МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2024 р.

***КВАЛІФІКАЦІЙНА РОБОТА***

**на здобуття ступеня вищої освіти «магістр»**

**на тему:** Математична модель для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity та розробка програми для її реалізації

Виконав: студент групи 6151м

_____ Семенчук І.М.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н, доцент.

(посада, науковий ступень вчене звання)

_____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2024 р.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
« 14 » жовтня 2024 р.

***ЗАВДАННЯ***
***НА КВАЛІФІКАЦІЙНУ РОБОТУ***
**на здобуття ступеня вищої освіти «магістр»**

Студенту Семенчуку Івану Миколайовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Математична модель для оцінювання складності об'єктно-орієнтованого проекування через зв'язки між класами 2D ігор на рушії Unity та розробка програми для її реалізації

Керівник роботи к.т.н. доц. Макарова Л.М.

Затверджені наказом ректора № 1070-УЧ від « 11 » 10 2024 р.

2. Термін подання роботи: 09.12.2024 р.

3. Вихідні дані по роботі:

4. Перелік питань, що належать до розробки (найменування розділів)

Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Розділ з розробки програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів Титульний слайд. Актуальність теми. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації. Існуючі математичні моделі для оцінювання складності ООП через зв'язки між класами. Рівняння еліпсу передбачення для нормалізованих метрик RFC та CBO 2D ігор на рушії Unity. Рівняння трансформованого еліпсу передбачення для метрик RFC та CBO. Постановка задачі на розробку проекту програми. Діаграма варіантів використання. Діаграма діяльності. Графічне вікно програми з інтерфейсними компонентами. Результати випробувань програми. Висновки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 14.10.2024 р.

**К А Л Е Н Д А Р Н И Й  П Л А Н**

| Номер | Назва етапів роботи | Терміни виконання | Примітка |
|---|---|---|---|
| 1. | Підготовка розділу ВСТУП | 16.10.2024 | НС* |
| 2. | Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою | 18.10.2024 | НС* |
| 3. | Підготовка розділу (ів) за результатами власних досліджень | 21.10.2024 | НС* |
| 4. | Підготовка розділу з розробки програмного забезпечення | 27.11.2024 | |
| 5. | Підготовка організаційно-економічного розділу | 29.11.2024 | |
| 6. | Підготовка розділу з охорони праці | 02.12.2024 | |
| 7. | Підготовка розділу з охорони навколишнього середовища | 04.12.2024 | |
| 8. | Підготовка розділу ВИСНОВКИ | 05.12.2024 | |
| 9. | Оформлення списку використаних джерел та додатків | 06.12.2024 | |
| 10. | Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.) | 09.12.2024 | не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку) |
| 11. | Підготовка презентації та доповіді | 13.12.2024 | |
| 12. | Попередній захист роботи на засіданні кафедри ПЗАС | 16.12.2024 | |
| 13. | Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді | 19.12.2024 | |

* - за результатами наукового стажування (НС), яке було з 02.09.2024 по 13.10.2024 р.

**Студент** _____ Семенчук І.М.
(підпис) (ПІБ)

**Керівник роботи** _____ Макарова Л.М.
(підпис) (ПІБ)

# РЕФЕРАТ

Семенчук Іван Миколайович

«Математична модель для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на Unity та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2024 р.

**Обсяг роботи:** 96 стор., 9 табл., 15 рис., 30 використаних джерел, 5 додатків.

**Актуальність теми роботи:** в парадигмі розробки ігор високі значення метрик RFC і CBO, які значно перевищують рекомендовані значення є частим явищем. Тому розробка математичної моделі, яка дозволить оцінювати складність об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity за значеннями метрик FRC і CBO з урахуванням їх кореляції на рівні застосунку є актуальною.

**Мета та завдання дослідження.** Метою роботи є підвищення достовірності оцінювання складності об’єктно-орієнтованого проєктування на основі аналізу зв’язків між класами 2D ігор з відкритим кодом, створених на ігровому рушії Unity. Завданнями дослідження є аналіз існуючих моделей для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами, розробка математичної моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity.

**Об’єкт дослідження** – процес оцінювання складності об’єктно-орієнтованого проєктування через зв’язки між класами у 2D іграх з відкритим кодом на Unity.

**Предмет дослідження** – математична модель для оцінювання складності об’єктно-орієнтованого проєктування на основі зв’язків між класами 2D ігор з відкритим кодом на рушії Unity.

**Методи дослідження.** Для розв'язання поставленої задачі було застосовано методи теорії ймовірності, математичної статистики та багатовимірного статистичного аналізу.

**Наукова новизна отриманих результатів**. Удосконалено рівняння еліпсу передбачення для десяткових логарифмів нормалізованих метрик RFC та CBO для оцінювання складності ООП через зв’язки між класами 2D ігор розроблених на рушії Unity на основі нормалізуючого перетворення у формі десяткового логарифму із врахуванням викидів у даних. Рівняння еліпсу передбачення враховує кореляцію між метриками RFC та CBO, що дозволяє підвищити достовірність оцінювання складності ООП через зв’язки між класами 2D ігор розроблених на рушії Unity**.**

**Практичне значення одержаних результатів.** Створено програмне забезпечення для оцінювання складності ООП через зв’язки між класами 2D ігор розроблених на рушії Unity, яке реалізовано мовою програмування Python.

**Апробація результатів роботи.** Основні положення і результати досліджень пройшли апробацію на XIV Міжнародній науково-технічної конференції «Інновації в суднобудуванні та океанотехніці» (20-21 вересня 2023, м. Миколаїв) та V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS-2024)» (30-31 жовтня 2024, м. Миколаїв).

**Публікації.** За результатами досліджень опубліковано дві наукові праці, а саме матеріали конференції.

**Ключові слова:** об’єктно-орієнтоване проектування, оцінювання складності проектування, зв’язки між класами, 2D ігри, Unity, еліпс передбачення.

# ABSTRACT

Semenchuk Ivan Mykolayovych

" Mathematical model for estimating the complexity of object-oriented design through class relationships in 2D games on the Unity engine and development of a program for its implementation"

Qualification work for obtaining a master's degree in the speciality 121 «Software Engineering». Admiral Makarov National University of Shipbuilding. Mykolaiv, 2024.

Volume of work: 96 pages, 9 tables, 15 figures, 30 references, 5 appendices.

**Relevance of the topic of the work:** In the game development paradigm, high values of RFC and CBO metrics, which significantly exceed recommended thresholds, are a common occurrence. Therefore, the development of a mathematical model that enables the evaluation of object-oriented design complexity through class interrelationships in 2D games developed on the Unity engine, based on RFC and CBO metrics while considering their correlation at the application level, is highly relevant.

**The goal and objectives of the study.** The objective of this work is to enhance the accuracy of evaluating the complexity of object-oriented design based on the analysis of interrelationships between classes in open-source 2D games developed on the Unity game engine. The research tasks include analyzing existing models for evaluating the complexity of object-oriented design through class interrelationships.and developing a mathematical model for evaluating the complexity of object-oriented design through the interrelationships between classes in 2D games developed on the Unity engine.

**The object of the study** is the process of evaluating the complexity of object-oriented design through class interrelationships in open-source 2D games developed on Unity.

**The subject of the study** is the mathematical model for evaluating the complexity of object-oriented design based on class interrelationships in open-source 2D games on the Unity engine.

**The methods of the study**. The solution of the posed problem involved methods from probability theory, mathematical statistics, and multivariate statistical analysis.

**The scientific novelty of obtained results.** The prediction ellipse equation for the decimal logarithms of the RFC and CBO metrics has been improved for evaluating the complexity of OOD through class interrelationships in 2D games developed on the Unity engine. This improvement is based on a normalizing transformation in the form of a decimal logarithm while accounting for outliers in the data. The prediction ellipse equation incorporates the correlation between RFC and CBO metrics, enhancing the reliability of evaluating the complexity of OOD through class interrelationships in 2D games created on Unity.

**The practical significance of obtained results** is that software application for evaluating the complexity of OOD through class interrelationships in computer games has been developed and implemented using the Python programming language.

**Approval of the research results.** The main provisions and results of the study were presented at the XIV International Scientific and Technical Conference "Innovations in Shipbuilding and Ocean Engineering" (September 20–21, 2023, Mykolaiv) and the V All-Ukrainian Scientific and Practical Online Conference "Information Technologies: Models, Algorithms, Systems (ITMAS-2024)" (October 30–31, 2024, Mykolaiv).

**Publications**. The research results were published in two scientific papers, specifically in conference proceedings.

**Keywords:** evaluation of object-oriented design complexity; class interrelationships; 2D games; Unity; prediction ellipse.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

CBO – Coupling Between Objects classes;

MD – Mahalanobis distance;

OOD – object-oriented design;

RFC – Response for Class;

SMD – squared Mahalanobis distance.

# ВСТУП

**Актуальність теми.** Щороку на рушії Unity розробляється велика кількість ігор, яка постійно зростає. Понад 50% всіх мобільних ігор розробляються на цьому рушії, а загальна кількість нових ігор на різних платформах може сягати тисяч. На Unity виходить від 6000 до 10000 ігор щороку тільки на платформі Steam. Якщо врахувати мобільні платформи, наприклад, App Store та Google Play, інді-ігри та менші платформи, ця цифра значно зростає і може перевищувати 20000 – 30000 нових ігор щороку [1].

Сучасна індустрія розробки ігор стикається з викликами, пов’язаними із складністю об’єктно-орієнтованого проектування. Оцінювання складності через зв’язки між класами є важливим аспектом, оскільки воно безпосередньо впливає на подальші витрати та ефективність підтримки, оновлення ПЗ. Відомо, що метрики ПЗ, такі як CBO та RFC, можуть слугувати інструментами для такого оцінювання [2]. Однак традиційний аналіз цих метрик зазвичай виконується окремо, без урахування їхньої кореляції на рівні проекту. Відповідно до [3] рекомендоване значення RFC має складати від 0 до 50, високе значення RFC ускладнює тестування, клас стає складним для розуміння. Рекомендоване значення CBO від 1 до 5 [4], якщо значення CBO перевищує 14 – це свідчить про надмірну зв’язаність, що є показником невдалої архітектури [5].

В ряді робіт [6 - 8] було запропоновано підхід для визначення складності об’єктно-орієнтованого проектування через зв’язки між класами на основі еліпсу передбачення для нормалізованих метрик RFC та CBO, зокрема, в роботі [6] - на рівні застосунку для проектів з відкритим кодом на Java, в роботі [7] - для перевірки взаємозв'язків між частинами Java-застосунків, в роботі [8] - для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами комп’ютерних ігор.

Проте, в парадигмі розробки ігор високі значення метрик RFC і CBO, які значно перевищують рекомендовані значення є частим явищем, значення RFC для

певних класів може дорівнювати сотням, а CBO десяткам. Тому розробка математичної моделі, яка дозволить оцінювати складність об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity за значеннями метрик FRC і CBO з урахуванням їх кореляції на рівні застосунку є актуальною.

**Мета і завдання дослідження.** Метою роботи є підвищення достовірності оцінювання складності об'єктно-орієнтованого проєктування на основі аналізу зв’язків між класами 2D ігор з відкритим кодом, створених на ігровому рушії Unity.

**Завдання дослідження:** Завданням дослідження є розробка математичної моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity, аналіз зв’язків між метриками RFC та CBO з врахуванням їхньої кореляції та створення програмного забезпечення для реалізації цієї моделі.

**Об’єкт дослідження** – процес оцінювання складності об'єктно-орієнтованого проєктування через зв’язки між класами у 2D іграх з відкритим кодом на Unity.

**Предмет дослідження** – математична модель для оцінювання складності об'єктно-орієнтованого проєктування на основі зв’язків між класами 2D ігор з відкритим кодом на рушії Unity.

**Методи дослідження.** Для розв'язання поставленої задачі було застосовано методи теорії ймовірності, математичної статистики та багатовимірного статистичного аналізу.

**Наукова новизна отриманих результатів**. Удосконалено рівняння еліпсу передбачення метрик RFC та CBO для оцінювання складності ООП через зв’язки між класами 2D ігор, розроблених на рушії Unity на основі нормалізуючого перетворення у формі десяткового логарифму із врахуванням викидів у даних та врахування кореляції між метриками RFC та CBO, що дозволяє підвищити достовірність оцінювання складності ООП через зв’язки між класами 2D ігор розроблених на рушії Unity**.**

**Практичне значення одержаних результатів.** Створено програмне забезпечення для оцінювання складності ООП через зв’язки між класами 2D ігор розроблених на рушії Unity**.**

**Особистий внесок здобувача**. Всі отримані результати є результатом самостійної роботи автора. У роботах, які були опубліковані у співавторстві, автором виконано: збір метрик програм, що використовують 2D графічний рушій Unity та попередня обробка даних [9]; побудова рівняння еліпсу передбачення метрик RFC та CBO для оцінювання складності ООП через зв’язки між класами 2D ігор, розроблених на рушії Unity та перевірка його якості [10].

**Апробація результатів роботи.** Основні положення і результати досліджень пройшли апробацію на XIV Міжнародній науково-технічної конференції «Інновації в суднобудуванні та океанотехніці» (20-21 вересня 2023, м. Миколаїв) та V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS-2024)» (30-31 жовтня 2024, м. Миколаїв).

**Публікації.** За результатами досліджень опубліковано дві наукові праці, а саме матеріали конференції.

# 1 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЕКТУВАННЯ ЧЕРЕЗ ЗВ'ЯЗКИ МІЖ КЛАСАМИ 2D ІГОР НА РУШІЇ UNITY ТА ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ ПРОВЕДЕННЯ ДОСЛІДЖЕНЬ ЗА ОБРАНОЮ ТЕМОЮ

У першому розділі було виконано аналіз існуючих моделей для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор та здійснено обґрунтування необхідності проведення досліджень за обраною темою.

## 1.1 Аналіз існуючих моделей для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами

Оцінювання складності об'єктно-орієнтованого проектування (ООП) є ключовим аспектом у галузі розробки програмного забезпечення, оскільки рівень складності архітектури має вагомий впливав на якість та ефективність продукту в майбутньому. Оцінювання складності об'єктно-орієнтованого проектування є критично важливим завданням для розробників ігор, оскільки тісно пов'язане з витратами на подальший розвиток і підтримку гри. Невірна оцінка складності проектування може призвести до ускладнень у майбутньому, пов'язаних із додатковими витратами, проблемами у підтримці, модифікації та масштабуванні продукту.

Метрики, такі як Coupling Between Objects (CBO) та Response for Class (RFC), є ефективними інструментами для аналізу складності окремих класів. Однак, на практиці ці показники часто розглядаються без врахування їх взаємного впливу, що ускладнює адекватне оцінювання загальної складності проектування на рівні всього застосунку. Крім того, допустимі значення цих метрик для окремих класів

не завжди дозволяють адекватно оцінити складність проектування всього програмного продукту [3].

Для вирішення цієї проблеми існує потреба в розробці моделі, яка враховує взаємозв'язки між метриками CBO та RFC на рівні всього застосунку.

Дану проблему детально розглянуто в роботі [6], де представлено математичну модель, яка враховує кореляцію між метриками CBO та RFC для Java-застосунків. У роботі розглядається перспективний підхід з використанням еліпсу передбачення, що дозволяє оцінювати зв'язок між метриками та ідентифікувати класи з високою складністю. Це дає змогу визначити критичні класи, які можуть створювати труднощі під час підтримки програмного забезпечення. Така модель сприяє точнішому оцінюванню складності проектування та виявленню аномальних зв'язків між класами, що допомагає завчасно виявити потенційні проблеми на ранніх етапах розробки.

На особливу увагу  заслуговує дослідження проблеми взаємозв'язків між класами для оцінювання складності ООП комп'ютерних ігор, яке представлене в роботі [8]. В ній для оцінювання складності ООП через зв'язки між класами комп'ютерних ігор був застосований еліпс передбачення для нормалізованих метрик RFC та CBO. За результатами роботи автори визначили, що для оцінювання складності ООП через зв'язки між класами комп'ютерних ігор може бути застосований еліпс передбачення для нормалізованих метрик RFC та CBO. Також було визначено, що побудову еліпсу передбачення для нормалізованих метрик RFC та CBO для оцінювання складності ООП через зв'язки між класами комп'ютерних ігор краще здійснювати із застосуванням нормалізуючих перетворень із врахуванням викидів у даних. Застосування критерію на основі квадрата відстані Махаланобіса до даних без їх нормалізації показало, що дані містять двовимірні викиди. Такий результат пояснюється тим, що зазначені дані мають розподіл, який не є гаусівським.

Підсумовуючи, розглянуті моделі мають кілька ключових переваг. По-перше, використання еліпсу передбачення є ефективним інструментом для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами за

допомогою метрик CBO та RFC та дозволяє глибше аналізувати структуру програмного забезпечення, виявляючи класи з високою складністю, які можуть становити потенційні труднощі для подальшої підтримки та розвитку. По-друге, застосування нормалізуючих перетворень підвищує надійність результатів, тому що дані мають негаусівський розподіл. Крім того, критерій, заснований на відстані Махаланобіса, надає можливість ефективно ідентифікувати двовимірні викиди, що важливо для отримання точної моделі і надійних результатів.

Підстановка тестових значень метрик RFC і CBO 2D ігор в розглянуті вище математичні моделі показала, що їх використання для оцінювання складності об’єктно-орієнтованого проектування 2D ігор, розроблених на рушії Unity, не є виправданим і потрібна нова модель яка враховуватиме всі аспекти і особливості розробки 2D ігор на рушії Unity.

## 1.2 Обґрунтування необхідності проведення дослідження за обраною темою

Щоб обґрунтувати необхідність проведення нового дослідження, перевіримо, наскільки достовірні результати оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами дають існуючі моделі для 2D ігор, розроблених на рушії Unity. Порівняємо з моделлю, запропонованою в роботі [8].

Для перевірки була використана вибірка двовимірних негаусівських даних, а саме середні значення метрик RFC та CBO для 50 проєктів 2D ігор з відкритим кодом, розроблених на рушії Unity мовою програмування C#. Проєкти були взяті з відкритого джерела - репозиторію GitHub (https://github.com). Дані представлено в таблиці 1.1.

Таблиця 1.1 - Значення метрик проєктів 2D ігор, розроблених на рушії Unity

| Назва проекту | Кіль-кість класів | Серед-нє значен-ня RFC | Серед-нє значен-ня CBO | Посилання на проекти |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2DRPG-master | 23 | 6,0870 | 0,2609 | https://github.com/BlainMaguire/2DRPG |
| 2dtouch-ctr-master | 6 | 13,8333 | 0,5000 | https://github.com/joaoborks/unity-2dtouch-ctr/tree/main |
| AngryBirdsStyleGame-master | 13 | 10,3077 | 0,4615 | https://github.com/dgkanatsios/AngryBirdsStyleGame |
| Asteroids-master | 47 | 14,9149 | 0,9362 | https://github.com/antfarmar/Unity-3D-Asteroids/ |
| Asteroids-tutorial-main | 5 | 14,6000 | 0,8000 | https://github.com/zigurous/unity-asteroids-tutorial |
| Bad-Manor-master | 52 | 11,5192 | 0,5192 | https://github.com/mohan-cao/Bad-Manor/tree/master |
| Blockbreaker-master | 11 | 30,0000 | 0,8182 | https://nuno-faria.itch.io/blockbreaker2 |
| Brick-breaker-tutorial-main | 5 | 11,8000 | 0,6000 | https://github.com/zigurous/unity-brick-breaker-tutorial/blob/main |
| CardGameTutorial-main | 25 | 13,2000 | 0,9600 | https://github.com/ddsurvivor/CardGameTutorial |
| CardGame-unity-2d-tutorial-master | 11 | 16,4545 | 0,7273 | https://github.com/sominator/unity-2d-card-game-tutorial/tree/master/ |
| Unity-Battle-City-master | 12 | 12,4167 | 0,4167 | https://github.com/Sonic853/Unity-Battle-City/tree/master/ |
| Cut_The_Rope_Replica-master | 5 | 6,2000 | 0,2000 | https://github.com/Brackeys/Cut-the-Rope-Replica/tree/master |
| DeathtrapDungeon-master | 32 | 16,6563 | 1,5313 | https://github.com/SouthBegonia/DeathtrapDungeon |
| RedRunner-master | 61 | 11,4918 | 0,6066 | https://github.com/BayatGames/RedRunner/tree/master |
| FlappyBird-unity-tutorial-main | 5 | 11,8000 | 0,4000 | https://github.com/zigurous/unity-flappy-bird-tutorial |
| UnityTradingCardGame-master | 7 | 76,0000 | 2,2857 | https://github.com/islam0talha/UnityTradingCardGame/tree/master |
| Happy-Village-Scary-Dungeons-master | 36 | 8,8333 | 0,4444 | https://github.com/ChancePenner/Happy-Village-Scary-Dungeons/blob/master |
| InfiniteRunner-2d-master | 21 | 8,6667 | 0,4762 | https://github.com/joaokucera/unity-infinite-runner-2d/tree/master |
| KifuwarabeFighter2-master | 52 | 6,9038 | 0,7692 | https://github.com/muzudho/KifuwarabeFighter2/blob/master |
| Legend-of-z-master | 73 | 22,5205 | 1,1096 | https://github.com/hueter/legend-of-z |
| Lu-Racer-master | 5 | 7,8000 | 0,4000 | https://github.com/mindcandy/lu-racer |
| MachineLearningRoguelike-master | 41 | 16,0488 | 1,0732 | https://github.com/UnityTechnologies/MachineLearningRoguelike |
| Misaka-2Dgame-Unity-master | 41 | 13,4634 | 1,1951 | https://github.com/voidarea97/Misaka-2Dgame-Unity |
| Pacman-unity-tutorial-main | 15 | 13,9333 | 1,8667 | https://github.com/zigurous/unity-pacman-tutorial |

Продовження таблиці 1.1

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Pacman-unity-tutorial-main | 15 | 13,9333 | 1,8667 | https://github.com/zigurous/unity-pacman-tutorial |
| Pinball-master | 8 | 22,6250 | 0,2500 | https://github.com/zwu002/Unity-Pinball |
| Piranhan-prototype | 23 | 7,5652 | 0,5652 | http://unity3d-jp.github.io/piranhan/ |
| Card-Game-Simulator-develop | 120 | 29,0667 | 29,0667 | https://github.com/finol-digital/Card-Game-Simulator |
| PompaDroid-master | 36 | 20,2500 | 1,0833 | https://github.com/dylanmpeck/PompaDroid/blob/master/ |
| Pong-tutorial-main | 7 | 5,8571 | 0,5714 | https://github.com/zigurous/unity-pong-tutorial |
| Ragnarok-master | 46 | 8,6304 | 0,2391 | https://github.com/JasonVanRaamsdonk/Ragnarok |
| RetroFutureGame-master | 88 | 19,4432 | 1,0341 | https://github.com/FelixEder/RetroFutureGame |
| Slider-main | 339 | 21,1917 | 1,4454 | https://github.com/randomerz/Slider |
| newbark-unity-master | 43 | 20,3721 | 0,9302 | https://github.com/itsjavi/newbark-unity/tree/master |
| Snake-tutorial-main | 2 | 19,5000 | 0,5000 | https://github.com/zigurous/unity-snake-tutorial/ |
| Sokoban.UnityClient | 119 | 8,3109 | 0,9412 | https://github.com/peteschmitz/Graphite.Unity.Sokoban |
| SpaceInvaders-tutorial-main | 7 | 19,1429 | 1,1429 | https://github.com/zigurous/unity-space-invaders-tutorial |
| Space-Shooter-Game-master | 12 | 5,5833 | 0,3333 | |
| Match3-master | 21 | 14,6667 | 1,5238 | https://github.com/gemfile0/Match3/tree/master |
| StrategyGameCourse-master | 11 | 22,0000 | 1,0909 | https://github.com/PeaPodAndPuffin/StrategyGameCourse/tree/master |
| Super-mario-tutorial-main | 18 | 12,7778 | 0,5556 | https://github.com/zigurous/unity-super-mario-tutorial |
| Super-polyspawn-wars-master | 22 | 44,7727 | 1,1818 | |
| Sycophant-master | 93 | 17,3548 | 1,0645 | https://github.com/Xenomega/Sycophant |
| TankHero-2D-master | 64 | 9,8594 | 0,6719 | https://github.com/bitzhuwei/TankHero-2D |
| Techbook-unity-2d-action-game-master | 37 | 34,3514 | 0,7027 | https://github.com/baba-s/techbook-unity-2d-action-game/tree/master |
| Tetris-unity-tutorial-main | 6 | 16,0000 | 1,1667 | https://github.com/zigurous/unity-tetris-tutorial |
| Tiptop-master | 13 | 16,3077 | 0,6923 | https://github.com/geraked/game-tiptop/tree/master |
| TurboTrack2D-master | 14 | 15,7857 | 0,7857 | https://github.com/h8man/TurboTrack2D |
| Tutorial-2d-game-unity-master | 13 | 11,4615 | 1,1538 | https://github.com/pixelnest/tutorial-2d-game-unity/tree/master |
| Unity2DGameTutorial-master | 19 | 6,7368 | 0,1579 | https://github.com/hussien89aa/Unity2DGameTutorial/tree/master |
| VGS-Tetris-master | 7 | 17,2857 | 0,7143 | https://github.com/VTSTech/VGS-Tetris |

Протестуємо отримані значення FRC та CBO підставивши їх в модель [8]. Результати представлено на рисунку 1.1. Можна зробити висновок, що використання моделі [8] для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор, розроблених на рушії Unity, не є виправданим, тому існує потреба у проведені нового дослідження.

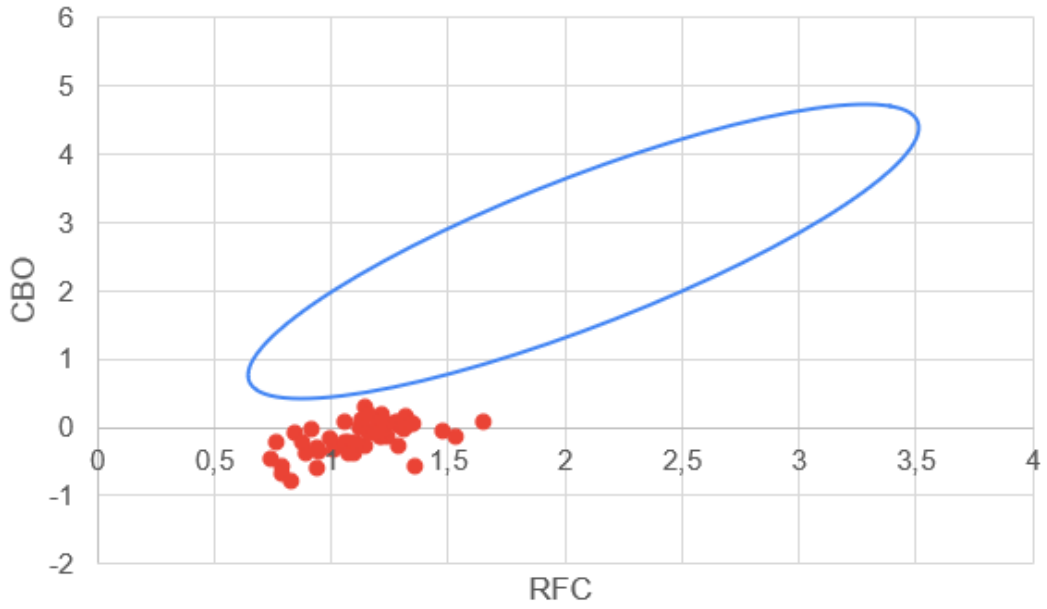


Рисунок 1.1 – Тестування моделі [8] з даними проектів 2D ігор, розроблених на рушії Unity

Актуальність нового дослідження полягає в необхідності створення математичної моделі для оцінювання складності об’єктно-орієнтованого проєктування спеціально для 2D ігор, розроблених на рушії Unity. Традиційні підходи до оцінювання складності об’єктно-орієнтованого проєктування не враховують специфіку ігрових проєктів, що призводить до неточних та неповних результатів.

Незважаючи на це, існує обмежена кількість спеціалізованих підходів до аналізу якості об’єктно-орієнтованого проєктування ігор, створених на Unity. Це створює прогалину в практичних інструментах, які могли б бути використані розробниками для покращення архітектури ігор та оптимізації процесу їхньої розробки.

Крім того, розробка математичної моделі, яка враховуватиме взаємозв’язки між ключовими метриками, такими як RFC та CBO, дозволить створити більш точний і надійний механізм оцінювання складності. Врахування кореляцій між цими метриками, а також можливих аномалій у даних.

Практичне значення цього дослідження полягає у можливості його застосування для прогнозування складності проєктів ще на етапі їхнього планування, що дозволить заздалегідь ідентифікувати ризики, пов’язані зі складністю реалізації. Це може бути корисним не лише для індивідуальних розробників, але й для команд, які працюють над великими ігровими проєктами, де критично важливо оптимізувати зв’язки між класами для досягнення високої продуктивності та легкості підтримки коду.

Таким чином, створення математичної моделі для оцінювання складності об’єктно-орієнтованого проєктування 2D ігор на рушії Unity є важливим кроком для розвитку сучасних методів аналізу архітектури програмного забезпечення, особливо в контексті ігрової індустрії.

## 1.3 Висновки за розділом 1

В даному розділі було розглянуто існуючі математичні моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами, приведено обґрунтування необхідності проведення дослідження за обраною темою.

1) Для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами можна використовувати еліпс передбачення для нормалізованих значень метрик RFC та CBO.

2) Для більш точної оцінки отриманих результатів можна використовувати трансформований еліпс передбачення, який базується на еліпсі передбачення для нормалізованих даних з використанням зворотнього нормалізуючого перетворення.

3) Існує необхідність в подальшому удосконаленні математичної моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity.

# 2 МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЕКТУВАННЯ ЧЕРЕЗ ЗВ'ЯЗКИ МІЖ КЛАСАМИ 2D ІГОР НА РУШІЇ UNITY

В цьому розділі було виконано перевірку зібраних даних за допомогою критерію Мардіа на те, чи мають ці дані нормальний розподіл, було виконано нормалізацію зібраних даних за допомогою нормалізуючого перетворення десятковим логарифмом, також було виконано перевірку на наявність викидів та їх видалення. Надалі на основі нормалізованих значень метрик RFC та CBO було побудовано еліпс передбачення для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор, розроблених на рушії Unity. На основі отриманого еліпсу передбачення для нормалізованих даних за допомогою зворотнього нормалізуючого перетворення було побудовано трансформований еліпс передбачення для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор розроблених на рушії Unity.

## 2.1 Методика перевірки багатовимірної нормальності даних та побудови еліпсу передбачення

Перед початком побудови треба перевірити, чи мають отримані дані нормальний розподіл. Виконати це можна за допомогою тесту Мардіа [11].

Перевірка багатовимірної нормальності полягає в тому, щоб перевірити, чи багатовимірна асиметрія та ексцес узгоджуються з багатовимірним нормальним розподілом. Для вибірки $X_1$, $X_2$, …, $X_n$, що складається з $1 \times k$ векторів:

$$\beta_{1,k} = \frac{1}{n^2} \sum_{i-1}^{n} \sum_{j=1}^{n} m_{ij}^3 , \tag{2.1}$$

$$\beta_{2,k} = \frac{1}{n} \sum_{i=1}^{n} m_{ii}^2 , \tag{2.2}$$

де:

$$m_{ij} = (X_i - \bar{X})^T S^{-1}(X_j - \bar{X}) ,$$

$$S = \frac{1}{n}\sum_{j=1}^{n}(X_j - \bar{X})(X_j - \bar{X})^{\mathrm{T}} .$$

Фактично використовуються вибіркові значення асиметрії та ексцесу, які отримуються шляхом множення значення багатовимірних асиметрії і ексцесу, описаних вище, на $(n/(n-1))^3$ та $(n/(n-1))^2$ відповідно.

Якщо вибірка походить із багатовимірного нормального розподілу, тоді:

$$\frac{\mathrm{n}}{6}\,\beta_{1,k} \sim X^2(df) , \tag{2.3}$$

$$\left[\beta_{2,k} - \ k(k+2)\right]\sqrt{\frac{n}{8k(k+2)}} \sim N(0,1) ,$$

де:

*a* – рівень значущості;

*k* – розмірність

*df* - параметр можна розрахувати за формулою:

$$df = \frac{k(k+1)(k+2)}{6} .$$

Критичне значення для багатовимірної асиметрії може бути отримане з формули (2.3)

$$\beta_{1,critical} = X^2{}_{a,df} . \tag{2.4}$$

Для багатовимірного ексцесу існує 2 кордони, нижній та верхній які розраховуються за наступними формулами [12]:

$$\beta_{2,lower} = E[\beta_2] - z_a * \sigma[\beta_2], \quad (2.5)$$

$$\beta_{2,upper} = E[\beta_2] + z_a * \sigma[\beta_2], \quad (2.6)$$

де:

$E[\beta_2]$ = k(k+2),

$\sigma[\beta_2] = \sqrt{\frac{8k(k+2)}{n}}$ ,

$z_a = \Phi^{-1}(1 - a/2)$ .

Практично доведено, якщо дані мають багатовимірний нормальний розподіл, то розподіл квадрату відстані Махаланобіса поводиться як розподіл $\chi^2$. Для великих вибірок даних, відстань $d_i^2$ повинна вести себе приблизно як незалежні випадкові величини $\chi^2_{m,a}$. Тут $\chi^2_{m,a}$ – це квантиль розподілу $\chi^2$ з рівнем значущості *a*. Якщо значення величини квадрату відстані Махаланобіса для певної пари даних більше за квантиль розподілу $\chi^2$, то такі значення розглядаються як викиди і відкидаються. Цей процес повторюється доки всі значення квадрату Махаланобіса не будуть меншими чи дорівнювати квантилю розподілу $\chi^2$.

Значення квадрату відстані Махаланобіса для кожної багатовимірної точки даних *i, i =1,2…,N,* визначається як:

$$d_i^2 = (Z - \bar{Z})^T S_N^{-1} (Z - \bar{Z}) . \quad (2.7)$$

де: $\bar{Z}$ – вектор вибіркових середніх;

$S_N$ - вибіркова коваріаційна матриця:

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(Z_i - \bar{Z})(Z_i - \bar{Z})^T, \quad (2.8)$$

де: $Z_i$ – гаусівський випадковий вектор $Z = \{Z_1, Z_2, ..., Z_m\}^T$ для кожної багатовимірної точки даних $i$, $i = 1,2,..,N$;

$m$ – кількість компонентів вектору $Z$.

Вектор $Z$ отримують шляхом перетворення негаусівського випадкового вектору $X = \{X_1, X_2, ..., X_m\}^T$ за допомогою нормалізуючого перетворення десяткового логарифма

$$Z = \psi(X) , \tag{2.9}$$

$$Z_i = LG(X_i) . \tag{2.10}$$

Для десяткового логарифму формула оберненого перетворення матиме вигляд:

$$X_i = 10^{Zi}. \tag{2.11}$$

Рівняння еліпсу передбачення для нормалізованих даних визначається як [13]:

$$(Z - \bar{Z})^T S_N^{-1} (Z - \bar{Z}) = \frac{2(N^2-1)}{N(N-2)} F_{2, N-2, a}, \tag{2.12}$$

де $Z$ – це гаусівський випадковий вектор, $Z = \{Z_1, Z_2\}^T$;

$\bar{Z}$ – вектор вибіркових середніх, $\bar{Z} = \{\bar{Z}_1, \bar{Z}_2\}$;

$N$ – кількість точкових даних (в нашому випадку $N$=33);

$F_{2, N-2, a}$ – квантиль $F$ розподілу з 2 та $N$-2 ступенями вільності;

$a$ – рівень значущості;

$S_N$ – вибіркова коваріаційна матриця.

Ліва частина рівняння (2.12) є квадратом відстані Махаланобіса. Відповідно до [14], тестова статистика для квадрату відстані Махаланобіса $d_i^2$ може бути розрахована наступним чином:

$$\frac{N(N-2)d_i^2}{2(N^2-1)}\ , \tag{2.13}$$

яка приблизно має *F* розподіл з 2 та *N*-2 ступенями свободи.

## 2.2 Передобробка даних для побудови математичної моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity

Для побудови математичної моделі для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор, розроблених на рушії Unity, була використана вибірка двовимірних негаусівських даних, а саме значення метрики RFC на рівні всього застосунку, та значення метрики CBO на рівні всього застосунку з 50 проєктів 2D ігор, розроблених на рушії Unity (див. табл. 1.1).

Виконавши оцінку за критерієм Мардіа за формулами (2.1), (2.2), можна зробити висновок, що двовимірний розподіл даних з таблиці 1.1 не є гаусівським. Результати тесту представлено в таблиці 2.1. За рівня значущості $a$ = 0,005 розраховане значення асиметрії $\beta_{1,2}$ = 57,08, відповідно до формули (2.4) критичне значення $\beta_{1,2,0.005,50}$ = 14,86. Розраховане значення ексцесу $\beta_{2,2}$ = 21,52, розрахована за формулою (2.5) нижня критична межа для ексцесу $\beta_{2.2.0.005,50,lower}$ = 4,76, а верхня межа, розрахована за формулою (2.6), $\beta_{2.2.0.005,50,upper}$ = 11,24. Отримані значення асиметрії і ексцесу значно більші за критичні значення, й розподіл суттєво відрізняється від нормального. Тому для виявлення викидів у даних використовуємо метод, оснований на нормалізуючих перетвореннях і квадраті відстані Махаланобіса.

Таблиця 2.1 – Результати виконання тесту Мардіа для вихідного набору даних

| Величини | Значення |
|---|---|
| $b_{1,2}$ | 57,08 |
| $b_{2,2}$ | 64,04 |
| $b_{1,critical}$ | 14,86 |
| $b_{2,lower}$ | 4,76 |
| $b_{2,upper}$ | 11,24 |
| Normal | No |

Тому для подальшого аналізу треба виконати нормалізацію зібраних даних за допомогою десяткового логарифму за формулою(2.9) та видалити викиди за допомогою квадрату відстані Махаланобіса за формулою (2.7).

Нормалізовані дані метрик RFC та CBO та значення квадрату відстані Махаланобіса, представлено в таблиці 2.2

Таблиця 2.2 – Значення нормалізованих метрик RFC та CBO та квадрату відстані Махаланобіса

| № | LG(RFC) | LG(CBO) | $d_i^2$ | № | LG(RFC) | LG(CBO) | $d_i^2$ |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0,7844 | -0,5836 | 2,7818 | 26 | 0,8788 | -0,2478 | 1,5063 |
| 2 | 1,1409 | -0,3010 | 0,3803 | **27** | **1,4634** | **1,4634** | **24,4420** |
| 3 | 1,0132 | -0,3358 | 0,4671 | 28 | 1,3064 | 0,0348 | 0,4735 |
| 4 | 1,1736 | -0,0286 | 0,0820 | 29 | 0,7677 | -0,2430 | 3,2571 |
| 5 | 1,1644 | -0,0969 | 0,0069 | 30 | 0,9360 | -0,6214 | 2,1543 |
| 6 | 1,0614 | -0,2846 | 0,2412 | 31 | 1,2888 | 0,0146 | 0,3721 |
| 7 | 1,4771 | -0,0872 | 2,7813 | 32 | 1,3262 | 0,1600 | 0,8186 |
| 8 | 1,0719 | -0,2218 | 0,1249 | 33 | 1,3090 | -0,0314 | 0,5064 |
| 9 | 1,1206 | -0,0177 | 0,2326 | 34 | 1,2900 | -0,3010 | 1,5216 |
| 10 | 1,2163 | -0,1383 | 0,1531 | 35 | 0,9196 | -0,0263 | 2,0842 |
| 11 | 1,0940 | -0,3802 | 0,6299 | 36 | 1,2820 | 0,0580 | 0,3932 |
| 12 | 0,7924 | -0,6990 | 3,3461 | 37 | 0,7469 | -0,4771 | 3,0263 |
| 13 | 1,2216 | 0,1850 | 0,8883 | 38 | 1,1663 | 0,1829 | 1,1070 |
| 14 | 1,0604 | -0,2171 | 0,1507 | 39 | 1,3424 | 0,0378 | 0,7028 |
| 15 | 1,0719 | -0,3979 | 0,6787 | 40 | 1,1065 | -0,2553 | 0,1529 |
| **16** | **1,8808** | **0,3590** | **10,2872** | 41 | 1,6510 | 0,0726 | 5,4122 |
| 17 | 0,9461 | -0,3522 | 0,8107 | 42 | 1,2394 | 0,0272 | 0,2243 |
| 18 | 0,9379 | -0,3222 | 0,8380 | 43 | 0,9938 | -0,1727 | 0,5367 |
| 19 | 0,8391 | -0,1139 | 2,7562 | 44 | 1,5359 | -0,1532 | 4,4731 |
| 20 | 1,3526 | 0,0452 | 0,7785 | 45 | 1,2041 | 0,0669 | 0,3240 |
| 21 | 0,8921 | -0,3979 | 1,2753 | 46 | 1,2124 | -0,1597 | 0,1816 |

Продовження таблиці 2.2

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 22 | 1,2054 | 0,0307 | 0,2059 | 47 | 1,1983 | -0,1047 | 0,0530 |
| 23 | 1,1292 | 0,0774 | 0,6202 | 48 | 1,0592 | 0,0621 | 1,0366 |
| 24 | 1,1441 | 0,2711 | 2,0590 | 49 | 0,8285 | -0,8016 | 4,0635 |
| 25 | 1,3546 | -0,6021 | 6,3277 | 50 | 1,2377 | -0,1461 | 0,2732 |

Для рівня значимості 0,005 максимальне значення $\chi^2$ становить 10,6. Виходячи з цього, видаляємо набори з найбільшим значенням $d_i^2$, які більші за 10,6. На першій ітерації було знайдено й видалено проект №27, значення $d_i^2$ якого склало 24,442. На наступній ітерації було знайдено і видалено проект №16, значення $d_i^2$ якого на другому етапі склало 10,62. На третій ітерації набору даних з значенням $d_i^2$, більшим за максимальне значення $\chi^2$, знайдено не було, з чого можна зробити висновок, що викиди з вибірки успішно видалені.

Виконаємо тест Мардіа ще раз, для перевірки нормальності розподілу отриманого набору даних. Результат тесту Мардіа для нормалізованих і очищених від викидів даних наведено в таблиці 2.3. За наведеним результатом можна зробити висновок, що отриманий набір даних не відхиляється суттєво від двовимірного нормального розподілу. Розраховане значення асиметрії $\beta_{1,2}$ = 0,92, що менше за критичне значення $\beta_{1,2,0.005,48}$ = 14,86. Розраховане значення ексцесу $\beta_{2,2}$ = 9,39, отримане значення знаходиться в проміжку між нижнім і верхнім критичним значенням ексцесу [4,76; 11,24]. Отже, отримані значення асиметрії та ексцесу відповідають нормальному розподілу при рівні значущості $a$ = 0,005.

Таблиця 2.3 - Результат тесту Мардіа для нормалізованих і очищених від викидів даних

| Величини | Значення |
|---|---|
| $b_{1,2}$ | 0,92 |
| $b_{2,2}$ | 9,39 |
| $b_{1,critical}$ | 14,86 |
| $b_{2,lower}$ | 4,76 |
| $b_{2,upper}$ | 11,24 |
| Normal | Yes |

В подальшому данні, наведені в таблиці 2.2, використовуються для побудови рівняння еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор, розроблених на рушії Unity.

## 2.3 Побудова рівняння еліпсу передбачення та трансформованого еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity

Для побудови рівняння еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity в попередньому розділі ми нормалізували дані за допомогою десяткового логарифму та видалили викиди за допомогою квадрату Махаланобіса. Далі рівняння еліпсу передбачення для нормалізованих метрик RFC та CBO визначається за формулою (2.12).

Коваріаційна матриця $S_N$ має вигляд згідно з (2.8):

$$S_N = \begin{pmatrix} 0{,}0047 & 0{,}0308 \\ 0{,}0308 & 0{,}0613 \end{pmatrix}.$$

Для цієї матриці обернена матриця є наступною:

$$S_N^{-1} = \begin{pmatrix} 31{,}7496 & -15{,}9632 \\ -15{,}9632 & 24{,}3419 \end{pmatrix}.$$

Вектор випадкових середніх $\bar{Z}$ має вигляд:

$$\bar{Z} = \{1{,}1219;\ -0{,}1614\}.$$

Рівень значущості $a$ приймаємо на рівні 0,005.

Тестову статистику (2.13) було використано для побудови рівняння еліпса передбачення для нормалізованих даних метрик RFC і CBO. Побудований еліпс передбачення представлений на рисунку 2.1.

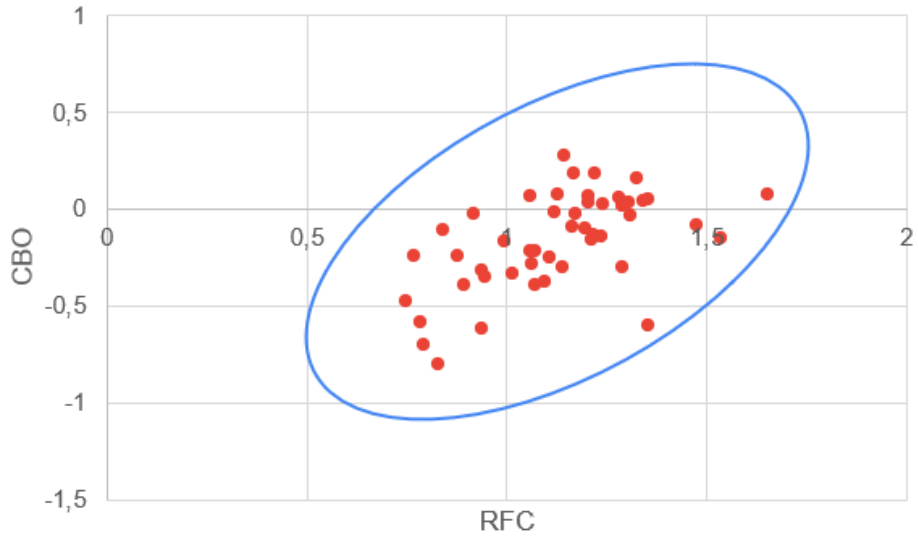


Рисунок 2.1 – Еліпс передбачення для нормалізованих даних метрик RFC та CBO

Відповідно до [15] трансформований еліпс передбачення може бути отриманий на основі побудованого еліпсу передбачення для нормалізованих даних з використанням оберненого перетворення за формулою (2.10).

В підрозділі 2.1 в якості нормалізуючого перетворення було використано десятковий логарифм, формула (2.10). Функція оберненого перетворення для натурального логарифму (2.11). Використаємо цю функцію до еліпсу передбачення для нормалізованих даних RFC та CBO, які ми отримали, та побудуємо трансформований еліпс передбачення для метрик RFC та CBO.

Трансформований еліпс передбачення для метрик RFC та CBO для оцінювання складності об'єктно-орієнтованого проетування 2D ігор на рушії Unity наведено на рисунку 2.2.

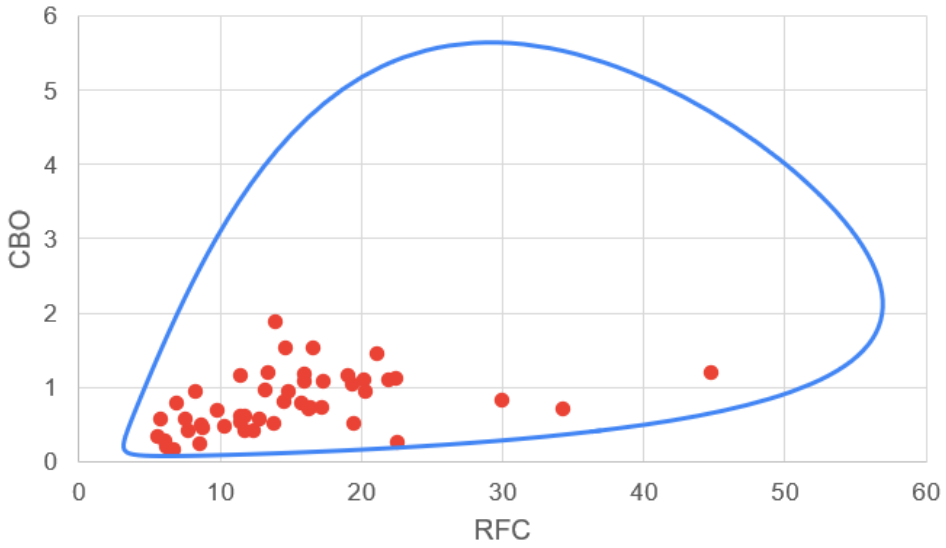


Рисунок 2.2 – Трансформований еліпс передбачення для метрик RFC та CBO

На рисунку 2.3 продемонстровано практичне використання отриманої моделі на прикладі 10 незалежних проектів які не брали участі у побудові даної моделі. Дані метрик проектів вказані в таблиці 2.4.

Таблиця 2.4 – Набір проектів для перевірки якості побудованої моделі

| Назва проєкту | RFC avg | CBO avg |
|---|---|---|
| 2D-TowerDefense | 16,125 | 0,75 |
| FantasyTrip | 10,2 | 0,4223 |
| **BulletRaid** | **18,5556** | **0,6667** |
| Faceless | 22,0108 | 0,6559 |
| **Unity-Wander-In-Color-Game** | **2,6866** | **0,1493** |
| Innerclash | 9,7872 | 0,6809 |
| Unfinished (business) | 12,2941 | 1,0294 |
| Ezra's Escape | 14,9091 | 1,0909 |
| Unity-2D-Game-Template | 18,8 | 0,0923 |
| Glitch-Garden | 9 | 0,3 |

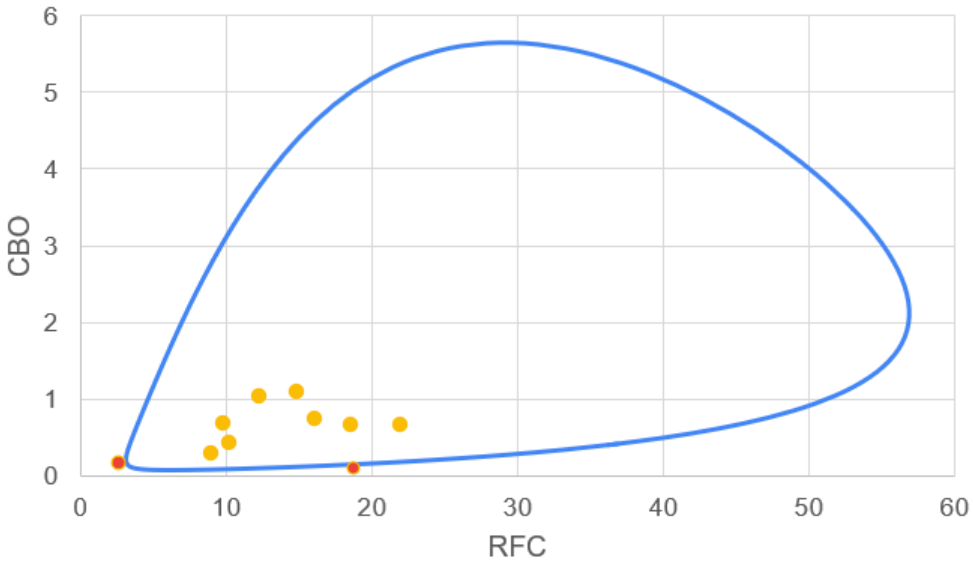


Рисунок 2.3 – Тестування моделі на даних реальних проектів

За результатами тестування можна зробити висновок що два проекти мають недосконалі архітектуру, а саме BulletRaid та Unity-Wander-In-Color-Game.

## 2.4 Висновки за розділом 2

У даному розділі виконано перевірку даних на наявність викидів для побудови математичної моделі для оцінювання складності ООП через зв’язки між класами 2D ігор на рушії Unity, здійснено побудову рівняння еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity для нормалізованих даних та рівняння трансформованого еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity.

1) Перевірку даних для побудови математичної моделі на наявність викидів виконано за допомогою нормалізуючого перетворення натуральним логарифмом і квадрату відстані Махаланобіса (SMD). За допомогою квадрату відстані Махаланобіса в нормалізованому наборі даних було знайдено і видалено 8 викидів. Застосування критерію Мардіа до отриманого набору даних показало, що зазначені дані мають гаусівський розподіл.

2) Було розраховано і побудовано рівняння еліпса передбачення для нормалізованих даних метрик RFC і CBO, яке було використано для побудови трансформованого еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity.

3) Удосконалено рівняння трансформованого еліпсу передбачення для значень метрик RFC та CBO для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity на основі нормалізуючого перетворення у формі десяткового логарифму із врахуванням викидів у даних. Зазначене рівняння у порівнянні з існуючими математичними моделями враховує як кореляцію між метриками RFC та CBO, так і те, що їх розподіл не є гаусівським.

4) Також було перевірено практичне використання моделі на прикладі 12 незалежних проектів, які не брали участі у побудові даної моделі.

5) Отримані результати свідчать про те, що для побудови рівняння трансформованого еліпсу передбачення для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity, потрібно використовувати нормалізуючі перетворення з урахуванням викидів у даних. Це дозволяє підвищити достовірність оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity.

# 3 РОЗРОБКА ПРОЕКТУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ СКЛАДНОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОЕКТУВАННЯ ЧЕРЕЗ ЗВ'ЯЗКИ МІЖ КЛАСАМИ 2D ІГОР НА РУШІЇ UNITY

В розділі розроблено проект програми для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity. Виконано постановку задачі на розробку програми, виконано ескізне, технічне та робоче проектування.

## 3.1 Постановка задачі на розробку проекту програми для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity

На основі аналізу існуючих моделей для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами, які було розглянуто в розділі 1.1, сформулюємо постановку задачі на розробку на розробку проекту програми. Потрібно розробити програму для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор, які розроблені на рушії Unity та відповідатиме наступним вимогам.

*Вимоги до функціональних характеристик програми.*

Програма має виконувати наступні функції:

1) Зчитувати з .xlsx файлу дані: Назву гри, середня значення метрики RFC для гри, середня значення CBO для гри;

2) Здійснювати побудову трансформованого еліпсу передбачення для метрик RFC та CBO;

3) Обчислювати значення квадрату відстані Махаланобіса для досліджуваних ігор;

4) Здійснювати оцінювання складності об'єктно-орієнтованого проектування гри через зв'язки між класами з довірчою ймовірністю 0,005;

5) Виводити на екран значення квадрату відстані Махаланобіса для метрик RFC та CBO, результати оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами з довірчою ймовірністю 0,005;

6) Зберігати в .xlsx файл результати оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами, значення квадрату відстані Махаланобіса для кожного проекту. Зберігати фото побудованого графіку.

*Вимоги до складу, параметрів та технічних засобів.*

Програма має використовуватися на обладнанні з наступними мінімальними параметрами: процесор intel core i3 (або вище), оперативна пам’ять обсягом 4 гб, вільний простір на жорсткому диску 400 Мб.

*Вимоги до інформаційної та програмної сумісності.*

Для роботи програми необхідна операційна система Windows 10 (або вище) із встановленим пакетом Microsoft Office.

На основі поставленої задачі було розроблено технічне завдання, яке наведено в додатку А.

## 3.2 Розробка ескізного проекту програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

В даному пункті було виявлено варіанти використання та розроблено їх сценарії, розроблено специфікацію та діаграму станів програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

У цьому проекті передбачається наявність лише одного актора – користувача, яким матиме змогу виконувати всі функції програми. Виявлені варіанти використання представлені в таблиці 3.1.

Діаграма варіантів використання [16] програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор зображена на рисунку 3.1. Сценарії варіантів використання виконані у вигляді діаграми діяльності [17], яка представлена на рисунку 3.2.

Таблиця 3.1 - Виявлені варіанти використання

| Актори | Варіант використання | Опис |
|---|---|---|
| Користувач | Обрати файл з даними | Користувач обирає файл з даними rfc та cbo для аналізу. |
| Користувач | Побудувати еліпс передбачення | Будує еліпс передбачення для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор та поміченими досліджуваними проектами на графіку. |
| Користувач | Оцінити складність об'єктно-орієнтованого проектування гри | Оцінює складність об'єктно-орієнтованого проектування гри через зв'язки між класами з ймовірністю 0.005 |
| Користувач | Зберегти результати | Зберігає результати оцінювання складності складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор та рисунок отриманого графіку. |
| Користувач | Обрати файл для збереження результатів | Користувач обирає файл в який будуть збережені файл, або вказує ім'я нового файлу куди будуть збережені результати |
| Користувач | Закрити програму | Закриває всі віконні форми і фонові процеси програми |

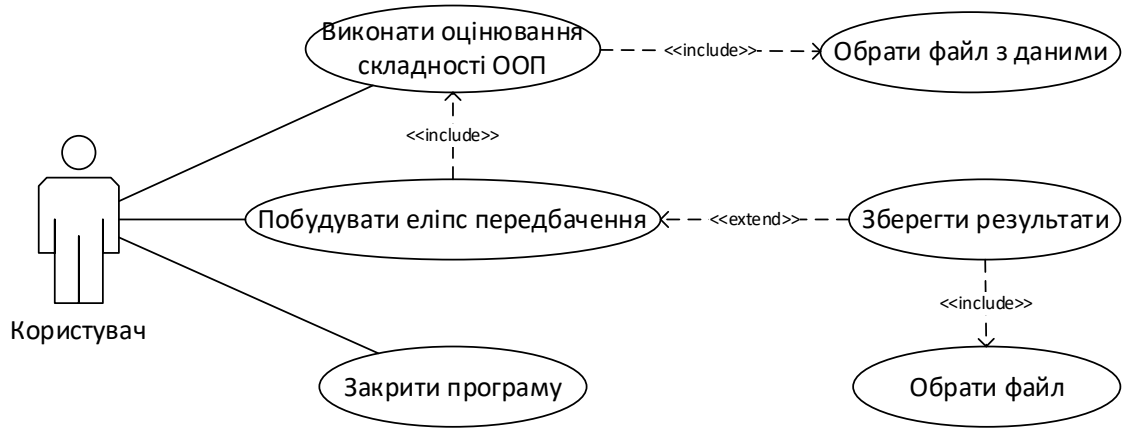


Рисунок 3.1 – Діаграма варіантів використання програми для оцінювання складності ООП через зв'язки між 2D ігор на рушії Unity

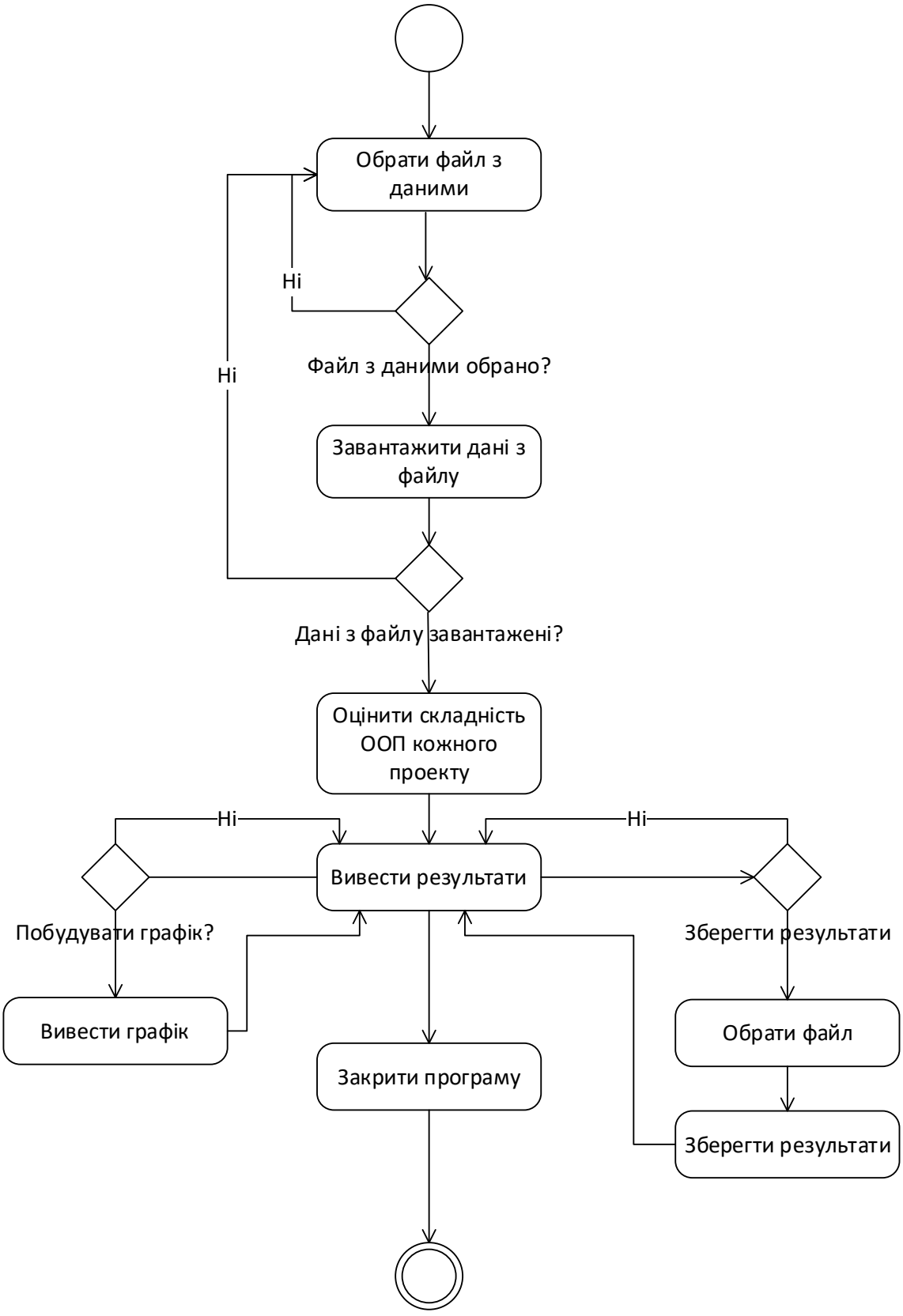


Рисунок 3.2 – Діаграма діяльності ПЗ для оцінювання складності ООП через зв’язки між класами комп’ютерних ігор

Зовнішня специфікація програми.

*Назва програми*: Unity 2D Game OOD Complexity Estimator.

*Функції, які виконує програма:*

1) Завантажує з файлу дані метрик RFC та CBO ігрових проектів;

2) Оцінює складність об’єктно-орієнтованого проектування через зв’язки між класами для кожного проекту;

3) Виводить результати оцінювання об’єктно-орієнтованого проектування через зв’язки між класами для кожного проекту;

4) Будує графік трансформованого еліпсу передбачення для оцінюваних даних;

5) Зберігає результати оцінювання у вказаний файл;

*Вхідні дані:*

- Назва проекту, середнє значення метрики RFC, середнє значення метрики CBO;
- Файл з даними математичної моделі;

*Вихідні значення:*

- Файл з результатами оцінювання складності об’єктно-орієнтованого проектування;
- .png файл, який містить графік еліпсу передбачення і результатів оцінювання;

*Зовнішні ефекти:* поява графічного вікна для обрання файлу з даними, поява графічного повідомлення про успішне або невдале виконання операції, поява графічного вікна з зображенням результатів оцінювання у вигляді еліпсу передбачення, вихід з програми (закриття програми і всіх її вікон).

Для проектування інтерфейсу програми, яка оцінює складність об'єктно-орієнтованого програмування через зв'язки між класами 2D-ігор на рушії Unity, представимо стани програми у вигляді діаграми станів. Діаграма станів для

оцінювання складності об'єктно-орієнтованого проектування через зв’язки між класами наведена на рисунку 3.3.

На діаграмі станів [18] визначені такі стани програми: 0 – початковий стан, 1 – графічне вікно головного модулю, 2 – графічна форма обрання файлу, 3 – графічне вікно із результатами оцінювання складності об’єктно-орієнтованого проектування 2D ігор на рушії Unity, 4 – графічне вікно з результатами побудови трансформованого еліпсу передбачення, 5 – кінцевий стан.

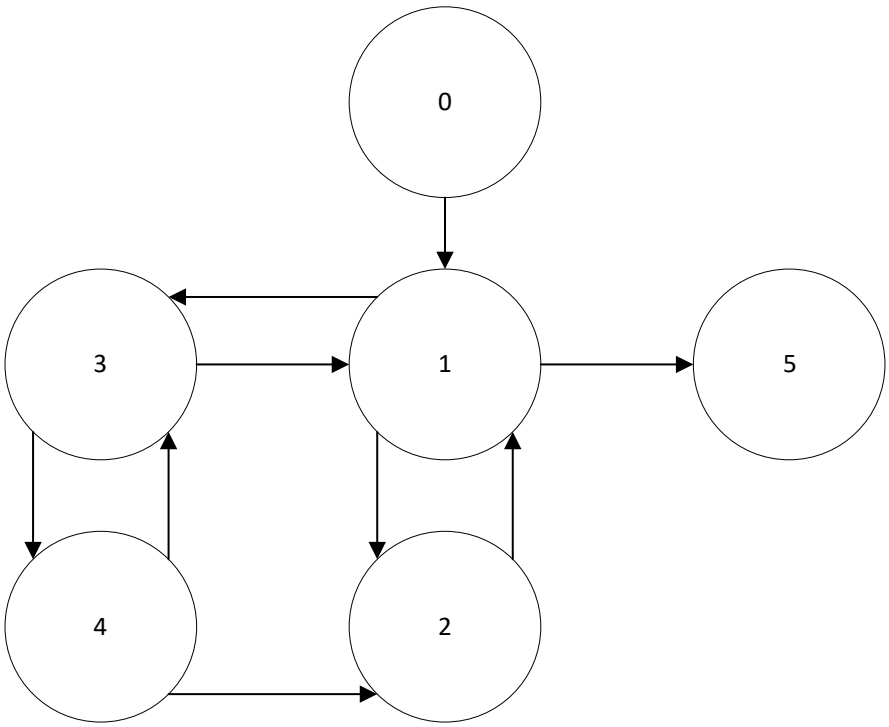


Рисунок 3.3 – Діаграма станів програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity

Користувацький інтерфейс є важливою частиною програмного забезпечення. Від нього залежить ефективність роботи працівника й точність управління машиною з сторони людини.

На прототипі можливо тестувати сприйняття системи користувачем та її основну логіку. Користь паперової версії прототипування у виключній простоті модифікації за результатами тестування й в можливості безболісно знаходити представників цільової аудиторії. На паперовому етапі прототип може пережити

багато виправлень і, відповідно, багато версій [19]. Прототипи вікон програми приведено на рисунках 3.4 – 3.7.

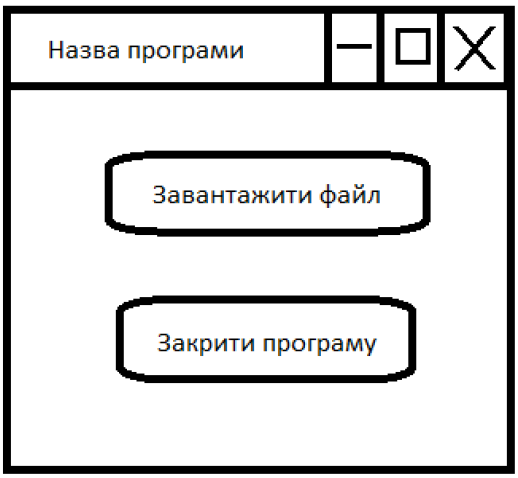


Рисунок 3.4 - Прототип стартового вікна програми

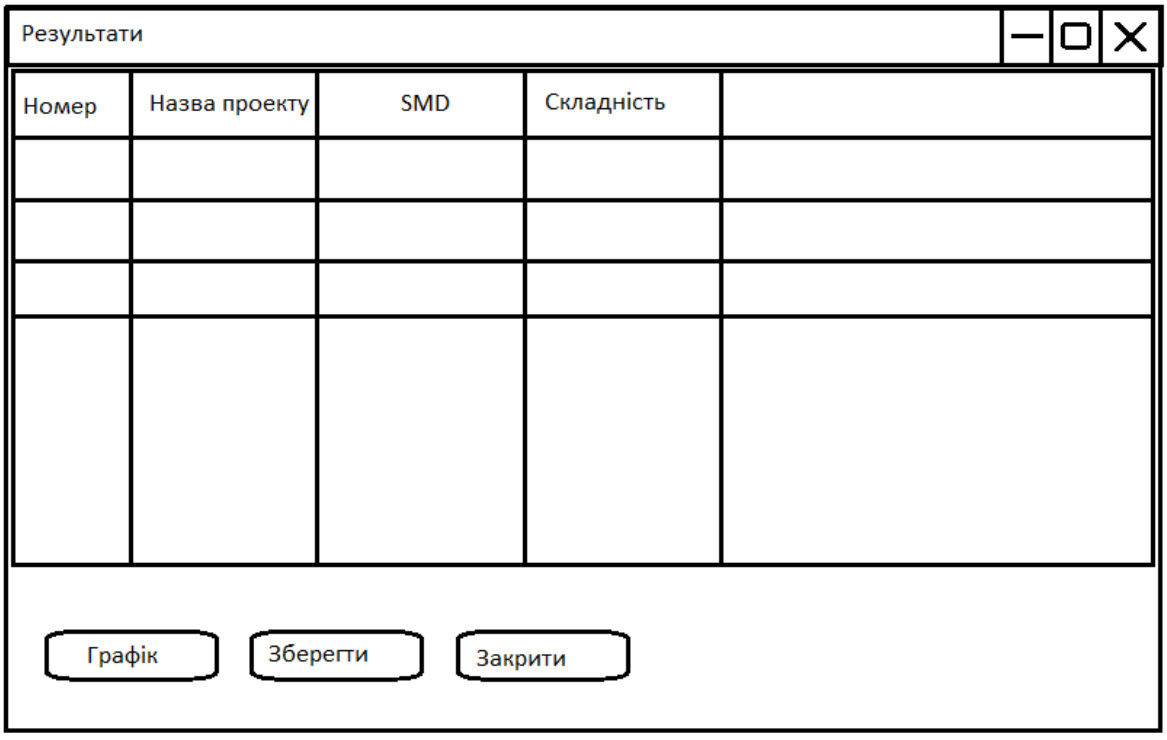


Рисунок 3.5 – Прототип віконної форми з результатами розрахунку складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

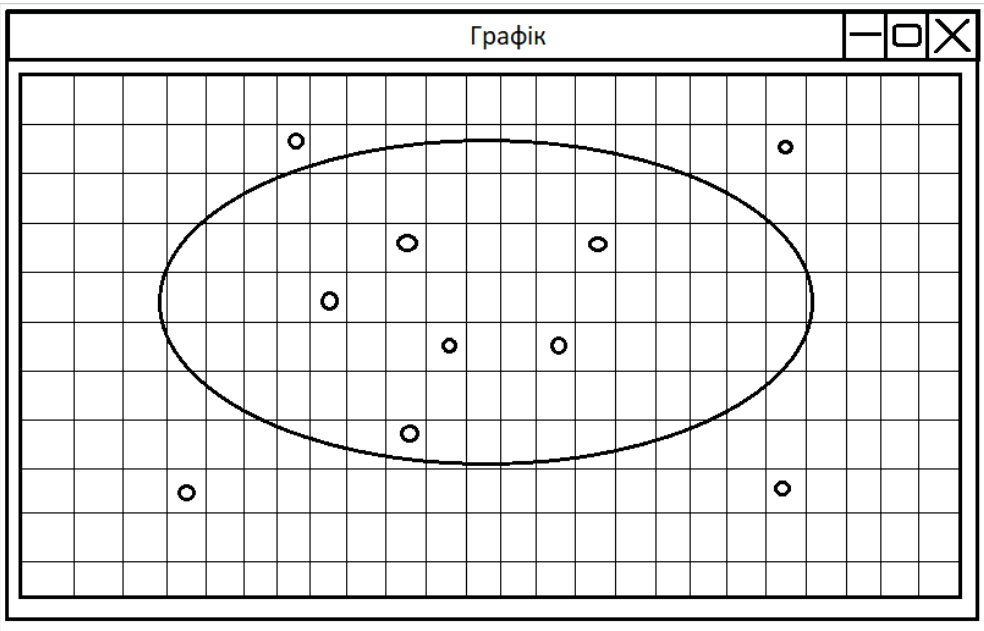


Рисунок 3.7 – Прототип вікна з графічними результатами оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

Підберемо стилі для програмного інтерфейсу на прикладі віконної форми з графічним інтерфейсом (див. рис. 3.8).

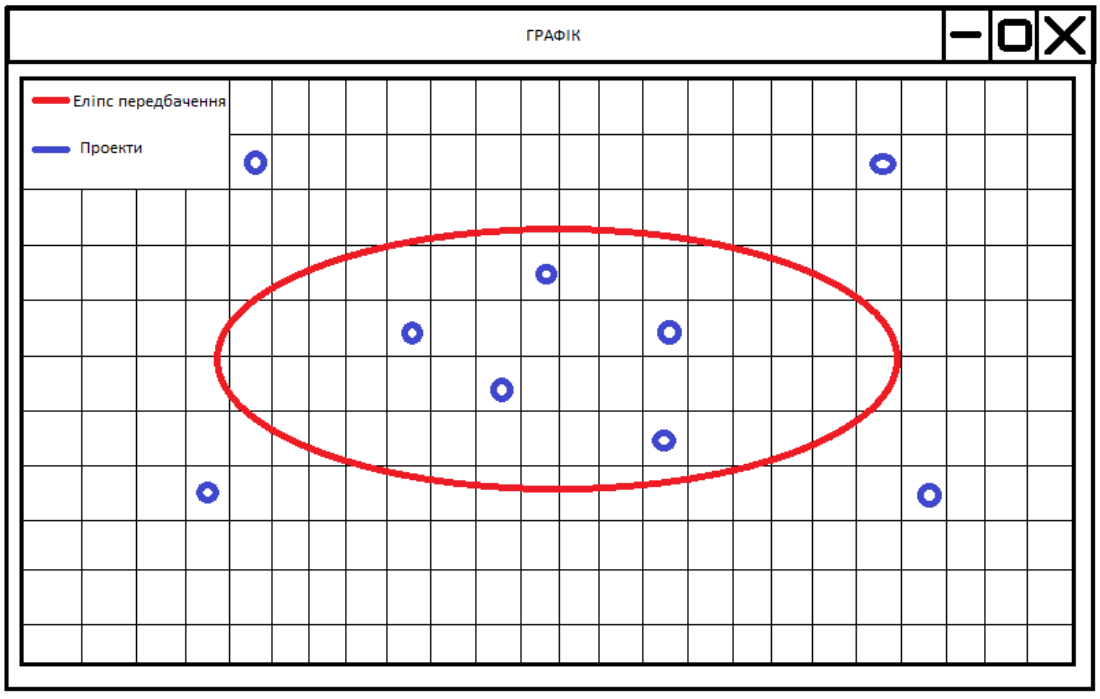


Рисунок 3.8 – Підбір стилю для віконної форми з графічним результатами оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

Приклад фінального варіанту стилю оформлення вікон програми наведено на рисунку 3.9

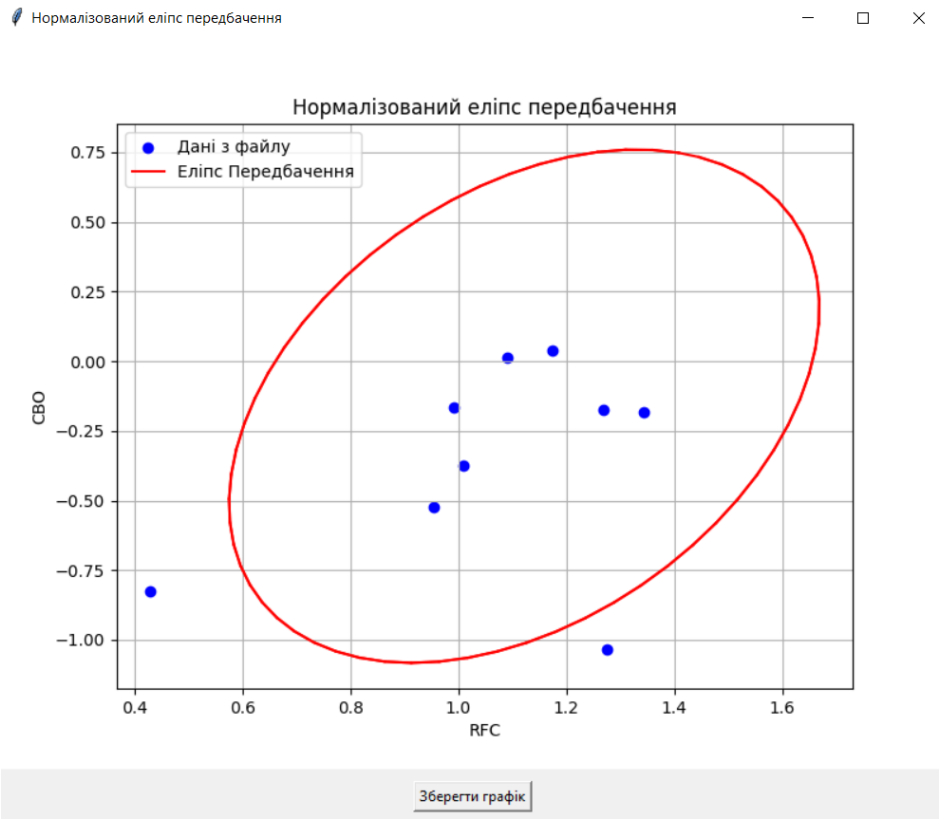


Рисунок 3.9 – Фінальний варіант стилю оформлення віконної форми з графічним результатами оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор

В цьому розділі було розроблено ескізний проект програми, включаючи специфікацію, діаграму станів і варіанти використання. Користувач має змогу виконувати всі основні функції програми, такі як вибір файлів, оцінка складності та збереження результатів. Надалі було здійснено розробку технічного проекту програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity.

## 3.3 Розробка технічного проекту програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D на рушії Unity

В межах технічного проекту було спроектовано програму для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity.

Програма складається з наступних модулів: Application - головний модуль, який відповідає за запуск програми та організацію користувацького інтерфейсу; DataLoader - модуль для завантаження даних з Excel-файлів; SMDCalculator - модуль для математичних обчислень, пов'язаних із оцінюванням складності об’єктно-орієнтованого проектування; PlotBuilder – модуль для побудови графіків; ResultsManager - модуль для відображенням та збереженням результатів.

*Специфікація модулів*

*Назва модуля:* Application

*Функції модуля*:

1) Ініціює головне вікно програми;

2) Обробляє вибраний користувачем файл;

3) Завантажує дані моделей через модуль DataLoader;

4) Викликає обчислення складності об’єктно-орієнтованого проектування через модуль SMDCalculator;

5) Відображає результати через модуль ResultManager;

*Вхідні дані:*

- Файл .xlsx з значеннями метрик RFC та CBO для аналізу;

*Вихідні дані:*

- значення квадрату відстані Махаланобіса для нормалізованих значень метрик RFC та CBO;

- висновок по оцінюванню складності архітектури проекту.

*Зовнішні ефекти*: поява графічного вікна для обрання файлу з даними, поява графічного вікна з результатами оцінювання складності об’єктно-орієнтованого проектування, вихід з програми (закриття всіх віконних форм програми).

*Назва модуля:* DataLoader

*Функції модуля*:

1) завантажує в програму характеристики математичної моделі еліпсу передбачення;

2) завантажує в програму дані метрик RFC та CBO проектів для аналізу.

*Вхідні дані:*

- файл .xlsx з значеннями метрик RFC та CBO для аналізу;

Вихідні дані:

- імена проектів, значення RFC та CBO досліджуваних проектів;

- дані для побудови нормалізованого еліпсу передбачення;

- дані для побудови трансформованого еліпсу передбачення.

*Зовнішні ефекти*: Графічне повідомлення у випадку помилки завантаження даних з файлів.

*Назва модуля:* SMDCalculator

*Функції модуля*:

1) Оцінювання складності об’єктно-орієнтованого проектування проекта;

2) Обчислення значення квадрату відстані Махаланобіса для кожного проекту;

*Вхідні дані:*

- Досліджувані значення метрик RFC та CBO;

- Нормалізовані значення метрик RFC та CBO моделі;

Вихідні дані:

- імена проекту;

- номер проекту;

- значення квадрату відстані Махаланобіса;

- результат оцінювання складності архітектури проекту;

*Назва модуля:* PlotBuilder

*Функції модуля*:

1) Будує графік моделі еліпсу передбачення;

2) Будує графік точок досліджуваних проектів за значенням метрик RFC та CBO;

3) Надає можливість зберегти отриманий графік у вигляді файлу .png.

*Вхідні дані:*

- координати моделі еліпсу передбачення;
- дані метрик RFC та CBO досліджуваних проектів;

Вихідні дані:

- файл формату .png.

*Зовнішні ефекти*: Графічне вікно з побудованим еліпсом передбачення та даними метрик RFC та CBO.

*Назва модуля:* ResultsManager

*Функції модуля*:

1) Відображає результати розрахунків у графічному вікні;

2) Зберігає результати розрахунків у вигляду файлу форму .xlsx.

*Вхідні дані:*

- номер проекту;
- назва проекту;
- значення квадрату відстані Махаланобіса для проекту;
- результат оцінювання складності об’єктно-орієнтованого проектування.

Вихідні дані:

- файл з результатами формату .xlsx.

*Зовнішні ефекти*: Графічне вікно з таблицею результатів аналізу проектів.

## 3.4 Розробка робочого проекту програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity

В робочому проекті розроблено програму для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор розроблених на рушії Unity, здійснено тестування програми, розроблено програмну документацію, здійснено випробування програми.

Програма написана мовою програмування Python [20] з використанням бібліотек scipy [21], numpy [22], matplotlib [23], openpyxl [24], tkinter [25]. Текст програми наведений в додатку Б. Приклад кодування модулю SMDCalculator, що виконує оцінювання складності об’єктно-орієнтованого проектування та розраховує значення квадрату відстані Махаланобіса наведено нижче.

```
class SMDCalculator:
    @staticmethod
    def is_inside_ellipse(x, y, elips_rfc, elips_cbo):
        points = np.column_stack((elips_rfc, elips_cbo))
        hull = ConvexHull(points)
        new_point = np.array([x, y])
        return all(
            np.dot(eq[:-1], new_point) + eq[-1] <= 0
            for eq in hull.equations
        )

    @staticmethod
    def calculate_complexity_and_smd(names, rfc, cbo, normalized_rfc,
normalized_cbo, elips_rfc, elips_cbo):
        results = []

        rfc_mean = np.mean(normalized_rfc)
        cbo_mean = np.mean(normalized_cbo)
        rfc_std = np.std(normalized_rfc)
        cbo_std = np.std(normalized_cbo)

        for i in range(len(names)):
            normalized_rfc_i = np.log10(rfc[i]) if rfc[i] > 0 else 0
            normalized_cbo_i = np.log10(cbo[i]) if cbo[i] > 0 else 0

            smd = ((normalized_rfc_i - rfc_mean) / rfc_std) ** 2 +
((normalized_cbo_i - cbo_mean) / cbo_std) ** 2
            difficulty = "Простий" if SMDCalculator.is_inside_ellipse(rfc[i],
```

```
cbo[i], elips_rfc, elips_cbo) else "Складний"
            results.append((i + 1, names[i], smd, difficulty))
        return results
```

В програмі були використані такі інтерфейсні компоненти: графічні вікна, кнопки та контекстні меню. Приклад віконної форми програми наведено на рис.3.4.

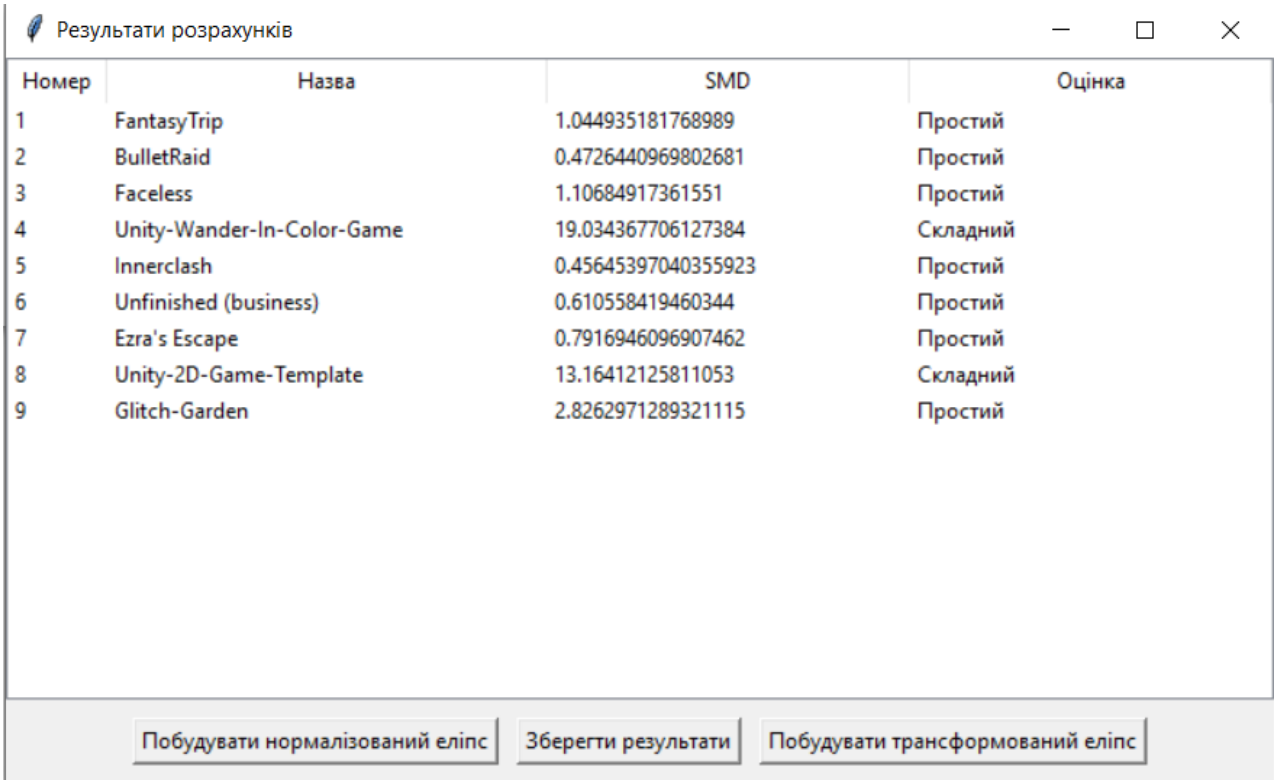


Рисунок 3.4 – Віконна форма з результатами оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity

Було виконано тестування окремих модулів. У якості прикладу наведемо результати тестування модулю PlotBuilder. Було виконано побуду рівняння еліпсу передбачення на основі нормалізуючого перетворення у формі десяткового логарифму. Результати візуалізації емпіричних даних з метрик CBO та RFC 10 комп’ютерних ігор наведені на рис.3.5.

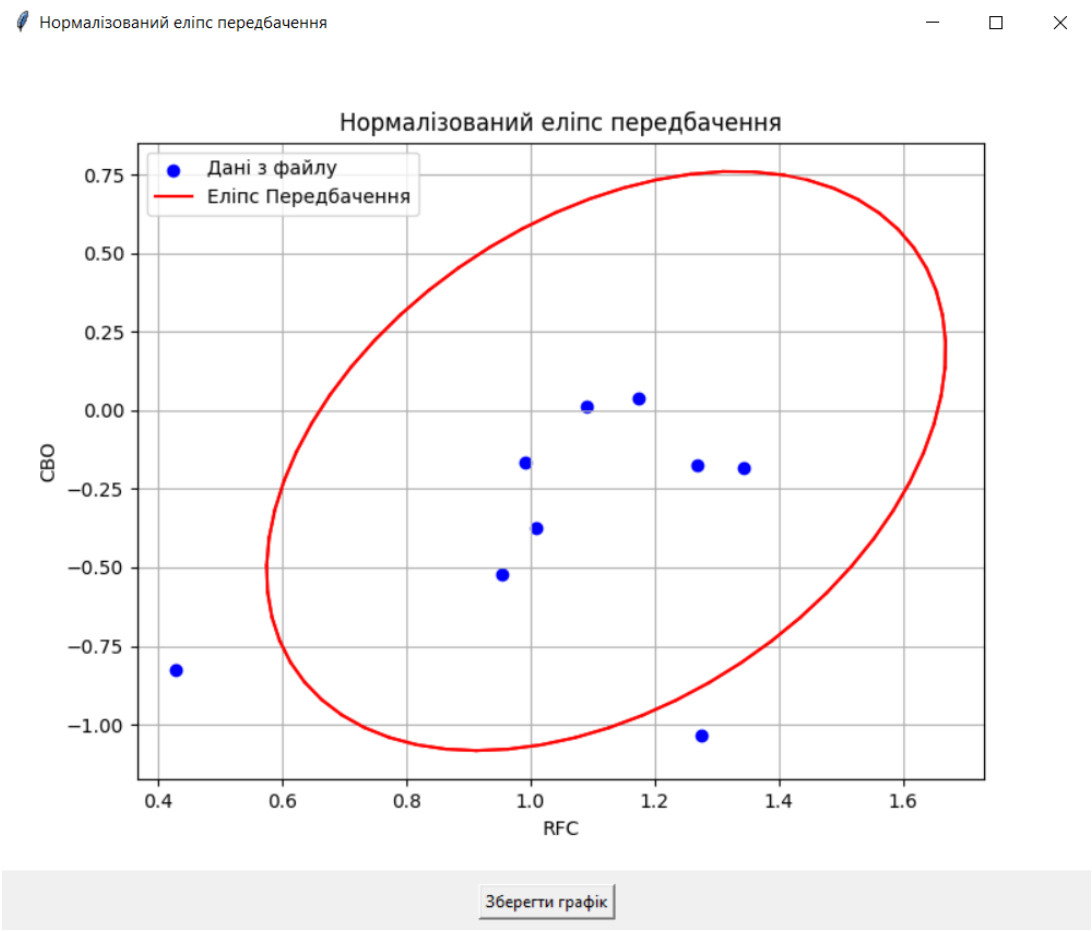


Рисунок 3.5 – Віконна форма з графічним зображенням отриманих результатів оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity в вигляді нормалізованого еліпсу передбачення

Як зазначалося раніше проекти, для яких значення нормалізованих метрик RFC та CBO, що використовуються як координати точки, виходять за межі еліпсу передбачення, вважаються такими, що мають підвищену складність об’єктно-орієнтованого проектування через зв’язки між класами порівняно з іншими іграми, на основі яких побудовано цей еліпс. І навпаки, якщо координати точки залишаються всередині еліпсу передбачення, гра вважається такою, що має просту складність об’єктно-орієнтованого проектування через зв’язки між класами порівняно із зазначеними іграми.

Як ми бачимо на рисунку 3.5 два досліджуваних проекти мають підвищену складність проектування, а саме Unity-2D-Game-Template та Unity-Wander-In-

Color-Game. Ці проекти знаходяться поза кордонами еліпсу передбачення й значення квадрату відстані Махаланобіса для цих проектів більше критичного значення 10,6.

Було виконано побуду рівняння трансформованого еліпсу передбачення. Результати візуалізації емпіричних даних з метрик CBO та RFC 10 комп'ютерних ігор наведені на рис.3.6.

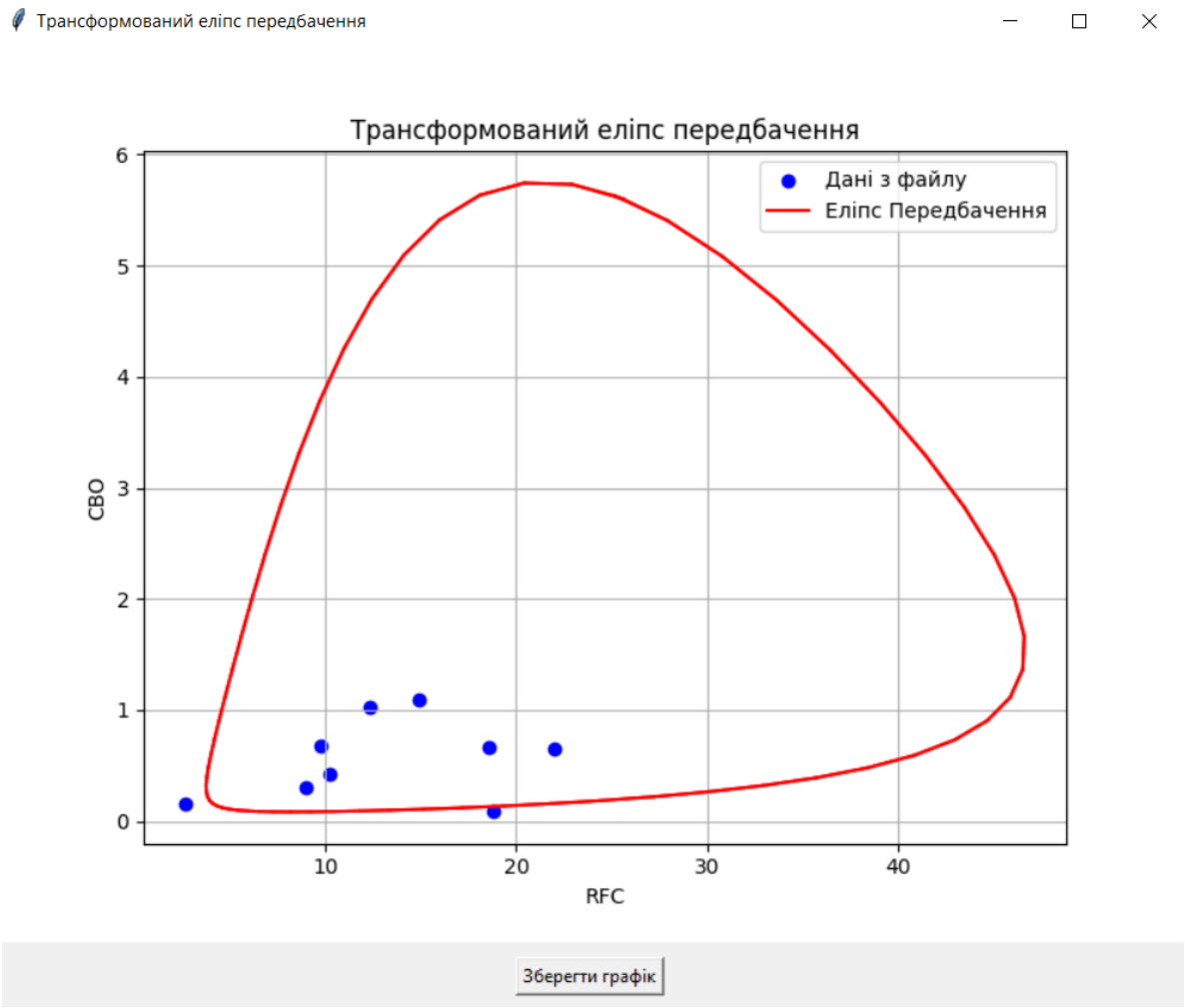


Рисунок 3.6 - Віконна форма з графічним зображенням отриманих результатів оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity в вигляді трансформованого еліпсу передбачення

На рисунку 3.6 було отримано аналогічний результат. Два досліджуваних проекти мають підвищену складність проектування, а саме Unity-2D-Game-Template та Unity-Wander-In-Color-Game.

Отримані результати практично збігаються з результатами, що наведені в таблиці 2.5. Це свідчить про працездатність розробленої програми.

## 3.5 Висновки за розділом 3

У даному розділі виконано постановку задачі на розробку проекту програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity, розроблено ескізний, технічний та робочий проекти програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity.

1) В ескізному проекті було виявлено варіанти використання та побудовано їх сценарії, розроблено зовнішню специфікацію програми для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity та стани цієї програми при виконанні.

2) В технічному проекті було спроектовано програму для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity на основі покрокової деталізації зверху вниз, розроблено специфікації модулів програми, побудовано алгоритми модулів програми.

3) В робочому проекті було розроблено програму для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity, здійснено тестування програми, розроблено програмну документацію, здійснено випробування програми. При випробуванні програми також застосовувалися дані метрик 10 2D ігор розроблених на рушії Unity, які представлені в таблиці 2.5. Результати випробовування програми практично збігаються з результатами отриманими на етапі аналізу. Це свідчить про працездатність розробленої програми.

# 4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМИ

Найважливішим аспектом під час проєктування комп'ютерної програми для розробника, з точки зору економіки, є процес визначення її вартості. Очевидно, що комп'ютерна програма є специфічним продуктом із низкою унікальних характеристик.

Розробка програми потребує одноразових витрат на її створення, придбання необхідного обладнання, а також регулярних витрат, пов'язаних з її експлуатацією. Економічний ефект від використання програми розраховується з урахуванням витрат на її функціонування. Співвідношення отриманого економічного ефекту до витрат на розробку програми визначає економічну ефективність капіталовкладень.

Розрахунок економічних показників здійснюється з урахуванням чинних на момент обчислення оптових цін, тарифів та ставок заробітної плати.

## 4.1 Розрахунок витрат на створення та впровадження програмного забезпечення

Витрати на розробку програмного забезпечення складаються з витрат на заробітну плату розробника, на амортизацію та експлуатацію ЕОМ, на якій виконується розробка, на засоби розробки та витрати на матеріальні комплектуючі [26]. Розробка програмного забезпечення виконується спеціалістом, місячний оклад якого складає 18000 грн. Додаткова заробітна плата складає 10% від основної. Виходячи з цього, основна і додаткова заробітна плата розробника системи 19800 грн./міс, а вартість сучасної ПЕОМ складає 13000 грн. Вартість кіловат-години електроенергії дорівнює 1,68 грн. Витрати на допоміжні матеріали наведені в таблиці 4.1.

Таблиця 4.1 - Витрати на допоміжні матеріали

| Пункти витрат | Сума, грн. |
|---|---|
| Папір | 120 |
| Заправлення картриджа до принтера | 300 |
| Непередбачені витрати | 200 |
| Разом матеріали і комплектуючі вироби | 620 |

Вартість розробки системи розраховується за формулою [27]:

$$C_{пр} = (З_{ЗП} + З_{СЗ} + З_{ЗГ} + З_{е}) * Т + З_{М'} \quad (4.1)$$

де:

$Т$ – тривалість розробки, міс.;

$З_{ЗП}$ - основна і додаткова заробітна плата обслуговуючого персоналу, грн.;

$З_{СЗ}$ – відрахування на соціальні заходи ( 38% від основної і додаткової заробітної плати), грн.;

$З_{ЗГ}$ – загальногосподарські витрати (10% від заробітної плати), грн.;

$З_{е}$ – витрати на електроенергію;

$З_{М}$ – витрати на основні і допоміжні матеріали.

Розрахуємо $З_{е}$ при споживанні потужності 0,3 Вт, тривалості роботи на місяць рівної 21* 8 = 168 годин і вартості кіловат-години 1,44 грн. до 250 кВт включно та 1,68 після 250 кВт, отримаємо:

$$З_{е} = 168 * 1,44 * 0,3 = 72,58 \text{ грн.}$$

Представимо всі поточні витрати на розробку програмного забезпечення в таблиці 4.2

Таблиця 4.2 - Всі поточні витрати на розробку програмного забезпечення

| Найменування витрат | Одиниця виміру | Кількість |
|---|---|---|
| Тривалість розробки | міс. | 3 |
| Основна і додаткова заробітні плати | грн. | 19800 |
| Відрахування та соціальні заходи | грн. | 7524 |
| Загальногосподарські витрати | грн. | 1980 |
| Витрати на допоміжні матеріали | грн. | 620 |
| Витрати на електроенергію | грн. | 72,58 |

Відповідно до формули (4.1) вартість розробки ПЗ складає:

$C_{пр}$ = (19800 + 7524 + 1980 + 72,58) * 3 + 620 = 88749,74 грн.

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$A_{об} = 1300 * 0,6 = 7800 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали визначаються в розмірі 5% вартості основного устаткування:

$$B_M = 13000 * 0,05 = 650 \text{ грн.}$$

Річний обсяг робіт ПЕОМ у годинах визначається в такий спосіб:

$$Ф_M = 253,3 * T_З,$$

де $T_З$ – це середнє місячне завантаження устаткування (близько 7 годин), 253,3 – середня кількість робочих днів у році:

$$Ф_M = 253,3 * 7 = 1773,1 \text{ год.}$$

Витрати на електроенергію $З_е$ при 1773,1 годинах роботи устаткування в рік складуть:

$$З_е = 1773,1 * 0,3 * 1,44 = 765,98 \text{ грн.}$$

Експлуатаційні витрати для ПЕОМ за рік складуть:

$$З_{ЗР} = 7800 + 650 + 765,98 = 9215,98 \text{ грн.}$$

В перший рік витрати на створення й експлуатацію програмного забезпечення складуть:

$$З_{СЕ} = 88749,74 + 9215,98 = 97965,72 \text{ грн.}$$

## 4.2 Розрахунок економічної ефективності розробки

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного забезпечення вивільняє одного працівника(за експертною оцінкою фахівців підприємства).

Зарплата одного працівника в рік складає:

$$18000 * 12 * 1 = 216000 \text{грн.}$$

Річний економічний ефект розраховується по формулі:

$$Э_{рік} = \Delta C_n - E_n \cdot k, \quad (5.2)$$

де $\Delta C_n$ – вивільнені кошти після впровадження системи (216000 грн.) мінус експлуатаційні витрати (9215,98 грн.);

$E_n$ – коефіцієнт ефективності (дорівнює коефіцієнту амортизації (0,6));

$k$ – одноразові витрати на впровадження продукту (97965,72).

$$Э_{рік} = 216000 - 9215,98 - 97965,72 * 0,6 = 148004,59 \text{грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = 97965{,}72\,\frac{97965{,}72}{216000 - 9215{,}98} \approx 0{,}47.$$

Отже строк окупності програмного забезпечення складає 5,6 місяців.

У цьому розділі були здійснені розрахунки на створення й експлуатацію програмного забезпечення, а також розрахунки економічної ефективності та строку окупності програмного забезпечення. Виходячи з розрахунків, впровадження такого програмного забезпечення окупить себе протягом першого року експлуатації за 5,6 місяців. Отже, можна зробити висновок, що дане програмне забезпечення економічно вигідне та обґрунтоване.

# 5 ОХОРОНА ПРАЦІ

## 5.1 Основні положення Закону України «Про охорону праці»

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Охорона праці містить три основних складових частини: правові норми трудового законодавства, виробничу санітарію, гігієну та техніку безпеки, а також протипожежний захист і електробезпеку .

Мета охорони праці – забезпечення безпечних, нешкідливих і сприятливих умов праці через вирішення багатьох складних завдань, основними з яких є:

- проектування підприємств, технологічних процесів і конструювання обладнання з обов'язковим виконанням вимог охорони праці;

- знаходження оптимальних співвідношень між різними факторами виробничого середовища, що дозволяє забезпечити мінімум несприятливого впливу їх на здоров'я працівників;

- встановлення, законодавче оформлення визначених норм кожного з несприятливих або небезпечних факторів, систематичний король за їх застосуванням;

- розробка конкретних заходів щодо покращення умов праці та забезпечення її безпеки на основі застосування у виробництві новітніх досягнень науки і техніки;

- застосування раціональних засобів захисту працівників від впливу несприятливих факторів виробничого середовища, а також втілення організаційних заходів, які нейтралізують або послаблюють ступінь їх впливу на організм людини;

- розробка та застосування методів і засобів оцінки ефективності заходів з охорони праці, що плануються і здійснюються .

Законодавство про працю містить норми і вимоги з техніки безпеки і виробничої санітарії, норми, що регулюють робочий час і час відпочинку, звільнення та переведення на іншу роботу, норми праці щодо жінок, молоді, гігієнічні норми і правила тощо.

Правовою основою законодавства щодо охорони праці є Конституція України, Закони України: «Про охорону праці» [28], «Про охорону здоров'я»[29], «Про пожежну безпеку», «Про використання ядерної енергії та радіаційний захист», «Про забезпечення санітарного та епідеміологічного благополуччя населення», а також Кодекс законів про працю України.

Основоположним законодавчим документом в галузі охорони праці є Закон України «Про охорону праці», дія якого поширюється на всі підприємства, установи і організації незалежно від форм власності та видів їх діяльності, на усіх громадян, які працюють, а також залучені до Праці на цих підприємствах.

Верховна Рада України 14 жовтня 1992 року прийняла Закон України «Про охорону праці». Цей Закон визначає основні положення щодо реалізації конституційного права громадян про охорону їх життя і здоров'я в процесі трудової діяльності, регулює за участю відповідних державних органів відносини між власником підприємства, установи і організації або уповноваженим ним органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

## 5.2 Аналіз шкідливих та небезпечних факторів на робочому місці

Шкідливі виробничі фактори – фактори середовища і трудового процесу, які можуть викликати професійну патологію, тимчасове або стійке зниження працездатності. В таблиці 5.1 перераховані шкідливих та небезпечних виробничих факторів.

Таблиця 5.1 – Перелік шкідливих та небезпечних виробничих факторів

| Найменування факторів | Можливі джерела їх виникнення | Характер дії |
|---|---|---|
| Небезпека ураження електричним струмом | Мережа живлення | Небезпечний |
| Пожежонебезпечність приміщень | Наявність матеріалів, що загорають і джерел запалення (електроапаратура) | Небезпечний та шкідливий |
| Електромагнітне випромінювання в тому числі і рентгенівське | ЕПТ (Дисплей є джерелом рентгенівського, радіочастотного, ультрафіолетового й інфрачервоного випромінювання та випромінювання звукового діапазону) | Шкідливий |
| Статична електрика | ЕПТ монітору і діелектрична поверхня екрана | Шкідливий |
| Іонізація повітря | Статична електрика і рентгенівське випромінювання | Шкідливий |
| Підвищений рівень шуму | Шум створюється перетворювачем напруги ЕОМ, її технічною периферію, а також людьми, що працюють в аудиторії | Шкідливий |
| Несприятлива освітленість | Недостатнє штучне і природне освітлення | Шкідливий |
| Незадовільні параметри мікроклімату | Незадовільний стан системи опалення і вентиляції | Шкідливий |
| Психофізіологічні напруження | Монотонність праці, перенапруженість зорових аналізаторів | Шкідливий |

Робота на персональному комп'ютері супроводжується постійною і значною напругою функцій зорового аналізатора. Розлад органів зору різко збільшується при роботі понад чотири годин на день .

Нервово-емоційне напруження при роботі на ПК виникає внаслідок дефіциту часу, великого об'єму і щільності інформації, особливостей діалогового режиму спілкування людини і ПК, відповідальності за безпомилковість інформації. Тривала робота на дисплеї, особливо в діалоговому режимі, може призвести до нервово-емоційного перенапруження, порушення сну, погіршення стану, зниження

концентрації уваги та працездатності, хронічного головного болю, підвищеної збудливості нервової системи, депресії.

Підвищені статичні і динамічні навантаження у користувачів ПК призводять до скарг на болі в спині, шийному відділі хребта і руках. З усіх нездужань, обумовлених роботою на комп'ютерах, частіше зустрічаються ті, які пов'язані з використанням клавіатури. У період виконання операцій введення даних кількість дрібних стереотипних рухів кистей і пальців рук за зміну може перевищити 60 тис., що відповідно до гігієнічної класифікації праці відноситься до категорії шкідливих і небезпечних. Більшість працюючих рано чи пізно починають пред'являти скарги на болі в шиї і спині. Ці нездужання накопичуються поступово і отримали назву «синдром тривалих статичних навантажень».

Іншою причиною виникнення СДСН може бути тривале перебування в положенні «сидячи», яке призводить до сильного перенапруження м'язів спини і ніг. Основною причиною перенапруги м'язів спини і ніг є нераціональна висота робочої поверхні столу і сидіння, відсутність опорної спинки і підлокітників, незручне розміщення монітора, клавіатури та документів, відсутність підставки для ніг.

Ще одним небезпечним фактором може стати недостатність освітлення, що приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. неправильний напрям світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть привести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Тривала і інтенсивна робота на комп'ютері може стати джерелом важких професійних та звичайних захворювань таких як:

- неправильна постава – це призводить до подальшого розвитку викривлення хребта сколіозу, лордозу, кіфозу, а як результат головні болі, болі в області шиї і всього хребта, болі в області таза. Неправильно положення ніг може привести до артриту (запалення суглобів), артрозу (деформації);

- тунельний синдром – найвідоміше захворювання людей працюють за комп'ютером. Воно ж синдром зап'ястного каналу. Тунельний синдром проявляється після кількох годин напруженої роботи за комп'ютером;

- геморой - хвороба аналогічна варикозного розширення вен. Причиною хвороби може бути застій крові у венах;

- розвиток короткозорості – через те, що екран монітора за контрастом вище, ніж навколишні об'єкти розвивається короткозорість;

- порушення фокусування – наслідком напруженої роботи за монітором є порушення фокусування, яка може бути викликана перенапруженням очних м'язів;

- сухість очей – через рефлексу, заснованого на тому, що при погляді на джерело світла очей починає менше моргати виникає «обсушування» рогівки ока яке призводить до очних болів.

## 5.3 Заходи щодо зменшення впливу шкідливих факторів роботи за комп’ютером

Працюючи за комп’ютером, рекомендуємо дотримуватися правил тривалості роботи, правильної постави, розміру шрифтів та зображень, вимог до приміщення тощо.

Принципи правильної роботи за комп’ютером:

- у робочому приміщенні (кімнаті), де встановлені комп’ютери, щодня потрібно виконувати вологе прибирання;

- приміщення, у якому знаходяться комп’ютери, потрібно провітрювати щогодини;

- після кожного часу роботи рекомендується робити десяти хвилинну перерву, яку зручно суміщати з провітрюванням. За будь-яких умов безперервна робота за комп’ютером для дорослої людини не повинна перевищувати двох годин. Під час перерви не варто читати або дивитися телевізор;

- необхідно постійно слідкувати за станом екрану монітора: він має бути

чистим, без плям та пилу;

- потрібно слідкувати за поставою: ноги повинні твердо стояти на підлозі чи на спеціальній підставці; стегна повинні бути розташовані під прямим кутом до тулуба, а гомілки – під прямим кутом до стегон; сидіти потрібно прямо або злегка нахилившись вперед; відстань від очей до екрану монітора – не менше 55-60 см; центр екрану має знаходитися на рівні очей чи трохи нижче; рекомендується хоча б раз на день виконувати гімнастику для очей;
- щоб попередити „синдром сухого ока”, потрібно моргати кожні 3-5 секунд;
- не використовувати звичайний телевізор замість монітору;
- у процесі роботи рекомендується періодично (приблизно раз на 20-30 хвилин) переводити погляд з екрану на найбільш віддалений предмет у кімнаті, а ще краще – на віддалений об’єкт за вікном;
- якщо з’явилося відчуття втоми, напруження, сонливості, тяжкості в очах, потрібно припинити роботу та хоча б трохи відпочити.

Рекомендовані умови для роботи за комп'ютером:

- твердий стілець. край стільця не повинен тиснути на судини під колінами.;
- відстань до монітора повинна бути 50-70см;
- перерву в сидячій роботі, 15-20 хвилин кожні 1-2 години;
- правильне освітлення робочого місця. при слабкому світлі очі напружуються і болять;
- потрібно закривати очі для відпочинку. час від часу відводите очі вбік, щоб дати відпочити своєму зору.

# 6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

У сучасних умовах збереження навколишнього середовища є невід'ємною складовою будь-якої діяльності, що стосується технічних або інформаційних систем. Екологічна свідомість суспільства вимагає інтеграції принципів сталого розвитку у всі сфери, включаючи інформаційні технології. При створенні комп’ютерних програм та проєктів необхідно враховувати потенційний вплив на навколишнє середовище, особливо щодо витрат ресурсів, енергії та поводження з відходами.

У міру зростання темпів індустріалізації людство стикається з новими екологічними викликами, такими як забруднення атмосфери, знищення природних екосистем, накопичення небезпечних відходів і вплив електромагнітного випромінювання. Ефективне управління ресурсами та мінімізація негативного впливу на природу є ключовими завданнями сучасних підприємств, зокрема в сфері інформаційних технологій.

## 6.1 Аналіз існуючої проблеми необхідності охорони навколишнього середовища

Охорона навколишнього природного середовища, раціональне використання природних ресурсів та забезпечення екологічної безпеки життєдіяльності людини є основою сталого економічного та соціального розвитку України. Сучасний науково-технічний прогрес значно впливає на природні процеси, змінюючи баланс між суспільством і природою. Втручання у природні екосистеми часто супроводжується накопиченням токсичних речовин, які не включаються у природні цикли та викликають серйозні екологічні наслідки.

Україна реалізує екологічну політику, яка спрямована на збереження безпечного навколишнього середовища для життя людей, захисту їхнього здоров'я

та досягнення гармонійної взаємодії суспільства і природи. Екологічна політика регулюється Законом України "Про охорону навколишнього природного середовища" [30] та низкою інших спеціалізованих законодавчих актів, які забезпечують правові, економічні та соціальні основи екологічної діяльності.

Однією з ключових проблем є збільшення екологічного навантаження внаслідок поширення інформаційних технологій. Електромагнітне випромінювання від комп'ютерної техніки, а також проблема утилізації електронних відходів потребують комплексного підходу. Вплив електромагнітного поля може негативно позначатися на здоров'ї людей, викликаючи головний біль, порушення сну, перевтому тощо. Сучасні дослідження також вказують на те, що неправильна утилізація електроніки спричиняє значне забруднення ґрунтів та водних ресурсів.

Крім того, швидке зростання кількості електронних пристроїв призводить до зростання енергоспоживання. Це вимагає від розробників програмного забезпечення враховувати енергоефективність алгоритмів та оптимізацію роботи обладнання. Використання хмарних технологій може бути одним із рішень для зменшення енергетичних витрат.

Створення комп’ютерних програм зазвичай пов’язане зі створенням та використанням технічних систем, що може потребувати значних енергетичних та сировинних ресурсів. Ефективне використання ресурсів є важливим фактором у зменшенні екологічного навантаження. Зокрема, оптимізація виробничих процесів, зменшення енергоспоживання та впровадження принципів "зеленого" програмування можуть суттєво знизити негативний вплив на навколишнє середовище.

Енергоефективність комп’ютерних систем має прямий вплив на екологічну ситуацію. Наприклад, впровадження енергоощадного обладнання та перехід на відновлювані джерела енергії, такі як сонячна або вітрова, є важливими кроками до сталого розвитку. Водночас важливо враховувати економічну доцільність таких заходів, щоб забезпечити їх реальне впровадження в комерційній діяльності.

Однією з головних проблем, пов’язаних з інформаційними та комп’ютерними технологіями, є утилізація електроніки та інших технічних засобів. Щорічно у світі утворюється понад 50 мільйонів тонн електронних відходів, значна частина яких не переробляється належним чином. Це призводить до забруднення ґрунтів важкими металами та іншими токсичними речовинами.

Для вирішення цієї проблеми необхідно запроваджувати програми з повторного використання або безпечної утилізації обладнання. Особлива увага повинна приділятися правильному обранню сировини та матеріалів, що легко утилізуються або підлягають рециклінгу. Утилізація побутових відходів, таких як пластик, папір і скло, повинна здійснюватися за допомогою спеціалізованих підприємств, які забезпечують вторинну переробку.

Розробники програмного забезпечення також можуть сприяти зменшенню кількості відходів, створюючи продукти, які є сумісними з більш екологічними технологіями. Крім того, популяризація цифрових рішень, таких як електронні документи замість паперових, сприяє зниженню впливу на навколишнє середовище.

При розробці комп’ютерних програм важливо враховувати енергозберігаючі технології та методи. Оптимізація алгоритмів для зменшення споживання потужності або використання енергоефективного обладнання сприяє зменшенню витрат та навантаження на навколишнє середовище. Використання хмарних технологій або споруд "зелених" дата-центрів дозволяє суттєво знизити енергоспоживання.

Крім того, слід впроваджувати політику мінімізації електромагнітного впливу, вимикаючи техніку, коли вона не використовується, та забезпечуючи регулярні перерви для працівників, що працюють з комп’ютерами. Впровадження стандартів енергоефективності в компаніях та використання обладнання з низьким енергоспоживанням є важливими аспектами сталого розвитку.

Особливу увагу слід приділяти впровадженню інноваційних технологій, таких як штучний інтелект, який може оптимізувати процеси розробки та тестування програмного забезпечення, зменшуючи витрати часу та ресурсів.

Використання таких технологій не лише підвищує ефективність роботи, але й сприяє зменшенню впливу на навколишнє середовище.

## 6.2. Розробка заходів щодо зменшення забруднення

Одним із ключових напрямків охорони навколишнього середовища є розробка комплексних заходів, спрямованих на зменшення забруднення. У сучасному світі це завдання охоплює широкий спектр дій, які можуть бути реалізовані як на рівні державної політики, так і в межах окремих підприємств та організацій. У цьому підрозділі розглянемо основні підходи до зменшення забруднення у контексті інформаційних технологій та суміжних галузей.

*Управління відходами електронної техніки.*

Один з найбільших викликів сучасності - це утилізація електронних відходів. Щорічно у світі накопичується понад 50 мільйонів тонн електронних відходів, більшість з яких не піддається належній переробці. Ефективна система управління відходами має включати:

- створення центрів збору та переробки електронної техніки;
- впровадження програм повторного використання компонентів;
- розробку стандартів для виробництва електроніки з екологічно безпечних матеріалів.

*Оптимізація енерговитрат.*

Інформаційні технології є значним споживачем енергії, що сприяє підвищенню рівня викидів вуглекислого газу. Зменшення енергоспоживання можливе завдяки:

- впровадженню енергоощадних технологій у дата-центрах;
- оптимізації алгоритмів програмного забезпечення;
- переходу на використання відновлюваних джерел енергії.

*Реалізація політики "зеленого" програмування.*

Принципи "зеленого" програмування передбачають розробку програмного забезпечення, що мінімізує використання апаратних ресурсів. Це включає:

- оптимізацію коду для скорочення навантаження на процесор та пам'ять;
- використання вбудованих інструментів моніторингу енергоспоживання;
- створення програм, які враховують енергозберігаючий режим роботи обладнання.

*Популяризація цифрових рішень.*

Скорочення використання паперових документів через перехід на цифрові технології сприяє зменшенню вирубки лісів і зниженню рівня забруднення. Це включає:

- розвиток платформ для електронного документообігу;
- впровадження електронних підписів та хмарних сховищ даних;
- стимулювання підприємств до переходу на безпаперові офісні процеси.

*Зменшення впливу електромагнітного випромінювання.*

Тривале використання електронних пристроїв може впливати на здоров'я людей і довкілля. Для мінімізації цього впливу пропонується:

- впровадження стандартів безпеки для електронного обладнання;
- регулярна перевірка робочих місць на відповідність нормам електромагнітного випромінювання;
- проектування пристроїв із зниженим рівнем випромінювання.

*Освіта та підвищення екологічної свідомості.*

Формування екологічної культури серед розробників і користувачів технологій сприяє довгостроковим позитивним змінам. До заходів у цьому напрямі належать:

- проведення тренінгів та семінарів на тему екологічної безпеки;

- впровадження екологічних стандартів у навчальні програми вищих навчальних закладів;

- створення інформаційних кампаній, які популяризують екологічно дружні технології.

У підсумку, реалізація заходів щодо зменшення забруднення потребує злагоджених дій на всіх рівнях суспільства. Інтеграція екологічних принципів у розробку та використання інформаційних технологій допоможе забезпечити сталий розвиток і збереження природних ресурсів для майбутніх поколінь.

Охорона навколишнього середовища є важливою складовою діяльності у сфері інформаційних технологій. Забезпечення екологічної безпеки, ефективне використання ресурсів та впровадження енергоощадних технологій сприяють не лише збереженню природи, але й підвищенню ефективності роботи підприємств. Сучасні виклики вимагають інтеграції екологічних принципів у всі етапи розробки програмного забезпечення, що дозволить досягти сталого розвитку та зберегти природні ресурси для майбутніх поколінь.

# ВИСНОВКИ

В кваліфікаційній роботі вирішене практичне завдання з підвищення достовірності оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity. Для вирішення поставленого завдання було розроблено рівняння еліпсу передбачення для нормалізованих метрик RFC та CBO, а також рівняння трансформованого еліпсу передбачення для цих метрик.

1) В сучасних методах оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами застосовуються метрики RFC та CBO. Для оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity було застосовано еліпс передбачення для нормалізованих метрик RFC та CBO. Ефективність такого підходу підвищується за рахунок використання нормалізуючих перетворень та видалення викидів із набору даних. Також для оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity може бути застосований трансформованих еліпс передбачення для метрик RFC та CBO, який будується на основі нормалізованого еліпсу передбачення з використанням зворотнього нормалізуючого перетворення.

2) Перевірку даних на наявність викидів було виконано за допомогою методу, що базується на нормалізуючих перетвореннях та квадраті відстані Махаланобіса. В ході дослідження було встановлено, що початковий набір даних має негаусівський розподіл та містив два викиди, видалення яких приблизило двовимірний розподіл до нормального.

3) Удосконалено рівняння еліпсу передбачення для десяткових логарифмів метрик RFC та CBO. Рівняння враховує кореляцію між метриками, що дозволяє підвищити точність оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity.

4) На основі запропонованого рівняння еліпсу передбачення було створено програму для оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity. Підготовлена програмна документація: технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань (додатки А, Б, В, Г, Д). Проведені випробування підтвердили працездатність програми та її ефективність у оцінюванні складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Read S. Gaming is booming and is expected to keep growing. URL: https://www.weforum.org/agenda/2022/07/gaming-pandemic-lockdowns-pwc-growth/ (дата звернення: 08.09.2024).

2. Chidamber S., Kemerer C. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*. 1994. vol. 20 issue 6, P. 476-493. DOI: https://doi.org/10.1109/32.295895.

3. Objectscript_Q. Response for Class (RFC). URL: https://www.cachequality.com/docs/metrics/response-for-class (дата звернення: 10.08.2024).

4. Sahraoui H.A., Godin R., Miceli T. Can metrics help to bridge the gap between the improvement of OO design quality and its automation? *Software Maintenance : 2000 International Conference ICSM* (San Jose, CA, USA). P. 154-162. DOI:10.1109/ICSM.2000.883034.

5. Chidamber S., Kemerer C. Object-oriented metrics suite. URL: https://www.aivosto.com/project/help/pm-oo-ck.html (дата звернення: 15.08.2024).

6. Prykhodko S., Prykhodko N. A Technique for Detecting Software Quality Based on the Confidence and Prediction Intervals of Nonlinear Regression for RFC Metric. *Computer Sciences and Information Technologies: 2022 IEEE 17th International Conference (CSIT-2022, Lviv, Ukraine)*. P. 499-502. DOI: https://doi.org/10.1109/CSIT56902.2022.10000532.

7. Приходько С.Б., Смикодуб Т.Г. Математична модель для перевірки взаємозв'язків між частинами Java-застосунків за метриками RFC та CBO. *Матеріали II Всеукраїнської науково-практичної інтернет конференції "Інформаційні технології: моделі, алгоритми, системи: (ITMAS – 2021)"*, Миколаїв, 2021.С. 25-26.

8. Папакін М.М., Приходько С.Б. Математична модель для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами

комп'ютерних ігор та розробка програми для її реалізації. *Матеріали III Всеукраїнської науково-практичної інтернет конференції «Інноваційні технології: моделі, алгоритми, системи (ITMAS-2024)» м. Миколаїв 26-28 жовтня 2022 р.* Миколаїв, 2022. С. 50-51.

9. Семенчук І.М., Макарова Л.М. Попередня обробка інформації для оцінювання складності розробки 2D ігор, що створювалися на ігровому рушії Unity. *Матеріали XIV Міжнародної науково-технічної конференції «Інновації в суднобудуванні та океанотехніці» м. Миколаїв 20-21 вересня 2023 р.* Миколаїв, 2023. С. 467-469.

10. Семенчук І.М., Макарова Л.М. Математична модель для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity. *Матеріали V Всеукраїнської науково-практичної інтернет конференції «Інноваційні технології: моделі, алгоритми, системи (ITMAS-2024)» м. Миколаїв 30-31 жовтня 2024 р.* Миколаїв, 2024. С. 61-62.

11. Zaiontz C. Real statistics using Excel. Multivariate Normality Testing (Mardia). URL: https://real-statistics.com/multivariate-statistics/multivariate-normal-distribution/multivariate-normality-testing/ (дата звернення: 08.09.2024).

12. Wulandari D., Sutrisno, Bayu Nirwana M. Mardia's Skewness and Kurtosis for Assessing Normality Assumption in Multivariate Regression. *Enthusiastic international journal of statistics and data science*. 2021. vol. 1, issue 1, P. 1-6. DOI: http://dx.doi.org/10.20885/enthusiastic.vol1.iss1.art1.

13. Chew V. Confidence, prediction and tolerance regions for the multivariate normal distribution. *Journal of the American Statistical Association*. 1966. vol. 61, issue 315, P. 605-617. DOI: https://doi.org/10.2307/2282774.

14. Afifi A.A., Azen S.P. Statistical analysis: a computer-oriented approach. New York; London: Academic Press, 1979. 442 p.

15. Приходько С.Б., Резниченко М.В. Математична модель для оцінювання складності об'єктно-орієнтованого проектування через зв'язки між класами застосунків з відкритим кодом на Java. *Матеріали III Всеукраїнської науково-практичної інтернет конференції "Інформаційні технології: моделі, алгоритми,*

*системи: (ITMAS – 2022)" м. Миколаїв 26-28 жовтня 2022 р.* Миколаїв, 2022. С. 6-7.

16. Паркер А. Діаграма варіантів використання UML. URL: https://www.guru99.com/uk/use-case-diagrams-example.html (дата звернення 17.10.2024).

17. Паркер А. Діаграма діяльності в UML. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення 17.10.2024).

18. Паркер А. Діаграма стану автомата та діаграма стану UML. URL: https://www.guru99.com/uk/state-machine-transition-diagram.html (дата звернення 17. 10.2024).

19. Поняття інтерфейсу. Типи користувацького інтерфейсу та етапи їх розробки. URL: https://allcompositions.at.ua/temy_1-2.pdf (дата звернення: 22. [10.2024).

20. Python Software Foundation. Мова програмування Python. URL: https://docs.python.org/uk/3/tutorial/index.html (дата звернення 25. 10.2024).

21. SciPy. Бібліотека SciPy. URL: https://scipy.org/install/ (дата звернення 25. 10.2024).

22. NumPy. Бібліотека NumPy. URL: https://numpy.org/ (дата звернення 25. 10.2024).

23. The Matplotlib development team. Бібліотека Matplotlib. URL: https://matplotlib.org/ (дата звернення 25. 10.2024).

24. OpenPyXl. Бібліотека OpenPyXl. URL: https://openpyxl.readthedocs.io/en/stable/ (дата звернення 25. 10.2024).

25. Python Software Foundation. Бібліотека Tkinter. URL: https://docs.python.org/uk/3/library/tkinter.html (дата звернення 25. 10.2024).

26. Визначення економічної ефективності програмного виробу. URL: https://studwood.net/1325190/ekonomika/viznachennya_ekonomichnoyi_efektivnosti_programnogo_virobu? (дата звернення 07.11.2024).

27. Лозовська Л.І., Дудник В.В. Сучасні підходи до вартісної оцінки програмних продуктів. URL: https://eurodev.duan.edu.ua/images/PDF/2014/2/16.pdf (дата звернення 07.11.2024).

28. Закон України «Про охорону праці». URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення 07.11.2024).

29. Закон України «Про охорону здоров’я». URL: https://zakon.rada.gov.ua/laws/show/2801-12#Text (дата звернення 07.11.2024).

30. Закон України № 1268-XII від 26.06.91 р. «Про охорону навколишнього природного середовища» (із змінами). URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення 07.11.2024).

# Додаток А – Технічне завдання на розробку програми для оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity

*Назва програми*: “Програма для оцінювання складності об’єктно-орієнтованого проектування через зв'язки між класами 2D ігор на рушії Unity”, надалі програма.

*Сфера застосування:* Програма застосовується в сферах пов’язаних з розробкою програмного забезпечення, зокрема 2D ігор розроблених на рушії Unity. Програма застосовується для оцінювання якості архітектури 2D ігор на Unity, виявлення складних зв’язків між класами та оптимізації проєктування, що полегшує підтримку й масштабування коду.

*Підстави для розробки:* наказ на затвердження тем кваліфікаційних робіт магістрів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

*Експлуатаційне призначення*: Забезпечення автоматизованого оцінювання складності складності об’єктно-орієнтованого проектування. Програма дозволяє розробникам та дослідникам оцінювати складність проєктів, виявляти потенційно проблемні зв’язки між класами та підвищувати ефективність проектування, підтримки та модифікації 2D ігор на рушії Unity.

*Функціональне призначення:* Функціональне призначення програми полягає в автоматизованому оцінюванні значень метрик RFC та CBO для визначення складності об'єктно-орієнтованого проєктування через зв’язки між класами 2D ігор на рушії Unity. Програма будує еліпс передбачення для нормалізованих значень метрик RFC та CBO та трансформований еліпс передбачення для цих метрик, обчислює значення квадрату відстані Махаланобіса та оцінює складність проектування з урахуванням виявлених аномалій, надаючи результати у вигляді звітів і графіків.

*Вимоги до програми:*

Вимоги до функціональних характеристик:

1) Завантажувати з файлу дані метрик RFC та CBO ігрових проектів;

2) Оцінювати складність об’єктно-орієнтованого проектування через зв’язки між класами для кожного проекту;

3) Виводити результати оцінювання об’єктно-орієнтованого проектування через зв’язки між класами для кожного проекту;

4) Будувати графіки рівняння нормалізованого та трансформованого еліпсу передбачення для оцінювання складно об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity;

5) Зберігати результати оцінювання у вказаний файл;

*Вхідні дані:*

- Файл з даними (Назва проекту, середнє значення метрики RFC, середнє значення метрики CBO);
- Файл з даними математичної моделі;

*Вихідні дані:*

- Файл з результатами оцінювання складності об’єктно-орієнтованого проектування(Номер проекту, Назва проекту, значення квадрату відстані Махаланобіса, результат оцінювання складності об’єктно-орієнтованого проектування);
- .png файл, який містить графік рівняння еліпсу передбачення і результатів оцінювання;

*Зовнішні ефекти:* поява графічного вікна для обрання файлу з даними, поява графічного повідомлення про успішне або невдале виконання операції, поява графічного вікна з зображенням результатів оцінювання у вигляді еліпсу передбачення, вихід з програми (закриття програми і всіх її вікон).

*Вимоги до надійності*: не висуваються.

*Вимоги до умов експлуатації*: не висуваються.

*Вимоги до складу й параметрів технічних засобів.*

Програма має використовуватися на обладнанні з наступними мінімальними параметрами:

- процесор intel core i3,
- оперативна пам’ять обсягом 4 гб,
- вільний простір на жорсткому диску 400 Мб.

*Вимоги до інформаційної та програмної сумісності.*

Для роботи програми необхідна операційна система Windows 10 (або вище) із встановленим пакетом Microsoft Office.

*Вимоги до програмної документації.*

Програмна документація маж містити:

- технічне завдання на розробку програми,
- текст програми,
- опис програми,
- інструкцію користувача,
- програму та методику випробувань ПЗ.

*Техніко-економічні показники*: не розраховуються.

*Стадії та етапи розробки.*

Стадії та етапи розробки наведено в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

| Стадія розробки | Етапи робіт | Термін виконання |
|---|---|---|
| 1.Технічне завдання | Розробка та затвердження технічного завдання | 10.10.2024 |
| 2.Ескізний проект | Розробка ескізного проекту | 12.10.2024 |
| | Затвердження ескізного проекту | 14.10.2024 |
| 3.Технічний проект | Розробка технічного проекту | 19.10.2024 |
| | Затвердження технічного проекту | 21.10.2024 |
| 4.Робочий проект | Розробка програми | 05.11.2024 |
| | Розробка програмної документації | 11.11.2024 |
| | Випробування програми | 13.11.2024 |

*Порядок контролю і приймання.*

Порядок контролю та приймання програми здійснюється представниками замовника у присутності представника розробника. Для приймання програми надається програмна документація, яка була розроблена відповідно до технічного завдання. Представники замовника установлюють відповідність програми технічному завданню. За результатами приймання програми представниками замовника складається акт приймання програми

# Додаток Б – текст програми

```
from tkinter import filedialog, messagebox, ttk
from openpyxl import Workbook, load_workbook
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
import numpy as np
from scipy.spatial import ConvexHull

# Глобальна константа для кількості рядків у моделі
MODEL_DATA_NUM = 66

class DataLoader:
    @staticmethod
    def load_elips_data():
        try:
            wb = load_workbook("ELIPS.xlsx")
            ws = wb["transformed"]

            elips_rfc = [ws.cell(row=i, column=1).value for i in range(2,
MODEL_DATA_NUM)]
            elips_cbo = [ws.cell(row=i, column=2).value for i in range(2,
MODEL_DATA_NUM)]
            return elips_rfc, elips_cbo
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зчитати ELIPS.xlsx: {e}")
            return [], []

    @staticmethod
    def load_normalized_data():
        try:
            wb = load_workbook("ELIPS.xlsx")
            ws = wb["normalized"]

            normalized_rfc = [ws.cell(row=i, column=1).value for i in range(2,
49)]
            normalized_cbo = [ws.cell(row=i, column=2).value for i in range(2,
49)]
            return normalized_rfc, normalized_cbo
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зчитати normalized дані:
{e}")
            return [], []

    @staticmethod
    def load_normalized_elips_data():
        try:
            wb = load_workbook("ELIPS.xlsx")
            ws = wb["normalized"]

            normalized_elips_rfc = [ws.cell(row=i, column=3).value for i in
range(2, 65)]
            normalized_elips_cbo = [ws.cell(row=i, column=4).value for i in
range(2, 65)]
            return normalized_elips_rfc, normalized_elips_cbo
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зчитати нормалізовані
дані еліпса: {e}")
```

```
            return [], []

    @staticmethod
    def load_main_data(file_path):
        try:
            workbook = load_workbook(filename=file_path)
            sheet = workbook.active

            names, rfc, cbo = [], [], []
            for row in sheet.iter_rows(min_row=2, min_col=1, max_col=3,
values_only=True):
                if row[0] is not None and row[1] is not None and row[2] is not
None:
                    names.append(row[0])
                    rfc.append(row[1])
                    cbo.append(row[2])
            return names, rfc, cbo
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зчитати файл: {e}")
            return [], [], []

class SMDCalculator:
    @staticmethod
    def is_inside_ellipse(x, y, elips_rfc, elips_cbo):
        points = np.column_stack((elips_rfc, elips_cbo))
        hull = ConvexHull(points)
        new_point = np.array([x, y])
        return all(
            np.dot(eq[:-1], new_point) + eq[-1] <= 0
            for eq in hull.equations
        )

    @staticmethod
    def calculate_complexity_and_smd(names, rfc, cbo, normalized_rfc,
normalized_cbo, elips_rfc, elips_cbo):
        results = []

        rfc_mean = np.mean(normalized_rfc)
        cbo_mean = np.mean(normalized_cbo)
        rfc_std = np.std(normalized_rfc)
        cbo_std = np.std(normalized_cbo)

        for i in range(len(names)):
            normalized_rfc_i = np.log10(rfc[i]) if rfc[i] > 0 else 0
            normalized_cbo_i = np.log10(cbo[i]) if cbo[i] > 0 else 0

            smd = ((normalized_rfc_i - rfc_mean) / rfc_std) ** 2 +
((normalized_cbo_i - cbo_mean) / cbo_std) ** 2
            difficulty = "Простий" if SMDCalculator.is_inside_ellipse(rfc[i],
cbo[i], elips_rfc, elips_cbo) else "Складний"
            results.append((i + 1, names[i], smd, difficulty))
        return results

class PlotBuilder:
    @staticmethod
    def build plot(rfc, cbo, elips rfc, elips cbo, title):
        plot_window = tk.Toplevel()
        plot_window.title(title)
        plot_window.geometry("800x600")

        fig, ax = plt.subplots(figsize=(8, 6))
        ax.scatter(rfc, cbo, color="blue", label="Дані з файлу")
        ax.scatter(elips_rfc, elips_cbo, color="red", label="Дані ELIPS.xlsx")
```

```
        points = np.column_stack((elips_rfc, elips_cbo))
        hull = ConvexHull(points)
        for simplex in hull.simplices:
            ax.plot(points[simplex, 0], points[simplex, 1], 'green')

        ax.set_title(title)
        ax.set_xlabel("RFC")
        ax.set_ylabel("CBO")
        ax.legend()
        ax.grid(True)

        canvas = FigureCanvasTkAgg(fig, master=plot_window)
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
        canvas.draw()

        def save_plot():
            file_path = filedialog.asksaveasfilename(defaultextension=".png",
filetypes=[("PNG файли", "*.png")])
            if file_path:
                fig.savefig(file_path)
                messagebox.showinfo("Успіх", "Графік збережено успішно!")

        save_button = tk.Button(plot_window, text="Зберегти графік",
command=save_plot)
        save_button.pack(pady=10)

class ResultsManager:
    @staticmethod
    def show_results_window(results, rfc, cbo, normalized_elips_rfc,
normalized_elips_cbo, elips_rfc, elips_cbo):
        results_window = tk.Toplevel()
        results_window.title("Результати розрахунків")
        results_window.geometry("700x400")

        tree = ttk.Treeview(results_window, columns=("Номер", "Назва", "SMD",
"Оцінка"), show="headings")
        tree.heading("Номер", text="Номер")
        tree.heading("Назва", text="Назва")
        tree.heading("SMD", text="SMD")
        tree.heading("Оцінка", text="Оцінка")
        tree.pack(fill=tk.BOTH, expand=True)

        for row in results:
            tree.insert("", tk.END, values=row)

        button_frame = tk.Frame(results_window)
        button_frame.pack(pady=10)

        plot_normalized_button = tk.Button(button_frame, text="Побудувати
нормалізований еліпс",
                                           command=lambda: PlotBuilder.build_plot(
                                               [np.log10(value) if value > 0 else
0 for value in rfc],
                                               [np.log10(value) if value > 0 else
0 for value in cbo],
                                               normalized_elips_rfc,
                                               normalized_elips_cbo,
                                               'Нормалізований графік'))
        plot_normalized_button.pack(side=tk.LEFT, padx=5)

        save_button = tk.Button(button_frame, text="Зберегти результати",
                                command=lambda:
ResultsManager.save_results_to_excel(results))
        save_button.pack(side=tk.LEFT, padx=5)
```

```
        plot_transformed_button = tk.Button(button_frame, text="Побудувати
трансформований еліпс",
                                            command=lambda:
PlotBuilder.build_plot(rfc, cbo, elips_rfc, elips_cbo, 'Трансформований графік'))
        plot_transformed_button.pack(side=tk.LEFT, padx=5)

    @staticmethod
    def save_results_to_excel(results):
        try:
            file_path = filedialog.asksaveasfilename(
                defaultextension=".xlsx",
                filetypes=[("Excel файли", "*.xlsx")]
            )
            if not file_path:
                messagebox.showwarning("Попередження", "Файл не збережено!")
                return

            wb = Workbook()
            ws = wb.active
            ws.title = "Результати"

            ws.append(["Номер", "Назва", "SMD", "Оцінка"])
            for row in results:
                ws.append(row)

            wb.save(file_path)
            messagebox.showinfo("Успіх", "Результати успішно збережено!")
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зберегти файл: {e}")

class Application:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Обробка Excel файлу")
        self.root.geometry("300x200")

    def run(self):
        button2 = tk.Button(self.root, text="Зчитати Excel і показати результати",
command=self.process_file)
        button2.pack(pady=10)

        button3 = tk.Button(self.root, text="Закрити", command=self.root.destroy)
        button3.pack(pady=10)

        self.root.mainloop()

    def process_file(self):
        file_path = filedialog.askopenfilename(
            title="Виберіть Excel файл",
            filetypes=[("Excel файли", "*.xlsx *.xlsm")]
        )

        if not file_path:
            messagebox.showwarning("Попередження", "Файл не вибрано!")
            return

        names, rfc, cbo = DataLoader.load_main_data(file_path)
        normalized_rfc, normalized_cbo = DataLoader.load_normalized_data()
        elips_rfc, elips_cbo = DataLoader.load_elips_data()
        normalized_elips_rfc, normalized_elips_cbo =
DataLoader.load_normalized_elips_data()

        results = SMDCalculator.calculate_complexity_and_smd(names, rfc, cbo,
```

```
normalized_rfc, normalized_cbo, elips_rfc, elips_cbo)
        ResultsManager.show_results_window(results, rfc, cbo,
normalized_elips_rfc, normalized_elips_cbo, elips_rfc, elips_cbo)

if __name__ == "__main__":
    app = Application()
    app.run()
```

# Додаток В – Опис програми

Загальні відомості

Програма для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity розроблена мовою програмування Python. Файл програми  Unity2DGameOODCE.exe має розмір 10 Мб.

Функціональне призначення

Програма Unity2DGameOODCE призначений для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity на операційній системі Windows 10(або вище).

Технічні засоби, що використовуються

Програма має використовуватися на обладнанні з наступними мінімальними параметрами: процесор intel core i3 (або вище), оперативна пам’ять обсягом 4 гб, вільний простір на жорсткому диску 400 Мб.

Виклик та завантаження

Для запуску та роботи з програмою необхідна операційна система Windows 10 (32 або 64 біти) із встановленим пакетом Microsoft Office. Завантаження програми здійснюється шляхом відкриття файлу Unity2DGameOODCE.exe стандартними засобами ОС.

# Додаток Г – Інструкція користувача

## Загальний опис роботи програми

Після завантаження файлу Unity2DGameOODCE.exe відкриється стартове вікно програми з двома кнопками: «Завантажити Ексель файл» та «Закрити програму». Натискання кнопки «Закрити програму» закриває всі створені програмою активні віконні форми. Стартове вікно програми наведено на рисунку Г.1.

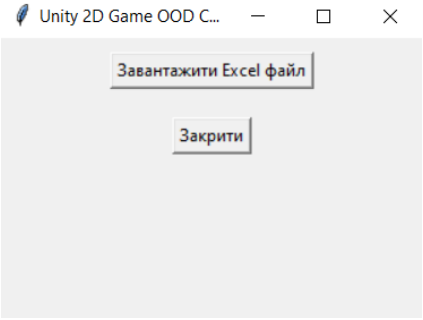


Рисунок Г.1 – Стартове вікно програми

При натисканні кнопки «Завантажити Ексель файл» відкривається нова віконна форма, яка пропонує обрати файл з даними, для яких треба виконати оцінювання складності об’єктно-орієнтованого проектування. Якщо файл не обрано, або наразі файл використовується іншою програмою ви отримаєте повідомлення про це. Віконну форму з попередженням наведено на рисунку Г.2.

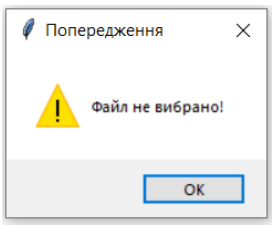


Рисунок Г.2 – Попередження про помилку обрання файлу з даними для аналізу

При успішному завантаженні файлу програма виконає оцінювання складності об’єктно-орієнтованого проектування отриманих з файлу проектів і відкриє нову віконну форму з результати оцінювання. Вікно результатів оцінювання наведено на рисунку Г.3

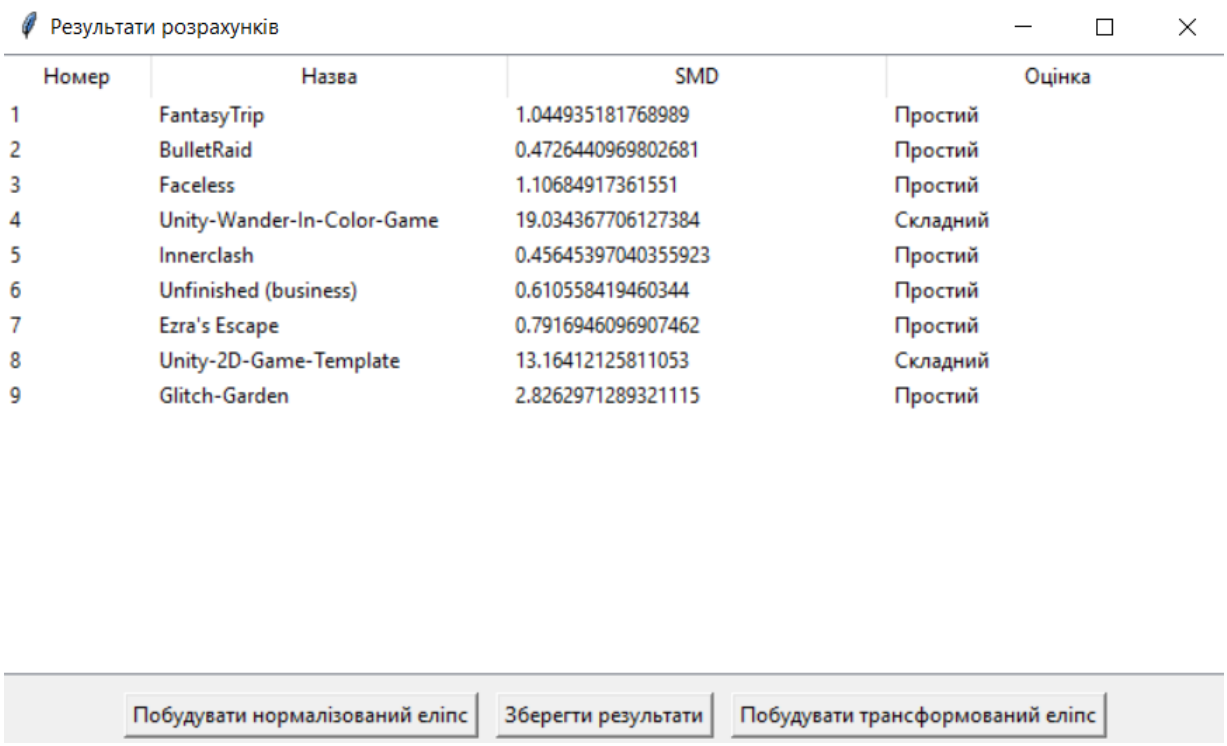


Рисунок Г.3 – Вікно результатів оцінювання складності об’єктно-орієнтованого проектування досліджуваних проектів

Для збереження результатів оцінювання натисніть кнопку «Зберегти результати». Програма запропонує вам обрати файл куди буде збережено результати оцінювання складності об’єктно-орієнтованого проектування досліджуваних проектів.

Для візуальної оцінки отриманих результатів натисніть кнопку «Побудувати нормалізований еліпс», або «Побудувати трансформований еліпс». Програма відкриє нову віконну форму з графічним зображенням еліпсу передбачення для нормалізованих даних метрик RFC та CBO та трансформований еліпс передбачення для метрик RFC та CBO відповідно. Віконну форму з графічним зображенням рівняння еліпсу передбачення для нормалізованих значень метрик

RFC та CBO та нормалізованими даними проектів у вигляді точок представлено на рисунку Г.4.

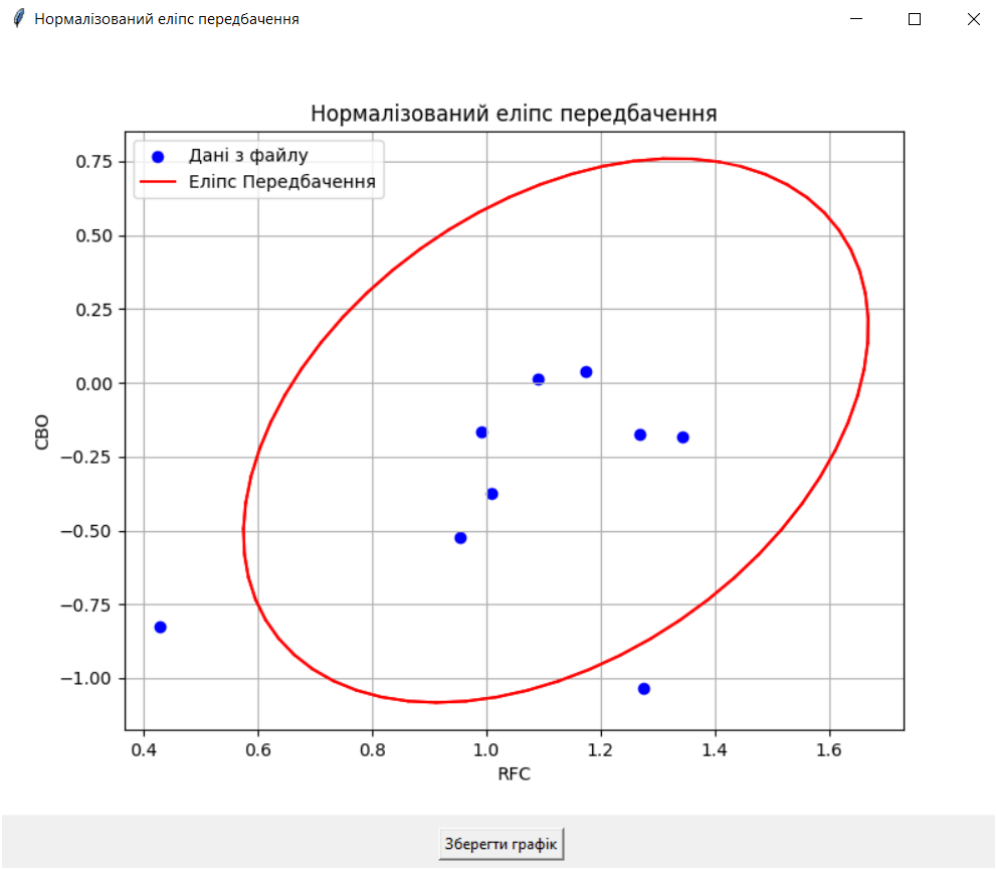


Рисунок Г.4 – Віконна форма з графічним зображенням рівняння еліпсу передбачення для нормалізованих значень метрик RFC та CBO та нормалізованими даними проектів у вигляді точок

На графіках досліджувані проекти позначені точками. Вихід точки за межі еліпсу передбачення свідчить про його більш складну структуру відносно проектів, на яких була побудована модель, і навпаки, якщо проект знаходиться в еліпсі – це свідчить що він має нормальну складність об’єктно-орієнтованого проектування відносно проектів на яких була розроблена модель.

Щоб зберегти отримані графічні результати натисніть кнопку «Зберегти», яка знаходиться під побудованим графіком. Програма вам запропонує обрати шлях і назву файлу, який ви хочете зберегти.

Щоб закрити програму закрийте всі активні віконні форми за допомогою символ «хрестик», або скористайтеся кнопкою «Закрити» на стартовому вікні програми, яка зачинить всі створені програмою активні віконні форми.

# Додаток Д – Програма та методика випробувань ПЗ

*Об’єкт випробувань*: Програма для оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами 2D ігор на рушії Unity Unity2DGameOODCE.

*Ціль випробувань*: перевірка відповідності програми Unity2DGameOODCE до вимог технічного завдання і повноти реалізації.

*Склад програмної документації:*

- технічне завдання;
- інструкція користувача;
- опис програми;
- текст програми;
- програма та методика випробувань ПЗ.

*Технічні вимоги.*

Програма має використовуватися на обладнанні з наступними мінімальними параметрами: процесор intel core i3 (або вище), оперативна пам’ять обсягом 4 гб, вільний простір на жорсткому диску 400 Мб.

*Методи випробувань.*

Випробування програми виконується представником замовника в присутності представника розробника. Відповідно до опису програми виконується запуск програми Unity2DGameOODCE. Далі виконується перевірка роботи програми. Виконується перевірка всіх екранних форм програми, а також всіх допустимих для виконання операцій всередині екранних форм згідно до посібника користувача. Якщо програма демонструє адекватне функціонування згідно до вимог ТЗ та представленої програмної документації, то робота з розробки програми вважається виконаною, і виконується впровадження програми до експлуатації. В протилежному випадку програма має бути відправлено на доробку розробникові.

Використовуються випробовування за методологією „чорної скриньки” (рис.Д.1 - Д.17).

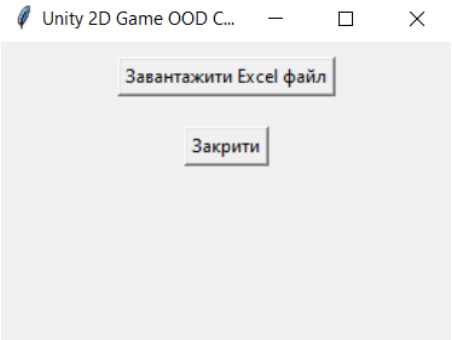


Рисунок Д.1 – Стартове вікно програми

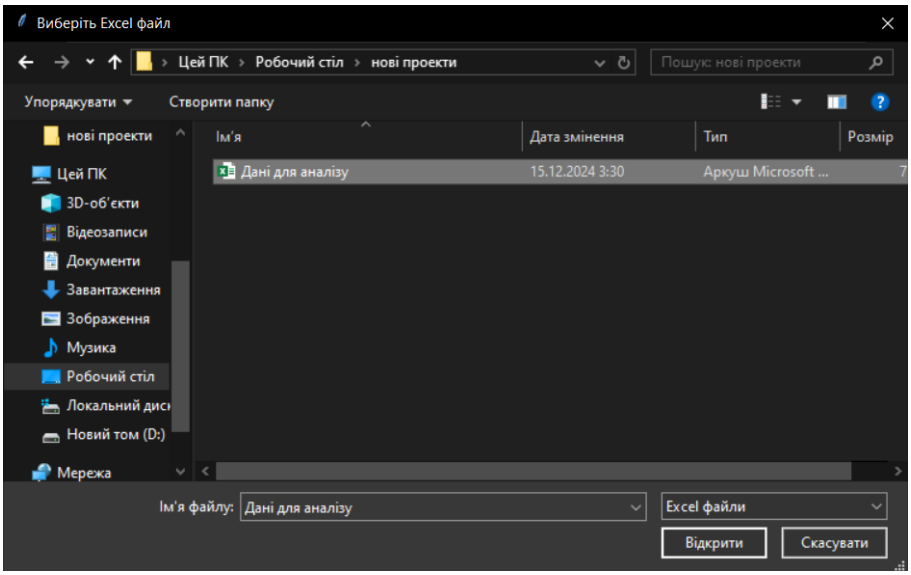


Рисунок Д.2 – Віконна форма вибору файлів для аналізу

Якщо користувач обирає файл не правильного формату, або не вибирає файл, отримує повідомлення від програми про помилку.

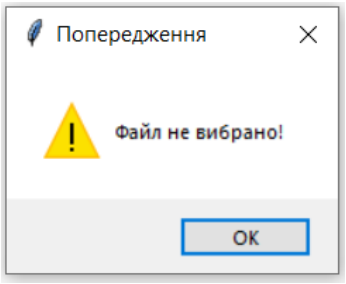


Рисунок Д.3 – Попередження про помилку обрання файлу з даними для аналізу

При обранні фалу з даними без помилок програма відкриває користувачу віконну форму з результатами оцінювання складності об’єктно-орієнтованого проектування.

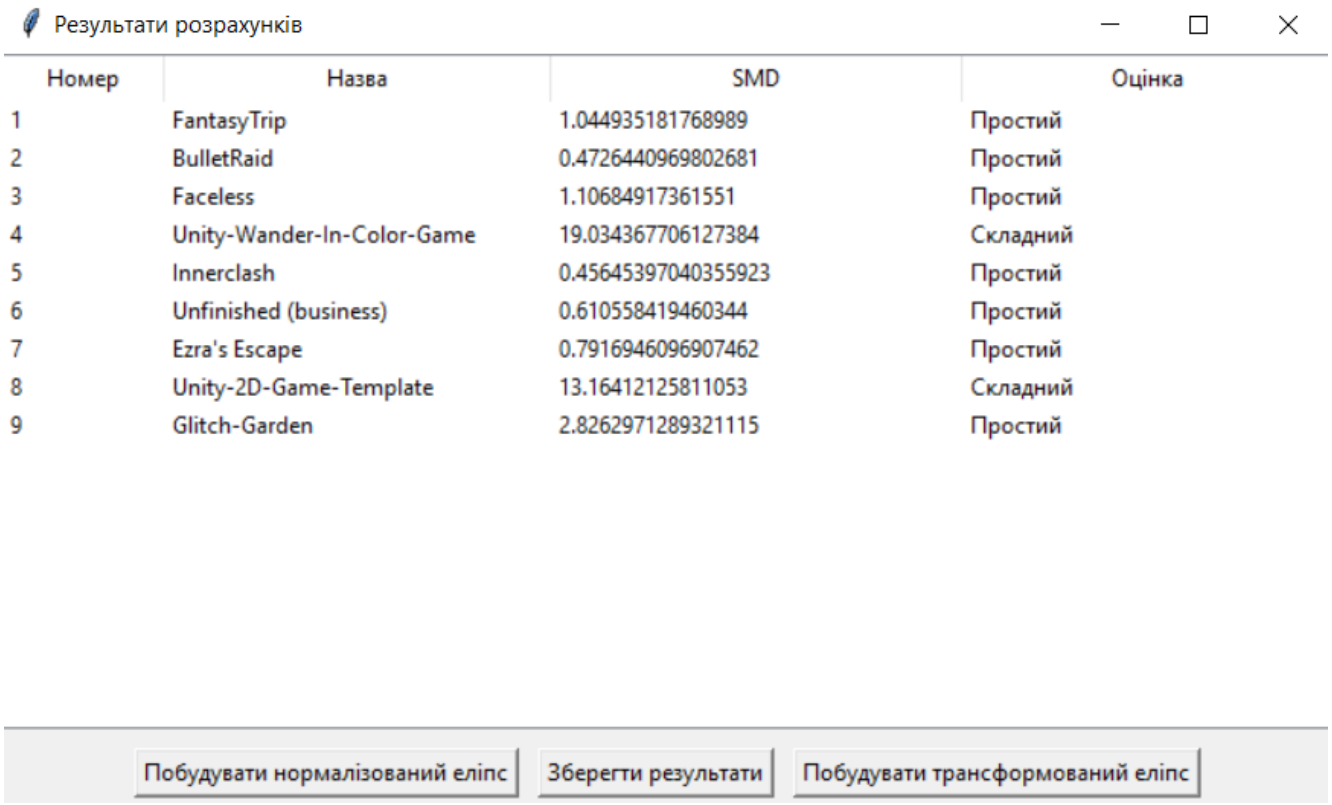


Рисунок Д.4 – Вікно результатів оцінювання складності об’єктно-орієнтованого проектування досліджуваних проектів

На вікні з результатами розрахунків користувач натискає кнопку «Побудувати нормалізований еліпс». Програма відкриває нову віконну форму з графічною інтерпритацією отриманих результатів у вигляді побудованого рівняння еліпсу передбачення для нормалізованих значень метрик RFC та CBO та нормалізованими даними проектів у вигляді точок.

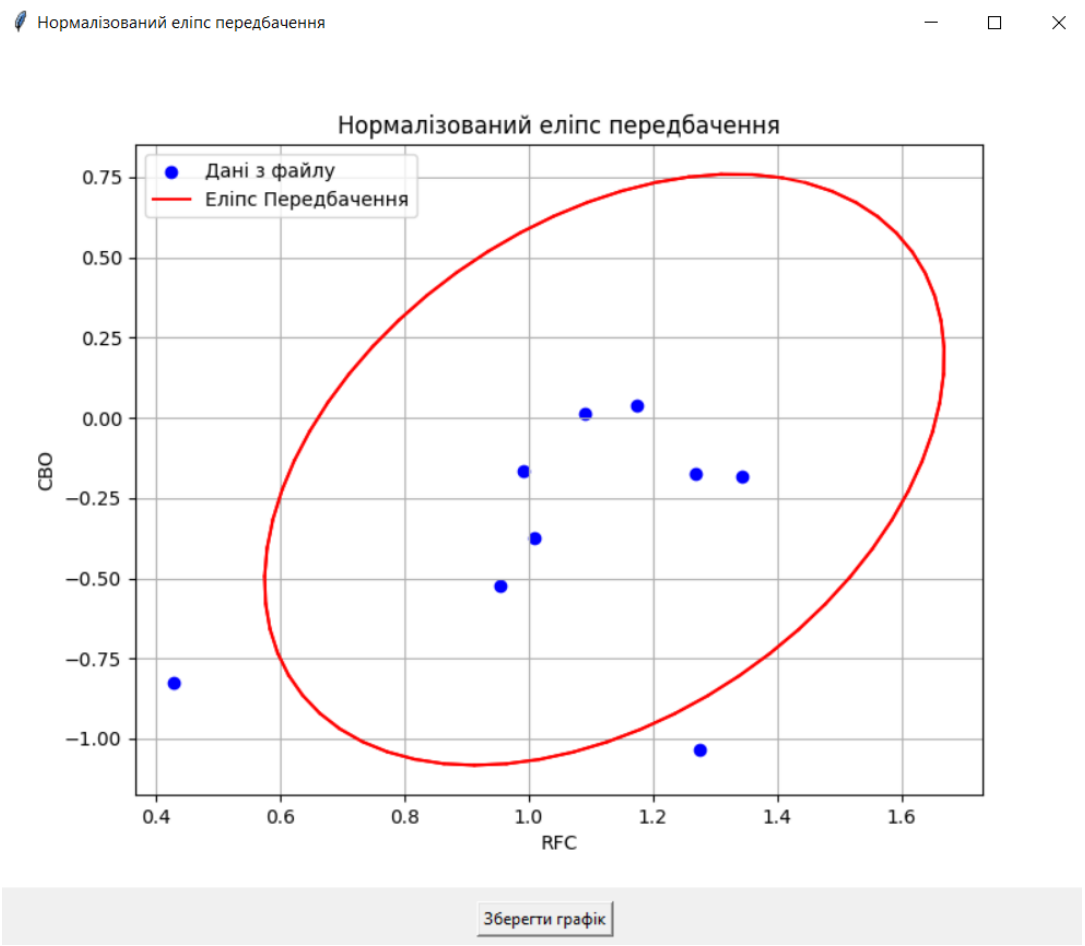


Рисунок Д.5 – Віконна форма з графічним зображенням рівняння еліпсу передбачення для нормалізованих значень метрик RFC та CBO та нормалізованими даними проектів у вигляді точок

Користувач натискає кнопку «Зберегти графік». Програма відкриває нову віконну форму для збереження .png файлу отриманого графіку (див. рис. Д.6). У разі успішного збереження файлу користувач отримає повідомлення про успішне збереження файлу (див. рис. Д.7).

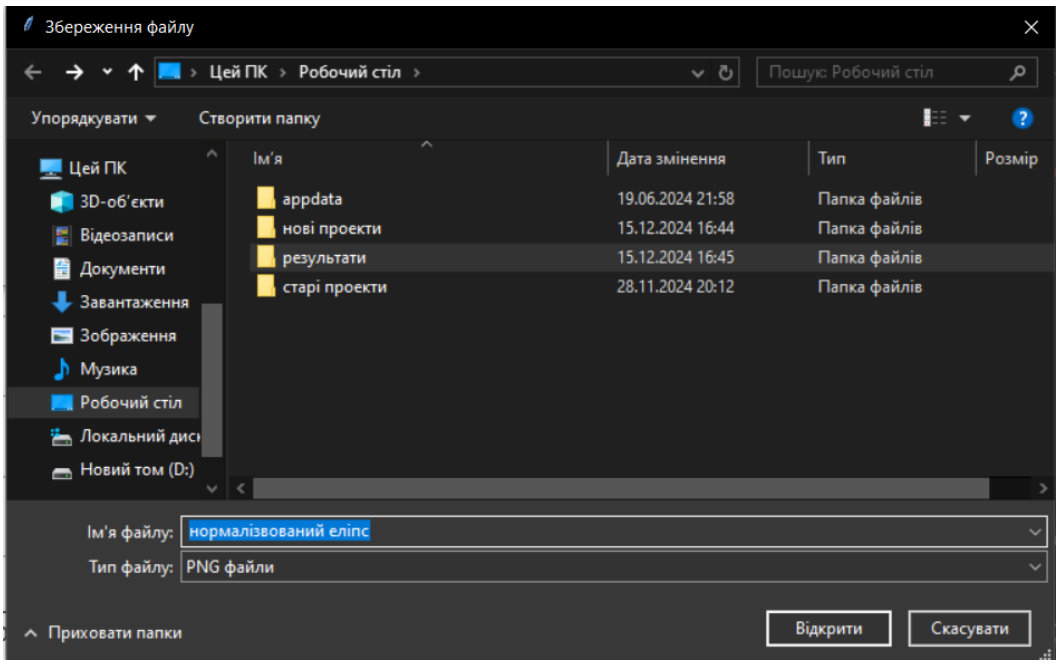


Рисунок Д.6 – Віконна форма для збереження .png файлу графіку рівняння еліпсу передбачення та результатів оцінювання складності об'єктно-орієнтованого проектування досліджуваних проектів у вигляді точок

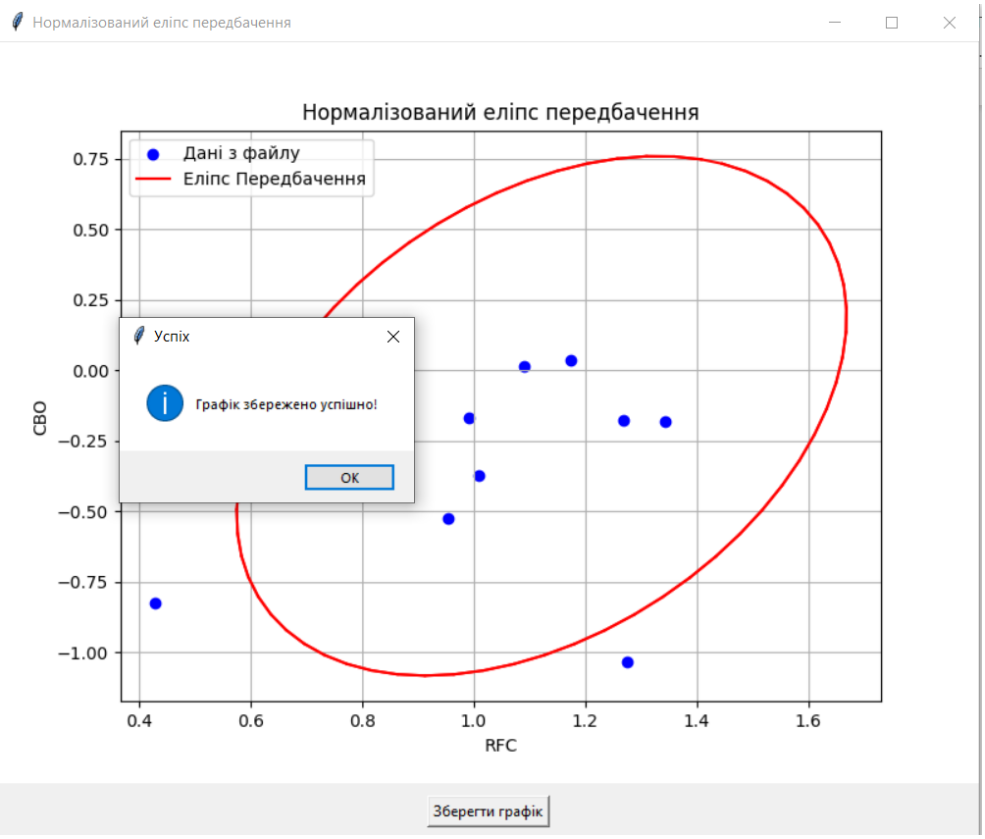


Рисунок Д.7 – Повідомлення про успішне збереження .png файлу побудованого графіку

На вікні з результатами розрахунків користувач натискає кнопку «Побудувати трансформований еліпс». Програма відкриває нову віконну форму з графічною інтерпритацією отриманих результатів у вигляді побудованого рівняння трансформованого еліпсу передбачення для значень метрик RFC та CBO та даними досліджуваних проектів у вигляді точок.

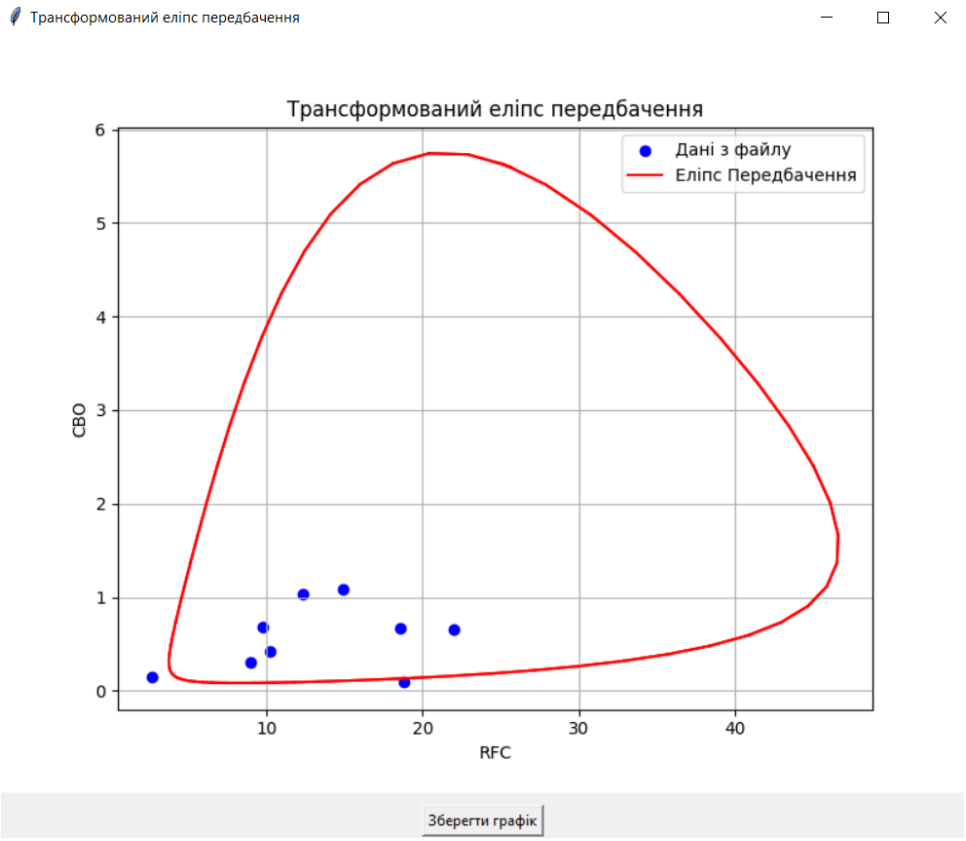


Рисунок Д.8 - Віконна форма з графічним зображенням рівняння трансформованого еліпсу передбачення для значень метрик RFC та CBO та даними досліджуваних проектів у вигляді точок

Користувач натискає кнопку «Зберегти графік». Програма відкриває нову віконну форму для збереження .png файлу отриманого графіку (див. рис. Д.9). У разі успішного збереження файлу користувач отримає повідомлення про успішне збереження файлу (див. рис. Д.10).

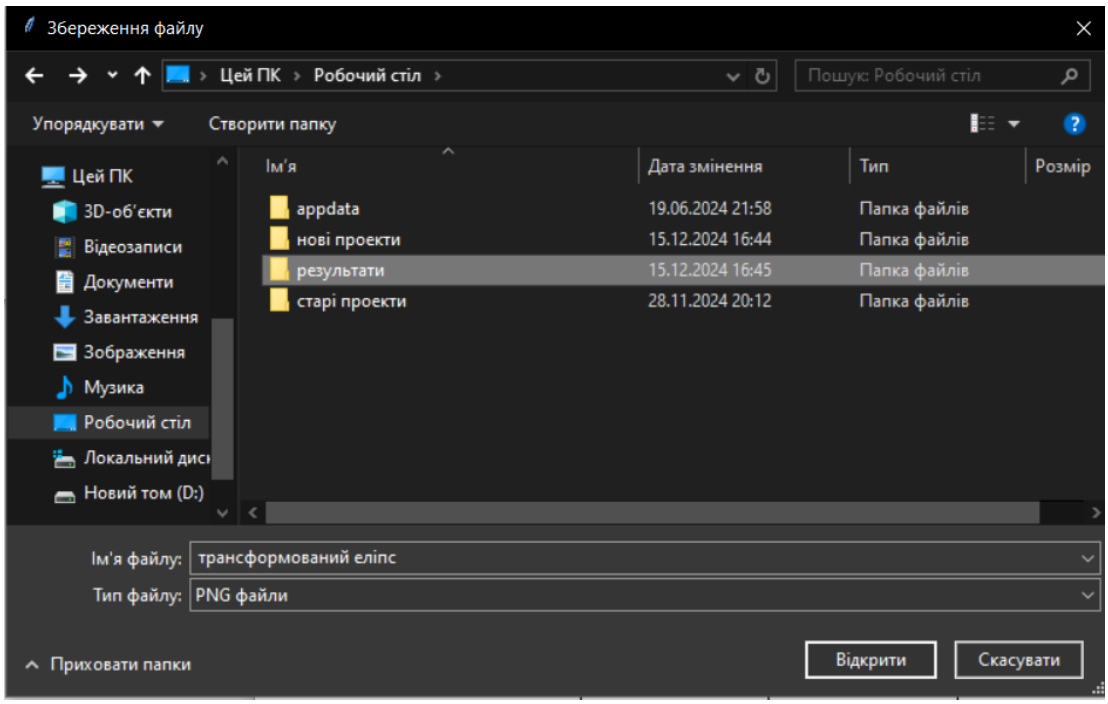


Рисунок Д.9 – Віконна форма для збереження .png файлу графіку рівняння еліпсу передбачення та результатів оцінювання складності об’єктно-орієнтованого проектування досліджуваних проектів у вигляді точок

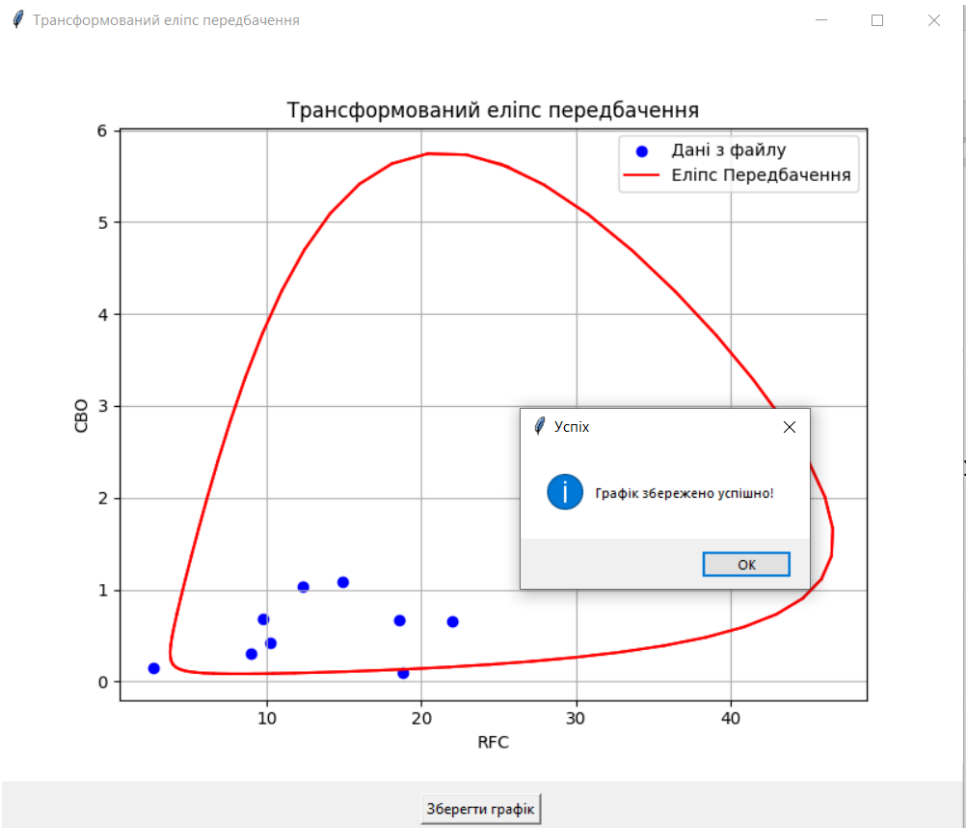


Рисунок Д.10 – Повідомлення про успішне збереження .png файлу побудованого графіку

На вікні з результатами розрахунків користувач натискає кнопку «Зберегти результати». Програма відкриває нову віконну форму для збереження .xlsx файлу отриманих результатів (див. рис. Д.11). У разі успішного збереження файлу користувач отримає повідомлення про успішне збереження файлу (див. рис. Д.12).

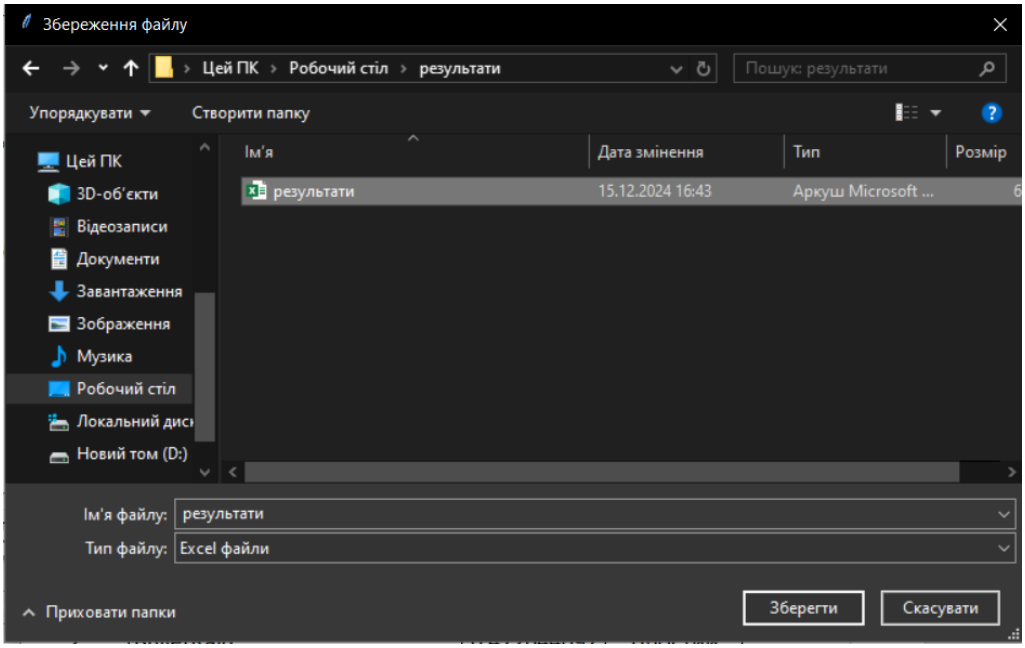


Рисунок Д.11 - Віконна форма для збереження .xlsx файлу отриманих результатів

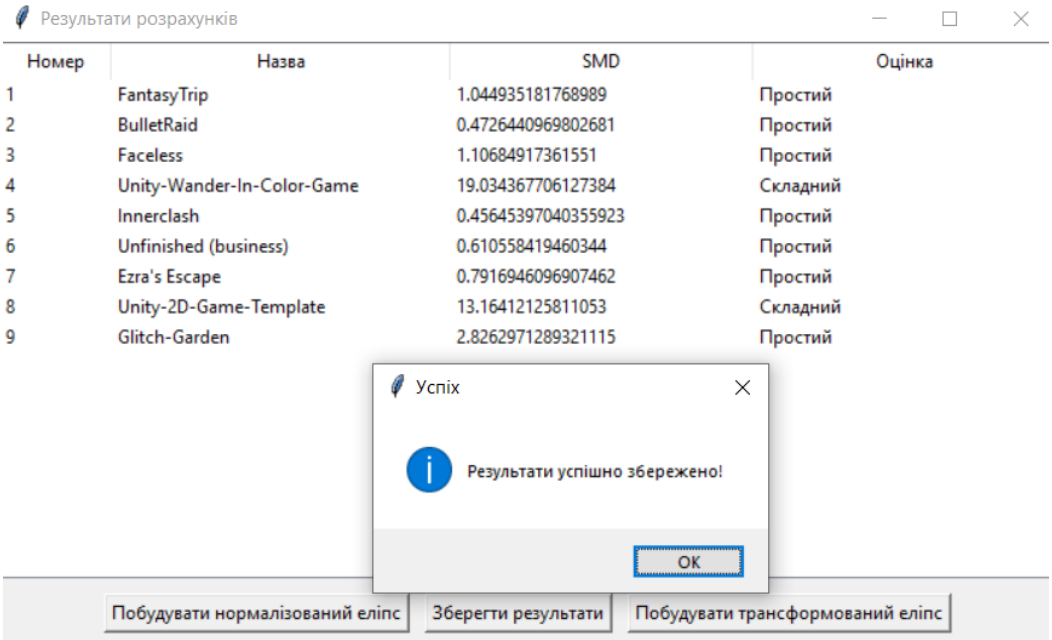


Рисунок Д.12 - Повідомлення про успішне збереження .xlsx файлу з результатами оцінювання складності об’єктно-орієнтованого проектування через зв’язки між класами для досліджуваних проектів

Зразок файлів, які були створені в процесі роботи програми наведено на рисунку Д.13. Структуру збережених результатів в .xlsx файлі наведено на рисунку Д.14.

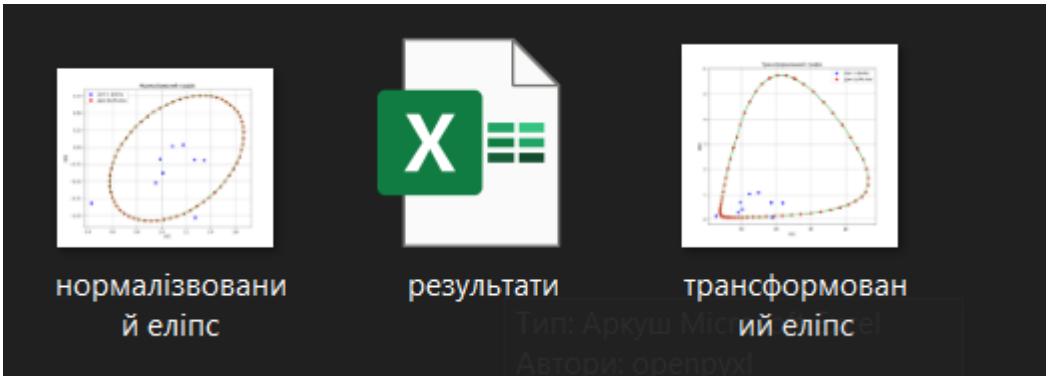


Рисунок Д.13 – Файли які були створені в процесі роботи програми

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Номер** | **Назва** | **SMD** | **Оцінка** | |
| 2 | 1 | FantasyTrip | 1,044935182 | Простий | |
| 3 | 2 | BulletRaid | 0,472644097 | Простий | |
| 4 | 3 | Faceless | 1,106849174 | Простий | |
| 5 | 4 | Unity-Wander-In-Color-Game | 19,03436771 | Складний | |
| 6 | 5 | Innerclash | 0,45645397 | Простий | |
| 7 | 6 | Unfinished (business) | 0,610558419 | Простий | |
| 8 | 7 | Ezra's Escape | 0,79169461 | Простий | |
| 9 | 8 | Unity-2D-Game-Template | 13,16412126 | Складний | |
| 10 | 9 | Glitch-Garden | 2,826297129 | Простий | |
| 11 | | | | | |
| 12 | | | | | |

Рисунок Д.14 – Структура збережених в .xlsx файл отриманих результатів

Для випробування програми треба використовувати набір проектів з таблиці 2.5