТАНІ ДЖЕРЕЛА** ..... **48**

# ВСТУП

## Актуальність

Головоломка (англ. Puzzle) - назва жанру комп'ютерних ігор, метою яких є вирішення логічних завдань, що вимагають від гравця задіяння логіки, стратегії і інтуїції або в інших випадках деякого наявності удачі.

### Історія розвитку

Предтечею жанру були настільні, графічні і механічні головоломки - від кросвордів до кубика Рубіка, який побачив світ у середині 1970-х. Ці головоломки вимагали від гравця логіки і спритності в рішенні, які також стали відігравати важливу роль в прототипах жанру як Q*bert і Boulder Dash. Еталоном жанру стала гра Тетріс, що з'явилася в 1985 році і поєднувала в собі простий і захоплюючий ігровий процес.

На початку 1990-х років такі ігри як Lemmings і The Lost Vikings оживили жанр головоломок. Поява тривимірної графіки посприяло розвитку головоломок на ігрових приставках. Щодо дешеві для виробництва гри знайшли свою нішу на портативних ігрових системах.

У 2000 році гри Pikmin, Meteos, Polarium, LocoRoco і Lumines знову відродили принципи жанру. В Японії серія ігор Brain Training (розвиваючі логічні ігри) - один з найбільш великих успіхів індустрії комп'ютерних ігор за 2005 рік.

**Різновиди**

У деяких головоломках гравцеві даються випадкові блоки або шматочки, які потрібно зібрати в певній послідовності і формі, наприклад, до таких ігор належать Тетріс, Klax і Lumines. З них Тетріс вважається однією з найважливіших відеоігор цього жанру, що породила безліч продовжень, варіацій і клонів за участю «падаючих блоків».

Інші ігри, такі як Bomberman і The Incredible Machine, вдають із себе площа з окремих деталей, де рішення задачі вимагає від гравця досягнення місця призначення, шляхом послідовних дій, обмежених часом. Деякі з таких головоломок мають режим гри, зворотний принципом Тетріс і подібним. Так, наприклад, в Tetrisphere і Tetris Attack, гравець повинен очистити площу від певних деталей за обмежену кількість ходів.

Також заслуговує на увагу створена в 1980 році Хіроюкі Імабаясі гра Сокобан, в якій гравець пересуває ящики по лабіринту з метою поставити їх на задані кінцеві позиції. 'Багато квести і екшен-адвенчури містять в собі елементи головоломки. Наприклад, Portal, Resident Evil, Silent Hill, Little Big Planet і серія ігор The Legend of Zelda.

Ігри-головоломки часто настільки труднопонімаеми, що термін іноді використовується для загального позначення ігор з незвичайним, не піддається опису ігровим процесом, як, наприклад, в грі Every Extend Extra.

Зазвичай головоломки не викликають складнощів для поширення і адаптації, їх можна зустріти на таких платформах як аркадні автомати, ігрові приставки, кишенькові комп'ютери і мобільні телефони.

Таким чином, актуальною є задача розробки ігрового програми в

жанрі аркада.

## Мета і задачі

Метою даної роботи є розробка програми гри - японської головоломки Hashi Wokakero - через використання мови програмування Ruby, що включає в себе графічний інтерфейс користувача (бібліотека GTK+), різні рівні складності відображення результату і різні підказки в процесі гри.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. виконати аналіз вимог і розробити зовнішні специфікації;
2. виконати проектування ігрового програми;
3. реалізувати ігрове додаток;
4. протестувати розроблене ігрове додаток.

## Структура роботи

Робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Обсяг роботи становить 40 сторінок, обсяг бібліографії – 15 джерел.

У першому розділі роботи, «Аналіз предметної області», проведено огляд аналогів, розроблений ескізний проект, визначені вимоги до гри і зовнішні специфікації.

У другому розділі роботи, «Проектування», виконано проектування ігрового додатку та реалізовано його.

Третя глава роботи, «Функціонал», описує документацію усі вимог, детальний опис графичного інтерфейсу та взаємодії.

Четвертий розділ, «Тестування», присвячений тестуванню ігрового програми.

У висновку описані основні результати, отримані при виконанні дипломної роботи.

## Апробація результатів випускної кваліфікаційної бакалаврської роботи

Даний ігровий додаток був використаний в рамках програми французького уряду у школі імені Жана-Люка Лайе у місті Ле Ман у квітні 2021 року. Учні 2го класу, в рамках уроку інформатики, мали можливість протестувати ігру. Відгуки викладачів та учнів, взявших участь у апробації, були розподілені наступним чином:

- 74% вважають додаток легким до використання, а гру – цікавою.
- 16% мали складності при запуску гри, але знайшли її досить цікавою.
- 10% дали оцінку “задовільно”.

# РОЗДІЛ 1
# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Принципи ігрового процесу

Hashi Wokakero (Мости) - це японська логічна головоломка, розроблена компанією Nikoli і опублікована в 1990 році (рис. 1.1) . Завдання гравця полягає в тому, щоб з'єднати лініями острови, і при цьому число мостів повинно відповідати зазначеному на острові числу.

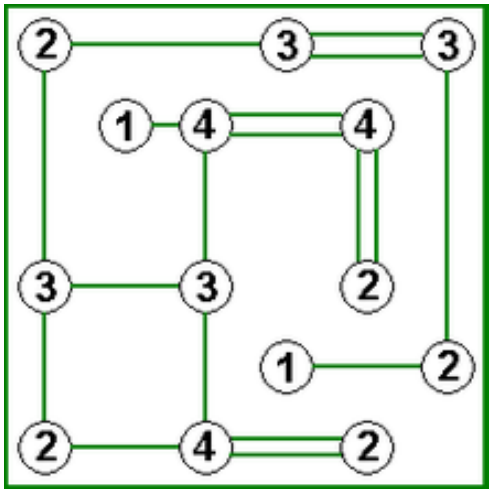


Рис 1.1. Стандартний вид пазлу Hashi

Головоломка також відома під іншими назвами, серед яких Hashiwokakero, Bridges, Chopsticks, Ai-Ki-Ai.

Метою гри є зв’язати конфігурацію даних островів, дотримуючись кількості мостів, що відходять з кожного острова; це число вказано на кожному острові. Хаші грають на прямокутній сітці, є кола, що містять число, від 1 до 8 включно.

## 1.2 Огляд аналогів

Прямими аналогами гри Hashi (мости) є інші пазли, видані японським видавцем Nikoli. Найбільш відомим є сучасна версія Sudoku, всесвітньо відома математична головоломка.

Для персональних комп'ютерів існує досить багато варіантів цієї гри, що випускаються різними компаніями-розробниками для різних платформ, наприклад,

• Sudoku (з набору ігор GNOME Games) або KSudoku (з набору ігор KDE Games) - для Linux-платформ (рис. 1.2). Програма написана на мові програмування Python. Рік випуску - 2006.
Розробник - Thomas Hinkle..

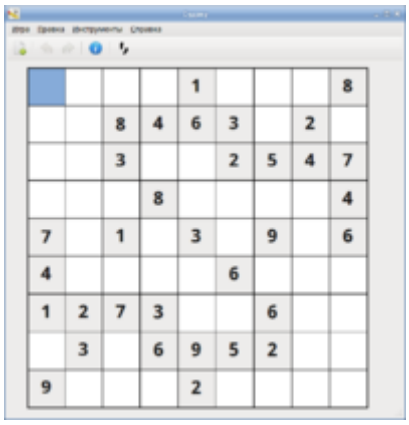

Рис 1.2. Фрагмент Sudoku

- Microsoft Sudoku - програма, створена корпорацією Microsoft (рис. 1.3). Сьогодні вона пройшла безліч взаємодій з боку розробників - корпорації Microsoft, з останньою поточною версією 2.4.4261, яка була офіційно випущена 2021-05-07. Як свідчення популярності

програми, лише в Apple App Store вона зібрала 1927 відгуків із середнім рейтингом користувачів 4,65489 із 5 можливих зірок.

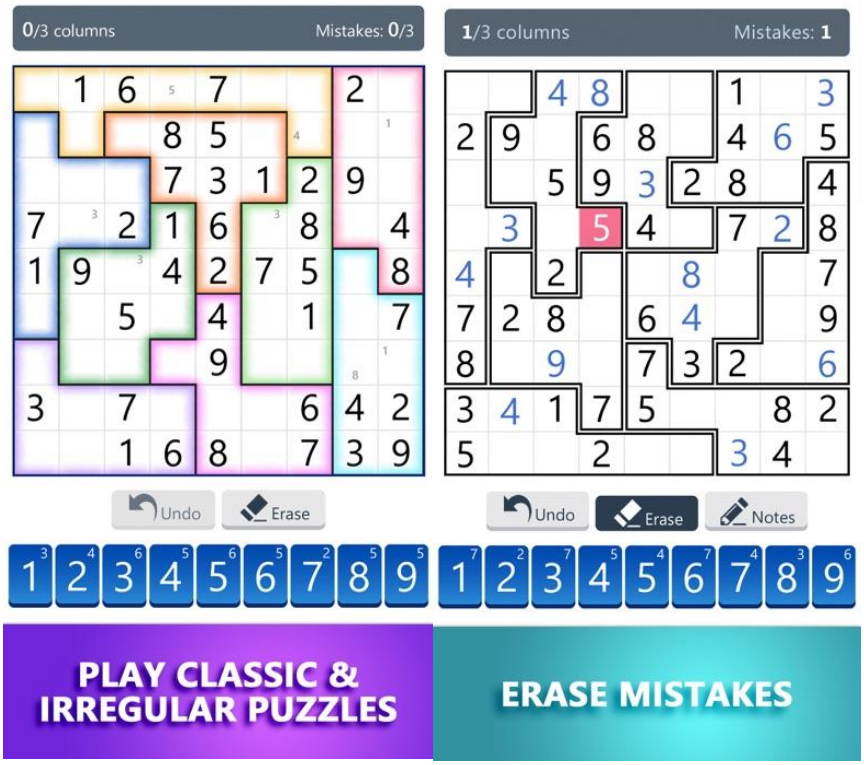


Рис 1.3. Ігрове вікно Microsoft Sudoku

- Pac-Man - комп'ютерна гра в жанрових аркадах, розроблена компанія Namco і вперше вийшла в 1980 році в Японії. Гра складається із 255-ти рівнів, розсчитаних на проходіння одним ігроком. Пак-ман (Pac-Man) - кругле жовте сутність лише з одним ртом. Задача ігор- рока - зібрати (з'їсти) всі білі точки на рівні, уникнувши стовцьковості з привидениями. Рівень закінчується, коли поєднуються всі точки. Гра має просте і понятне управління.

## 1.3 Пілотний проект

Ігровий додаток розрахований на його використання одним гравцем. Гра починається з меню, в якому необхідно вибрати режим гри, рівень складності, розмір поля, а також поле для введення імені гравця. Потім перед користувачем з'являється вікно безпосередньо гри, згідно із заданими параметрами в меню.

Необхідна функціональність: лічильник очок, таймер, збереження гри і чекпоінти, розділ з правилами гри і підказками.

Безпосередньо переможні умови відсутні. Мета гравця полягає в тому, щоб набрати якомога більше очок, за побудову правильних комбінацій мостів між числовими острівцями. Гра завершується, якщо головоломка була вирішена, і користувач натиснув кнопку «Перевірити».

Поставити гру на паузу неможливо. Гравець має лише можливість дострокового припинення гри шляхом її зупинки.

В ході розгадування кросвордів здійснюється підрахунок очок і часу, витраченого гравцем для вирішення кросворду.

Правильність побудови кросворду гравець може оцінити візуально шляхом простеження надійшов зображення. Однак передбачається також програмна перевірка.

По завершенню гри пропонується ввести ім'я гравця для вибудовування рейтингу.

Таким чином, завдання гравця полягає в тому, щоб знайти вірну комбінацію і чисто всіх мостів, і правильно розташувати їх між острівцями.

## 1.4 Діаграма прецедентів

В ході аналізу вимог до додатка була розроблена UML-діаграма варіантів використання (рис. 1.4).

Гра розрахована на використання одним гравцем.

У головному меню гравець може виконати наступні дії:

- *Вибрати режим гри:* у ігрового програми повинна бути настроюється складність генерації ігрового поля (нормальна і «авантюра»).

- *Вибрати розмір ігрового поля:* розміри 7x7, 10x10, 15x15, згідно зі стандартами творця HASHI.

- *Подивитися правила гри:* концептуальним рішенням є введення довідки. Таким чином, будь-який користувач, незалежно від знання правил виконання завдання, має можливість користуватися додатком.

- *Введення псевдоніма:* гравець має можливість свій результат з іншими гравцями в кінці гри.

- *Почати гру:* приступити до ігрового процесу, розпочати новий параметрований раунд.

**USE CASE DIAGRAM**

Рис 1.4. Діаграма прецедентів

## 1.5 Зовнішні специфікації

В результаті аналізу предметної області розроблений інтерфейс ігрового програми. Він складається з трьох основних екранів - головного меню, безпосередньо гри і кінця гри (рис. 1.5-1.7).

У головному меню знаходяться параметри гри, вікна правил гри і довідки, а також.

На ігровому екрані розташована сітка головоломки з островами, на кожному з яких зазначено номер - число мостів, а також таблиця з збереженими чекпоінти поточного раунду. Також присутні кнопки повернення в головне меню і перевірки результату.

На кінцевому екрані відображається час, витрачений на дозвіл головоломки, а також результати інших гравців. Також пропонується перейти в головне меню для вибору параметрів і нового раунду.

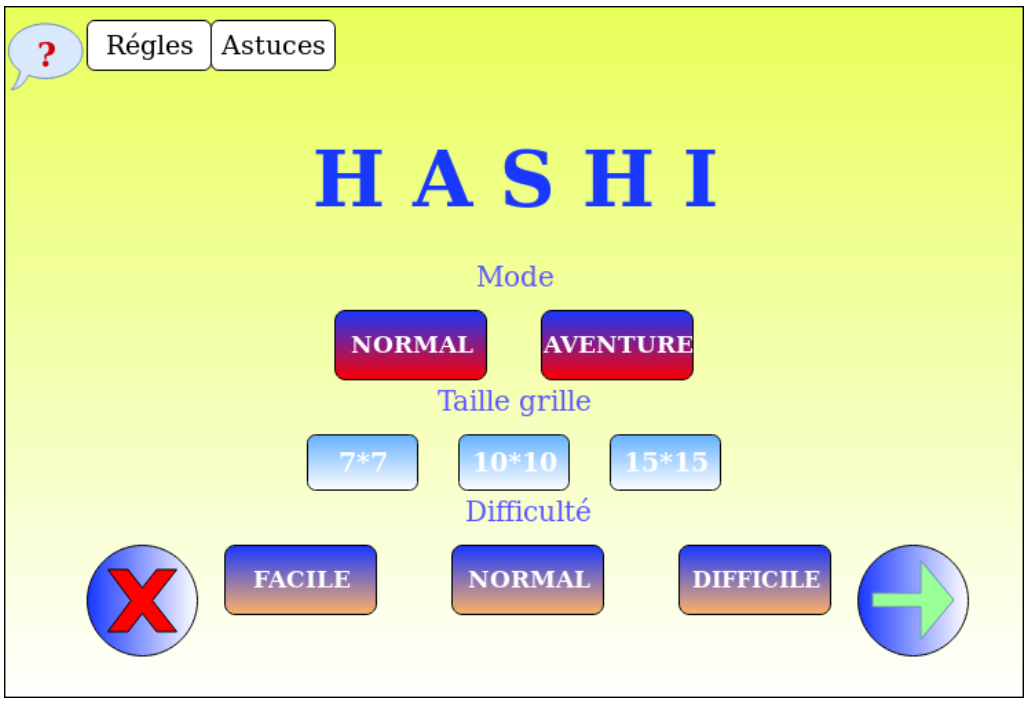


Рис. 1.5. Прототип головного меню

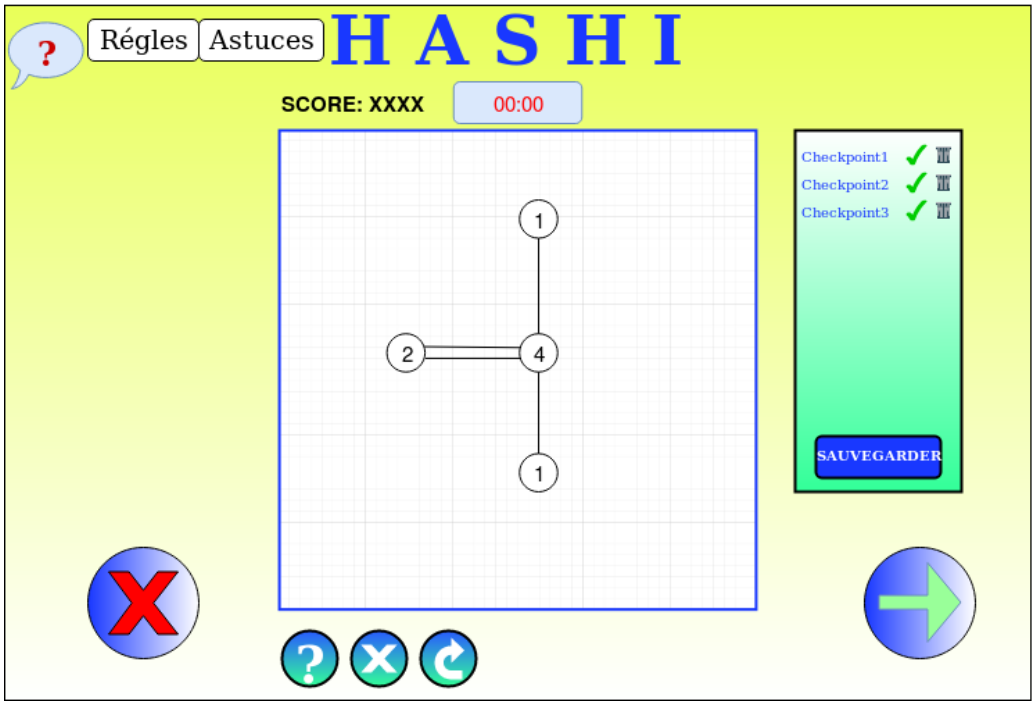


Рис 1.6. Прототип ігрового екрану.

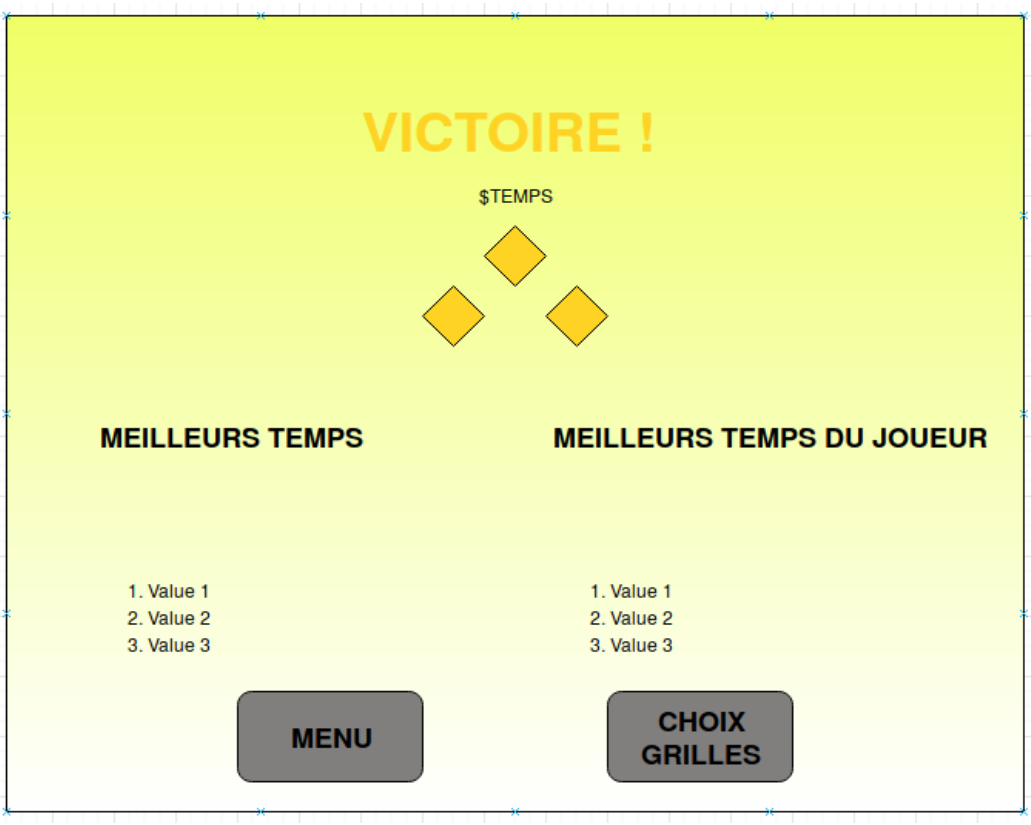


Рис 1.7. Прототип екрану кінця раунду

# РОЗДІЛ 2

# 2 ПРОЕКТУВАННЯ

## 2.1 Архітектура програмного забезпечення

Нам потрібно створити програмне забезпечення, яке дозволяє користувачеві використовувати ПЗ задля проходження раундів гри Hashi. Також необхідно впровадити правила гри, доступні гравцеві, а також допоміжні засоби, що полегшують роздільну здатність для користувача, якщо гра здається занадто складною.

### *2.1.1 Мова програмування Ruby*

Ruby (англ. Ruby - рубін, вимовляється [ 'ru: bɪ] - рубай) - динамічний, рефлективний, що інтерпретується високорівнева мова програмування. Мова має незалежної від операційної системи реалізацією багатопоточності, сильної динамічною типізацією, складальником сміття і багатьма іншими можливостями. За особливостями синтаксису він близький до мов Perl і Eiffel, по об'єктно-орієнтованого підходу - до Smalltalk. Також деякі риси мови взяті з Python, Lisp, Dylan і Клу.

Основне призначення Ruby - створення простих і в той же час зрозумілих програм для вирішення завдань, в яких час розробки, зрозумілість і простота важливіше, ніж швидкість роботи.
Принципи пристрою Ruby і програмування на ньому іноді виділяються в термін «Шлях Ruby» (англ. Ruby Way):

✓ *Мова для людини, а не для комп'ютера.*

Пріоритетом є зручність і мінімізація витрат праці програміста при розробці програми, звільнення програміста від рутинної роботи, яку комп'ютер може виконувати швидше і якісніше. На противагу машинно-орієнтованим мовам, які працюють швидше, Ruby - мова, найбільш близька до людини. Будь-яка робота з комп'ютером виконується людьми і для людей, і необхідно піклуватися в першу чергу про витрачених зусиль людей.

✓ *Просто, але не дуже просто.*

Спрощення є благом, але воно не повинно переходити певні межі, за якими перетворюється в самоціль і шкодить кінцевому результату.

✓ *Принцип найменшої несподіванки*

Програма повинна вести себе так, як очікує програміст. Але в контексті Ruby це означає найменше диво не при знайомстві з мовою, а при його ґрунтовному вивченні.

✓ *Ортогональность важлива, але природність важливіше.*

Надмірність допустима, якщо вона зручна. Ruby успадкував ідеологію мови програмування Perl в частині надання програмісту можливостей досягнення одного і того ж результату кількома різними способами.

✓ *Чи не бути рабом продуктивності.*

Якщо продуктивність для конкретного випадку неприпустимо низька, то це вимагає виправлення, а якщо заздалегідь відомо, що вона буде істотна - необхідно спочатку проектувати програму з урахуванням цього. Але

слід надавати перевагу елегантність ефективності в тих випадках, коли ефективність не дуже критична.

✓ *Не боятися змін під час виконання.*

Наявність в мові динамічних засобів, аж до самомодіфікації програми під час виконання, дають можливості, які дуже корисні для ефективного програмування. Зниження продуктивності, на яке припадає піти заради них, в більшості випадків цілком допустимо.

✓ *Слідувати простим і строгим правилам, але не доходити до педантизму.*

В Ruby ми бачимо не "педантичну несуперечливість", а суворе дотримання набору простих правил». Правила і угоди (зокрема, угоди про іменування, прийняті в мові) потрібні для того, щоб зробити розуміння програми простіше. Якщо відступ від правила в конкретному випадку логічно і зрозуміло - воно виправдано.

✓ *«Не потрібно з цим боротися».*

Ruby такий, яким він придуманий. Програмісту не слід чекати, що Ruby буде вести себе так само, як інший звичний йому мову. Програміст може слідувати своїм уявленням і звичкам, сформованим під впливом інших мов (див. «Принцип найменшої несподіванки» [⇨]), але якщо очікування виявляються невірні, це потрібно просто прийняти і використовувати.

Ruby - повністю об'єктно-орієнтована мова. У ньому всі дані є об'єктами, на відміну від багатьох інших мов, де існують примітивні типи. Кожна

функція - метод. Реалізується ідеологія «все - об'єкт», тобто будь-яка одиниця даних є об'єктом - екземпляром деякого класу, до якого можна застосувати всі синтаксичні засоби, призначені для роботи з об'єктами. У цьому сенсі мова не містить вбудованих примітивних типів даних. Умовно такими можна вважати типи, що надаються інтерпретатором і системної бібліотекою, використовувані найбільш часто і не потребують для використання спеціального вказівки імені класу.

Для розробки графічного інтерфейсу користувача (UI), був вики GTK + 3.0. GTK + - це фреймворк для створення кроссплатформенного графічного інтерфейсу користувача (GUI). Поряд з Qt він є однією з двох найбільш популярних на сьогоднішній день бібліотек для X Window System.

Спочатку ця бібліотека була частиною графічного редактора GIMP, але пізніше стала незалежною і набула популярності. GTK + - це вільне ПЗ, яке розповсюджується на умовах GNU LGPL і дозволяє створювати як вільне, так і власницьке програмне забезпечення.

Саме доступність і адаптивність даної бібліотеки до всіх існуючих платформ стали основними критеріями для вибору інструменту реалізації інтерфейсної частини.

### *2.1.2 Модель розробки ПЗ*

Після аналізу предметної області, встановивши відповідні критерії и требования в программному обеспечению, была выбрана V-модель проектировки.

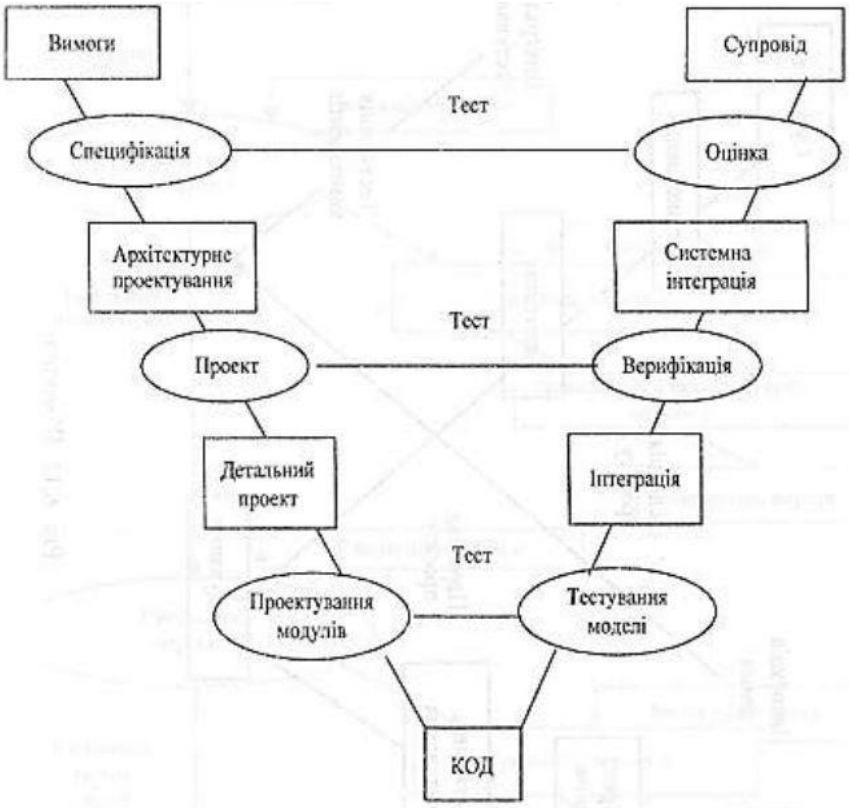


Рис 2.1. Структура V-моделі

*V-модель життєвого циклу* була введена для ідентифікації дій, пов'язаних з тестуванням на всіх стадіях розробки програмного продукту. Лівий бік моделі містить традиційні фази каскадної моделі, проте окрім робочого продукту виробляється відповідний тест. Правий бік моделі пов'язаний з інтеграцією і тестуванням.

Оскільки при розробці додатку були необхідні постійні тестування та калібровки функціоналу, ця модель виявилася оптимальною, хоча далеко не найбільш економічною. Вона забезпечує всі переваги та простоту моделі "каскада" і дозволяє додатково уточнити всі інтеграційні випробування та обслуговування з мінімальним залученням клієнтів, щоб заощадити час та інтегрувати продукт якнайкраще.

### *2.1.3 Діаграма класів (Class diagram) та її опис*

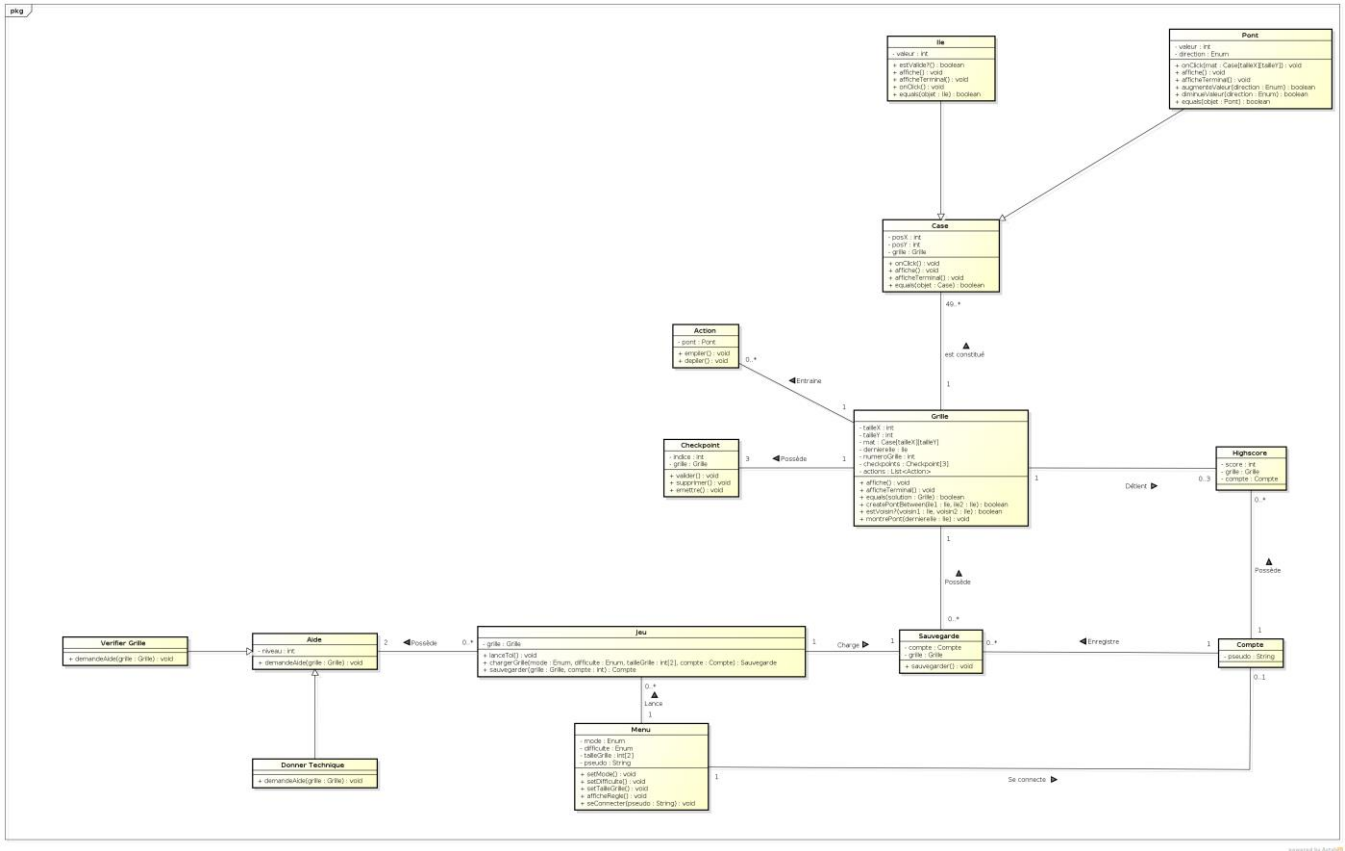


Рис 2.2. Фрагмент Sudoku

*Клас Jeu* відповідає за ігровий процес (2ге вікно після головного меню). Завантажує решітку із головоломкою.

*Клас Menu* – найперше вікно, головне меню. Запускається користувачем. Описує параметрування майбутнього раунду.

*Клас Aide* – ініціалізує вікно допомоги в меню та прогягом гри.

```
def initialize()
        @builderAide = Gtk::Builder.new
```

```
@builderAide.add_from_file("lib/Hashiparmentier/glade
/aide.glade")

        @window                                          =
@builderAide.get_object("windowAide")

        @window.signal_connect('destroy') { |_widget|
Gtk.main_quit }

@window.style_context.add_provider(@@CSS_AIDE,
Gtk::StyleProvider::PRIORITY_USER)

        @btnQuitter                                      =
@builderAide.get_object("btnQuitter")
        @btnQuitter.signal_connect('clicked')        {
|_widget| @window.destroy() }

@btnQuitter.style_context.add_provider(@@CSS_BUTTON_R
OSE, Gtk::StyleProvider::PRIORITY_USER)
```

*Клас Sauvegarde* - цей клас дозволяє зберігати в базі даних прогрес гравця на сітці та проводити трекінг збережень.

```
class Sauvegarde < ActiveRecord::Base
```

*Клас Grille* - клас представляє сітку гри з коробками, стеком дій, гіпотезами тощо. Сітки з “островами” не генеруються, а завантажуються із шаблонів, створених в окремій папці (формат .txt).

```
def Grille.chargerGrilles(dossier)
```

```
    ret = Array.new()
    Dir.each_child(dossier) do |fichier|
      text = File.open(dossier + "/" + fichier).read
      text.gsub!(/\r\n/, "\n");
      ret.push(Grille.creer(text))
    end
    return ret

  end
```

Цей метод дозволяє визначити, чи вирішена сітка (відповідає рішенню). True – решітка співпадає, false – ні.

```
def fini?()
    for i in (0..(@tailleX-1))
      for j in (0..(@tailleY-1))
        if(@mat[i][j] != @matSolution[i][j])
          return false
        end
      end
    end
    return true
  end
```

*Клас Compte* - цей клас представляє облікові записи користувачів.

Цей метод дозволяє отримати обліковий запис у базі даних або створити його, якщо він не існує:

```
def Compte.recuperer_ou_creer(pseudo)
  compte = Compte.find_by(name: pseudo);
  if(compte == nil)
```

```
      Compte.creer(pseudo)
    else
      return compte
    end
  end
```

*Клас Action* - цей клас представляє дії, що виконуються гравцем (розміщення, зняття мостів)

```
def Action.creer(ile1, ile2, methode)
    new(ile1, ile2, methode)
  end
  #:nodoc:
  def initialize(ile1, ile2, methode)
    @ile1 = ile1
    @ile2 = ile2
    @methode = methode
  end
```

*Клас Case* - цей клас представляє квадрати нашої сітки, конструює клітини, визначає на них острови та симулює “клік” на них.

*Клас Pont* – репрезентація мостів, їх кольорів та дії користувачів на них.

*Клас Ile* – представлення острова на решітці.

### *2.1.4 Діаграма послідовностей (Sequence diagram)*

Послідовність дій в грі HASHI (рис. 2.3-2.4):

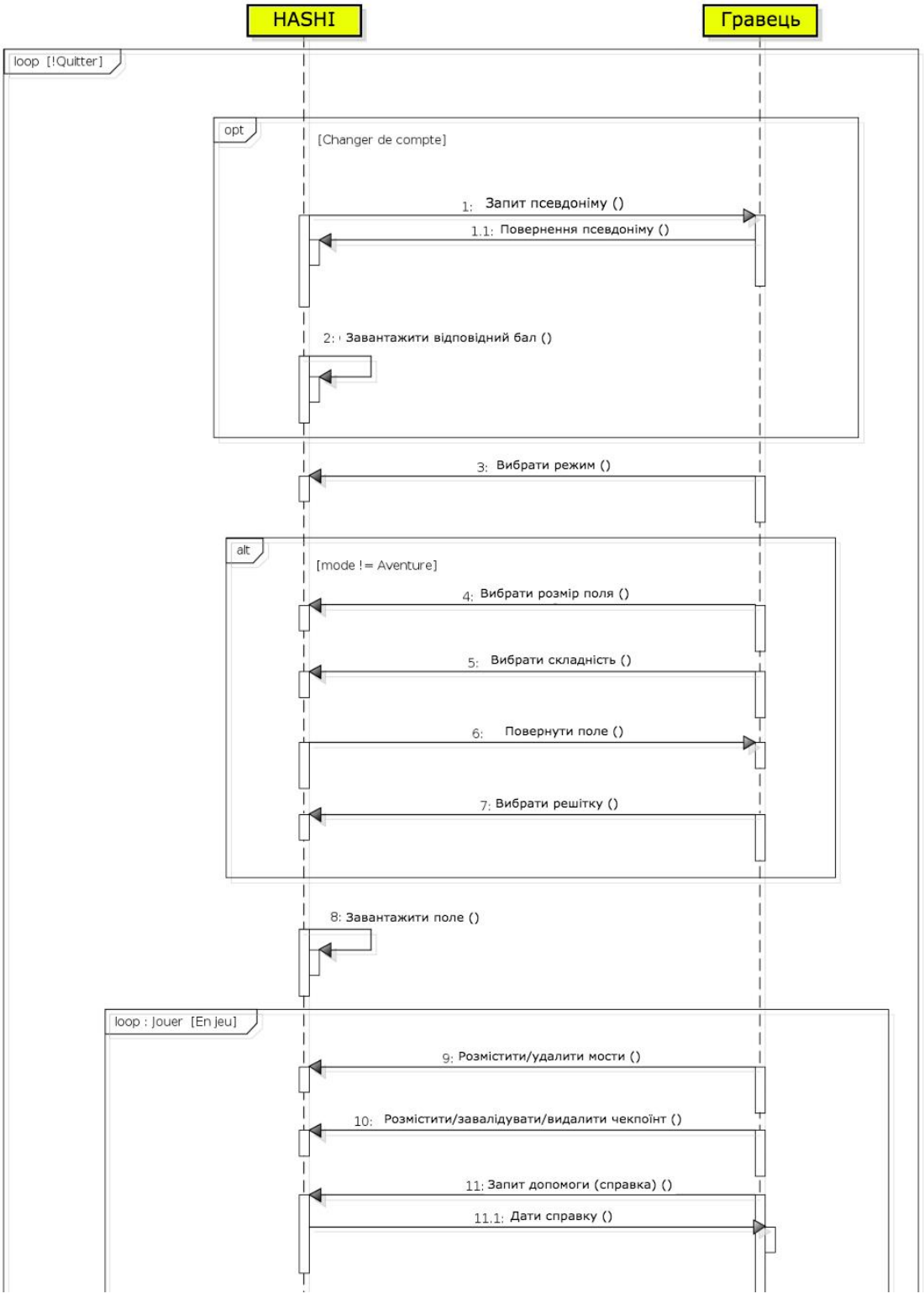


Рис. 2.3. Діаграма послідовностей (1 частина)

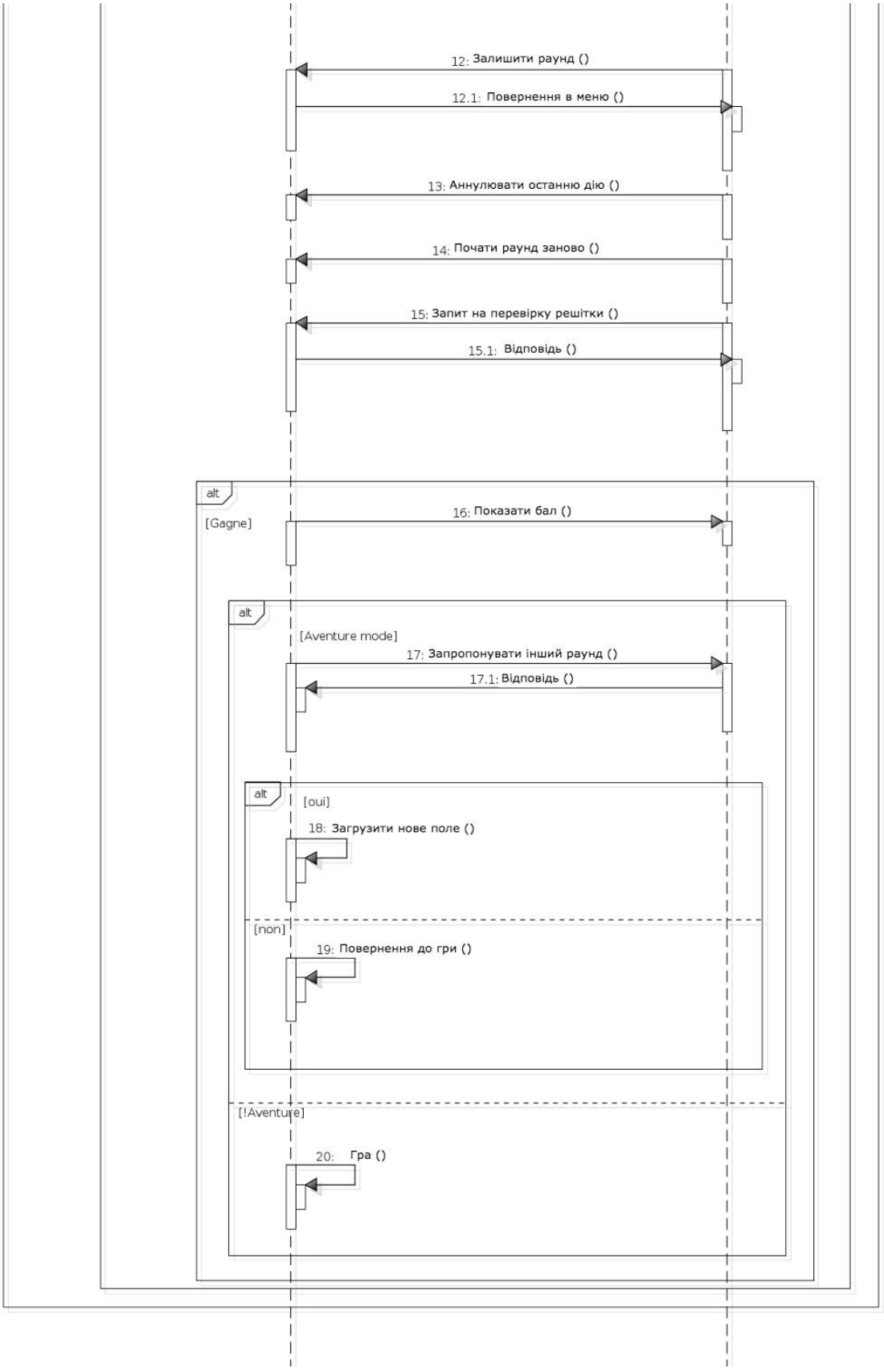


Рис. 2.4. Діаграма послідовностей (2 частина)

# РОЗДІЛ 3

# 3 ФУНКЦІОНАЛЬНІСТЬ ПЗ

## 3.1 Діаграма активностей (Activity diagram)

Надається загальна деталізована діграма. Детальний опис функціонування представлений у формі документації в розділі 3.2.

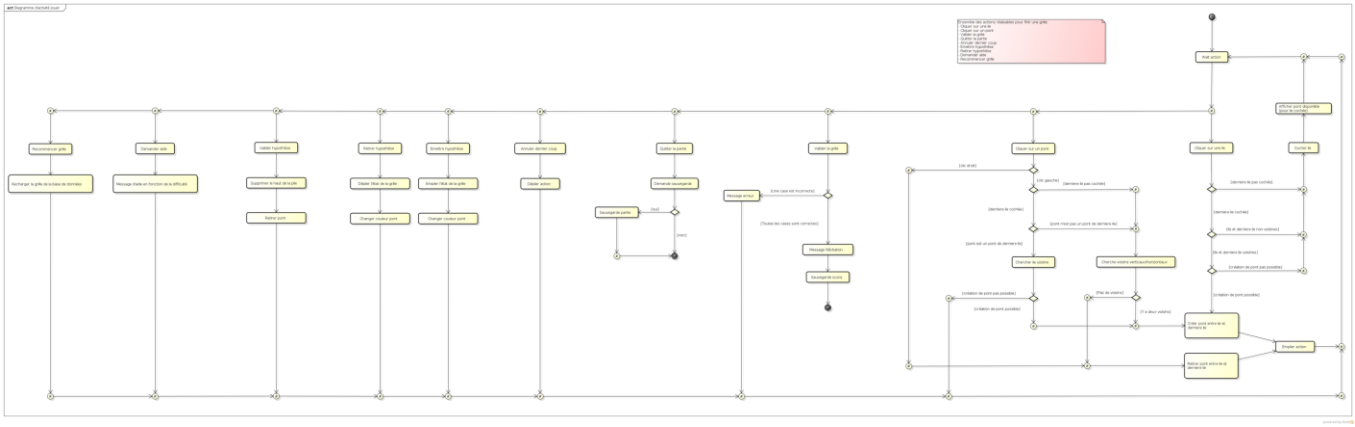

Рис. 3.1. Діаграма активностей

## 3.2 Огляд функціоналу ПЗ

### *3.2.1 Документація SAIF BF*

Поведінкова структура (Behavioral Framework - BF) надає набір конструкцій для визначення поведінкової семантики специфікацій, які забезпечують робочу взаємодію. Як результат, у центрі уваги BF - підзвітність - опис "хто що робить":

Таблиця 3.1.

Документація SAIF BF

| Посилання | Функції |
|---|---|
| **BF 01 :** | **Управління раундом:**<br>• BF 1.1: Керування секундоміром.<br>• BF 1.2: Дозволити висувати / перевіряти / видаляти гіпотези.<br>• BF 1.3: Дозволити повернутися назад / вперед (скасувати / повторити).<br>• BF 1.4: Зберегти розвиток сітки після виходу з гри.<br>• BF 1.5: Надати допомогу з вирішенням.<br>• BF 1.6: Дати оцінку відповідно до часу вирішення та запитуваної допомоги.<br>• BF 1.7: Дозволити перезапуск мережі. |
| **BF 02 :** | **Взаємодія сітки / користувача:**<br>• BF 2.1: Дозволити створити / видалити мости.<br>• BF 2.2: Дозволити перевірити / анулювати острови |
| **BF 03 :** | **Управління резервними копіями :**<br>• BF 3.1 : Зберегти прогресію решіток<br>• BF 3.2 : Зберегти найкращий результат (для кожного гравця / решітки) |
| **BF 04 :** | **Управління сітками:** |

| | |
|---|---|
| | • BF 4.1: Зберегти набір сіток для вирішення |
| **BF 05 :** | **Управління головним меню :**<br><br>• BF 5.1: Можливість вибору параметрів гри (ігровий режим / розмір сітки / складність сітки)<br>• BF 5.2: Можливість вибрати сітку серед запропонованих<br>• BF 5.3: Можливість відновити сітку, що триває або з самого початку<br>• BF 5.4: Мати можливість вийти з гри<br>• BF 5.5: Увійти під псевдонімом |

**BF 01 - Управління раундом:**

Ці функції дозволяють управляти допоміжними засобами та інструментами для вирішення сітки.

• BF 1.1 - Керувати секундоміром: ця функція дозволяє виміряти час роздільної здатності гри, гравець може призупинити свою гру в будь-який час.

• BF 1.2 - Дозволити висувати / перевіряти / видаляти гіпотези: ця функція дозволяє користувачеві керувати гіпотезами:
- Нова гіпотеза:
- Зберегти стан сітки.

- Змініть колір завершених мостів та островів, які будуть забарвлені пізніше.
- Підтвердити гіпотезу
- Збережіть удари, зроблені під час гіпотези.
- Видалити гіпотезу
- Повертається до стану сітки до гіпотези.

• BF 1.3.

• BF 1.4 - Зберегти прогрес сітки під час виходу з гри: ця функція автоматично зберігає прогрес гравця, коли він нормально залишає гру.

• BF 1.5 - Надайте допомогу з вирішенням: ця функція дозволяє користувачеві звернутися за допомогою до вирішення. Він має два рівні допомоги.

- Дати техніку: Відображає повідомлення з технікою, яку потрібно використовувати для розблокування програвача.

- Перевірити сітку: виділяє помилки сітки.

• BF 1.6 - Виставляйте оцінку відповідно до часу вирішення та запитуваних допоміжних засобів: ця функція дозволяє оцінювати гру гравця, щоб класифікувати найкращі ігри.

• BF 1.7 - Дозволити перезапустити сітку: ця функція дозволяє користувачеві перезапустити свою гру в будь-який час, натиснувши кнопку.

**BF 02 - Взаємодія сітки / користувача:**

Ces fonctionnalités permettent à l'utilisateur d'interagir avec la grille

- BF 2.1 - Дозволити створювати / видаляти мости: ця функція дозволяє користувачеві маніпулювати мостами трьома різними способами.

  - Створіть міст, клацнувши лівою кнопкою миші на острівці (який виділить наявні мости), а потім клацнувши лівою кнопкою миші один із виділених мостів

  - Збільште міст, клацнувши лівою кнопкою миші на ньому. (Якщо мостів вже два, тоді міст зникне).

  - Зменшіть міст, клацнувши правою кнопкою миші на ньому.
- BF 2.2 - Дозволити перевіряти / анулювати острови: ця функція дозволяє користувачеві розміщувати або видаляти хрести на островах, клацаючи лівою кнопкою миші або клацаючи правою кнопкою миші на них.

**BF 03 - Управління резервними копіями :**

Ці функції використовуються для управління системою резервного копіювання. Ця система заощадить прогрес користувача на сітці та набір балів, щоб оцінити найкращі ігри.

- BF 3.1 - Зберегти прогресії сітки: ця функція автоматично зберігає прогрес гравця, коли він нормально залишає гру.

- BF 3.2 - Збережіть найкращий результат у кожній сітці (для кожного псевдоніма в сітці): ця функція дозволяє системі зберігати оцінку, призначену гравцеві в кінці гри, для встановлення рейтингу.

**BF 04 - Управління резервними копіями :**

Ці функції дозволяють управляти усіма сітками, які гра може запропонувати користувачеві.

• BF 4.1 - Збережіть набір сіток для вирішення: ця функціональність дозволяє системі зберігати в базі даних набір сіток з різними розмірами та труднощами.

**BF 05 – Управління головним меню:**

Ці функції дозволяють користувачеві вибрати бажані налаштування гри.

• BF 5.1 - можливість вибору параметрів гри (режим гри / розмір сітки / складність сітки): ця функція дозволяє користувачеві вибирати різні параметри гри за допомогою кнопок.

• BG 5.2 - можливість вибрати сітку серед запропонованих: ця функція дозволяє користувачеві вибрати одну із сіток, збережених у базі даних, відповідно до вибраних параметрів.

• BF 5.3 - Щоб мати змогу відновити сітку, що триває або з самого початку: ця функція дозволяє користувачеві відновити гру, яка була нормально зупинена (користувач виходить з гри), або перезапустити гру з самого початку.

• BF 5.4 - Щоб мати можливість вийти з гри: ця функція дозволяє користувачеві вийти з гри, що триває, і мати можливість вийти з гри.

• BF 5.5 - Підключення з псевдонімом: ця функція дозволяє користувачеві з’єднуватися з псевдонімом для запису отриманих результатів.

### *3.2.2 Графічний інтерфейс користувача*

Екран головного меню (рис. 3.2) являє собою фон з кнопками для довідки за правилами гри і поясненнями про функціонал додатка (зверху в лівому кутку). По центру екрану розміщений фірмовий логотип програми. Нижче розташовані кнопки параметрованія майбутнього ігрового раунду: режим, розмір поля і складність. Під кнопками розташовано поле введення імені (псевдоніма) користувача. У самому низу розташовані кнопки виходу з гри і старту раунда.


Рис. 3.2. Головне меню фінального додатку

При натисканні кнопок довідки відкриваються додаткові вікна з відповідною інформацією (рис. 3.3). Функціонал цього меню реалізований у файлі Menu.rb, а графічний інтерфейс - у відповідних файлах .glade .


Рис. 3.3. Екран вікна допомоги

Вікно раунда гри. Крім аналогічних кнопок довідки зверху, тут присутні всі зазначені раніше елементи, а саме: картате поле раунди (15x15 на малюнку) таймер, нікнейм, лічильник очок (зверху), кнопки повернення, маніпуляцій над гратами і перевірки результату (знизу).

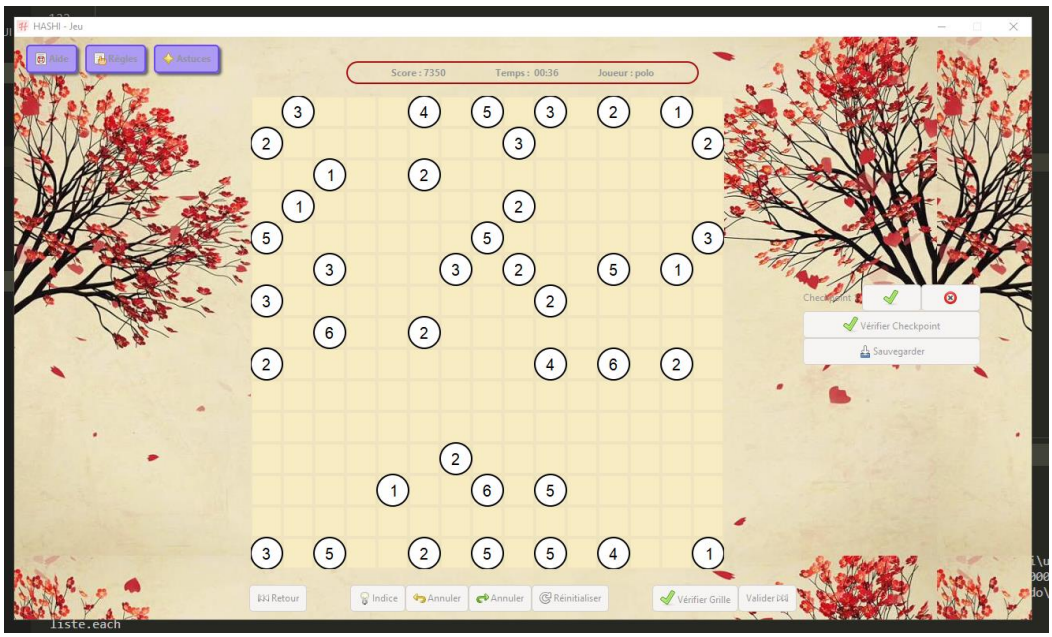


Рис. 3.4. Ігровий процес Hashi

Коли гра закінчена, з'являється вікно кінця раунду. Представлений результат раунду (час і окуляри), а також результати інших учасників і «нагорода» в якості медалі.

При натисканні кнопок довідки відкриваються додаткові вікна з відповідною інформацією. Функціонал цього меню реалізований в файлі Menu.rb, а графічний інтерфейс - у відповідних файлах .glade.

# РОЗДІЛ 4
# 4 ТЕСТУВАННЯ

Для перевірки працездатності ігрового програми було проведено функціональне тестування і юзабіліті-тестування.

## 4.1 Функціональне тестування

Функціональне тестування - це тестування ПО з метою перевірки можливості бути реалізованим функціональних вимог, тобто здатності ПО в певних умовах вирішувати завдання, потрібні користувачам.

### *4.1.1 Тестування на Linux*

Тест No 1. Зміна налаштувань ігрового програми.
Вхідні дані: гравець змінює параметри складності, розміру поля і складність (18 комбінацій).
Очікуваний результат: коректне відображення заданих параметрів в грі.
Отриманий результат: збігається з очікуваним.

Тест пройдено успішно.

Тест No 2. Відображення кращого результату.
Вхідні дані: кілька результатів по завершенні гри. Очікуваний результат: виведений кращий результат.
Отриманий результат: збігається з очікуваним.

Тест пройдено успішно.

Тест No 3. Збереження гри і чекпоінти

Вхідні дані: гравець зберігає свій прогрес при будівництві мостів в раунді.

Очікуваний результат: повернення до попередньої комбінації рішення.

Отриманий результат: збігається з очікуваним.

Тест пройдено успішно.

Тест No 4. Порівняння результатів.

Вхідні дані: гравець починає раунди під різними нікнейнамі.

Очікуваний результат: В кінці гри з'являється таблиця результатів, часу всіх никнеймов

Отриманий результат: збігається з очікуваним.

Тест пройдено успішно.

### *4.1.2 Тестування на Windows та Mac OS*

Тест No 5. Тестування на Windows і Mac OS

Вхідні дані: гравці тестують гру на Windows і Mac OS.

Отриманий результат наведено в таблиці 4.1 :

Таблиця 4.1

| ТЕСТУВАННЯ ОС | Linux (Ubuntu) | Windows | Mac OS |
|---|---|---|---|
| Тест No 1 |  |  |  |
| Тест No 2 |  |  |  |
| Тест No 3 |  |  |  |
| Тест No 4 |  |  |  |

Як можна побачити, згідно із тестуваннями, на Mac OS не працює бібліотека sqlite, відповідальна за базу даних для збереження нікнеймів та часу проходження. Скоріш за все, це пов’язано із старою версією системи (HighSierra).

## 4.2 Юзабілити-тестування

Юзабіліті-тестування (перевірка ергономічності) - метод оцінки зручності продукту у використанні, заснований на залученні користувачів в якості випробувачів і підсумовуванні отриманих від них висновків.

При проходженні даного тесту було залучено вісім пользова- телей. Тестувальникам було запропоновано вирішити такі завдання:

- ✓ змінити налаштування програми;
- ✓ почати гру;
- ✓ набрати тисячу або більше очок;
- ✓ переглянути рекорд.

Всі завдання були вирішені усіма тестувальниками без труднощів. Користувачі відзначили интуитивность управління і правил гри. Юзабілити-тестування пройшло успішно.

# ВИСНОВКИ

Для підведення висновків, пертинентним є підрахунок трудомісткісткості.

Таблиця 5.1

Розрахунок трудоємності задля розробки ігрового додатку HASHI

| № | Етапи проектування | В | К | Формула розрахунку | Трудоємність (люд.- г.) |
|---|---|---|---|---|---|
| 1 | Вивчення опису завдання $T_п$ | 1,5 | 1,3 | $T_п = \frac{O}{75 \cdot K} \cdot B$ | 3,2 |
| 2 | Алгоритм вирішення задачі $T_а$ | 1,5 | 1,3 | $T_а = \frac{O}{20 * K}$ | 8 |
| 3 | Блок-схема алгоритма $T_{бс}$ | 1,5 | 1,3 | $T_{бс} = \frac{O}{10 * K}$ | 16 |
| 4 | Розробка программи за блок-схемою $T_н$ | 1,5 | 1,3 | $T_н = \frac{O}{20 * K}$ | 18 |
| 5 | Відладка програми ЭВМ $T_{отл}$ | 1,5 | 1,3 | $T_{отл} = \frac{O}{4 * K}$ | 40 |
| 6 | Підготовка документації $T_д$ | 1,5 | 1,3 | $T_д = \frac{O}{15 * K}$ | 10,7 |
| | Итого: | | | | 95,9 |

Сума людино-годин на проектування гри: 95,9

В процесі виконання роботи були розглянуті види навчальних ігор, їх класифікація та особливості сприйняття ігор учнями.

При розробці гри поєднувалися лаконічність інтерфейсу з досить складною ігрової логікою завдань. Вдалося поєднати об'єктно-орієнтований

базис, графічний інтерфейс і базу даних в одній грі. Таким чином, всі поставлені в роботі завдання вирішені, і мета роботи досягнута.

Підводячи підсумки проведеного дослідження, можна сказати, що все поставлені завдання вирішені, отже, головна мета, поставлена в випускній кваліфікаційній роботі, а саме створення, тестування та апробація ігрового додатку, досягнута.

# ВИКОРИСТАНІ ДЖЕРЕЛА


1. Красноухов В. И. Занимательный мир механических головоломок
2. Pegg, Ed Jr. (2005-09-15). "Ed Pegg Jr.'s Math Games: Sudoku Variations". MAA Online. The Mathematical Association of America. Archived from the original on 2006-10-13. Retrieved 2006-10-03.
3. https://mycryptowiki.com/wiki/app/1462494195/microsoft-sudoku
4. https://expert.ru/2020/07/28
5. Брюс Тэйт Практическое использование Rails: Часть 4. Стратегии тестирования в Ruby on Rails. 01.07.2008.
6. Хэл Фултон, Андре Арко. Путь Ruby. — 3-е изд. — М.: ДМК Пресс, 2016. — С. 33—38.
7. Current UML Specification by Object Management Group (OMG)
8. Kaner C., Bach J., Pettichord B. Lessons Learned in Software Testing. – USA: Wiley, 2001. – 320 p.
9. Flower M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – USA: Addison-Wesley Publishing Company, 2004. – 150 p.
10. Ларман К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итера- тивную разработку. – М.: Вильямс, 2013. – 736 с.
11. Майерс Г., Баджет Т. Искусство тестирования программ. – М.: Вильямс, 2012. – 272 с.
12. Вершиніна Е.В., Гонченко М.С. (Сост.). Огляд моделей життєвого циклу розробки програмного забезпечення
13. Вершиніна Е.В., Гонченко М.С. (Сост.). Огляд моделей життєвого циклу розробки програмного забезпечення

14. GTK+ / Gnome Application Development, Havoc Pennington, Copyright © 1999 by New Riders Publishin
15. Russ Olsen. Design Patterns in Ruby
16. Hal Furton. The Ruby Way: Solutions and Techniques, Second Edition. Addison-Welsey, 2007
17.