Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка та тестування програмного забезпечення онлайн-застосунку для балансування відеоігор»

Виконав: студент IV курсу, групи СП-43
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Собків О.М..
(прізвище та ініціали)

Керівник (підпис) Бачинський М.В.
(прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М.
(прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р.
(прізвище та ініціали)

Рецензент (підпис)
(прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Петрик М.Р.
(підпис) (прізвище та ініціали)
« 06 » квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Собківу Олегу Михайловичу

1. Тема роботи Розробка та тестування програмного забезпечення онлайн-застосунку для балансування відеоігор

Керівник роботи Бачинський Михайло Володимирович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від « 06» квітня 2026 року № 4/9-171

2. Термін подання студентом роботи 22.06.2026

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Огляд предметної області.
2. Концепція функціонування інструменту.
3. Технічна реалізація та практичне застосування.
4. Безпека життєдіяльності, основи охорони праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність, мета, задачі дослідження
3. Результати порівняння існуючих рішень. 4. Функціональні вимоги. .
5. Нотація для опису сценарію процесу запуску симуляції. Діаграма послідовності
6. Візуалізація роботи деяких вузлів. 7. Аналіз даних.
8. Програмні засоби та технології. 9. Архітектура інструменту.
10. ER діаграма бази даних. 11. Скріншоти вікон роботи розробленого ПЗ
12. Процес отримання та обробка даних. 13. JSON, згенерований на основі прототипу..
14. Висновки по роботі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання _____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної галузі дослідження	06.04–12.04.26	Виконано
2.	Обґрунтування актуальності дослідження	13.04–26.04.26	Виконано
3.	Огляд існуючих механізмів та засобів	27.04–04.05.26	Виконано
4.	Проведення дослідження механізмів та засобів		
	опрацювання даних	05.05–10.105.26	Виконано
5.	Оформлення розділу «Огляд предметної області»	11.05–14.05.26	Виконано
6.	Оформлення розділу «Концепція функціонування		
	інструменту»	15.05–18.05.26	Виконано
7.	Оформлення розділу «Технічна реалізація та		
	практичне застосування»	19.06–23.05.26	Виконано
8.	Оформлення розділу «Безпека життєдіяльності,		
	основи охорони праці»	20.05–25.05.26	Виконано
9.	Нормоконтроль	06.06–15.06.26	Виконано
10.	Перевірка на плагіат	12.06–16.06.26	Виконано
11.	Попередній захист роботи	15.06–21.06.26	Виконано
12.	Захист кваліфікаційної роботи	24.06.26	

Студент _____ Собків О.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бачинський М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка та тестування програмного забезпечення онлайн-застосунку для балансування відеоігор // Кваліфікаційна робота освітнього рівня «бакалавр» // Собків Олег Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра програмної інженерії, група СП-43 // Тернопіль, 2026 // С. - 59, табл. - 1, рис. - 34, слайд. – 14, додат. - 1 , бібліогр. – 34.

Ключові слова: балансування, вузол, графічний редактор, ігрова механіка, інтерактивна дошка, прототипування, сценарій, JSON

Розроблено онлайн-інструмент для балансування відеоігор, який дозволить створювати прототипи, налаштовувати кількісні характеристики ігрових елементів, проводити симуляції, тестування та аналітику вибраних елементів.

У першому розділі здійснено огляд предметної області, представлені статистичні дані зростання кількості світових користувачів відеоігор. Наведено теоретичні моменти і підходи до балансування у відеоіграх. Проаналізовано можливості окремих програмних засобів для балансування відеоігор.

У другому розділі описана концепція роботи розроблюваного інструменту. Наведено сценарії його функціонування із діаграмами послідовностей. Виділені вузли із їх функціями, що наслідують основні процеси ігрової взаємодії. Представлено інструменти аналізування кількісних даних (загальних і в рамках однієї сесії).

У третьому розділ наведена технічна реалізація. Розроблено архітектуру, інтерфейси для авторизації і спільного функціонування, а також інтерактивної дошки. Розроблено модуль генерації. Інструмент дав змогу протестувати роботу прототипу і провести аналіз руху ресурсів, визначивши вади стратегії.

Четвертий розділ висвітлює важливі питання безпеки життєдіяльності та основ охорони праці.

ABSTRACT

Software development and testing of an online video game balancing application // Bachelor Thesis // Sobkiv Oleh // Ternopil Ivan Puluj National Technical University, Computer and Information Systems and Software Engineering Faculty, Department of Software Engineering, group SP-43 // Ternopil, 2026 // P. - 59, tab. - 1, fig. - 34, slid. - 14, annexes - 1, references - 34.

Keywords: balancing, node, graphic editor, game mechanics, interactive whiteboard, prototyping, script, JSON

An online tool for video game balancing has been developed, which will allow you to create prototypes, configure quantitative characteristics of game elements, conduct simulations, testing and analytics of selected elements.

The first chapter provides an overview of the subject area, presents statistical data on the growth of the number of global video game users. Theoretical points and approaches to balancing in video games are presented. The capabilities of individual software tools for balancing video games are analyzed.

The second chapter describes the concept of the developed tool. Scenarios of its operation with sequence diagrams are presented. Nodes with their functions that follow the main processes of game interaction are highlighted. Tools for analyzing quantitative data (general and within one session) are presented.

The third chapter presents the technical implementation. The architecture, interfaces for authorization and joint functioning, as well as an interactive board, have been developed. A generation module has been developed. The tool made it possible to test the operation of the prototype and analyze the movement of resources, identifying the shortcomings of the strategy.

The fourth chapter highlights important issues of life safety and the basics of occupational health and safety.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

Балансування гри - це процес підбору та налаштування кількісних параметрів механіки чи економіки в ігровому проекті з метою забезпечення рівноваги між різними елементами гри.

Геймдизайн - це процес створення та визначення основних елементів, правил, змісту ігрового процесу гри, що розробляється.

Ігрові механіки - це операції з різними ігровими елементами, які гравець збирає, використовує та продає (внутрішньоігрові ресурси: валюта, боєприпаси, показники здоров'я, сили заклинань та інші подібні елементи).

Ігровий прототип – схематична версія гри, створена для тестування та демонстрації основних ігрових елементів, механік та ідей до остаточного випуску продукту.

API (англ. Application Programming Interface) - це набір правил та інструментів (класів, процедур, функцій, структур чи констант), який визначає способи та формати обміну даними між різними програмами чи компонентами програмного забезпечен**ня.**

JSON (англ. JavaScript Object Notation) - це стандартний текстовий формат для зберігання та передачі даних, що базується на JavaScript**.**

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Створення прототипів ігрових процесів, включаючи механіку та економіку, є важливим етапом у створенні гри, оскільки дозволяє розробникам оцінити концепцію, протестувати геймплей і значно прискорити процес розробки. Особливу увагу слід приділити визначенням цих процесів. Під ігровою економікою розуміється організація та контроль за розподілом ресурсів у віртуальному світі. Поняття ігрової механіки має на увазі більш детальне розуміння.

Ігрова механіка описується як взаємодія гравців та інших фундаментальних частин гри, включаючи правила, завдання, цілі, дії, стратегії та ігрові стани. Особливу увагу слід приділити етапу досягнення балансу у взаємодії ігрових елементів. Некоректні значення кількісних параметрів, таких як кількість ресурсів або вартість предметів, можуть призвести до неправильної роботи гри і, зрештою, до повної втрати інтересу та відтоку користувачів.

Особливу увагу потрібно звернути на процес побудови збалансованого прототипу відеоігри. Для оптимізації кількісних параметрів необхідно використовувати методи комбінаторики та теорії ймовірностей. Процес балансування є трудомістким та ресурсозатратним. Тому розробка інструменту, що надає функціонал для прототипування та балансування гри, залишається актуальним завданням.

Метою роботи є розробка онлайн-інструменту для балансування відеоігор. Основні функції такого інструменту:

– створення схематичних прототипів для імітації ігрових процесів;

– симуляція та тестування ігрових сесій з різними кількісними параметрами;

– аналіз отриманих даних, балансування та експорт результатів.

Були сформульовані завдання, виконання яких необхідне для досягнення поставленої мети:

– вивчення можливих альтернативних рішень із подібним

функціоналом;

– опис концепції роботи інструменту;

– опис візуальної мови для створення та редагування ігрових прототипів з використанням JavaScript-бібліотек;

– розробка онлайн-інструменту, що надає інтерфейс користувача для створення, тестування та аналізу ігрових прототипів;

– розробка модуля генерації схем прототипів з урахуванням специфікації JSON.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

В умовах швидкого зростання конкуренції та підвищення вимог до якості ігрових продуктів, прагнення до створення високоякісних ігор при мінімальних часових та фінансових затратах стає все більш актуальною потребою, тому впровадження методів, що прискорюють процес розробки, таких як візуальне прототипування, є необхідною умовою для забезпечення конкурентоспроможності на ринку [2].

1.1 Статистична інформація

На основі даних аналітичної компанії Newzoo за 2025 рік, світовий ринок відеоігор демонструє стрімкий розвиток, залучаючи все більше гравців і забезпечуючи значні доходи даної індустрії [3]. Дослідження також свідчать про те, що до кінця 2025 року кількість користувачів відеоігор у світі перевищила 3,2 мільярди, що становить середньорічний темп зростання в 5,6% з 2015 року (рисунок 1.1).

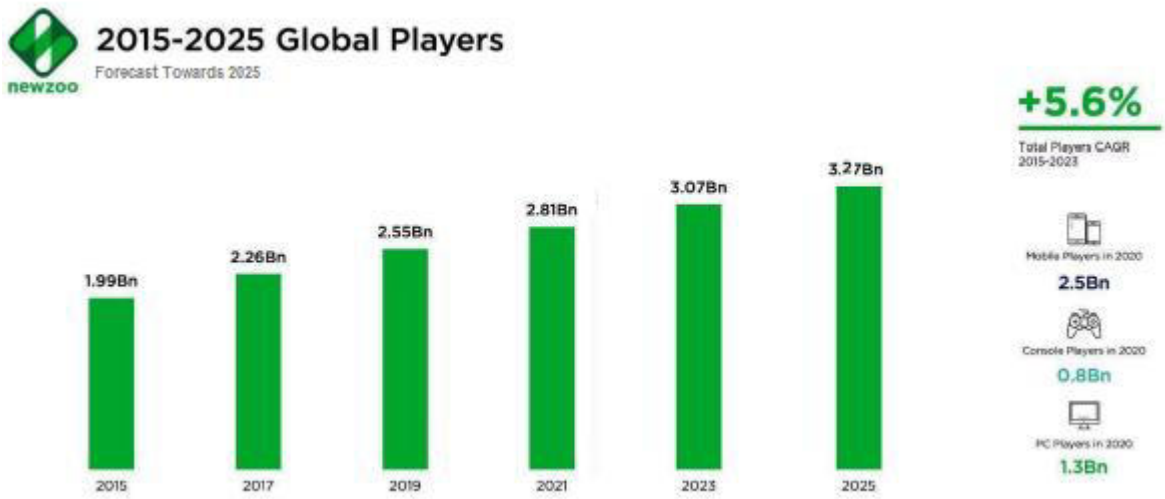


Рисунок 1.1 – Діаграма зростання користувачів відеоігор

За даними звіту організації з дослідження ринку Future Market Insights очікує,

що ринок відеоігор матиме колосальний середньорічний темп зростання в 13,5% протягом прогнозованого періоду з 2026 по 2033 рік [3].

1.2 Теоретичні основи балансу у відеоіграх

На сьогоднішній день поняття ігрового балансу має розмите визначення через відносну новизну області геймдизайну [4]. Основна мета будь-якої гри полягає у наданні гравцю ресурсів та стимулюванні його до їх використання, а завдання геймдизайнера полягає у визначенні необхідної кількості ресурсів, яку гравець повинен отримувати та витрачати, щоб досягти мети у контексті гри або її рівня. Були сформульовані різні погляду поняття балансу [5, 6].

Наприклад, у роботі [7] проведено аналіз концепції ігрового балансу з урахуванням інтерв'ю з ігровими дизайнерами. У ході дослідження автори наголошують на важливості досягнення гармонії між елементами інтерактивності, сюжету та ігрового прогресу, акцентуючи увагу на формування якісного ігрового досвіду.

У роботах [8, 9] концепція ігрового балансу розглядається не в контексті наративу та сюжету, а з більш технічної точки зору. Основна увага приділяється налаштуванню кількісних параметрів для врівноваження ігрових процесів.

В результаті аналізу вищезгаданих джерел пропонується наступне визначення: балансування відеоігор є процесом детального налаштування кількісних параметрів ігрових елементів, спрямований на створення справедливого ігрового досвіду. Головна мета балансування полягає в гарантуванні функціональної реалізації концепцій геймдизайну, при цьому забезпечуючи привабливість і цікавість гри для користувачів.

1.3 Підходи до балансування у різних ігрових жанрах

Перед тим як розпочинати балансування відеоігри, слід звернути увагу на її категорію та жанр, оскільки це визначає стиль ігрового процесу та цілі гравця.

Виділяють дві основні категорії:

– відеоігри, які розраховані на багато користувачів;

– однокористувацькі відеоігри.

Ті, що розраховані на багато користувачів, ґрунтуються на соціальній взаємодії, конкуренції та виконанні спільних завдань.

Однокористувальницькі надають розповідь, яка зосереджена на особистих досягненнях та розвитку персонажа.

Розглядаючи концепцію балансу в іграх, корисно звернутися до матеріалів [10]. У контексті однокористувальних ігор, основна увага приділена рівню складності — кожен наступний рівень має підвищувати складність, стимулюючи гравця до зростання його навичок та здібностей.

При дослідженні аспектів впливають на баланс у відеоіграх, котрі розраховані на багатьох користувачів, автор приділяє увагу визначенню переваг різних сторін, коли гравці отримують різне озброєння або вибирають персонажів різних класів. Гра вважається збалансованою, коли гравці мають рівні права та відсутні елементи, що надають комусь перевагу.

Вивчивши категорії ігор, слід перейти до аналізу жанрів. Виділяються такі види:

– рольові ігри – характеризуються прагненням гравців до постійного збільшення навичок та можливостей персонажа шляхом прогресу та розвитку;

– шутери від першої особи (FPS) - представляють жанр, де основна увага приділяється навичці прицілювання та стратегічному розміщенню персонажа. Гравці можуть очікувати на різноманітні варіанти зброї, тактик ігрового процесу та механік відродження персонажа;

– настільні ігри – це жанр, у якому ігровий процес моделюється на віртуальному столі чи дошці. Тут ключовими елементами є певні умови для перемоги та гнучкість правил, які можуть бути створені чи порушені у рамках ігрового процесу;

– симулятори – це жанр відеоігор, що створюється для імітації певного аспекту реального світу. Симулятори прагнуть передати гравцеві відчуття участі у

певному процесі, зазвичай шляхом відтворення реалістичних умов та правил.

Вивчення кожної категорії та жанру відеоігор сфокусовано на розумінні їхньої специфіки та особливостей, включаючи збалансованість ігрового процесу.

1.4 Альтернативні інструменти

Перед початком розробки інструменту необхідно провести дослідження існуючих аналогів для визначення їх переваг та недоліків, виявлення аналізу підходів до прототипування та балансування відеоігор, визначення ігрових жанрів, на які вони орієнтовані. Необхідно встановити ключові характеристики, які дозволять інструменту, що розробляється, успішно конкурувати з вже існуючими рішеннями.

Були виділені такі інструменти: табличні рішення Google Sheets та Excel, інструменти Miro та Machinations. Багато геймдизайнерів також віддають перевагу аркушу паперу та ручці для створення креслень, що підкреслює актуальність даної розробки. Розглянемо інструменти докладніше.

Безкоштовні інструменти для обробки даних у вигляді таблиць, таких як Excel та Google Sheets, надають функціонал для аналізу. Вони мають широкий спектр інструментів для роботи з даними, проте їх функціональність обмежена через їх математично орієнтовану природу. Цей висновок було зроблено з урахуванням досвіду професійних геймдизайнерів [11, 12].

Таблиці мають ряд переваг при використанні в процесі балансування відеоігор:

– балансування окремих параметрів - дозволяють керувати чисельними значеннями, такими як здоров'я персонажів, рівень складності тощо;

– формули та макроси - гнучкість при визначенні взаємозв'язків між параметрами гри та автоматизації певних процесів;

– різні жанри - можуть бути використані для балансування параметрів у рольових іграх, шутерах від першої особи, настільних іграх та симуляторах.

Табличні рішення ефективні з невеликою кількістю параметрів для простих

ігор. При підвищенні складності та збільшенні числа параметрів, залучених до процесу, геймдизайнерам потрібно створювати додаткові структури таблиць для полегшення розрахунків. Отже, для проектів підвищеної складності виникає потреба у більш інтуїтивно зрозумілих та візуально орієнтованих інструментах.

Інструмент Miro – це нескінченна інтерактивна онлайн-дошка, яка ефективно застосовується для візуального проектування рівнів та спільної роботи над проектами, але обмежена в аналізі даних. Однак варто виділити низку переваг інструменту:

– засоби візуалізації - надається широкий ряд інструментів для побудови графіків, різних діаграм, текстової інформації та інших елементів;

– спільна робота - команди можуть працювати в реальному часі, забезпечуючи онлайн-взаємодія між учасниками.

Miro може бути корисним для балансування параметрів у жанрах настільних ігор та симуляторів. У настільних іграх він може використовуватися для створення ігрових дощок, а також візуалізації ігрових правил і механік. Також інструмент може допомогти у створенні віртуальних світів, відображенні даних про статистику.

Miro зручний у використанні, коли акцент робиться на обговоренні та створенні концептуальних картин, але в ньому немає основного функціоналу для розрахунку та аналізу даних.

Станом на зараз Machinations [13] вважається одним із найбільш популярних інструментів для балансування параметрів у рольових іграх та симуляторах.

Результати недавнього дослідження показали, що Machinations є єдиним прикладним інструментом для створення прототипів ігрових механік поза ігровими рушіями [5]. Крім того, у дослідницькій роботі було проведено порівняння двох методів роботи: документування вимог до поведінки персонажу гри, який не знаходиться під контролем гравця та створення динамічних діаграм у Machinations. Результати експерименту показали, що використання цього інструменту суттєво скорочує загальний час роботи над проектом та дозволяє уникнути необхідності створення об'ємних текстових документів. Однак, незважаючи на переваги

Machinations, має ряд недопрацювань, виявлених в ході роботи, що підкреслюють дослідження [14, 15, 16]. В таблиці 1.1 представлено результати проведеного аналізу.

Таблиця 1.1 - Аналіз інструменту Machinations

Необхідний функціонал	**Реалізація інструментів**	**Складнощі**
Задання часу користувача для часового кроку	Не передбачено	Обмежує налаштування швидкості симуляції
Максимальна кількість кроків для тестування	100 кроків	Обмежує можливості аналізу тривалих процесів
Валідація полів для внесення значень	Не передбачено	Призводить до помилок тестування
Експорт у JSON	Не передбачено	Обмежує роботу за межами платформи
Видалення елементів у прототипі	Не переписує ланцюжок з’єднань	Необхідність в зміні схеми вручну
Робота з дробовими значеннями	Не передбачено	Потребує додаткових маніпуляцій
Відміна останньої дії	Не передбачено	Необхідність в зміні схеми вручну

При розробці власного інструменту слід враховувати функціональні можливості Machinations та його недоліки. Також запропоновано впровадити можливість генерації прототипу на основі готової специфікації у форматі JSON, що описує ігрову механіку чи економіку. Цей функціонал дозволив би користувачам у рази прискорити процес створення прототипу, особливо у випадку готової специфікації, яку можна було б легко вставити в редактор інструменту.

Таким чином, проаналізувавши існуючі альтернативи, можна зробити висновок про те, що властиво на сьогоднішній день існує лише один популярний інструмент, який здатний як проектувати, так і балансувати ігрові процеси в реальному часі. Однак цей інструмент має значні недопрацювання. Існуючі рішення мають ряд обмежень, таких як недостатня гнучкість у налаштуваннях параметрів та обмежені можливості для симуляції та експорту даних (рисунок 1.2).

Рисунок 1.2 – Результати порівняння існуючих рішень

1.5 Висновки до першого розділу

В цьому розділі виконано огляд предметної області, зокрема наведена статистика зростання користувачів відеоігор по 2025 рік включно із прогнозуванням до 2033 року.

Описано теоретичні аспекти балансу у відеоіграх та підходи до балансування у різноманітних ігрових жанрах.

Здійснено аналіз можливостей і характеристик окремих інструментів і засобів для балансування відеоігор.

2 КОНЦЕПЦІЯ ФУНКЦІОНУВАННЯ ІНСТРУМЕНТУ

Пропонований інструмент поєднуватиме основні функції для прототипування, підбору кількісних значень ігрових елементів та тестування, забезпечуючи наступні можливості (функціональні вимоги до розробки):

– налаштовує кількісні параметри в процесі створення або симуляції прототипів, що дозволить регулювати рух різних ігрових ресурсів;

– візуалізацію ігрових елементів у вигляді графів, що складаються з вузлів та ребер між ними, що спростить сприйняття логіки гри;

– можливості для колаборації з іншими учасниками у реальному часі, що забезпечить спільну взаємодію під час опрацювання ігрових процесів, дозволяючи вносити зміни синхронно;

– тестування в реальному часі, що дозволить спостерігати за змінами динаміки гри безпосередньо під час симуляції прототипу;

– генерація прототипів на основі JSON та подальше редагування ігрової механіки на основі цих даних, що скоротить час ручного проектування;

– аналітичні інструменти дозволять вивчати реалізованість та складність ігрових стратегій, а також оптимізувати розподіл;

– експорт даних у форматі JSON забезпечить швидке перетворення прототипів у код та їх використання у тестах, документації з дизайну гри (GDD) та подальшої розробки.

– експорт схеми прототипу у форматі PNG забезпечить наочне представлення процесу у презентаціях, обговореннях з командою та візуальної документації щодо проекту.

2.1 Сценарій функціонування інструменту

На основі розробленого концепту функціонування інструменту було сформовано сценарій взаємодії користувача із системою, який наведено нижче.

Аавторизація та реєстрація: користувач реєструється та входить до системи

через електронну пошту або з використанням облікового запису на Google чи GitHub.

Вибір команди та робочої дошки: після входу в систему користувач переходить на особисту сторінку, де вибирає команду та дошку для початку роботи. Користувач може надіслати запрошення іншим учасникам, вказавши їхню пошту.

Вибір режиму прототипування: користувач вибирає режим побудови прототипу – ручний або генерація.

Побудова та редагування: користувач створює прототип або редагує згенерований прототип, перетягуючи вузли з панелі інструментів та встановлюючи зв'язки між ними на інтерактивній дошці. У цьому вводяться кількісні характеристики щодо симуляції

Налаштування параметрів для аналізу: користувач встановлює кількість ітерацій, час у секундах та кількість ігрових сесій для запуску симуляцій шляхом заповнення відповідних полів на інтерактивній панелі (рисунок 2.1).

Аналіз результатів: після завершення симуляції користувач проводить аналіз отриманих значень шляхом вивчення графіків і підбирає необхідні кількісні характеристики для подальшої роботи. Шляхом зміни параметрів для аналізу користувач повторює процес до отримання потрібних результатів.

Експорт даних: користувач може експортувати код у форматі JSON або схему розширення PNG для подальшого процесу розробки.

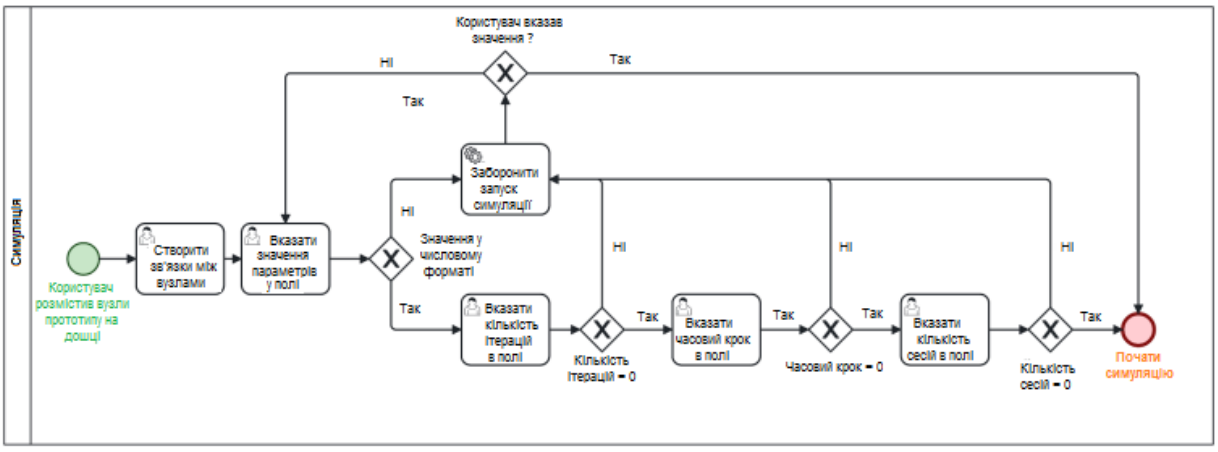


Рисунок 2.1 – BPMN -нотація для опису процесу запуску симуляції

У цей час інструмент працює за таким сценарієм.

Аутентифікація: інструмент перевіряє облікові дані, у разі успішної аутентифікації надає доступ до основного функціоналу інструменту (рисунок 2.2). Інакше пропонує форму реєстрації.

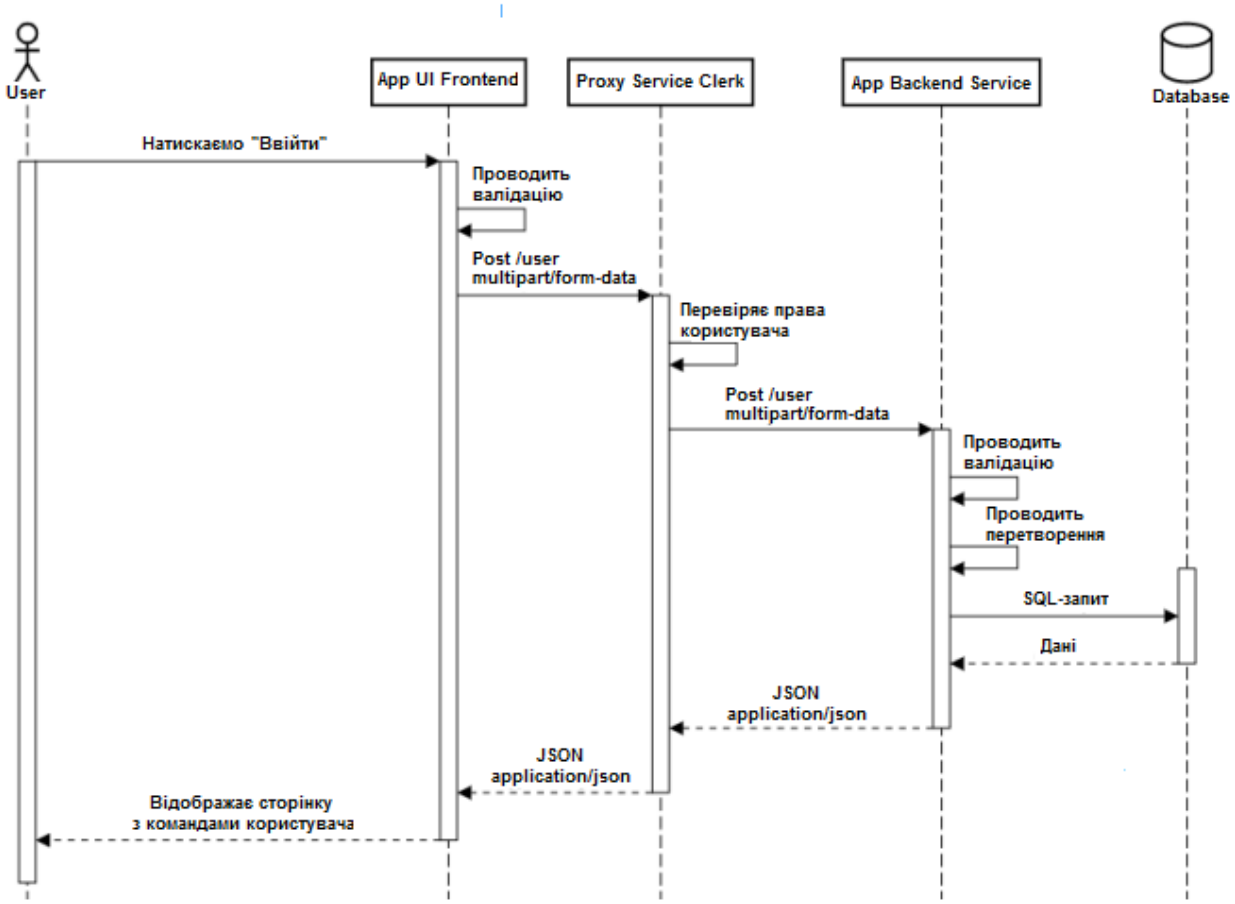


Рисунок 2.2 – Діаграма послідовності процесу аутентифікації

Команди та особисті дошки: при вході інструмент пропонує вибрати команду та дошку з існуючих або створити нову, додати дошку до обраного. Система надсилає GET-запит у систему, зіставляючи дані користувача з доступними йому командами та дошками.

Вибір режиму: інструмент реагує на запит користувача про вибір режиму побудови прототипу, надавши можливість вибору між інтерактивною дошкою та кодовим редактором.

Створення прототипу: інструмент відображає панель інструментів на дошці

для створення вузлів. Кожен вузол має різні функціональні можливості. Інструмент дозволяє змінювати їх розмір, встановлювати назви, змінювати типи з'єднань, а також надає кнопки скасування останніх дій.

Налаштування параметрів для аналізу: інструмент зберігає введені значення користувачів у стейтменеджері та запускає симуляцію прототипу із заданими параметрами через встановлений інтервал.

Виконання симуляції: система тимчасово зберігає дані після кожної ітерації симуляції до стейтменеджера. Після завершення кожної ігрової сесії вона надсилає POST-запит на сервер, надаючи інформацію про значення вузлів на кожному кроці ігрової сесії, вся інформація зберігається у базі даних.

Для розуміння необхідно з’ясувати визначення понять ігрової сесії та ітерації. Одна ігрова сесія - це сукупність ітерацій, де кожна ітерація є окремим проходом ресурсів по всьому графу. Одна ігрова сесія еквівалентна одній зіграній користувачем грі. Кількість вказаних користувачем ігрових сесій відображає можливі кінцеві значення вузлів після кількох зіграних ігор.

Аналіз значень: після вибору користувачем вузла шляхом натискання на ньому інструмент відправляє GET-запит на сервер з ідентифікатором потрібного вузла. Система зіставляє значення запиту з інформацією базі даних і відправляє дані клієнтові, клієнт, зі свого боку, відображає отримані дані як діаграму послідовності (рисунок 2.3).

Експорт даних: інструмент надає користувачеві можливість аналізувати отримані результати та вивантажувати код у JSON або схему в PNG.

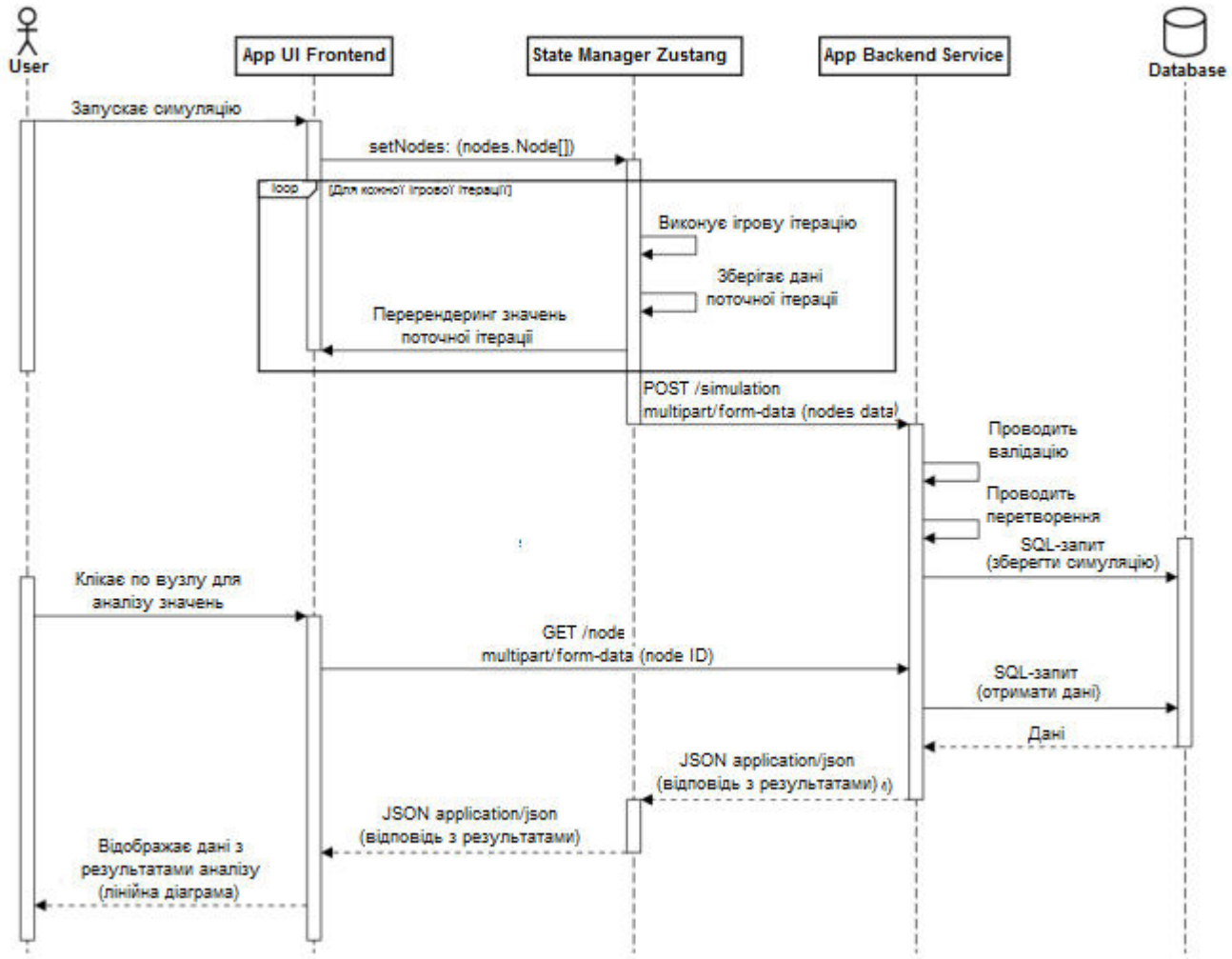


Рисунок 2.3 – Діаграма послідовності аналізу значень

2.2 Опис візуальної мови

У сучасній практиці візуальне проектування активно використовується з метою побудови моделей у різноманітних областях діяльності, а також для опису роботи різних систем. За результатами дослідження [17], найчастіше використовуваними візуальними мовами проектування є UML, IDEF, BPMN.

Автори поділяють мови візуального проектування на дві категорії:

– універсальні;

– предметно-орієнтовані.

Універсальні мови позбавлені спеціалізованої орієнтації і підтримують позначення, що використовуються. На думку дослідників, функціональність даних мов у конкретній галузі стає обмеженою та розмитою. Предметно орієнтовані мови

призначені для конкретних областей, використовуючи функціонал, адаптований під особливості проектованої області.

У рамках цієї роботи необхідно було створити візуальну мову, враховуючи специфіку ігрової продукції. Це має на увазі розробку предметно-орієнтованої мови з виділенням основних компонентів для моделювання ігрових процесів.

Використовуючи базу Machinations як основу, були виділені вузли з функціями, що імітують основні процеси взаємодії ігрових елементів, вони представлені на рисунку 2.4.

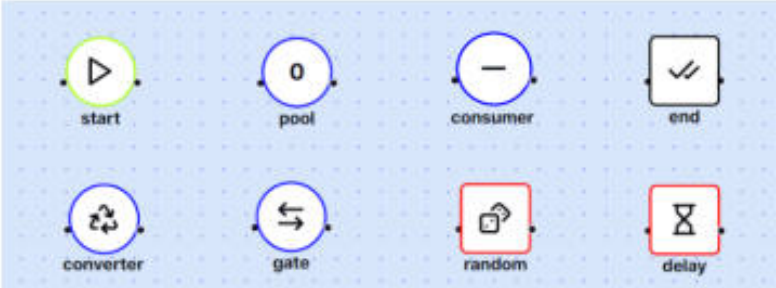


Рисунок 2.4 – Відображення основних елементів візуальної мови

Для кращого розуміння необхідно подати опис кожного вузла.

– Source (Виробництво): Генерує ресурси, необхідні для роботи інших вузлів.

– Pool (Сховище ресурсів): Служить для тимчасового зберігання ресурсів, щоб вони могли бути використані у подальшій передачі.

– Consumer (Споживання): Утилізує ресурси для виконання різних ігрових завдань.

– End (Кінцевий вузол): означає завершення всіх процесів у симуляції прототипу.

– Converter (Перетворення): Перетворює один вид ресурсу на інший.

– Gate (Вузол розподілу): регулює потік ресурсів між різними вузлами.

– Random (Подія на основі випадкового розподілу): Ініціює процеси на основі випадкових подій, вводячи елемент непередбачуваності.

– Delay (Вузол тимчасової затримки): Вводить тимчасову затримку перед

виконанням наступної дії.

Представлена мова має інтуїтивну зрозумілість, що спрощує її освоєння потенційними користувачами. Ясність і простота роблять її доступним широкому спектру учасників розробки.

2.3 Опис функціональності вузлів

Кожен створений вузол володіє своєю функціональністю, яка і визначає особливості його роботи. В цьому підрозділі описані функціональні можливості роботи вузлів.

2.3.1 Source node (вузол, котрий виробляє)

Стартовий вузол у графі представляється візуальним компонентом із двома ручками, через які можна встановити необмежену кількість з'єднань з іншими вузлами (рисунок 2.5).

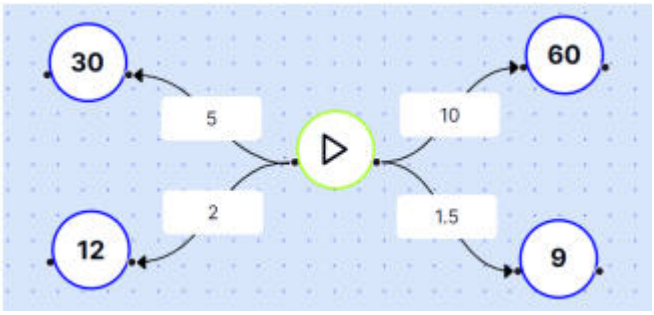


Рисунок 2.5 – Візуалізація роботи Source node

Без наявності стартового вузла неможлива симуляція прототипу. У рамках ігрового світу цей компонент є джерелом виникнення ресурсів. Кількість ресурсів, що виходять з компонента, вказується в ребрі з'єднання з наступним вузлом графа. Кількість таких стартових вузлів необмежена.

2.3.2 Pool node (сховище ресурсів)

Цей вузол виконує роль ключового елемента, забезпечуючи прийняття, збереження та передачі ресурсів. Особливістю вузла є можливість зберігання як цілих, так і дробових значень. За умовчанням значення пулу дорівнює нулю, демонструючи відсутність ресурсу. Візуалізація ресурсів у пулі здійснюється як числові значення, котрі можуть збільшуватись або зменшуватись в залежності від процесів, що проектуються в системі.

2.3.3 Consumer node (вузол, котрий споживає)

Дані вузли здатні одночасно утилізувати ресурси з кількох джерел та видаляти їх із ігрової економіки. Крім того, можливо встановити відсоткове відношення видалення, яке вказується на сполучному ребрі.

Як приклад, розглянемо прототип процесу гри, де гравець керує здоров'ям свого персонажа. Вузол "Damage" - шкода пов'язана з ресурсом "Health" - здоров'я. Якщо встановлено відсоткове значення на зв'язку, наприклад, 10%, це означає, що кожного разу, коли персонаж отримує шкоду, буде знищено 10% його поточного здоров'я (рисунок 2.6).

Рисунок 2.6 – Візуалізація роботи Consumer node

2.3.4 Converter node (вузол, котрий перетворює)

Цей вузол виконує функцію перетворення ресурсів із одного типу на інший. Функціональність вузла залежить від вхідних та вихідних ребер. Його завдання - приймати кількість ресурсів з одного сховища та передавати їх до іншого зі зміненим значенням. Цей процес здійснюється за одну ітерацію. Вузли, які перетворюють використовуються для моделювання різних процесів, наприклад, виробничих, де сировина або інші матеріали перетворюються на готову продукцію.

На рисунку 2.7 представлений процес будівництва будинку, в якому ресурси

"гроші" ("money"), "камінь" ("stone") та "дерево" ("wood") за допомогою вузла "building" перетворюються на готовий будинок

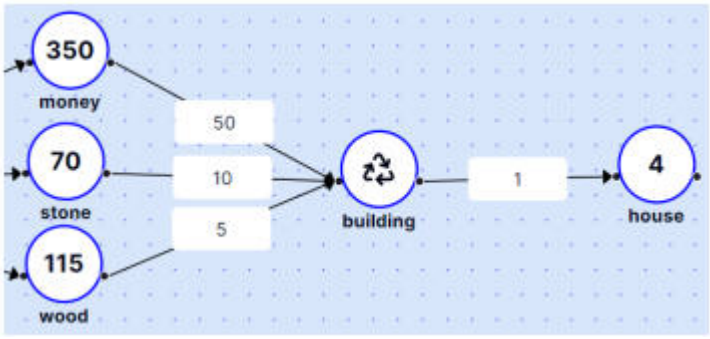


Рисунок 2.7 – Візуалізація роботи Converter node

2.3.5 Gate node (вузол розподілу)

Він приймає кількість ресурсів, що протягом однієї ігрової ітерації переноситься до іншого вузол. Цей компонент може мати лише одне вхідне ребро і як мінімум два вихідні ребра, максимальна кількість не обмежена.

Вузол розподілу функціонує так: кількість ітерацій, протягом яких ресурс буде направлений у кожен вузол, вказується на вихідних ребрах.

Допустимо, існує вузол розподілу з двома вихідними з’днаннями. Перше вихідне з'єднання має коефіцієнт 3, а друге – 1. Якщо за одну ігрову ітерацію у вузол-ворота надходить 1 одиниця ресурсу, то 3 наступні ітерації будуть направлені через перше вихідне з'єднання, 4 ітерація - через друге. Ітерації з 5 по 7 знову направлятимуть ресурс першим шляхом і так до закінчення ігрової симуляції (рисунок 2.8).

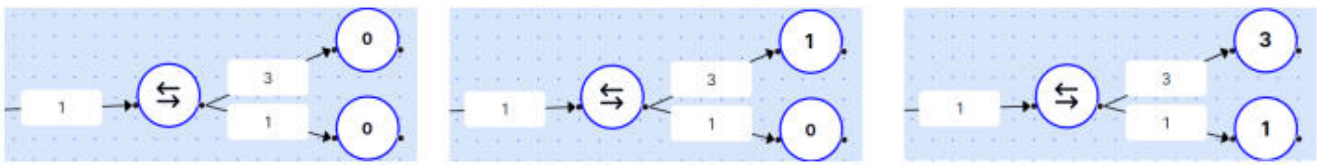


Рисунок 2.8 – Візуалізація роботи вузла розподілу

У контексті ігрових процесів цей вузол може бути розгалуженням в дорозі ігрових ресурсів або вибором шляху, який буде пройдений ігровим персонажем в

залежності від певних умов.

2.3.6 Random node (подія на основі випадкового розподілу)

При розробці даного вузла варто звернути увагу на поняття випадковості в іграх. Випадкові процеси, такі як кидання кубика або тасування карток, забезпечують непередбачуваність. Всі ігри вимагають певного рівня випадковості, щоби залишатися захоплюючими.

Згідно з думкою автора книг з ігрового дизайну, випадковість в іграх може бути класифікована на два основні типи: вхідну та вихідну [18]:

– вхідна випадковість (input randomness) впливає на стан гри до прийняття гравцем будь-яких рішень чи дій;

– вихідна випадковість (output randomness) впливає результат після того, як гравець зробив вибір.

Ігри, де відсутній будь-який елемент випадковості, називаються детермінованими. І тут результат повністю визначається діями гравця чи заздалегідь заданими умовами. Такі ігри характеризуються передбачуваним результатом.

В ігровій розробці вхідна випадковість відіграє ключову роль у створенні різноманітних сценаріїв, особливо в одиночних іграх. У інструменті, що реалізується, доступна можливість імітації випадкових процесів з упором на вхідну випадковість.

Для більш точного моделювання реальних умов гри було розроблено вузол, який дозволяє випадково розподілити ресурси (рисунок 2.9).

Цей вузол дозволяє встановити кількість ресурсних одиниць для розподілу на вході. На виході він зв'язується з іншими елементами, які приймають ресурси з різною ймовірністю. З'єднання, що виходять із цього компонента, показують пропорції розподілу ресурсів. Наприклад, при розподілі 10 одиниць ресурсу, 1/10 одиниці перенаправляється по першому з'єднанню, 2/10 - по другому та третьому, а 5/10 - по четвертому.

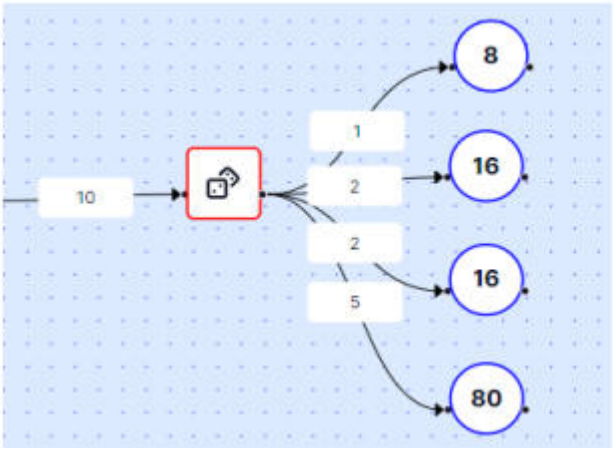


Рисунок 2.9 – Візуалізація роботи вузла для випадкового розподілу

Алгоритм випадкового розподілу ресурсів працює наступним чином.

На першому етапі обчислюється ймовірність відправки ресурсу по кожній із гілок. Для цього спочатку підсумовуються значення всіх ребер, щоб отримати загальне значення, доступне для розподілу. Потім кожне значення гілки поділяється на загальне значення, щоб визначити можливість для кожної гілки. Імовірності обчислюються за формулою (2.1)

$$p_i = \frac{e_i}{\sum_{i=1}^{n} e_i}, \tag{2.1}$$

де p_i. - ймовірність відправки ресурсу по і -й гілці, e_i - значення і -го ребра.

З урахуванням обчислених ймовірностей алгоритм виконує такі дії для розподілу ресурсів:

1. Генерується випадкове число від 0 до кількості ресурсів, що залишилася.

2. Отримане число множиться на ймовірність гілки визначення кількості ресурсів, які будуть відправлені по цій гілки. Результат округляється в меншу сторону за допомогою функції Math.floor.

3. Певна кількість ресурсів віднімається від загальної доступної кількості

та відправляється поточною гілкою.

4. Процес повторюється кожної гілки.

Якщо після основного розподілу залишилися нерозподілені ресурси, вони поступово розподіляються між усіма гілками.

Додатково використовуються випадкові числа для забезпечення різноманітного і випадкового розподілу ресурсів, що залишилися.

Після завершення всіх етапів розподілу повертається масив, де кожен елемент є кількістю ресурсів, надісланих з кожної гілки.

2.3.7 Delay node (вузол тимчасової затримки)

У грі деякі події повинні займати більше однієї ітерації задля досягнення мети. Для цього використовується вузол тимчасової затримки, який зупиняє передачу ресурсів на певну кількість ітерацій.

Вхідне ребро компонента визначає кількість ресурсів, готових до передачі, а вихідне ребро показує кількість ітерацій, на які ресурси чекатимуть, перш ніж бути переданими далі за схемою (рисунок 2.10).

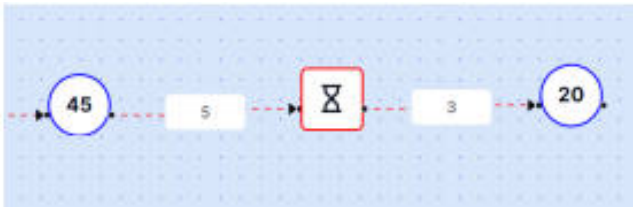


Рисунок 2.10 – Візуалізація роботи Delay node

Вузол забезпечує можливість затримки передачі ресурсів, що для реалізації багатокрокових ігрових подій.

2.3.8 End node (кінцевий вузол)

Як правило, гра завершується при досягненні гравцем встановленої мети. Кінцевий вузол використовується для завершення гри при настанні таких умов. Цей вузол приймає як вхідні дані тільки з'єднання станів. У ребрі вказується умова, при виконанні якої ігрова сесія скидається і починається заново без огляду на

кількість ітерацій зазначених користувачем у полі. Прикладом опису станів є вирази з використанням операторів порівняння (=, >, <) по відношенню до заданого числового значення.

Приклад: У стратегічній грі кінцевий вузол може бути налаштований на завершення гри, коли гравець накопичує 80 одиниць ресурсу (умова: = 80). Або гра може завершитися, якщо кількість ресурсу перевищує 100 одиниць (умова: > 100). За виконання цих умов ігровий процес завершується, всі значення скидаються, і починається нова ігрова сесія (рисунок 2.11).

Рисунок 2.11 – Візуалізація роботи кінцевого вузла

Кінцевий вузол є важливим для моделювання процесів, коли гра завершується при виконанні певних умов, проте його використання необов'язково при прототипуванні, на відміну від стартового вузла.

2.4 Опис інструментів для аналізу кількісних даних

У наукових статтях щодо досягнення балансу під час проектування ігрових процесів рекомендується починати з визначення ключових ресурсів [19, 20]. Застосування інструментів для власне аналітики дозволяє відстежувати сукупний потік ресурсів, допомагаючи виявити вузькі місця та надлишки.

У своїх лекціях з курсу "Принципи ігрового балансу" Ян Шрайбер розглядає кількісні дані як параметри, що відстежуються в грі, які можна використовувати для отримання необхідної інформації через статистичний аналіз [21]. Автор представив ряд інструментів, таких як середнє значення, медіана та середньоквадратичне відхилення.

У розробленому інструменті використовуються різні види діаграм для візуалізації потенційних наслідків ігрових процесів, він проводить обчислення на

основі даних симуляції та відображає результати на графіках. Ці обчислення проводяться як на основі всіх ігрових сесій загалом, так і на базі даних однієї ігрової сесії.

2.4.1 Інструменти аналізу загальних даних

Ці інструменти виконують обчислення, спираючись на дані про рух ресурсів на всіх ітераціях за кілька симульованих ігор.

До таких даних належать наступні.

1. Середнє значення вузла: корисне для загального розуміння стабільності розподілу ресурсів. Обчислення середнього значення відображає формула (2.2).

$$\overline{y} = \frac{\Sigma_{i=1}^{z}\Sigma_{j=1}^{x} y_{ij}}{z \cdot x}, \tag{2.2}$$

де z – кількість ігрових сесій, x – кількість ітерацій, зазначених користувачем, y_{ij} - кількість ресурсів, що проходять через вузол за час j-ї ітерації і-ї ігрової сесії.

2. Медіана: надає важливі відомості про наявність перекосу даних у більшу чи меншу сторону, або про їхню симетричність. Для обчислення медіанного значення упорядковуються всі значення у (кількість ресурсів для j-ї ітерації в і-й ігровій сесії) за зростанням та вибирається значення у середині упорядкованого списку.

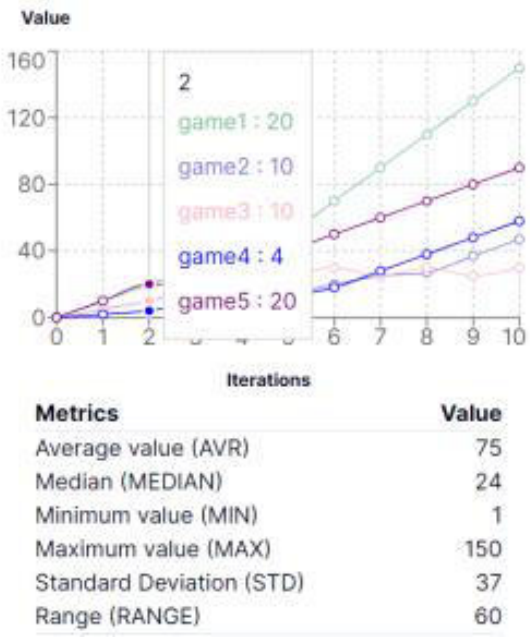
3. Максимальне та мінімальне значення.

4. Стандартне відхилення: оцінює розкид значень. Розраховується так: для кожного з показників вузла за 1 ігрову сесію віднімається середнє значення, результат підноситься до квадрату, після чого всі отримані значення сумуються і діляться на загальну кількість ігрових сесій, за формулою (2.3).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}, \quad (2.3)$$

де σ - середньоквадратичне відхилення, x_i - кожне із значень у вибірці, $\bar{x}$ – середнє значення вузлів за період всіх ігрових сесій, n – кількість ігрових сесій.

Різниця між максимальним та мінімальним значенням вузла допомагає виявити діапазон коливань. Загальна схема статистики роботи вузла показана на рисунку 2.12.



Metrics	Value
Average value (AVR)	75
Median (MEDIAN)	24
Minimum value (MIN)	1
Maximum value (MAX)	150
Standard Deviation (STD)	37
Range (RANGE)	60

Рисунок 2.12 – Графік, що відображає статистику значень вузла

2.4.2 Інструменти аналізу даних однієї сесії

В рамках аналізу однієї ігрової сесії передбачено обчислення наступних значень:

– середнє значення;

– медіана;

– максимальне та мінімальне значення.

Крім того, передбачено кругову діаграму, що відображає кількість ресурсів,

що надходять у вузол за одну ігрову сесію. Діаграма демонструє відсоткове співвідношення ресурсів, що проходять через вузол до загальної кількості ресурсів, згенерованих джерелами за період гри. Ці дані допомагають виявити вузькі місця у схемі, куди ресурс не надходить, що дозволяє виявити проблемні стратегії. Відображення статистики даних за одну сесію представлено на рисунку 2.13.

Metrics	Value
Average value (AVR)	80
Median (MEDIAN)	50
Minimum value (MIN)	10
Maximum value (MAX)	150

Рисунок 2.13 – Діаграми, що відображають статистику значень вузла

2.5 Висновки до другого розділу

У ході опису концепції роботи запропонованого інструменту наведені його основні можливості. Розроблено сценарії його функціонування, представлені діаграми послідовностей.

Застосовуючи базу Machinations як основу, були виокремлені вузли зі функціями, котрі імітують ключові процеси взаємодії ігрових елементів. Наведено їх опис функціональності із використанням предметно-орієнтованої мови моделювання. Описано інструменти для проведення аналізу кількісних даних, як загальних та і в межах однієї сесії.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ

Процес технічного втілення містить декілька складових частин, які і будуть представлені в цьому розділі. Для демонстрації роботи онлайн-інструменту було вирішено розробити модель економіки руху ресурсу на прикладі виробництва зброї.

3.1 Загальна архітектура

При проектуванні архітектури клієнт-серверного інструменту враховувалися можливості, які дозволяють користувачеві взаємодіяти з інтерактивною дошкою в режимі реального часу та отримувати інформацію без перезавантажень, а також імпортувати, редагувати, експортувати схеми. Розробка інтерактивних інструментів може супроводжуватися рядом типових проблем:

– тривале завантаження даних: важливо, щоб переходи між екранами та завантаження даних відбувалися швидко для скорочення затримок;

– повільне оновлення інтерфейсу: дані в програмі повинні оновлюватись непомітно для користувача.

Щоб уникнути зазначених проблем, було ухвалено рішення реалізації інструменту на основі підходу SPA. Single Page Application є веб-застосунком, який завантажує одну HTML- сторінку і динамічно оновлює її вміст у процесі взаємодії з користувачем. Архітектура інструмента представлена на рисунку 3.1.

Для реалізації SPA програми був обраний фреймворк Next.js [22], він підтримує просту інтеграцію з javaScript -бібліотеками для створення інтерфейсів користувача і використовує підхід BFF для взаємодії клієнтських і серверних компонентів через API. Це дозволяє попередньо завантажувати сторінки та асинхронно підвантажувати дані, забезпечуючи швидкі переходи та мінімізуючи тимчасові очікування.

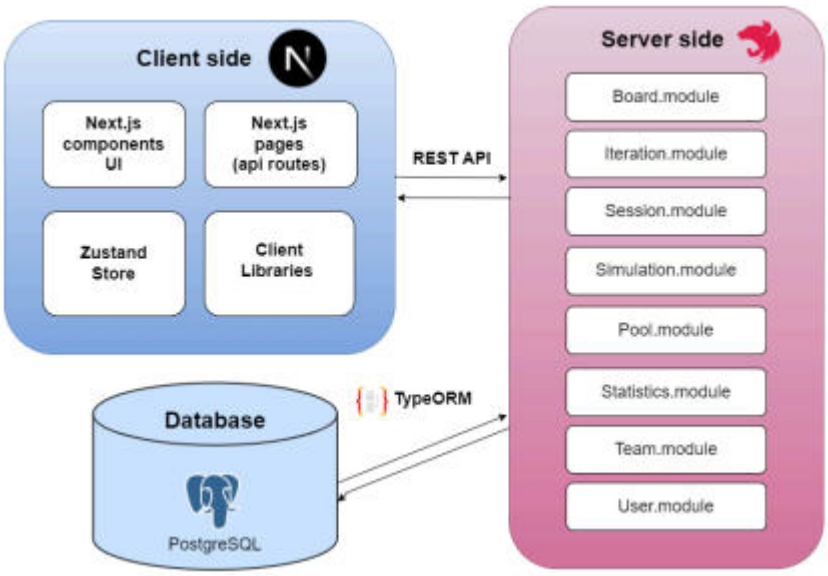


Рисунок 3.1 – Розроблена архітектура інструменту

Щоб реалізувати функціонал інтерактивної дошки, було вирішено використовувати рендеринг на стороні клієнта (CSR), який необхідний для динамічних оновлень у реальному часі без частих перезавантажень. Як менеджер пакетів був обраний npm.

Для глобального управління станом було ухвалено рішення скористатися бібліотекою Zustand [23]. Вона призначена для зберігання та оновлення тимчасових даних у процесі роботи програми. Завдяки мінімалістичному синтаксису, бібліотека дозволяє створювати модульні сховища даних для різних компонентів без громіздкої архітектури, забезпечуючи простоту використання та масштабованість.

Для виконання операцій із даними, взаємодії з базою даних, математичних обчислень та розрахунку статистики на серверній стороні було прийнято рішення використовувати фреймворк Nest.js [24]. Бізнес-логіка серверної частини розбита на модулі з провайдерами, контролерами та іншими компонентами, щоб забезпечити ізоляцію між ними та можливість масштабування. Для інкапсуляції використовуються послуги, які впроваджуються через механізм застосування залежностей. Серверна частина розпочинає свою роботу з отримання запиту від клієнтської сторони. Цей запит обробляється контролером, який потім

перенаправляє його у відповідний сервіс для виконання бізнес-логіки.

Для обробки та зберігання даних використовується PostgreSQL [25], до якої відбувається звертання із використанням TypeORM [26]. PostgreSQL забезпечує зберігання даних про стан ігрових елементів, одержуваних у ході симуляцій, щоб аналізувати та використовувати цю інформацію для статистичних розрахунків. Діаграма класів структури бази даних представлена на рисунку 3.2.

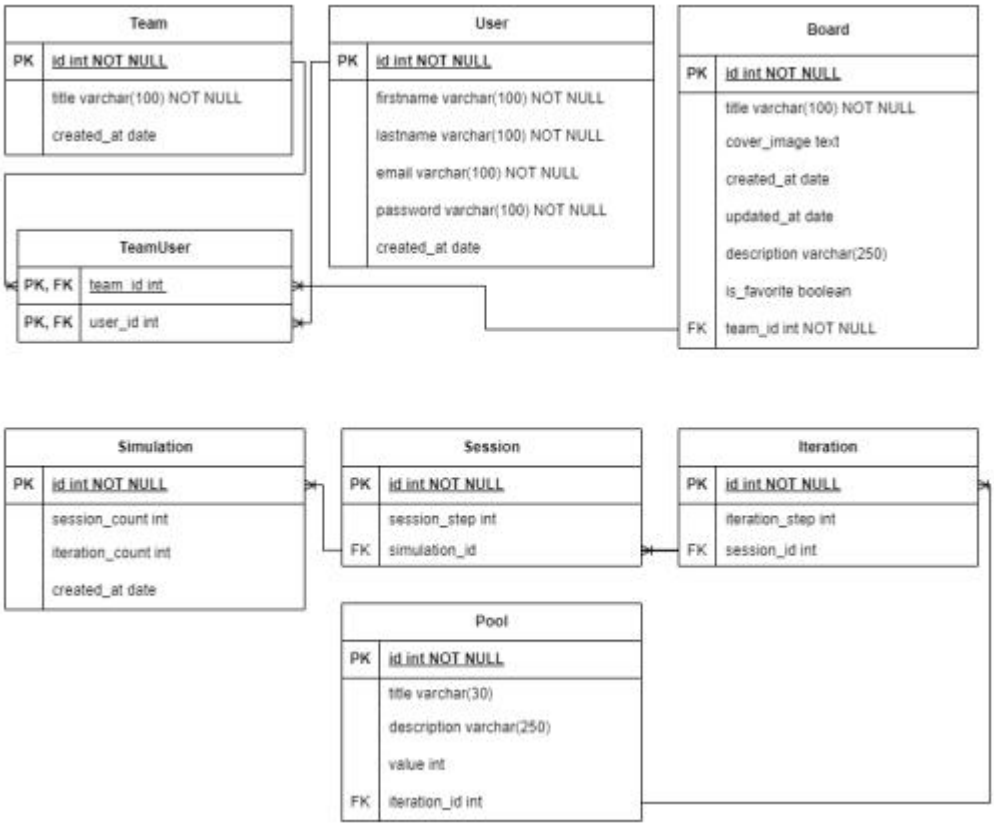


Рисунок 3.2 – ER діаграма бази даних

3.2 Розробка інтерфейсів для авторизації та спільної роботи

Для початку роботи з інструментом користувачеві надається форма реєстрації. Для спрощення реєстрації та спільної роботи інтегровано API інструменту Clerk [27], що надає широкий спектр ідентифікаційних провайдерів (Google, GitHub та електронна пошта). Clerk пов'язує облікові записи користувачів, автоматично прив'язуючи їх при вході через єдиний обліковий запис (рисунок 3.3).

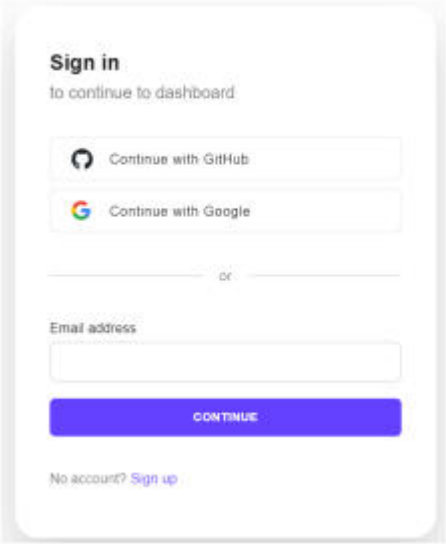


Рисунок 3.3 – Інтерфейс форми авторизації

В результаті успішної авторизації користувач відразу ж переходить на головну сторінку, власне яка є і інтерфейсом для роботи з різними командами та інтерактивними дошками. На цій сторінці на панелі зліва відображається список існуючих команд. Кожна команда має власний набір дощок. Користувач може створити нову дошку для прототипування або вибрати існуючу (рисунок 3.4).

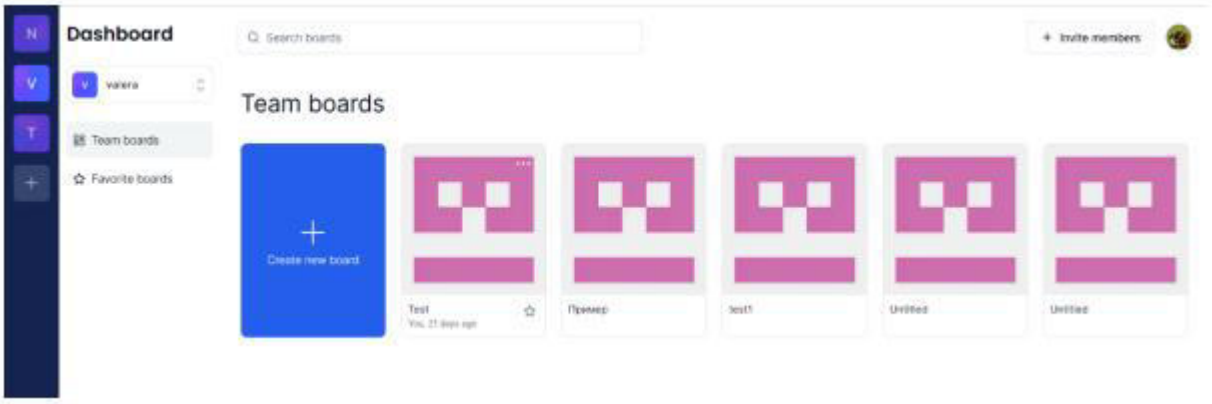


Рисунок 3.4 – Інтерфейс для роботи з інтерактивними дошками

Після створення або вибору дошки, користувач може надіслати запрошення іншому користувачеві для спільної роботи. Щоб забезпечити цей функціонал необхідно перерахувати пошти учасників через кому та вказати доступні ролі з можливих: учасник або адміністратор (рисунок 3.5).

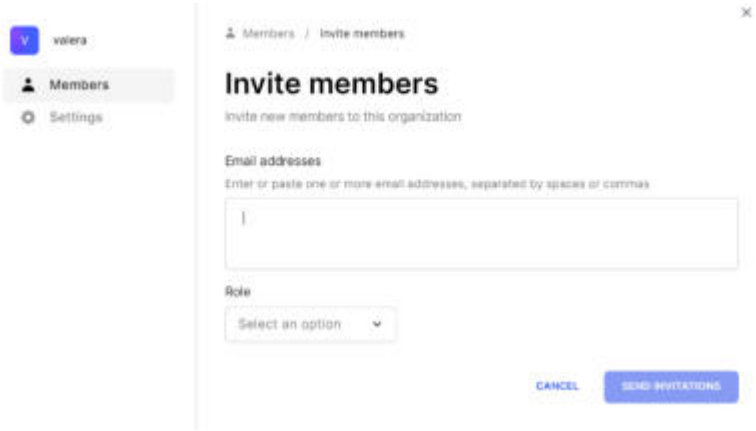


Рисунок 3.5 – Вікно для надсилання запрошення

Реалізований інструмент можна використовувати для колективної роботи в єдиному інтерактивному просторі завдяки бібліотеці Liveblocks [28], яка автоматично керує переходом в автономний режим та відновленням під'єднання після втрати зв'язку.

3.3 Розробка інтерфейсів інтерактивної дошки

Необхідно забезпечити програмне втілення інтерфейсів графічних елементів дошки.

3.3.1 Графічний редактор

Для розробки графічного редактора необхідно використовувати бібліотеку, за допомогою якої можна було б реалізувати архітектуру, що масштабується, на клієнтській стороні. У ході аналізу кількох варіантів були висунуті наступні критерії відбору:

– продуктивність (відображення великих обсягів даних без втрати швидкості виконання);

– сумісність з TypeScript (перевага віддавалася бібліотеці, що повністю підтримує типізацію, щоб уникнути непередбачених помилок у процесі розробки);

– документація та підтримка (необхідна наявність якісної документації та хорошої підтримки бібліотеки);

– гнучкість та масштабованість (необхідна можливість побудови гнучкої архітектури компонентів)

Виходячи з перерахованих вище критеріїв, для реалізації була обрана бібліотека ReactFlow [29], яка добре підходить для створення власних типів вузлів і ребер, а також для побудови архітектури, що масштабується.

Графічний редактор включає сітку, яка візуально допомагає розміщувати елементи на полотні. На дошці є панелі інструментів і кнопки, що забезпечують доступ до різних функцій. Загальний інтерфейс інтерактивної дошки представлено на рисунку. 3.6.

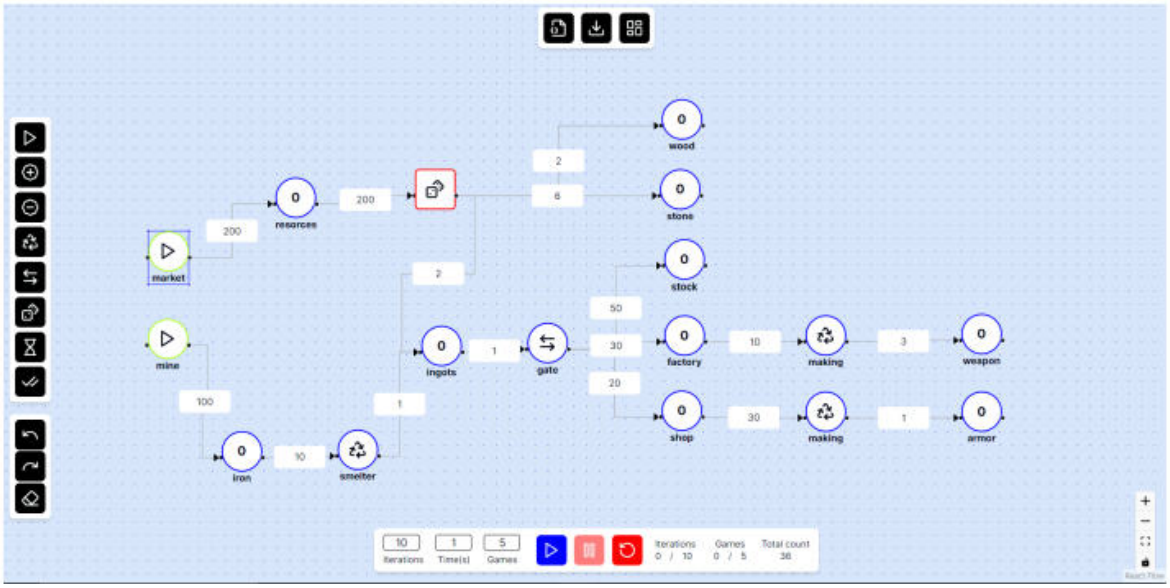


Рисунок 3.6 – Загальний інтерфейс інтерактивної дошки

На верхній панелі представлені три кнопки:

– кнопка "Editor", котра відкриває кодовий онлайн-редактор;

– кнопка із зображенням завантаження, дозволяє користувачеві завантажувати схему у форматі PNG;

– кнопка із зображенням меню, відображає бічну панель з повною інформацією про дошку та її учасників.

На лівій панелі відображаються всі вузли, за допомогою яких користувач може створити власну схему. Нижче розташовані кнопки для скасування останніх

дій в обидві сторони, а також кнопка для видалення всіх елементів на полотні (рисунок 3.7).

Рисунок 3.7 – Кнопки для роботи з діями всередині прототипу

На правому боці інтерактивної дошки в нижній частині розташована панель з контрольними кнопками, які спрощують процес роботи. Ці кнопки включають інструменти для збільшення та зменшення масштабу та блокування області перегляду (рисунок 3.8).

Рисунок 3.8 – Кнопки для керування масштабом

3.3.2 Панель для запуску симуляції

Щоб протестувати роботу прототипу, інструмент дозволяє запускати симуляції та налаштовувати тестові умови для їх виконання. Це робиться через нижню панель інтерактивної дошки (рисунок 3.9).

Рисунок 3.9 – Панель для запуску симуляцій

Під час кожного ітераційного кроку всі вузли на схемі змінюють свій стан

одночасно. Кожна ітерація є одним кроком руху ресурсу за схемою. Після завершення кожної ігрової сесії, яка є зіграною грою, значення всіх вузлів обнуляються, і прототип починає роботу заново з новими значеннями. Швидкість виконання діаграми може бути налаштована: значення за замовчуванням становить 1 секунду на крок. Зниження цього прискорює процес, а підвищення - уповільнює його.

Перше поле на панелі призначене для встановлення кількості ітерацій, які має виконати система протягом однієї ігрової сесії.

Друге поле визначає тривалість однієї ітерації в секундах.

Третє поле визначає кількість ігрових сесій, які слід симулювати. Кожна сесія включає раніше задану кількість ітерацій, що дозволяє аналізувати різноманітні розподілу ресурсів в ході гри.

Кнопки керування на панелі дозволяють почати візуалізацію симуляції, призупинити та відновити процес на певному етапі. Остання кнопка призначена для скидання всіх даних та початку симуляції з першої ітерації першої сесії.

Бічна частина панелі відображає лічильник виконаних ітерацій та ігрових сесій, а також показує загальну кількість елементів, що використовуються у схемі, включаючи з'єднання та компоненти.

У реалізованому інструменті було впроваджено перевірку валідації рядкових величин (рисунок 3.10).

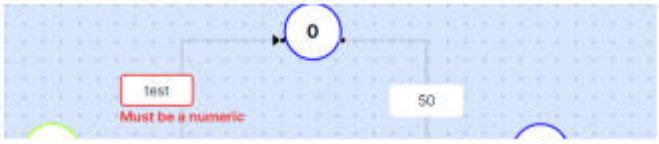


Рисунок 3.10 – Відображення помилки валідації

Якщо в полі ребра вказується не числове значення, з'являється помилка (див. рис. 3.10) і блокується можливість запуску симуляції. Кнопка запуску залишається неактивною, доки неправильне значення не буде виправлено.

3.3.3 Панель загальної інформації

На загальній інформаційній панелі дошки відображається назва проекту, що підлягає зміні, ім'я творця дошки, інформація про інших учасників під час спільної роботи з їхніми іменами та аватарами. Поле для опису процесів призначене для чіткого визначення цілей, напрямів та етапів розвитку проектованого процесу (рисунок 3.11).

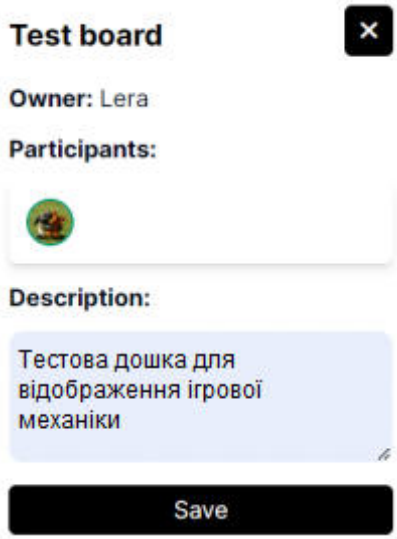


Рисунок 3.11 – Панель з відображенням загальної інформації про дошку

Нижче на панелі (рисунок 3.12) подано вибір кнопок для візуалізації типу з'єднань.

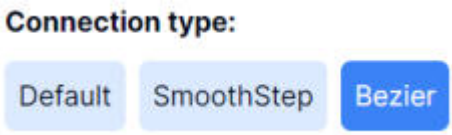


Рисунок 3.12 – Кнопки для вибору з'єднань

Доступні такі варіанти.

– стандартне з'єднання (Default) - прямі лінії без кутів;

– згладжений крок (SmoothStep) – з'єднання, які підлаштовуються під розташування вузлів, утворюючи прямі кути;

– з'єднання кроком Безьє (Bezier) – плавні лінії без кутів.

На рисунку 3.13 наведені основні види з’єднань.

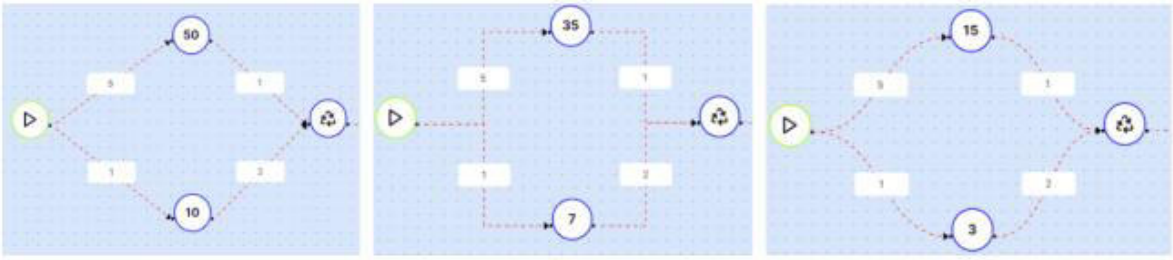


Рисунок 3.13 – Види з'єднання вузлів

3.4 Розробка модуля генерації на основі специфікації

В рамках розробки було додано додатковий функціонал, що відрізняє інструмент від аналогів. Суть полягає у використанні попередньо підготовлених JSON специфікацій, щоб усунути необхідність ручної побудови схеми. Для реалізації цієї функціональності було інтегровано кодовий онлайн-редактор Monaco Editor, який надає засіб для внесення змін.

Спочатку користувачеві надається можливість вибору режиму побудови прототипу: або вручну, або із застосуванням генерації. Часто перед етапом тестування доступна готова специфікація, або описана в документі GDD , або експортована з інших джерел. У таких випадках кращим варіантом є вибір генерації.

Процес роботи з редактором починається зі вставки JSON- специфікації в IDE (Integrated Development Environment), приклад роботи модуля представлений на рисунку 3.14.

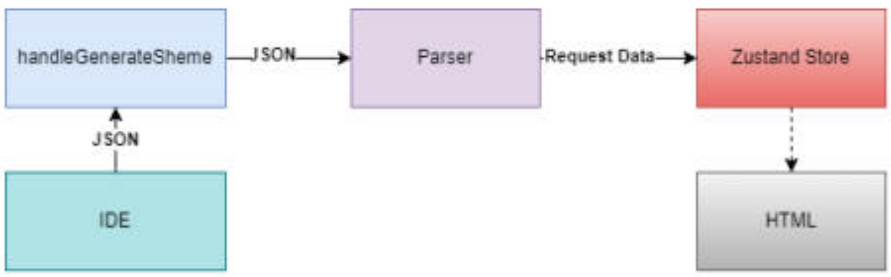


Рисунок 3.14 – Процес отримання та обробка даних

Після активації кнопки "Generate" спрацьовує функція handleGenerateBuild, яка відправляє дані у вигляді рядка в парсер, він зі свого боку здійснює вилучення необхідної інформації і передає її в сховище даних Zustand Store. Сховище взаємодіє з бібліотекою ReactFlow і додає елементи масив ребер і вузлів, що призводить до побудови відповідної схеми прототипу на основі вмісту без перерендерингу всередині однієї сторінки.

Однак для правильного опису специфікації необхідно дотримуватись наступних умов: обов'язкові значення: “edge_type” - описує візуальний тип відображення ребер графа, “iteration_counts” - кількість ітерацій, “time_step” - тимчасовий крок, “games_counts” - кількість ігрових сесій та “elements” - масив значень, який повинен мати помилки.

При описі масиву з елементами графа необхідно вручну вказувати "id" - ідентифікатор елемента та "element_type" - тип елемента, можливі значення - "node" або "edge". Якщо елемент є вузлом, необхідно вказати "struct" - структура вузла та, за бажання, "label" - назва вузла. Якщо елемент є ребром, необхідно вказати "target_id" - ідентифікатор цільового вузла, "source_id" - ідентифікатор вихідного вузла та "value" - значення, яке вводиться в ребро для з'єднання вузлів.

Після створення прототипу користувач отримує доступ до редагування схеми, зв'язків і вузлів як в режимі онлайн-редактора, так і вручну шляхом графічних елементів. Інтерфейс редактора представлено на рисунку 3.15.

Щоб переробити схему прототипу в JSON , реалізована і зворотна процедура. Після ручного прототипування та активації кнопки "Build" дані зі сховища витягуються з масиву вузлів та ребер. Потім вирушають до парсера, який далі передає їх на візуалізацію в кодовий редактор. Це дозволяє користувачам копіювати відповідний код для використання у подальших етапах розробки гри.

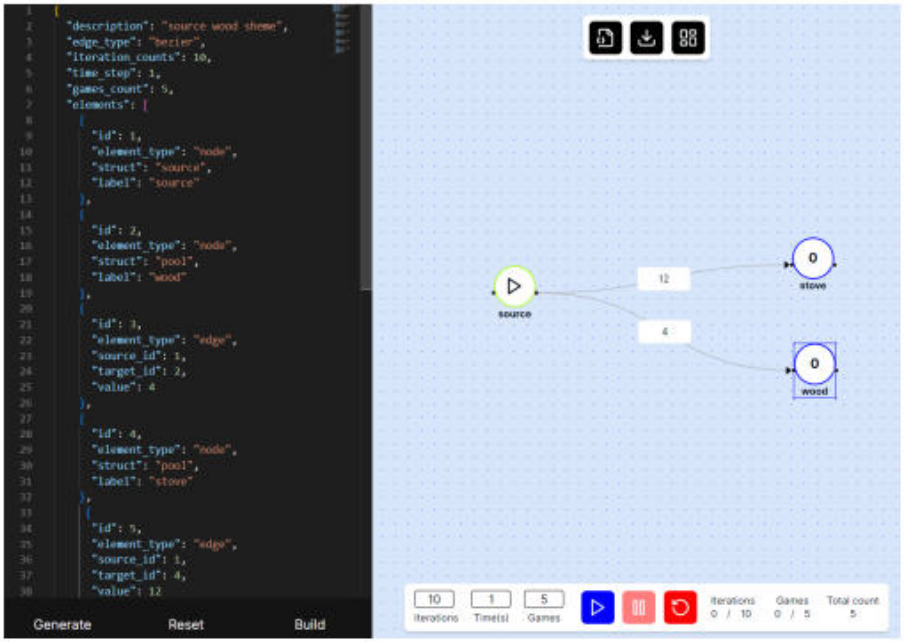


Рисунок 3.15 – Інтерфейс редактора для створення

3.5 Тестування. Застосування на практиці

Для проведення тестування та дослідження практичного застосування розробки було розроблено модель руху ресурсу для виробництва ігрової зброї.

Основні елементи, задіяні у схемі прототипу:

– копальня (Mine);
– ринок ресурсів (market);
– плавильна (smelter);
– центр розподілу ресурсів (Gate);
– склад (stock);
– фабрика зброї (factory);
– майстерня (shop).

У цій схемі передбачено два джерела ресурсів: копальня та ринок. Копальня видобуває 100 одиниць залізняку за ігрову ітерацію. Ринок виробляє 200 одиниць ресурсу, які потім випадковим чином розподіляються між деревиною, камінням та металевими зливками.

У процесі виробництва зброї використовуються металеві зливки, які

додатково виробляються шляхом обробки заліза у плавильні, забезпечуючи два джерела появи ресурсу. Далі за схемою металеві зливки розподіляються між складом, фабрикою та майстернею. За одну ігрову ітерацію 10 одиниць злитків на фабриці виробляють 3 одиниці зброї, а 30 одиниць злитків у майстерні – одну одиницю броні.

Необхідно проаналізувати реалістичність зазначеного розподілу ресурсів, вивчивши статистику їх надходження до кінцевих точок, для цього було встановлено 10 ігрових ітерацій для симуляції роботи 5 ігор. Схема прототипу представлена на рисунку 3.16.

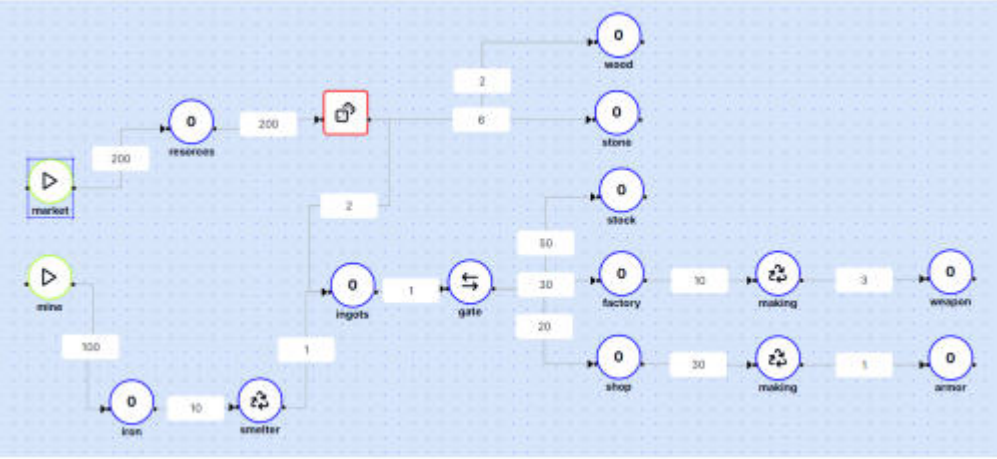


Рисунок 3.16 – Прототип роботи процесу виробництва зброї

В результаті аналізу кінцевих вузлів було виявлено незбалансовані значення схеми. Зокрема, у вузол "armor" не надходило необхідної кількості ресурсу. За 5 ігрових симуляцій максимальне значення досягло лише 2 одиниць, і найчастіше ресурси у вузол не надходили. Зі всіх вироблених за час симуляцій ресурсів у вузол "armor" за першу ігрову ітерацію потрапило лише 1/100 частина. Статистична схема подана на рисунку 3.17.

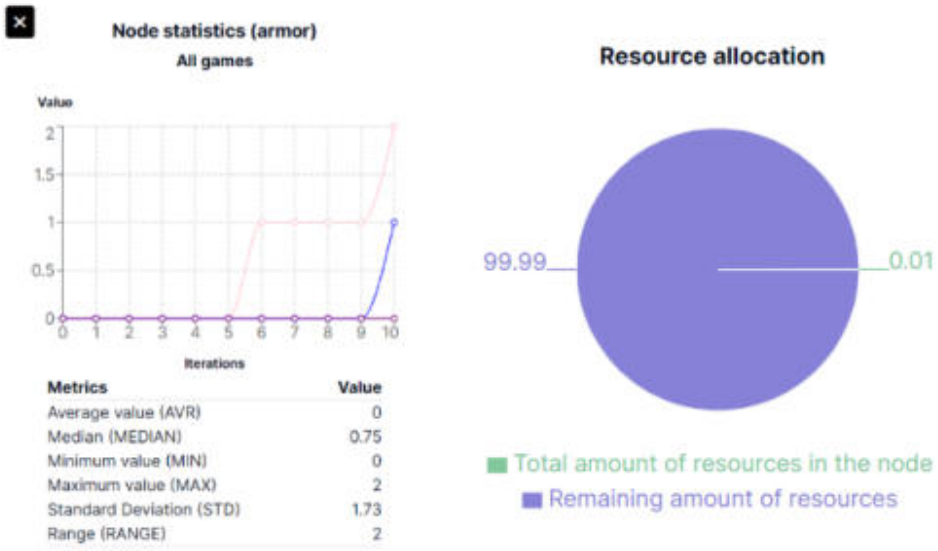


Рисунок 3.17 – Діаграми, що відображають статистику значень вузла

Дані з прототипу були перетворені на JSON для подальшого використання. Результат, що згенерував інструментом, показаний на рисунку 3.18.

Рисунок 3.18 – JSON, згенерований на основі прототипу

Інструмент дозволив протестувати роботу прототипу та проаналізувати рух

ресурсів по ньому, виявивши недоліки стратегії.

Крім того, інструмент забезпечив експорт даних для подальшого доопрацювання гри.

3.6 Висновки до третього розділу

Описано особливості технічної реалізації компонентів проєкту із вибором спеціалізованих програмних засобів. Розроблено архітектуру, представлені діаграми. Втілено інтерфейси для авторизації та спільної роботи, а також інтерактивної дошки. Розроблено модуль генерації на основі раніше сформованої специфікації

Проведене тестування інструменту дає змогу підібрати нові значення та дослідити прототип до досягнення потрібного балансу.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

В цьому розділі необхідно проаналізувати важливі питання, котрі стосуються безпеки життєдіяльності та основ охорони праці.

4.1 Долікарська допомога при ураженні електричним струмом

Удар електричним струмом є поширеною травмою і часто може закінчитися летальним випадком. Ураження електричним струмом відбувається підчас контакту тіла людини з джерелом електричної енергії під напругою. Це реакція організму людини на проходження електричного струму через тіло. Вона проявляється по різному, від легких потрясінь до небезпечних здоров'ю травм, які можуть вплинути на тканини в організмі.

Шкода завдана електричним струмом залежать від кількох факторів: наскільки висока була напруга, яка область тіла була вражена і від виду струму. Фізичні наслідки для людини можуть коливатися від опіків частин тіла до серйозних вражень внутрішніх органів [35].

Часто люди стикаються з підвищеним ризиком ураження високою напругою. Низька напруги не наносить серйозних травм для люди, а з іншої сторони висока напруги, яка має більше 500 В, може призвести до серйозного пошкодження тканин. У дітей або підлітків при враженні електричним струмом в діапазоні від 110 до 200 В можуть вникнути значні травми. Зазвичай це трапляється підчас порушення техніки безпеки при роботі з електричними приладами, які є в побуті. Це можуть бити електричні шнури, подовжувачі, несправні розетки та багато чого іншого.

Виділяють чотири основні фактори від яких залежить вплив електричного струму на організм: величина струму, що протікає через тіло; органи, через які проходить струм; час, протягом якого струм вражає тіло; частота струму.

Фізичні наслідки на пряму залежать від величини струму, яка вражає тіло людини. Струм менший за 1 мА не завдає жодної шкоди людина фізичного ефекту.

При 1 мА можуть відчуватися слабкі поколювання, а при 5 мА – легкий поштовх. Однак при струмі ведичною від 6 до 25 мА людина може відчувати больовий шок і деяку втрату контролю над м’язами.

При ураженні електричним струмом від 50 до 150 мА може спричинити у людини сильний біль, м’язове скорочення і навіть призвести до зупинку дихання. У певних випадках можливий летальний результат для людини. Струм від 1000 до 4300 мА призводить до ймовірної смерть, тому що дана напруга спричиняє пошкодження нервових зав’язків, м’язових скорочення та порушення ритму серця. До 10 000 мА електричний струм спричиняє важкі опіки шкіри людини та зупинку серця. Висока ймовірність смерті.

Найпоширеніші ознаки та симптоми враження електричним струмом включають:

- втрата свідомості;
- ускладнення або зупинка дихання;
- опіки, які виникають там, де струм входить і виходить з тіла;
- зупинка серця;
- слабкий і непостійний пульс або його припинення.

Перш за все, що потрібно зробити при першій допомозі, це відключити джерело живлення. Вимкнути електропостачання, від’єднати електроприлад від джерела електричного струму або вимкнути блок запобіжників, якщо він знаходиться неподалік. Не потрібно намагатися підходити близько до жертви, якщо не переконані, що це безпечно і живлення вимкнено.

Потрібно бути обережним у вологих місцях, тому що вода є електричним провідником і рятівник може стати теж жертвою. Якщо людина не впевнена щодо вологості поверхні, потрібно відключити основне електропостачання будинку. У випадку коли це неможливо, використати підручний предмет, який не є провідником і відокремити людину від джерела струму. Це може бути дерев’яна або пластмасова річ.

Після того як постраждалого відокремили від джерела електрики, потрібно викликати швидку і надати першу медичну допомогу. Далі потрібно визначити

стан жертви. Перевірити, чи людина у свідомості і дихає. У складних випадках у жертви може бути слабкий пульс або його відсутність. Можливо, що дихання зупинилося. Якщо людина втратила свідомість і перестала дихати, потрібно почати серцево-легеневу реанімацію. Руки розташовуємо в центрі грудної, одна на іншу. Сильно і швидко натиснути 30 раз приблизно до третини діаметра грудної клітки. Після кожного натискання на грудну клітку робиться два рятувальні вдихи. Потрібно відкинути голову потерпілого назад і підняти підборіддя. Затиснути ніс і створити повне ущільнення. Далі подути потерпілому в рот і подивитися, чи підніметься грудна клітка. Потрібно продовжувати робити натискань на грудну клітку та вдих, поки не прибуде медична допомога або людина не почне сама дихати. Якщо потерпілий живий, перемістити його у зручне йому положення подальше від небезпеки. Можна запобігти шоку, поклавши людину рівно на землю, з головою трохи нижче тіла [35].

При роботі з соціальною мережею користувач повинен знати і вміти як правильно поводитися з ПК, тому що людина перебуває у непосередньому контакті з джерелом напруги. Удар електричним струмом є потенційно смертельною травмою. Негайна медична допомога важлива, щоб запобігти серйозним травмам і смерті. Для запобігання уражень електричним струмом при роботі за ПК слід встановити додаткові захисні пристрої, що забезпечують недоступність токопровідних частин для дотику. Для зменшення небезпеки використовувати розділовий трансформатор для розв'язки з основною мережею.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Робоче місце – це ділянка простору, яка облаштована необхідним обладнанням, відповідно до трудової діяльності, для виконання поставлених завдань.

Правильно побудоване робоче місце повинне забезпечувати:

– найкраще розміщення обладнання і предметів праці;
– не допускати дискомфорту;

– підвищувати продуктивність праці;

– зменшувати втому працівника.

Розмір робочого місця повинен бути таким, щоб людина не виконувала лишніх рухів і не відчувала дискомфорту під час виконання роботи. Також для працівника важливо мати змогу змінити робочу позу, наприклад, положення тулуба, рук або ніг. Потрібно мінімізувати або звести до нуля всі незручності положення тіла [36].

Різні дослідження заявляють, що при правильному проектуванні робочого місця продуктивність людини може зрости від 15-25%. Такі фактори як рівень освітлення, вологість повітря, температура, шум, вібрація, токсичність, мають значний вплив на умови життєдіяльності і працездатності людини.

Антропометричні вимоги визначають відповідність робочого місця до фізіологічних параметрів тіла людини як зріст і розміри тіла. Індикатором цього є правильна робоча поза, відсутність дискомфорту, оптимальні зони досягнення, раціональні рухи. Психофізіологічні та фізіологічні вимоги формулюють відповідність обладнання і робочого місця можливостям співробітника щодо розуміння, обробки даних, пошук і реалізації рішень.

Організація робочого місця передбачає наступні пункти:

– раціональне положення робочого місця у приміщенні;

– вибір робочих меблів відповідно до фізіологічних характеристик працівника;

– правильне компонування і розміщення обладнання на робочих місцях;

– урахування особливостей та характеру професійної діяльності.

До загальних принципів організації робочого місця відносять:

– робоче місце повинне містити тільки ті предмети, які беруть учать у робочому процесі, але не заважати йому;

– предмети, які часто використовуються у роботі, розміщуються ближче, ніж ті речі, якими користуються рідше;

– предмети, які беруться лівою рукою, повинні розміщуватися зліва, а предмети, які використовуються правою рукою — справа;

– якщо при роботі з предметом працівник використовує дві руки, то він розміщується з урахуванням зручності захоплювання його двома руками;

– робоче місце не повинно бути засмічене;

– необхідна оглядовість повинна бути забезпечена при правильній організації робочого місця.

Робоча поза – це найбільш тривале положення тіла працівника протягом робочого дня. При зручній робочій позі забезпечується стійкість положення тулуба, ніг, рук, голови і витрачається мінімальний запас енергії та максимальну продуктивність [36].

Сидячи і стоячи – дві найпопулярніших пози у робочому процесі. При проектуванні робочого місця потрібно враховувати, що з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи. При роботі стоячи, людина стомлюється більше ніж сидячи. У відсотковому еквіваленті це на 10% більше енергії. При додатковому навантаженні підвищується артеріальний і венозний тиск крові, розширення вен, пошкоджуються ступені та викривляється хребет. У свою чергу при сидячій роботі нижня частина тіла розслаблена, а основне навантаження спрямоване на м'язи шиї, спини, таза, стегон. При неправильній сидячій позі розвивається застій крові у ногах, а якщо пальці виконують багато роботи можливе запалення суглобів.

Організація робочого місця при використанні ПК повинна відповідати усім ергономічним вимогам. Ключові ергономічні вимоги до проектування робочого місця оператора ПК зображені на рисунку 4.1.

При роботі з персональним комп’ютером потрібно:

– зменшувати кількість статичних напружень;

– розподіляти кількість і час статичних напружень;

– змінювати робочі пози під професійної діяльності.

Саме вибір правильної робочої пози визначається від впливу багатьох факторів. Одні з найважливіших це – кількість зусиль яка прикладається, велична робочої зони, відношення висоти робочої поверхні і ростом працівника.

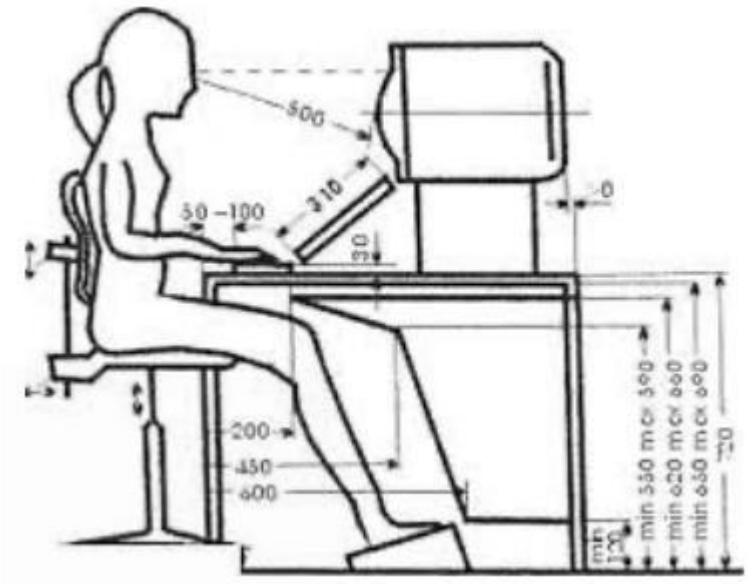


Рисунок 4.1 – Робочий стіл і розміщення користувача ПК

При використанні соціальної мережі користувач повинен дотримуватися вище зазначених правил, які будуть сприяти комфортній і продуктивній роботі. Важливо регулярно робити короткі перерви. Часта зміна заняття – кращий спосіб уникнути можливих неприємностей.

4.3. Висновки до четвертого розділу

В цьому розділі висвітлено важливі питання надання долікарської допомоги при ураженні електричним струмом та вимоги ергономіки до організації робочого місця оператора ПК.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було досягнуто мети - розроблено онлайн-інструмент для балансування відеоігор, що володіє функціями зі створення схематичних прототипів для імітації ігрових процесів, симуляції та тестування ігрових сесій з різними кількісними параметрами, аналізу та експорту отриманих даних.

У рамках виконання роботи для досягнення поставленої мети було виконано такі завдання:

- вивчені можливі альтернативні рішення з подібним функціоналом;
- описано концепцію роботи інструменту;
- описана візуальна мова для створення та редагування ігрових процесів з використанням JavaScript -бібліотек;
- розроблені інтерфейси для створення, тестування та аналізу ігрових прототипів;
- розроблено модуль генерації схем прототипів на основі специфікації JSON.

Розроблений у рамках даної роботи інструмент має необхідний функціонал для прототипування, усуває проблеми при запуску симуляцій та надає розширені інструменти аналітики. Таким чином, реалізований інструмент перевершує аналог Machinations, усуваючи його недоліки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Exploring the Origins of Game Design: A Comprehensive Look into Its Evolution and Development URL: https://www.rpgwizard.org/exploring-theorigins-of-game-design-a-comprehensive-look-into-its-evolution-anddevelopment/ (дата звернення 04.05.2026)

2. Games Market Engagement, Revenues & Trends data // A Gaming Report Newzoo. URL: https://newzoo.com/resources/blog/games-market-engagement-revenues-trends-2000-2026-gaming-report (дата звернення: 11.05.2026).

3. Future Market Insights. A report on the analysis of the video game market in the segment of mobile devices, consoles and computers. URL: https://www.futuremarketinsights.com/reports/video-game-market (дата звернення: 11.05.2026).

4. Brazy, A. Video Game Balance: The Complete Guide. Video Game Mechanics: A Beginner's Guide. URL: https://gamedesignskills.com/game-design/video-game-mechanics (дата звернення: 12.05.2026).

5. Jane Mitchell Embracing Iteration In Game Design: A Path To Perfection URL: https://www.watermelonpixel.com/a/embracing-iteration-in-gamedesign-a-path-to-perfection (дата звернення 13.05.2026).

6. Mihail Moroşan. Automated Game Balancing in Ms PacMan and StarCraft Using Evolutionary Algorithms. // Poli Riccardo. URL: https://www.researchgate.net/publication/315639209_Automated_Game_Balancing_in_Ms_PacMan_and_StarCraft_Using_Evolutionary_Algorithms (дата звернення: 14.05.2026).

7. Richard Rouse. Game design: Theory and practice. URL: https://gamifique.wordpress.com/wp-content/uploads/2011/11/5-game-design-theory-and-practice.pdf (дата звернення: 14.05.2026).

8. Ian Schreiber. Game Balance Concepts. A continued experiment in game design and teaching. Intro to Game Balance. // Romero Brenda. URL:

https://gamebalanceconcepts.wordpress.com/2010/07/07/level-1-intro-to-game-bal ance (дата звернення: 16.05.2026).

9. Fullerton T. Game design workshop: a playcentric approach to creating innovative games. 5th edition. Boca Raton, FL : CRC Press, 2024. 554 p.

10. Методологія балансування відеоігор. URL: https://gdcuffs.com/balance-methods (дата звернення: 18.05.2026).

11. Stevenson, J. Game designer at Tag Games. My Approach To Economy Balancing Using Spreadsheets. URL: https://www.gamedeveloper.com/design/my-approach-to-economy-balancing-usin g-spreadsheets (дата звернення: 19.05.2026).

12. War Robots Universe. Avoid the cell and table swamp: maintaining game balance with ease. URL: https://medium.com/my-games-company/avoid-the-cell-and-table-swamp-maintain ing-game-balance-with-ease-9f3e90bf45ac (дата звернення: 19.05.2026).

13. Machinations.io browser-based tool to design and balance game systems // Documentation. URL: https://machinations.io/docs/what-is-machinations (дата звернення: 20.05.2026).

14. How we used Machinations to simulate the economy of IndusTree. URL: https://well-done-games.com/blog/how-we-used-machinations-to-simulate-the-economy-of-industree (дата звернення: 21.05.2026)

15. Game systems: Battle passes and how to balance them. URL: https://medium.com/@machinations.io/game-systems-battle-passes-and-how-to-balance-them-machinations-io-4f11f8152db4 (дата звернення: 21.05.2026).

16. Machinations: Progression Systems for Game Economy Designers. URL: https://www.onlinetutorials.org/it-software/machinations-progression-systems-for-game-economy-designers/ (дата звернення: 21.05.2026).

17. Нотації для бізнес-процесів: види, переваги та недоліки. URL: https://todo.ltd/blog/notacziyi-dlya-biznes-proczesiv-vydy-perevagy-ta-nedoliky/ (дата звернення: 22.05.2026).

18. Burgun, K. Randomness and Game Design. URL: https://www.gamedeveloper.com/design/randomness-and-game-design (дата звернення: 24.05.2026)

19. Meet Sarvaiya. Achieving Game Economy Balance Through Data-Driven Insights.– URL: https://www.sonamine.com/blog/achieving-game-economy-balance-through-data-d riven-insights (дата звернення: 24.05.2026).

20. Yang, R. The Level Design Book // Andrew Yoder.– URL: https://book.leveldesignbook.com/process/combat/balance (дата звернення: 24.05.2026).

21. Ian Schreiber. Game Balance Concepts. A continued experiment in game design and teaching. Metrics and Statistics. // Romero Brenda.– URL: https://gamebalanceconcepts.wordpress.com/2010/08/25/level-8-metrics-and-statis tics (дата звернення: 25.05.2026).

22. Next.js by Vercel. – The React Framework Documentation // The library for web and native user interfaces. URL: https://nextjs.org (дата звернення: 25.05.2026).

23. Zustand Documentation. // Zustand is a small, fast and scalable state-management solution, it has api based on hooks. URL: https://docs.pmnd.rs/zustand/getting-started/introduction (дата звернення: 26.05.2026).

24. NestJS Documentation. // NestJS is a framework for building efficient, scalable Node.js server-side applications. URL: https://docs.nestjs.com (дата звернення: 27.05.2026).

25. PostgreSQL. // The World's Most Advanced Open Source Relational Database. URL: https://www.postgresql.org (дата звернення: 27.05.2026).

26. TypeORM. // ORM for TypeScript and JavaScript. URL: https://typeorm.io (дата звернення: 27.05.2026).

27. Clerk Authentication and User Management Documentation. // Clerk is a complete suite of embeddable UIs, flexible APIs, and admin dashboards to authenticate and manage users. URL: https://clerk.com/docs (дата звернення: 28.05.2026).

28. Liveblocks Documentation. // Liveblocks is a real-time collaboration infrastructure for building performant collaborative experiences. URL: https://liveblocks.io/docs (дата звернення: 28.05.2026).

29. React Flow Node-Based UIs in React Documentation. // Highly customizable React library for workflow builders, no-code apps, image processing, visualizers, and more. URL: https://reactflow.dev/ (дата звернення: 29.05.2026).

30. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі Михалик Д.М., Цуприк Г.Б., Бревус В.М. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. 45 с.

31. Stefanyshyn, I. , Pastukh, O., Stefanyshyn, V. , Baran, I. , Boyko, I. Robustness of AI algorithms for neurocomputer interfaces based on software and hardware technologies CEUR Workshop Proceedings, 2024, 3742, pp. 137–149.

32. Boyko, I. , Petryk, M. , Mudryk, I. , Stoianov, Y., Tsupryk, H. Mathematical Model of the Capacitor Based on Zeolite Material. Proceedings - International Conference on Advanced Computer Information Technologies, ACIT, 2021, pp. 45–48.

33. Boyko, I. , Petryk, M. , Mykhailyshyn, R . Excitons in resonant tunnelling structures based on AlN/GaN/AlN/AlGaN/AlN nitride: spectral dependences and intensities of interband optical transitions. Ukrainian Journal of Physical Optics, 2022, 23(3), pp. 180–191.

34. M. R. Petryk, A. Khimich, M. M. Petryk, and J. Fraissard, “Experimental and computer simulation studies of dehydration on microporous adsorbent of natural gas used as motor fuel,” Fuel, Vol. 239, 1324–1330 (2019). https://doi.org/https://doi.org/10.1016/j.fuel.2018.10.134

35. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звертання: 10.12.2025).

36. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.