МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«10» червня 2024р.	«10» червня 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Інформаційні системи та штучний інтелект

на тему: Розроблення інформаційної системи для курсів з вивчення комп'ютерних наук

Виконав: здобувач 4 курсу, групи КН-4-4

Анісімов Олексій Юрійович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Харкянен Олена Валеріївна _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти Олена Харкянен _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2024р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Інформаційні системи та штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
Сергій ГРИБКОВ

" 15 " квітня 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
Анісімов Олексій Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інформаційної системи для курсів з вивчення комп'ютерних наук

керівник роботи Харкянен Олена Валеріївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від 15 квітня 2024 року № 279-кс
2. Строк подання здобувачем роботи 03.06.2024 р.
3. Вихідні дані до роботи
Інформація про університет ті його структурні підрозділи, інформація про дисципліни, робочі програми дисциплін
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз об'єкта дослідження
2) Моделювання додатку
3) Створення додатку
4) Охорона праці та техніка безпеки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Харкянен О. В., доцент	15.03.2024	28.05.2024
2.	Харкянен О. В., доцент	15.03.2024	28.05.2024
3.	Харкянен О. В., доцент	15.03.2024	28.05.2024

7. Дата видачі завдання_____15 квітня 2023 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Аналіз предметної області	15.03.2024-20.03.2024	
	Розробка технічного завдання	15.03.2024-29.03.2024	
	Прототипування	27.04.2024-20.05.2024	
	Розробка додатку	21.05.2024-24.05.2024	
	Оформлення документації	23.05.2024-28.05.2024	

Здобувач _____ Олександр АНІСІМОВ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Олена ХАРКЯНЕН
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці системи для курсів з вивчення комп’ютерних наук у вигляді відеоігри. Метою роботи є розроблення додатку в ігровій формі для допомоги здобувачам і викладачам у вивченні та викладанні комп'ютерних наук.

У роботі досліджений процес навчання за спеціальністю 122 «Комп’ютерні науки» у Національному університеті харчових технологій.

Розроблена система включає три режими, один для вивчення логічних операцій та операцій над множинами, один для вивчення формування SQL запитів і один для вивчення розробки алгоритмів. Якщо користувач зареєстрований, то прогрес користувача буде зберігатися між сесіями. Рівні для даних режимів досить легко створювати, так що викладачі можуть створювати нові рівні за потребою.

Систему було розроблено за допомогою середовища Godot 4 .NET та мові програмування C#. Данні завантажуються за допомогою нестандартної системи.

Дипломна робота складається з 43 сторінок, 14 рисунків, 3 таблиць, 4 додатків і 14 літературних джерел.

КЛЮЧОВІ СЛОВА : GODOT, C#, .NET, DESKTOP, ПЗ, ОСВІТА, НАВЧАЛЬНА ГРА, ІНФОРМАЦІЙНА СИСТЕМА

SUMMARY

The graduate work is dedicated to the development of the system for computer science courses in form of a video game. The aim of the work is to develop an application in a game form to help students and teachers in the study and teaching of computer sciences.

The process of studying in the specialty 122 «Computer Science» at the National University of Food Technology was analysed.

The developed system includes three game modes, one for the study of logical operations and operations on sets, one for the study the formation of SQL requests and one for the study of the development of algorithms. In case of the user being signed in their progress would be stored and would carry between sessions.

Levels for these game modes are quite easy to create, so teachers can create new levels as needed.

The system was developed using the Godot 4 .NET and C# programming language. The data is loaded with a custom system.

The 51-page graduate work consists of 43 pages, 14 images, 3 tables, 4 appendices, and references to 14 literary sources.

KEYWORDS: GODOT, C#, .NET, DESKTOP, SOFTWARE, EDUCATION, EDUCATIONAL GAME, INFORMATION SYSTEM

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ОП — Освітня Програма

AST – Abstract Syntax Tree

НУХТ — Національний університет харчових технологій

АКС — факультет Автоматизації та комп’ютерних систем

ІТ – Інформаційні технології

ВСТУП

Інформаційні технології стали невід'ємною частиною сучасного світу. Вони охоплюють майже усі сфери людського життя, включаючи науку, освіту, бізнес, медицину, розваги, комунікацію, політику, тощо. ІТ дозволяють швидко та ефективно обмінюватися інформацією, автоматизувати процеси виробництва та документообігу, сприяє глобалізації і міжнародній кооперації. За допомогою інформаційних технологій можливо робити інструменти для оптимізації роботи, будь то програма для створення малюнків, текстові редактори, система автоматизованого проєктування, база даних тощо.

Через наявність інформаційних технологій у різних сферах людського життя є досить висока потреба у ІТ-спеціалістах і відповідно є потреба в навчанні комп'ютерним наукам.

Є багато шляхів організації навчання комп'ютерним наукам. Вони включають традиційну освіту, особисті проєкти, курси і навчальні додатки. Кожен з цих методів має свої переваги та недоліки.

Результатом цієї кваліфікаційної роботи є система для курсів з вивчення комп'ютерних наук у вигляді відеоігри. Вона базується на таких дисциплінах, як дискретна математика, організація баз даних та знань, основи програмування та розробка алгоритмів, і повинна навчати логічним операціям та операціями над множинами, формуванню запитів SQL, створенню алгоритмів.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ

1.1. Загальна характеристика

У Національному університету харчових технологій підготовка фахівців зі спеціальності 122 «Комп’ютерні науки» триває вже понад 20 років [3].

Факультет автоматизації та комп’ютерних систем був заснований у 2000 році у зв’язку з потребою у спеціалістах для комп’ютерних систем керування виробництвом у харчовій промисловості. До його складу увійшли кафедри автоматизації процесів управління, інтегрованих автоматизованих систем, інформатики та вищої математики [1].

У 2022 році кафедру інформаційних систем (заснована в 2002) і кафедру інформатики (заснована в 1973) було об’єднано у кафедру інформаційних технологій, штучного інтелекту і кібербезпеки[2].

Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки здійснює професійну підготовку фахівців за спеціальністю 122 «Комп’ютерні науки». Узагальненим об’єктом діяльності майбутніх фахівців є інформаційні та автоматизовані системи, засоби контролю, моніторингу і управління, їх математичне, інформаційне та програмне забезпечення, способи та методи їх проєктування, розробки, впровадження і експлуатації[2].

Це передбачає здатність до аналізу предметної області, формалізації даних та знань, створення логічного інтерфейсу, формування висновків для підтримки прийняття рішень в ординарних та позаштатних ситуаціях[3].

Основним фокусом освітньої програми та спеціалізації полягає у підготовці освітньо-професійних кадрів, які володіють сучасними методами та інформаційними технологіями для впровадження інформаційного, математичного, програмного, технічного та організаційного забезпечення інформаційно-управляючих систем та систем штучного інтелекту[3].

1.2. Організаційна структура процесу навчання

Перелік компонент освітньо-професійної програми та їхня логічна послідовність наведена у таблиці 1.1 [4].

Таблиця 1.1. Перелік компонент ОП

Код н/д	Компоненти освітньої програми	Кількість кредитів	Форма підсумкового контролю
Обов'язкові компоненти ОП			
ОК 1.	Історія і культура України	5	залік
ОК 2.	Іноземна мова (за професійним спрямуванням)	6	екзамен
ОК 3.	Вища математика	18	екзамен
ОК 4.	Дискретна математика	7	екзамен
ОК 5.	Чисельні методи	6	залік
ОК 6.	Теорія ймовірностей	6	екзамен
ОК 7.	Математичні методи дослідження операцій	7	екзамен
ОК 8.	Теорія прийняття рішень	6	екзамен
ОК 9.	Основи програмування та розробка алгоритмів (в тому числі курсова робота)	12	екзамен
ОК 10.	Об'єктно-орієнтоване програмування (в тому числі курсова робота)	8	екзамен
ОК 11.	Операційні системи (в тому числі курсова робота)	7	екзамен
ОК 12.	Організація баз даних та знань (в тому числі курсова робота)	8	екзамен
ОК 13.	Аналіз даних та машинне навчання	5	залік
ОК 14.	Web-технології та Web-дизайн	7	екзамен
ОК 15.	Комп'ютерна графіка	4	екзамен
ОК 16.	Системний аналіз	5	екзамен
ОК 17.	Паралельне програмування	5	екзамен
ОК 18.	Захист та безпека даних	4	екзамен
ОК 19.	Крос-платформне програмування	4	залік

ОК 20.	Проектування програмного забезпечення інформаційних систем (в тому числі курсовий проект)	6	екзамен
ОК 21.	Інтелектуальні системи	4	екзамен
ОК 22.	Комп'ютерні мережі	4	екзамен
ОК 23.	Управління ІТ-проектами (в тому числі курсова робота)	5	екзамен
ОК 24.	Моделювання систем	6	екзамен
ОК 25.	Інтернет речей	4	екзамен
ОК 26.	Виробнича практика	6	залік
ОК 27.	Переддипломна практика	3	залік
ОК 28.	Кваліфікаційна робота	12	
	Фізична культура і спорт		
Загальний обсяг обов’язкових компонент:		180 кредитів	
Вибіркові компоненти ОП			
ВБ 1.1.	Дисципліни гуманітарно-економічної підготовки	9	
ВБ 1.2.	Дисципліни природничо-наукової підготовки	7	
ВБ 1.3.	Дисципліни загальної фахової підготовки, професійного і практичного спрямування	44	
Загальний обсяг вибіркових компонент:		60 кредитів	
Загальний обсяг освітньої програми:		240 кредитів	

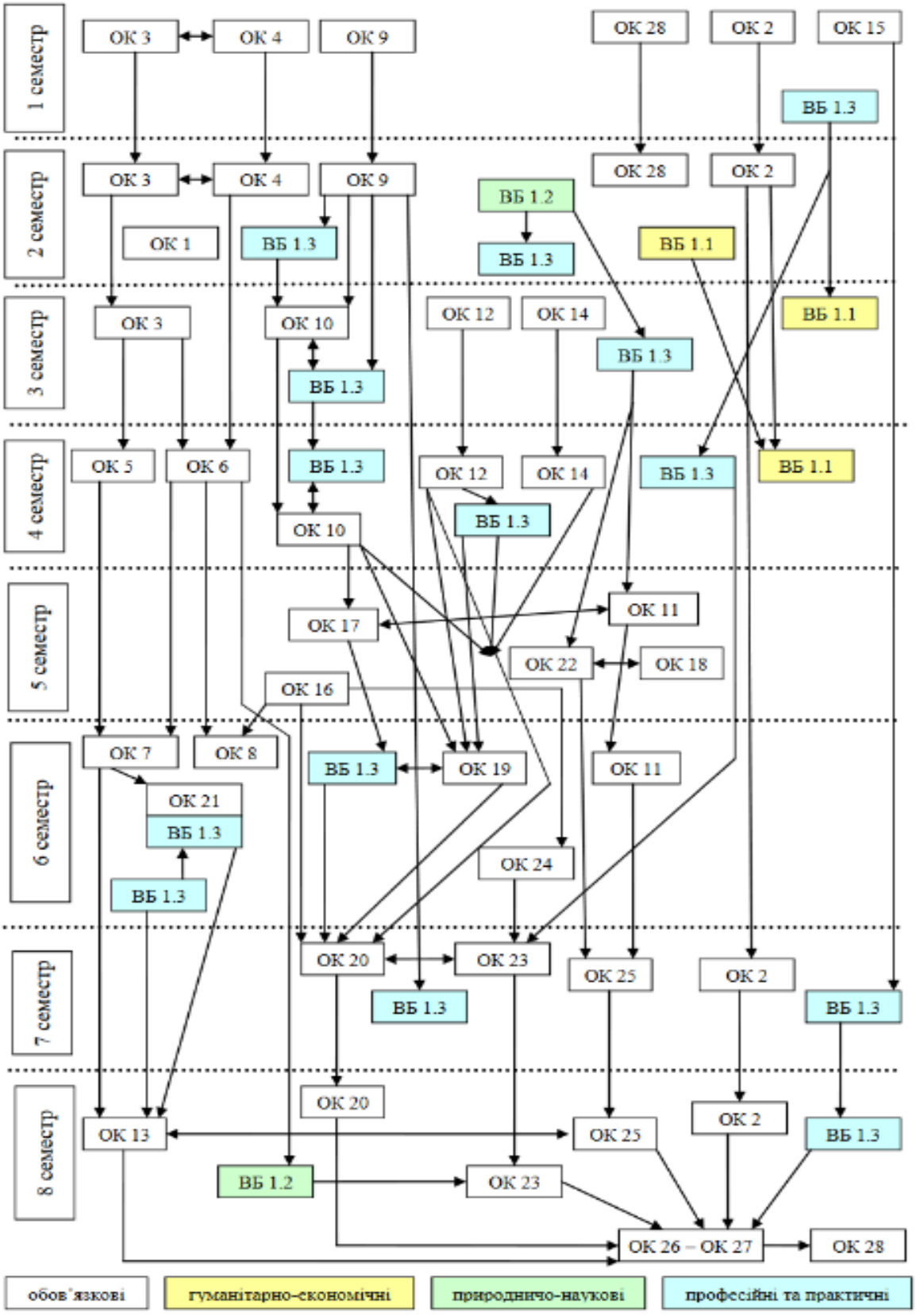

Рисунок 1.1 — Структурно-логічна схема освітньої програми

Як ми бачимо, різні дисципліни пов’язані між собою і потребують знань, отриманих при вивченні дисциплін, що їм передують.

1.3. Аналіз нинішнього стану комп’ютеризації

Серед програмного забезпечення, що використовується у процесі навчання здобувачів присутні:

- Zoom, для організації відеоконференцій, лекцій, співбесід.
- Moodle, для завантаження навчальних матеріалів, виконання лабораторних робіт, проходження тестів.
- Telegram, для комунікації між учасниками навчального процесу.

За допомогою цих програмних забезпечень організується процес дистанційного навчання, і вони можуть допомогти в очному навчанні. Цей набір імітує традиційні процеси очного навчання: відеоконференції в Zoom зазвичай імітують лекції, захист лабораторних та співбесіди, Moodle зазвичай імітує здачу лабораторних та тестів, в той час як Telegram дозволяє підтримувати контакт між учасниками навчального процесу.

На нашу думку, цей набір програмного забезпечення дозволяє забезпечувати повноцінну підтримку традиційного навчання, і допомагають при змішаній формі навчання, але усе одно залишаються деякі проблеми.

Одна з цих проблем — це те, що проведення лекцій у вигляді відеоконференції може призводити до зменшення концентрації здобувачів на лекції, оскільки вони не знаходиться в одному приміщенні з викладачем. При проведенні відеоконференції складніше організовувати інтеракції між здобувачами та викладачами і сама лекція сприймається складніше. До того ж, якщо здобувач знаходиться в аудиторії, то він сконцентрований на лекційному матеріалі, в той час як при відеоконференції може з’явитись бажання займатися іншими справами, що призводить до розподілення фокусу.

До того ж, умови військового стану добавляють особливих проблем для навчального процесу. Лекції, практичні роботи та захист лабораторних робіт можуть переноситись, відсутність інтернету або світла ускладнюють виконання лабораторних робіт та тестів, а стрес від військового стану може негативно впливати на психологічний стан учасників навчального процесу.

1.4. Розроблення функціональної моделі та аналіз існуючих бізнес-процесів

На початку курсу викладач формує його завдання. Під час проходження курсів з вивчення комп'ютерних наук постійно проводиться процес виконання та перевірки завдань.

Функціональна модель виконання та перевірки завдання наведена на рисунку 1.2:

Вхідні данні — сформоване завдання.

Обробка даних — виконання та перевірка завдання.

Вихідні данні — виконане завдання.

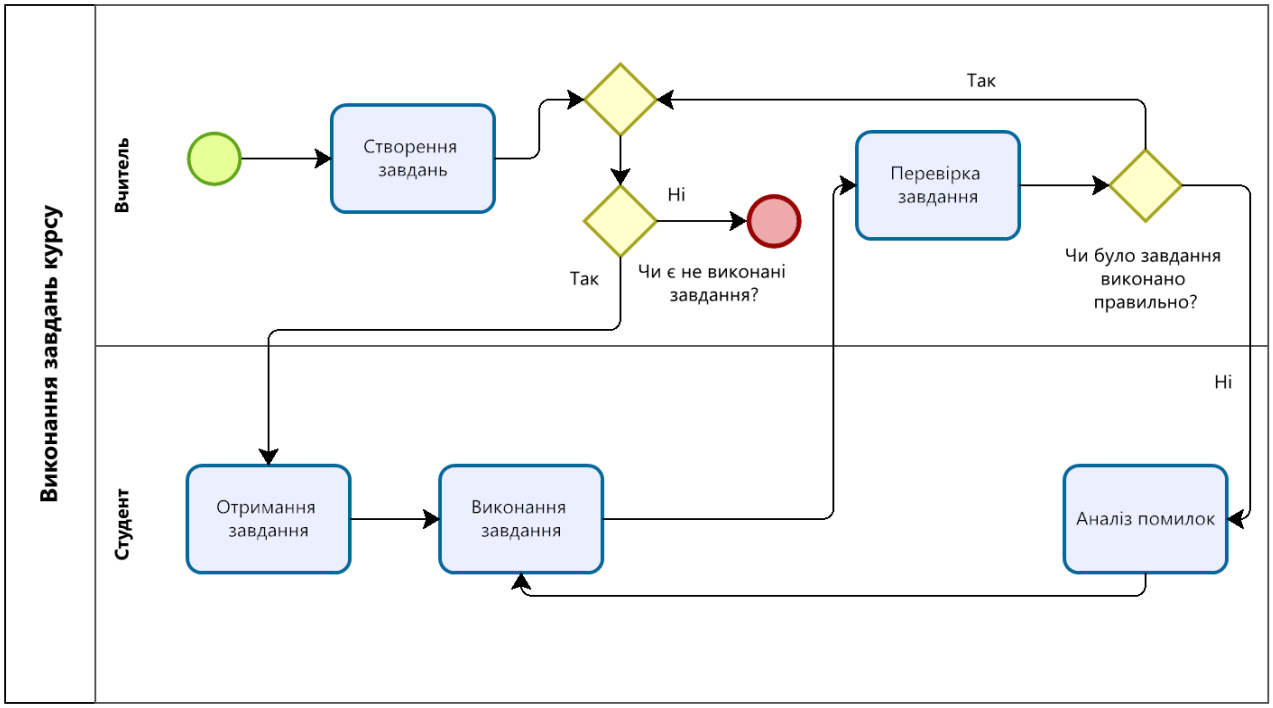

Рисунок 1.2 — Структурно-логічна схема виконання завдань курсу

Основними функціями, котрі можна автоматизувати, є перевірка виконання завдань.

Процес перевірки завдання потребує взаємодії здобувача та викладача, що може сповільнити процес виконання завдань, адже потрібно щоб здобувач виконав завдання, викладач його перевірив і надав відгук щодо виконання завдання, а перевиконати завдання якщо воно було виконано не правильно. Процес комунікування між здобувачем та викладачем займає досить багато часу, а в умовах військового стану кожен з етапів може бути зірваним, включаючи

комунікацію між здобувачем та викладачем, що може ще більше сповільнити процес навчання.

Таким чином, пропонуємо створення системи інформаційної системи для курсів з вивчення комп'ютерних наук у вигляді навчальної відеоігри, побудованої навколо легкості модифікації для можливості добавлення нових завдань без потреби в спеціального обладнання.

1.5. Огляд існуючих рішень

Огляд існуючих програмних забезпечень для вивчення комп’ютерних наук.

Scratch[5]

Scratch – візуальна мова програмування, де код створюється за допомогою блоків, та середовище, де використовується ця мова програмування. Scratch орієнтований на дітей віком від 8 до 16 років, і через це має простий та зрозумілий інтерфейс. Scratch є безплатним, open-source, і його код розміщено на GitHub. Scratch є доступним у вигляді веб-додатку і у вигляді настільного додатку.

CodeCombat[6]

CodeCombat – освітня відеоігра у вигляді веб-додатку, що навчає програмуванню. CodeCombat орієнтований на дітей віком від 9 до 16 років, і навчає мовам програмування JavaScript, Python, HTML і CoffeeScript.

CodeCombat складається з 11 розділів: три розділів по розробці ігор, двоє розділів по розробці веб-додатків, і шість розділів по комп’ютерним наукам. Перший розділ комп’ютерних наук (5 рівнів) є безплатним, а для доступу до інших (531 рівень) потрібно мати преміальний рахунок, котрий коштує 9.99$ за місяць або 99$ за рік.

CodeCombat має редактор рівнів. Код CodeCombat, за винятком рівнів, опублікований на GitHub з MIT ліцензією.

Brilliant [7]

Brilliant – це навчальна платформа, що має величезну кількість курсів у різних сферах, включаючи математику, аналіз даних, комп’ютерні науки, фізику, тощо. Ці курси складаються з головоломок, пов’язаних з темою навчання.

Безплатно доступна невелика кількість уроків, за інші потрібно платити 24.99$ в місяць або 13.49$ в рік.

Таблиця 1.2. Порівняння існуючих рішень

Характеристика	**Scratch**	**CodeCombat**	**Brilliant**
Тип	Середовище програмування	Відеоігра	Навчальна платформа
Основний шлях навчання	Програмування своїх проєктів	Створення алгоритму вирішення завдання	Вирішення головоломок
Чи працює без інтернету	Так	Ні	Ні
Чи є безплатним	Безплатно	Перші 5 рівнів безплатно, далі потрібна підписка	Невелика кількість уроків безплатна, за інші потрібна підписка
Чи ставить задачі перед користувачем	Ні	Так	Так
Чи є open-source	Так	Так, за винятком рівнів	Ні

Як ми бачимо, ці системи дуже сильно відрізняються, і кожна з них має свої переваги та недоліки, можливості та обмеження. Наприклад, Scratch діє скоріше як достатньо просте середовище щоб спробувати почати розробляти свої проєкти, і достатньо високий фокус є на спільноту користувачів. Scratch намагається визивати внутрішню мотивацію (наприклад, бажання самореалізації), і не має фокуса на зовнішню мотивацію. CodeCombat дає рівні, і вам потрібно розробити алгоритм, щоб пройти їх. Фокус на спільноту користувачів набагато слабший, аніж у Scratch, але він присутній. Brilliant дає

вам головоломки і вам потрібно вирішити їх, при цьому головоломки продумані так, щоб їхній розв’язок здавався інтуїтивно зрозумілим, і, так як ці головоломки базуються на реальних знаннях, ви починаєте розуміти ці знання. Фокус на спільноту у Brilliant відсутній. Це показує, що процес навчання можна організовувати різними шляхами і методами.

1.6. Обґрунтування доцільності проєктування та розроблення

Навчальні відеоігри можуть допомогти у викладенні матеріалу у цікавому вигляді, інтерактивність дозволяє краще запам’ятовувати інформацію, легше знайомитись з новою інформацією органічним шляхом, засвоювати матеріал у власному темпі, і для більшості видів ігор кількість людей які у них грають не обмежена. Гру достатньо розробити один раз і використовувати у процесі навчання доволі довгий період. Якщо гра навчає базовим необхідним навичкам, а не конкретним інструментам, то ці навички не застарівають з часу.

Бажано, щоб навчальна гра не залежала від підключення до інтернету, адже неможливо гарантувати постійне підключення до інтернету, бажано, щоб данні ігри було достатньо легко модифікувати без допомоги спеціалізованих інструментів, адже це полегшить додавання нових завдань, і бажано, щоб навички, котрі гра повинна розвивати, не були прив’язані до конкретного інструменту. Головоломки гарно підходять для розвитку математичних навичок, а створення алгоритмів для проходження рівнів гарно підходять для розвитку навичок розробки алгоритмів.

1.7. Концептуальна модель системи

При введенні даної системи в користування, для викладача достатньо лише створити завдання, після чого система може перевіряти правильність рішення без його втручання. Це дає можливість здобувачу виконувати завдання у своєму темпі, і перевірку можливо робити набагато частіше.

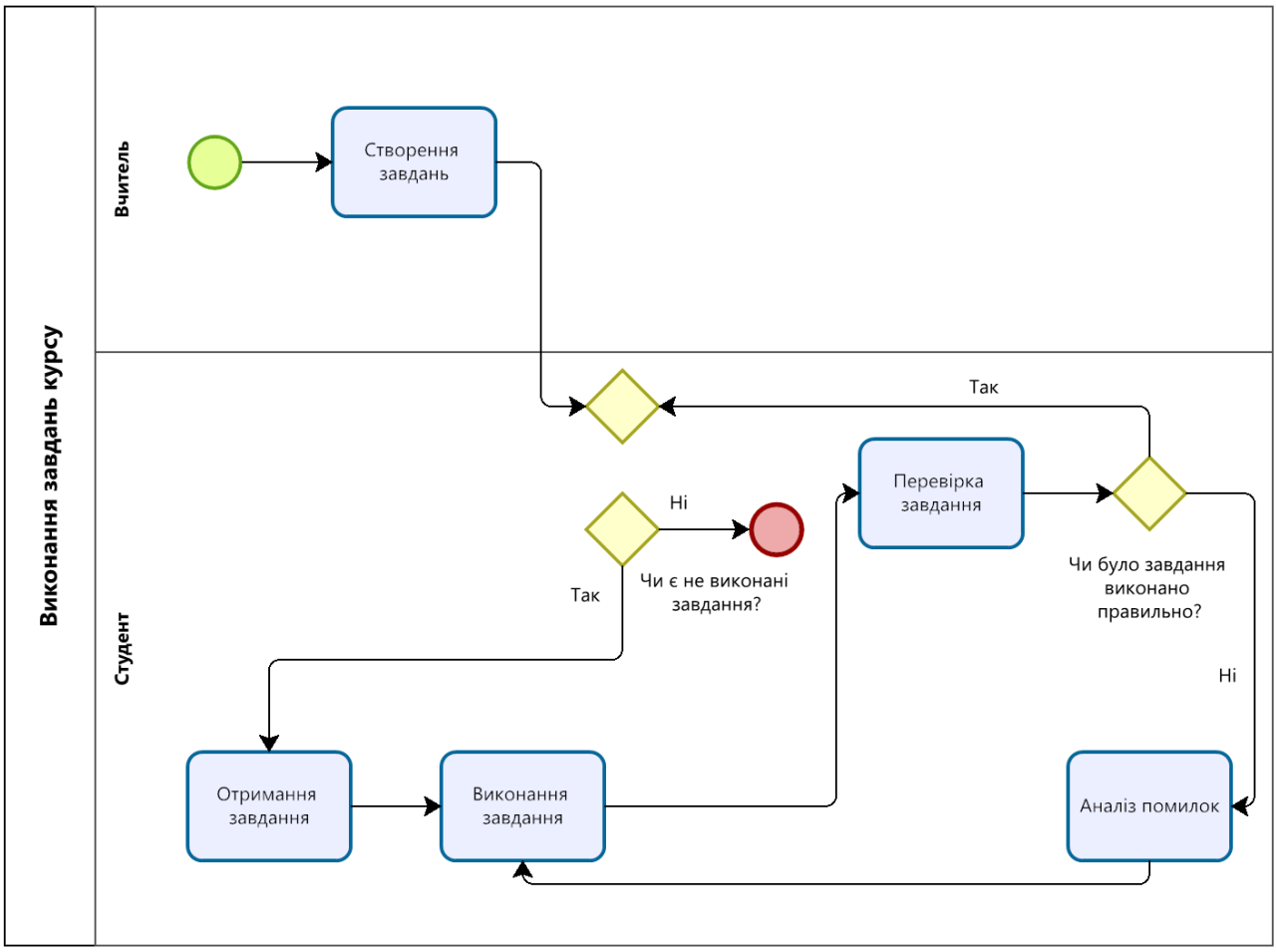

Рисунок 1.3 — Структурно-логічна схема виконання завдань курсу

Бажано, щоб було досить легко поповнювати список завдань новими завданнями. Звичайно, не усі завдання можна автоматизувати таким чином (наприклад, автоматизація перевірки складання звіту є занадто складною), але перевірку завдань ціль яких розвиток базових навичок, особливо у таких сферах як математика або створення алгоритмів, досить легко автоматизувати.

1.8. Висновок до розділу 1

Отже, пропонуємо розробити додаток для навчання та підтримки навичок пов’язаних з комп’ютерними науками у вигляді навчальної відеоігри. Дана відеогра не повинна потребувати підключення до інтернету і бути достатньо легко модифікованою.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ

2.1. Загальні положення

2.1.1. Найменування системи - «Система для курсів з вивчення комп'ютерних наук у вигляді навчальної відеоігри»

2.1.2. Результати робіт зі створення системи оформлюються згідно з вимогами ДСТУ на відповідні етапи розробки. Порядок оформлення і передачі результатів у даному випадку визначається змістом.

2.1.3. У випадку необхідності на наступних стадіях робіт по створенню системи окремі положення можуть уточнюватися і розвиватися.

2.2. Призначення і цілі створення системи

2.2.1. Призначення системи

Система призначена для допомоги учням у вивченні та закріпленні навичок, потрібних для комп’ютерних наук.

2.2.2. Цілі створення системи

Основною метою створення системи є розроблення додатку в ігровій формі для допомоги здобувачам і викладачам у вивченні та викладанні комп'ютерних наук.

2.3. Характеристики об’єкта автоматизації

2.3.1. Об’єктом автоматизації є навчальний процес здобувачів початкових курсів комп’ютерним наукам.

2.4. Вимоги до системи

2.4.1. Вимоги до системи в цілому

- Система повинна бути здатною працювати при відсутності мережі інтернету.

- Дані даної системи повинні бути у легкому для модифікації форматі.

- Завдання завдається «рівнем», структурою даних, котре включає в себе спеціалізовані данні завдання (фігури, таблиця даних, мапа рівня), опис та складність. Якщо завдання — це виділення деякого набору даних, то також повинні бути присутні цільові дані.

- У режимах, де завданням є виділення деякого набору даних, цільові дані повинні бути прописані як операція, що дає потрібний набір даних. Це потрібно для того, щоб не допустити неможливих завдань.

- Система повинна відстежувати, які рівні були пройдені, а які — ні. Якщо все ще не усі рівні були пройдені, то система повинна завантажувати непройдені рівні у порядку складності, а якщо усі рівні були пройдені, то система повинна завантажувати випадковий рівень. Якщо користувач зареєстрований, то список пройдених рівнів повинен зберігатися між сесіями.

2.4.2. Структура і склад системи

Дана система має включати в себе наступні режими:

Режим «PolyDraw», де користувачу потрібно складати логічні рівняння для того, щоб отримати потрібну фігуру.

Режим «SQL», де користувачу потрібно складати SQL запит для того, щоб отримати потрібні дані.

Режим «Algorithms», де користувачу потрібно складати алгоритм для того, щоб виконати задачу.

2.4.3. Вимоги до режиму «PolyDraw»

Ціллю даного режиму є розвиток навичок використання дискретної математики, а саме логічних операцій, булевої алгебри та алгебри множин.

Дається декілька фігур, і за допомогою логічних операцій потрібно отримати деяку множину точок.

Для цього використовуються операції: AND (перетин), OR (об’єднання), NOT (інверсія).

Для створення логічної операції потрібно використовувати візуальний інтерфейс у вигляді візуальних блоків. Блоки відповідають або логічній операції або фігурі. Блок фігури має текст кольору фігури, котрій цей блок відповідає. Блоки логічних операцій приймають у якості аргументів або блоки фігур, або блоки логічних операцій.

На екрані відображаються фігури, візуалізація потрібної множини точок та опис завдання.

Для отримання множини точок потрібно перенести візуальний блок у спеціальне поле та натиснути кнопку «Draw», після чого програма вираховує множину точок відповідну до вхідних даних та порівняє її зі заданою множиною точок. Після отримання множини точок ця множина точок візуалізується на екрані.

2.4.4. Вимоги до режиму «SQL»

Ціллю даного режиму є розвиток навичок використання мови запитів SQL.

Даються деякі дані, і потрібно за допомогою SQL запиту виділити деякий набір даних.

Для створення запиту SQL використовується текстове поле, і при натисненні кнопки здійснюється вибір даних та порівняння з цільовим вибором даних.

На екрані зображуються опис і дві таблиці, у першій виділений цільовий набір даних, у другій виділений набір даних, відповідний до зробленого запиту SQL.

2.4.5. Вимоги до режиму «Algorithms»

Ціллю даного режиму є розвиток навичок розробки алгоритмів.

Дається мапа, що складається з клітин. Клітини можуть бути землею (ground) або стіною (wall). На рівні знаходяться ключі. На мапі також знаходиться персонаж. Потрібно розробити алгоритм, за допомогою якого персонаж може зібрати усі ключі.

При відображенні рівня стіни зображуються як «#», земля як «_», персонаж як «@», і ключ як «$».

Персонаж може рухатися вперед та обертатися. Якщо персонаж встає на одну клітинку з ключем, то ключ збирається.

Розробка алгоритму робиться за допомогою візуальних блоків. Спеціальне поле приймає блоки, і при натисканні на кнопку «Start» починає програвати алгоритм.

Блоки алгоритмів можна розділити на три категорії: «Movement», «Condition and Loop», «Values».

«Movement» включає наступні блоки: Move, Rotate. Move рухає персонажа вперед, а Rotate повертає персонажа. Є два шляхи повертання персонажа: можна повертати декілька разів за годинниковою стрілкою, а можна повертати у деякий напрям.

«Condition and Loop» включає до себе такі блоки: If, Repeat.

«Values» включає наступні блоки: Print, IsWallAhead, Get, Set, Value. Print друкує до консолі, IsWallAhead перевіряє, чи на є на заданій відстані стіна, Set дозволяє створювати змінні та надавати їм значень, а Get дозволяє отримувати значення цих змінних, Value дозволяє встановити значення. Value, Get, IsWallAhead використовуються як аргументи в інших блоках.

2.4.5 Вимоги до технічного забезпечення

Операційна система Windows.

Процесор: x86_32 CPU з інструкціями SSE2, або будь-який x86_64 CPU.

Графічна карта з підтримкою OpenGL 3.3.

Монітор

Клавіатура та миша

2.5. Склад і зміст робіт по створенню системи

2.5.1. Стадії створення системи і терміни виконання робіт наведені в таблиці 2.1.

Таблиця 2.1. Найменування робіт при створенні системи

№ п/п	Найменування робіт	Строки виконання робіт
1	Аналіз області	20.03.2024
2	Розробка технічного завдання	29.03.2024
3	Прототипування	20.05.2024
4	Розробка додатку	24.05.2024
5	Оформлення документації	28.05.2024

2.6. Порядок контролю і приймання системи

2.6.1. Система вводиться на комп’ютері студента. До введення в дію система повинна пройти приймальні випробування згідно з ДСТУ 3974-2000.

2.6.2. Випробування для визначення працездатності і рішення про можливість приймання системи в дослідну експлуатацію проводять розробники разом із замовником. Програму випробувань складає розробник і затверджує замовник.

2.6.3. Здача в дослідну експлуатацію здійснюється на основі технічного завдання та інструкції користувача. За результатом дослідної експлуатації формується перелік доробок і рекомендовані строки їх виконання.

2.6.4. Введення в дію системи оформлюється актом здачі-прийому.

2.7. Вимоги до складу і змісту робіт із підготовки до введення системи в дію

Запустіть програму.

Дані повинні зберігатися у папці “users://”, що за замовченням знаходиться в “%USERPROFILE%\AppData\Roaming\Godot\app_userdata\2024,05,21”. Якщо папки “poly”, “sql”, “algo” в “users://levels” відсутні, то система згенерує та заповніть ці папки файлами за замовчуванням. При тому, якщо видалити файли з цих папок але не видаляти самі папки, то при наступному запуску система не буде перегенерувати ці файли. Папка “players” створюється, коли у перший раз зареєстрований користувач пройшов рівень.

Рівні зберігаються у форматі “.json”. Усі файли з форматом “.json”, що знаходяться у папках в “users://levels” вважаються за рівні. Рівні у папках “sql”, “algo” також мають посилання на файл з форматом “.csv”, котрий повинен знаходитися у тій самій папці.

2.8. Вимоги до документації

2.8.1. На систему розробляється комплекс документації у складі: технічне завдання та технічний проект.

2.8.2. Документація на систему розробляється у відповідності з вимогами Державних стандартів серії 19 «Єдина система програмної документації» та серії 24 «Єдина система стандартів автоматизованих систем управління».

2.9. Джерела розробки

2.9.1. При розробці технічного завдання на систему використано наступні документи:

ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлення.

ДСТУ 3973-2000. Система розроблення та поставлення продукції на виробництво.

ISO/IEC 23270:2018. Information technology. Programming languages C#.

ISO 21001. Educational organizations. Management systems for educational organizations.

РОЗДІЛ 3. ОПИС КОМПЛЕКСУ АВТОМАТИЗАЦІЇ

3.1. Інформаційне забезпечення системи

Система була створена у середовищі Godot 4 .NET з використанням бібліотеки SQLParser для C# від TylerBrinks[8].

Дані зберігаються у форматах json та csv. Дані управляються за допомогою нестандартної системи, створеною за допомогою методів Godot та C#. Усі файли формату json у папці players вважаються за данні користувачів, і усі файли у папках poly, csv, algo (котрі, у свою чергу, знаходяться у папці levels) вважаються за данні рівнів.

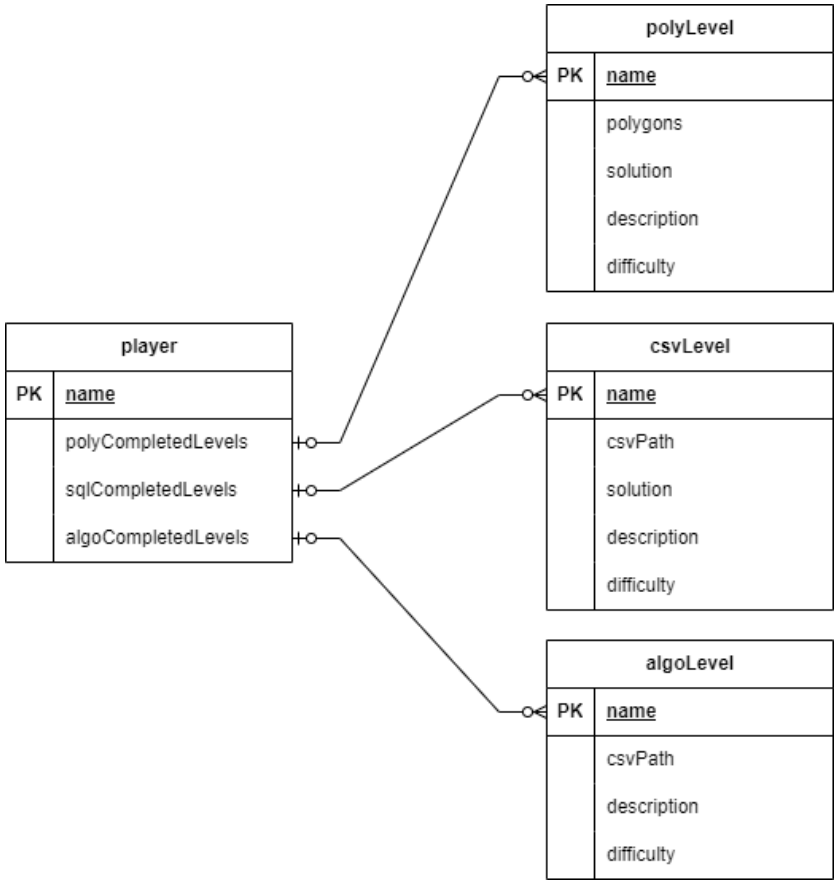

Рисунок 3.1 — Модель бази даних

3.2. Алгоритмізація та реалізація комплексу задач

Godot є ігровим рушієм, в основі якого лежить об’єктно-орієнтований дизайн та об’єктна композиція. Редактор Godot побудований на основі рушія Godot, і процес створення плагінів для Godot практично ідентичний до процесу створення ігор на Godot. Через це, Godot має зручніші методи побудови

інтерфейсу користувача для ігор у порівнянні з іншими ігровими рушіями, і є гарно придатним для створення неігрових додатків.

Після запуску програми запускається сінглтон AutoLoad, що буде працювати з моменту запуску до моменту закриття програми. Він відповідальний за імпорт та експорт даних. При запуску він імпортує дані про рівні, при реєстрації користувача він може імпортувати дані користувача, при завершенні рівня він оновлює списки завершених рівнів, і, якщо користувач зареєстрований, переписує зовнішні дані користувача, при скиданні даних він анулює списки завершених рівнів, і, якщо користувач зареєстрований, анулює дані користувача. Інші об’єкти можуть звертатися до AutoLoad для того, щоб завантажити рівень.

PolyDraw і Algorithms використовують систему візуальних блоків. Скрипт Draggable відповідає за можливість перетягувати блок. Блок можна перенести до об’єктів, до котрих прив’язані скрипті DropZone, DragZone, DragDisposer. DragZone дозволяє вільно переміщати блок у межах цього об’єкта. DropZone приймає один блок, і може дістатися до даних цього блоку. DragDisposer видаляє блок, котрий було перенесено до нього. DragCreate створює копію себе, але типа Draggable замість DragCreate. SizableDropZones створює дочірні об’єкти, до котрих прив’язані скрипти DropZone так, щоб був хоча б один пустий DropZone.

Скрипти DropZone, Draggable, DragCreate мають значення suffix у вигляді масиву рядків. Якщо suffix у DropZone має одне або більше значень, то у цей DropBox можна перенести лише блоки, котрі мають хоча б одне спільне значення у масиву suffix. За допомогою цієї системи можна обмежити додавання блоків у відповідні поля.

У режимі PolyDraw скрипт FigDrawer відповідальний за більшість подій: він запитує у AutoLoad данні рівня, генерує фігури, блоки та опис на основі отриманих даних, він же відповідальний за візуалізацію множин точок. Скрипт FigSummator відповідальний за інтерпретацію логічного виразу, створеного користувачем за допомогою блоків, і генерацію Abstract Syntax Tree (AST).

FigDrawer отримує від FigSummator данні про те, який логічний вираз зробив користувач, і вони разом генерують множини точок. FigDrawer порівнює множину точок користувача з цільовою множиною точок, і, якщо вони ідентичні, об’являє рівень завершеним.

У режимі SQL скрипт SceneControl запитує у AutoLoad данні рівня, генерує таблиці та опис на основі отриманих даних. Скрипт SqlEdit бере введений запит SQL, інтерпретує його за допомогою бібліотеки SqlParser, і порівнює з цільовими даними, і, якщо вони рівні, викликає метод Solved у SceneControl, котрий об’являє рівень завершеним.

У режимі Algorithms скрипт AlgorithmReader відповідальний за більшість подій: він запитує у AutoLoad данні рівня, генерує мапу та опис, він інтерпретує інструкції, створені користувачем за допомогою блоків, і генерує AST, після чого на базі цього AST виконує ці інструкції, і, якщо під кінець цих інструкцій персонаж зібрав усі ключики, об’являє рівень завершеним.

Шляхи подальшого вдосконалення системи полягають у наступному:

- Збільшити кількість режимів. Можна додати потребу пройти один режим щоб почати інший (наприклад, режими, пов’язані з теорією ймовірностей, відкриваються після проходження режимів, пов’язаних з дискретною математикою).

- Додавання теоретичного матеріалу, що дозволить пояснювати більш складну інформацію.

- Додавання додаткового сюжету та персонажів, що може зробити дану гру більш цікавою. Також можливо змінити графічне відображення рівнів та користувацького інтерфейсу для кращої інтеграції сюжету і завдань.

- Додавання більш складної системи оцінювання, замість бінарного «пройшов»/«не пройшов».

- Додавання різних рівней успішності проходження (наприклад, скільки блоків потрібно було для створення алгоритму, чи наскільки швидко алгоритм виконав свою дію).

- Додавання можливості експорту результатів гравця та генерацію «сертифікату».

Звичайно, потрібно мати на увазі, що усі компоненти взаємопов’язані, й зміна одного компонента може дуже сильно вплинути на відчуття усієї гри. Наприклад, створення конструкторів рівнів може допомогти у створенні нових рівнів і дозволить подалі створювати більш складні режими, але це може призвести до конфлікту з філософією «данні повинні не потребувати спеціалізованих інструментів для легшого модифікування або добавлення рівнів». Окрім цього, деякі ідеї вдосконалення можуть бути непоганими на папері, але на практиці конфліктувати зі філософією системи.

В цілому, у різних системах інтерактивного навчання є різні сильні та слабкі сторони, через що створення ідеальної системи для навчання є неможливим, адже різні системи мають різні методи та цілі навчання. Має сенс використовувати більше різних видів систем для інтерактивного навчання замість того, щоб намагатися розробити одну систему, що буде використовуватися в усіх випадках.

3.3. Інструкція користувача

Після запуску програми з’являється головне меню програми:

Рисунок 3.2 — Головне меню, реєстрація

Головне меню складається з двох частин. Перша частина — це реєстрація. Користувач може зареєструватися, а може продовжити без реєстрації. Реєстрація потрібна для збереження результатів користувача між сесіями, і якщо не зареєструватися, то результати не будуть зберігатися.

Після реєстрації або продовженні без реєстрації користувач переходить до другої частині головного меню, вибору режиму.

Рисунок 3.3 — Головне меню, вибір режиму, без реєстрації

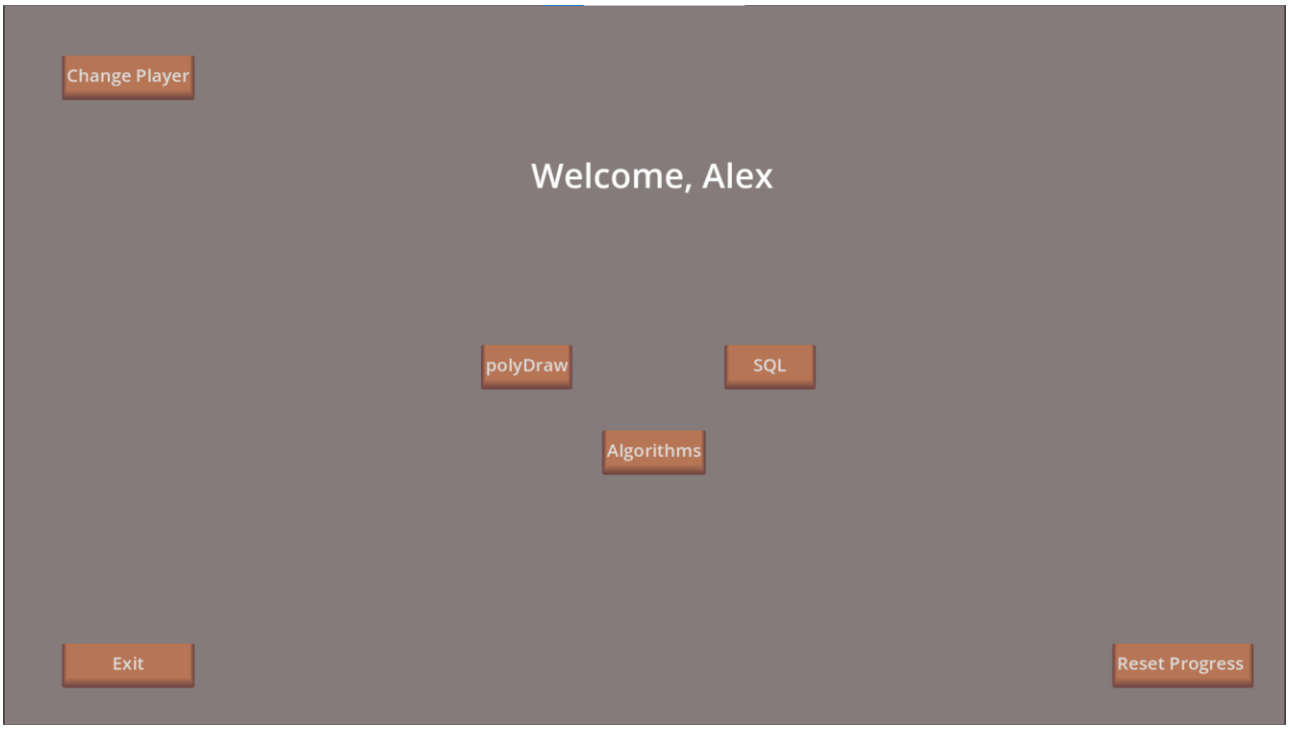

Рисунок 3.4 — Головне меню, вибір режиму, з реєстрацією

У вікні вибору режиму користувач може змінити гравця (перереєструватися), скинути прогрес, та вибрати режим поміж «PolyDraw», «SQL», «Algorithms».

При виборі режиму, система завантажує рівень. Який саме рівень завантажується залежить від того, які рівні пройшов користувач. Якщо користувач ще не пройшов усі рівні, то завантажується непройдений рівень з

найменшою складністю. Якщо є декілька непройдених рівнів з однаковою найменшою складністю, то з них вибирається і завантажується випадковий рівень. Таким чином користувач починає з більш простих рівнів, і з часом доходить до більш складних. Якщо користувач пройшов усі рівні, то завантажується випадковий рівень будь-якої складності.

В залежності від того, чи присутні у даному режимі непройдені рівні, змінюється вікно, котре з'являється при завершенні рівня та дає вибір чи завантажити новий рівень, чи повернутися до головного меню.

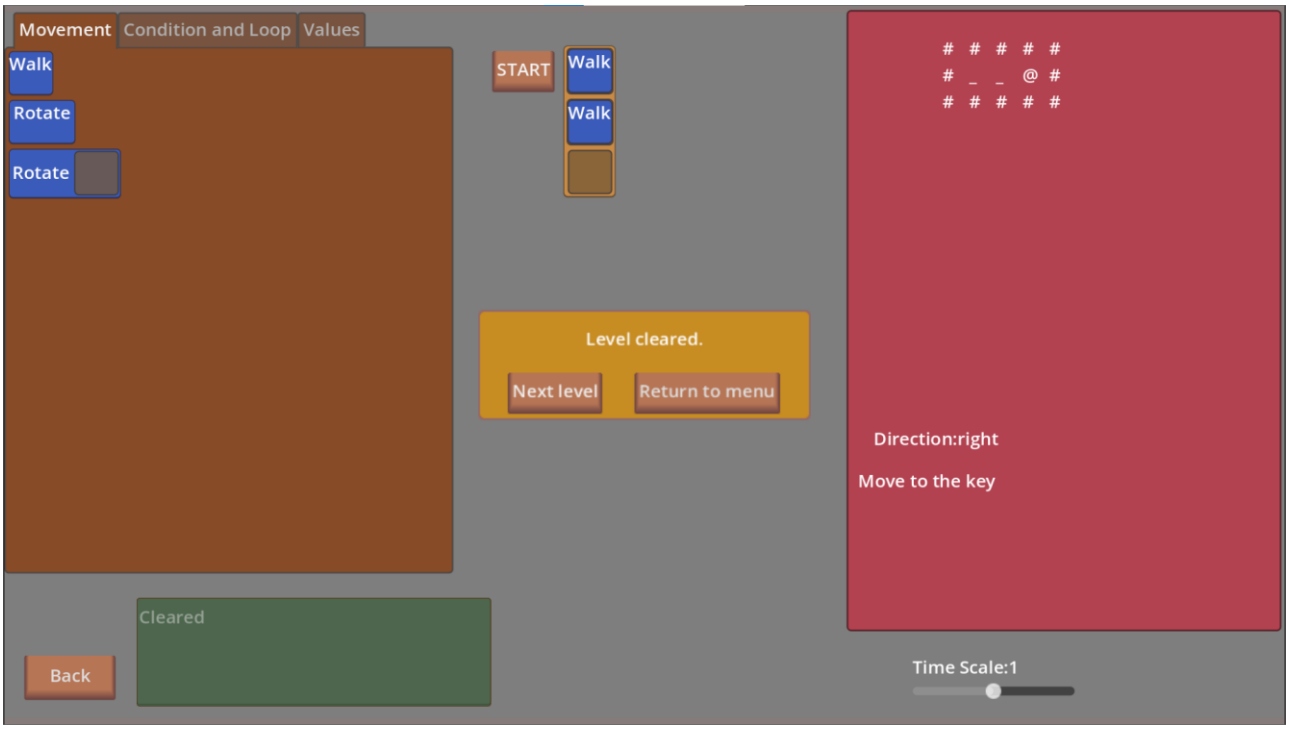

Рисунок 3.5 — Вікно повідомлення, не всі рівні пройдені

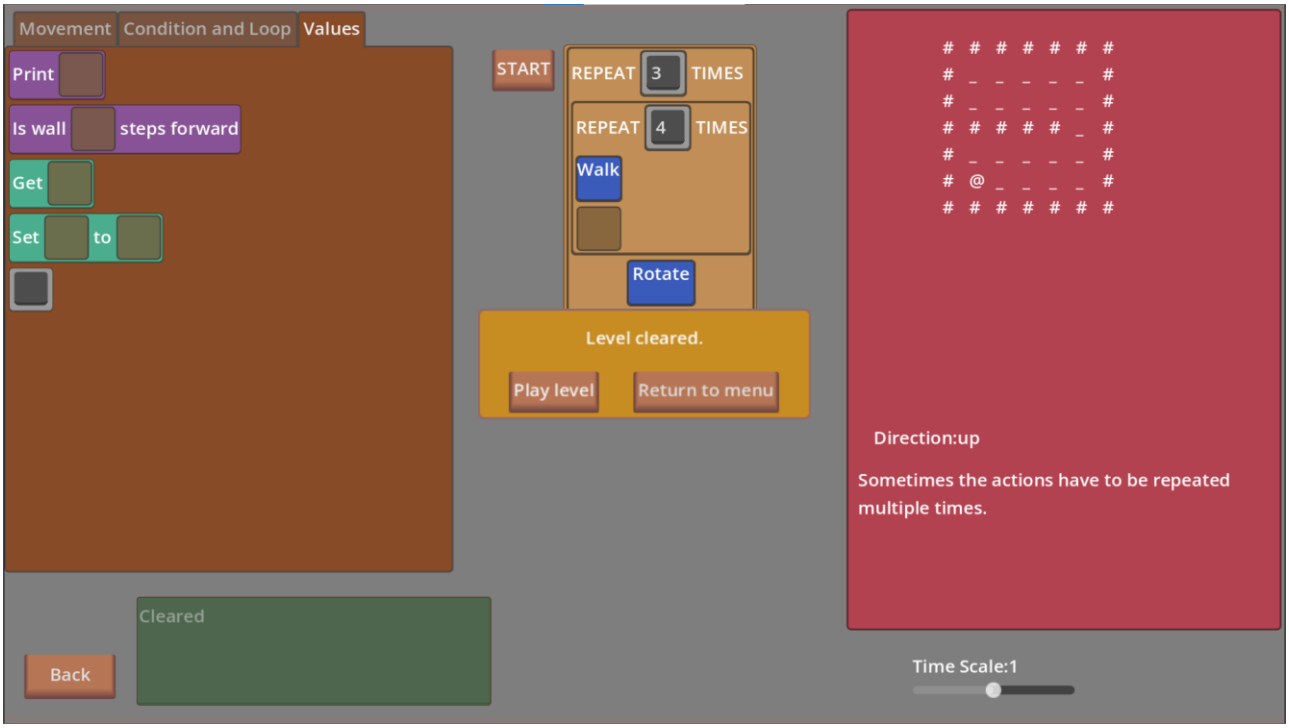

Рисунок 3.6 — Вікно повідомлення, не всі рівні пройдені

Режим PolyDraw — це режим, де користувачу потрібно складати логічні рівняння для того, щоб отримати потрібну фігуру. Ціллю даного режиму є розвиток навичок дискретної математики.

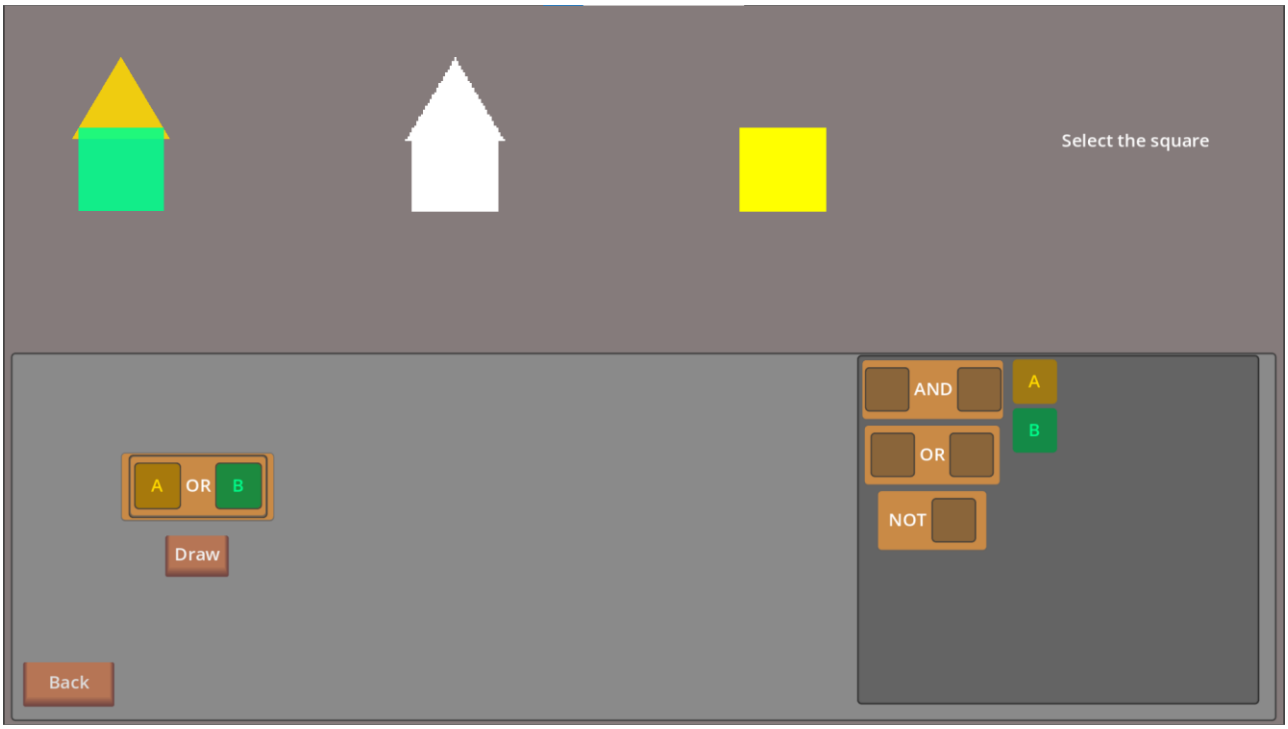

Рисунок 3.7 — Режим PolyDraw

У верхній половині екрану показуються фігури, множина обраних користувачем точок (на момент завантаження рівня вона пуста), цільова

множина точок та опис завдання. Множина обраних користувачем точок зображується білим кольором, а цільова множина точок зображується золотим кольором.

У нижній половині екрану присутні кнопка «Draw» та поле поряд з нею, у котру вставляються блоки для обирання множини точок, і при написканні кнопки «Draw» візуалізується множина обраних точок, і якщо вона відповідає до цільової множини, то рівень вважається пройденим. Блоки беруться з правої половині нижньої половині екрану. Ті блоки, що присутні при завантаженні рівня, замість того, щоб рухатися на інше місце при перетаскуванні, створюють аналогічні блоки на цільовому місці. Ці створені блоки, у свою чергу, не створюють дублікатів, і їх можна переміщувати у межах доступної зони. Якщо їх перемістити у більш темну зону у правій нижній половині екрану (котра включає до себе блоки-створювачі), то ці створені блоки будуть видалені.

У режимі PolyDraw доступні такі блоки: блоки фігур, кожен з котрих відповідає одній з фігур, та логічні блоки AND, OR, NOT. Логічні блоки мають поля, котрі повинні бути заповнені іншими блоками. Ці блоки можуть бути як логічними блоками, так і блоками фігур. Таким чином за допомогою невеликої кількості блоків можна виконувати досить велику кількість операцій.

Рівень вважається пройденим, коли множина обраних точок відповідає до цільової множини.

Режим SQL – це режим, де користувачу потрібно складати SQL запити для того, щоб обрати потрібні данні. Ціллю даного режиму є розвиток навичок складання SQL запитів.

Рисунок 3.8 — Режим SQL

У верхній половині екрану показуються таблиця з обраними цільовими даними, таблиця з даними, обраними користувачем та описом завдання. В цільовій таблиці обрані дані позначаються золотим кольором, в той час як в таблиці користувача обрані дані позначаються зеленим кольором.

У нижній половині екрану присутнє текстове поле, куди вводиться SQL запит, та кнопка, при натисканні на яку виконується SQL запит та обираються дані відповідно до запиту. Доступні такі команди та оператори SQL: SELECT, TOP, HAVING, AND, OR.

Рівень вважається завершеним, коли обрані користувачем дані відповідають цільовим даним.

Режим Algorithms – це режим, де користувачу потрібно складати алгоритм руху персонажу для того, щоб зібрати усі ключі.

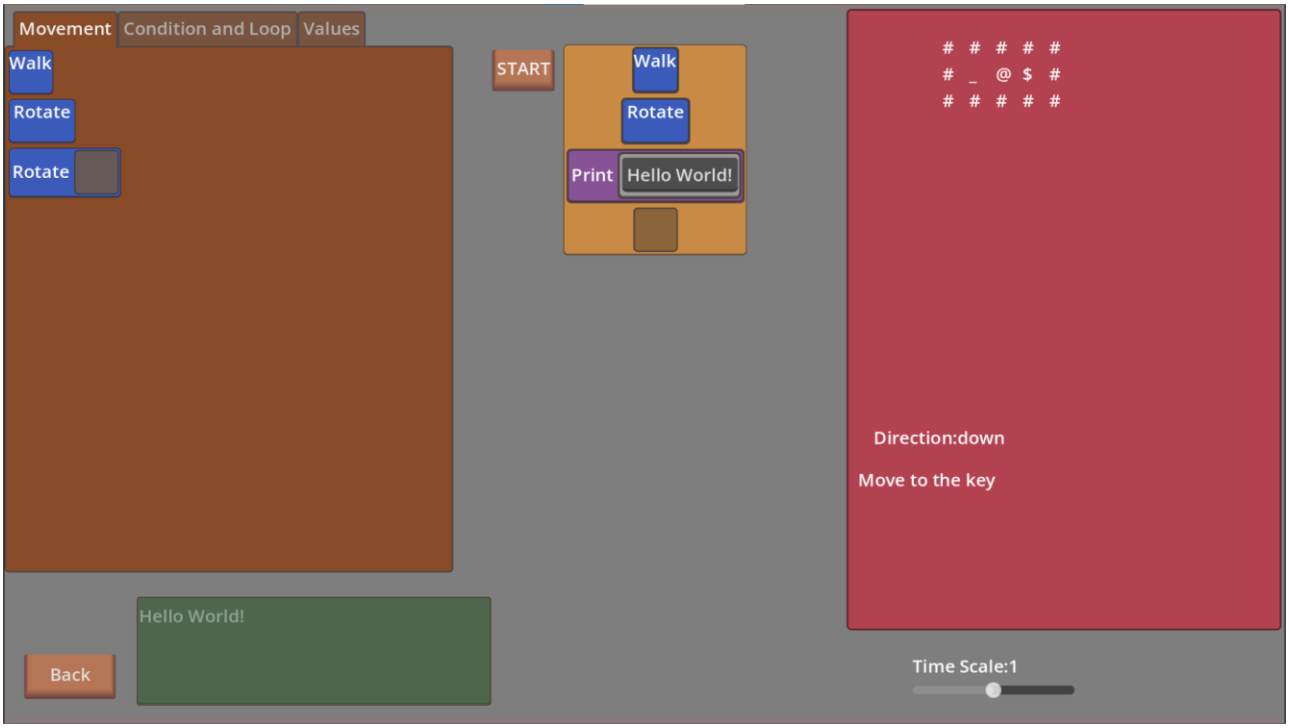

Рисунок 3.9 — Режим Algorithms, вкладка Movement

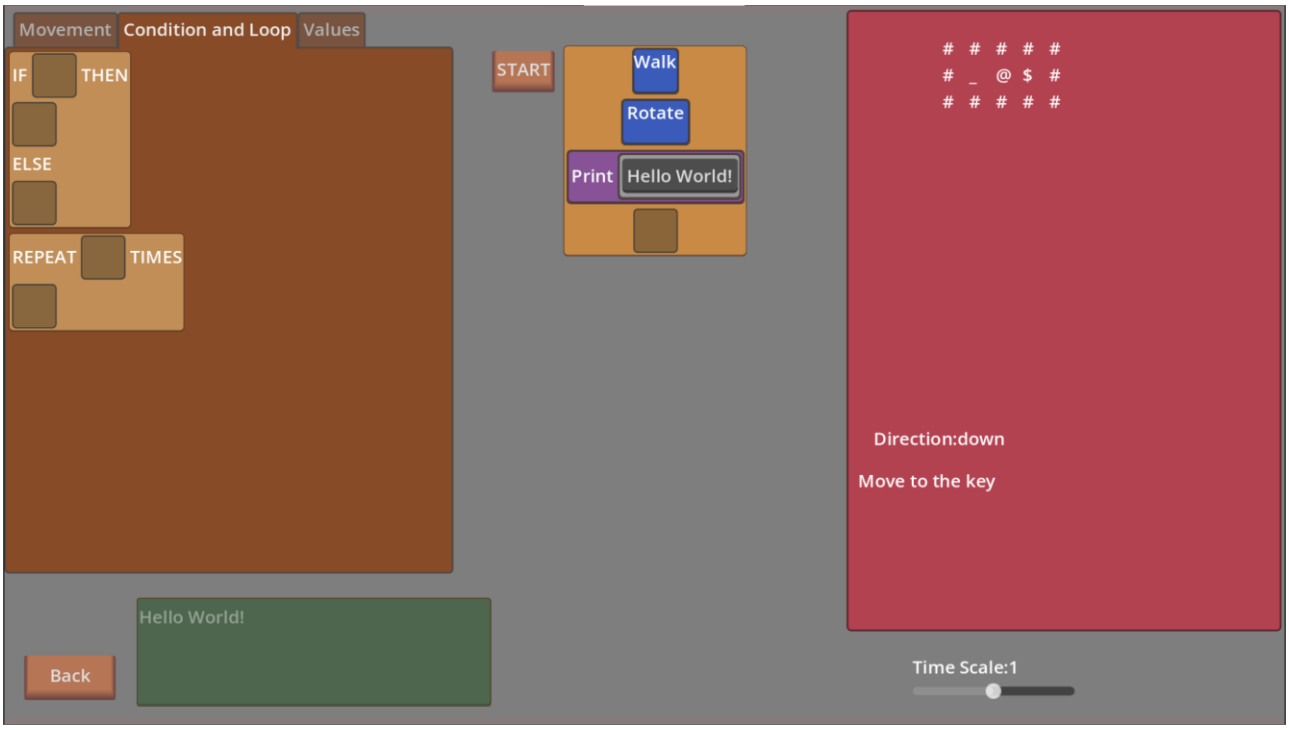

Рисунок 3.10 — Режим Algorithms, вкладка Condition and Loop

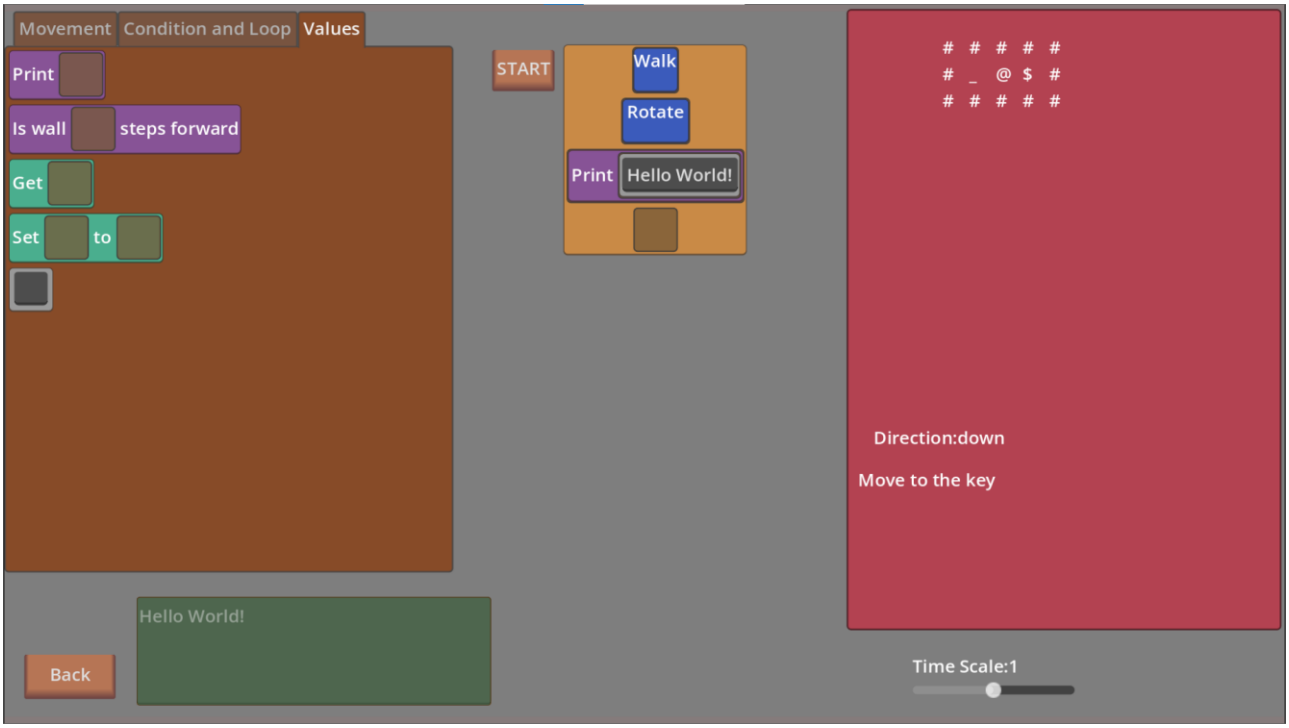

Рисунок 3.11 — Режим Algorithms, вкладка Values

Екран можна розділити приблизно на три частини.

У лівій частині екрану знаходяться блоки-створювачі, за допомогою яких можна створювати блоки візуального програмування. Ці блоки розподілені по трьох вкладках: Movement, Condition, Loop.

Доступні наступні блоки:

Move – персонаж йде один крок вперед;

Rotate – персонаж повертається. Має два варіанта: перший не потребує вказувати значення і персонаж повертається один раз на 90 градусів за годинниковою стрілкою. Другий потребує вказати значення, якщо це значення — це назва напряму, то повертає у цьому напрямку, а якщо це значення — це число, то персонаж повертається на 90 градусів заданою кількістю разів (якщо це число має позитивне значення, то персонаж повертається за годинниковою стрілкою, якщо негативне — то проти годинникової стрілки). Значення може задаватися також за допомогою змінної.

If – якщо виконується умова, то виконується один набір інструкцій, якщо не виконується, то інший. Умова може бути задана за допомогою блока IsWallAhead, змінної, або значення.

Repeat – повторює набір інструкцій заданий набір разів. Можна задати за допомогою значення або змінної.

Print – виводить текст у консоль.

Is Wall [] Steps Forward – блок IsWallAhead, перевіряє, чи є на заданій відстані стіна. Приймає параметр відстані у вигляді значення або змінної.

Get – повертає значення змінної. Приймає ім’я змінної, котру потрібно повернути, у вигляді значення або змінної.

Set – задає значення змінної. Якщо відсутня дана змінна, то створює її. Має два параметри, ім’я та значення, і приймає у вигляді значення або змінної.

Значення — блок, котрий використовується для задання значень.

Блоки Значення, Get, та IsWallAhead не можна використовувати як інструкції, їх можливо використовувати лише як значення для параметрів інструкцій.

Знизу від блоків-створювачів знаходиться консоль, у котру виводяться повідомлення, котрі або були виведені за допомогою інструкції Print, або виводяться через помилки в алгоритму.

У правій частині екрану показується мапа, де стіни позначаються як «#», земля як «_», персонаж як «@», і ключ як «$». Під картою знаходиться текст, котрий показує направлення персонажу, та опис завдання.

У центрі екрану знаходяться кнопка «Start» та поля для вводу інструкцій. Якщо останнє поле для вводу інструкцій заповнене, то створюється ще одне, а якщо два останніх поля для вводу інструкцій пусті, то одне з них прибирається. Таким чином завжди присутнє хоча б одне пусте поле для вводу інструкцій. При натисненні кнопки «Start» введені інструкції виконуються одна за одним.

Рівень вважається завершеним, якщо персонаж зібрав усі ключі.

Окрім цього, також у кожному режимі присутня кнопка «Back», котра дозволяє вийти до головного меню, а у головному меню присутня кнопка «Exit», котра дозволяє закрити гру.

3.4. Обґрунтування вибору технічних засобів

Godot 4 .NET було обрано за наступними причинами: це безплатний open-source ігровий рушій, яким достатньо легко користуватися, він гнучкий, надає можливість зручної розробки інтерфейсу користувача і підтримує C#. Не дивлячись на це, на даний момент у Godot 4 .NET є недолік, що полягає у неможливості експорту проекту у вигляді веб-додатку.

Для збереження даних обрано формати json та csv, що сприяє простоті розробки та модифікації системи.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Режим праці

Робота з комп’ютером нерідко включає до себе розумову напругу і напругу очей та може призводити до болі у спині. Через це досить важливим є дотримання правильного режиму праці та відпочинку. Необхідні перерви для відпочинку та прийняття їжі і перерви для відпочинку та особистих потреб. Бажано мати хоча б 15 хвилин перерви після кожної години роботи за комп’ютером. Тривалість безперервної роботи не повинна бути вищою за 4 годин. Для зменшення монотонності роботи є сенс чергувати різні види робіт та діяльностей. Бажано час від часу під час відпочинку займатися гімнастичними вправами.

4.2. Безпека робочого місця

4.2.1. Пожежна безпека

Потрібно дотримуватися вимог пожежної безпеки, такі як наявність і розташування пожежних виходів, наявність пожежної системи, наявність протипожежного обладнання і проведення тренінгів по пожежній безпеці.

4.2.2. Електробезпека

Потрібна регулярні перевірка та обслуговування електрообладнання, кабелі та проводи не повинні мати пошкоджену ізоляцію, кабелі і проводи з неізольованими провідниками не повинні залишатися під напругою.

4.3. Організація робочого місця

4.3.1. Освітлення

Потрібне достатнє освітлення робочого місця, воно не повинно бути занадто яскравим або тьмяним і не повинно створювати блисків на моніторі. Освітлення повинно забезпечувати видимість у робочому місці.

4.3.2. Шум

Шум може негативно впливати на здоров'я людини, викликати головну біль та стрес, проблеми з пам’яттю та апетитом, підвищену втому, погіршується концентрація, тощо. Через це бажано держати рівень шуму на мінімумі. Одна

проблема — більшість комп’ютерів створюють шум. Через це бажано, якщо можливо, облицьовувати звукопоглинальними матеріалами приміщення, якщо там створюється забагато шума.

4.3.3. Мікроклімат

Мікроклімат робочого місця включає такі параметри: температура, вологість, напрям і швидкість руху повітря, якість повітря, тощо. Бажано дотримуватися такого мікроклімату, при котрому людині буде комфортно і при котрому комп’ютер буде працювати оптимально. Для забезпечення мікроклімату використовуються вентиляція, кондиціонери, опалювальна система, тощо.

ВИСНОВКИ

В кваліфікаційній роботі розроблена система у вигляді додатку в ігровій формі для допомоги здобувачам і викладачам у вивченні та викладанні комп'ютерних наук. Всі задачі, поставлені в технічному завданні, було виконано.

Даний програмний продукт може використовуватися як додатковий засіб самостійної підготовки здобувачів, для самонавчання в домашніх умовах, а також в ролі розважальної програми.

Проєкт було виконано у середовищі Godot 4 .NET з використанням мови програмування C#.

Під час розробки були розроблені дані системи: система візуальних блоків, зроблена за допомогою методів для розробки Drag-and-Drop інтерфейсів в Godot; інтерпретація візуальних блоків і створення Abstract Syntax Tree на основі цих візуальних блоків; перевірка, чи знаходиться точка у межах полігону за допомогою класу Geometry2D в Godot; створення множини точок за допомогою AST, згенерованого на основі візуальних блоків, що відповідають логічним операціям; візуалізація цих множин точок; інтеграція бібліотеки SQLParser для C# у середовище Godot; система для управління даними тощо.

За допомогою даної системи можна організувати навчання та підтримку навичок, пов’язаних з комп’ютерними науками. Якщо представляти нові елементи поступово, то опису завдань достатньо для навчання даним навичкам. Видача завдань в залежності від складності призводить до того, що користувачу потрібно пройти спочатку легші завдання, а потім складніші, і таким шляхом можна представляти нові елементи поступово.

Можливі шляхи подальшого вдосконалення у вигляді добавлення нових сюжетів, кращої графіки, сюжету, тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Факультет АКС НУХТ - Історія. *Факультет АКС НУХТ*. URL: https://acs.nuft.edu.ua/about/history (дата звернення: 24.05.2024).

2. Кафедра ІТШІК НУХТ - Історія. *Кафедра ІТШІК НУХТ*. URL: https://it.nuft.edu.ua/about/history (дата звернення: 23.05.2024).

3. Кафедра ІТШІК НУХТ - Інформаційні системи та штучний інтелект. *Кафедра ІТШІК НУХТ*. URL: https://it.nuft.edu.ua/entrance/opp/opp_isai (дата звернення: 23.05.2024).

4. Освітні програми 2020 року | НУХТ. *Національний університет харчових технологій | НУХТ*. URL: https://nuft.edu.ua/spivrobitnyku/osvitni-programi/bakalavri/2020-rik (дата звернення: 23.05.2024).

5. Scratch - Imagine, Program, Share. *Scratch - Imagine, Program, Share*. URL: https://scratch.mit.edu/ (дата звернення: 27.05.2024).

6. CodeCombat - Coding games to learn Python and JavaScript. *CodeCombat*. URL: https://codecombat.com/ (дата звернення: 27.05.2024).

7. Brilliant | Learn interactively. *Brilliant | Learn interactively*. URL: https://brilliant.org/ (дата звернення: 27.05.2024).

8. GitHub - TylerBrinks/SqlParser-cs: A Friendly SQL Parser for .NET. *GitHub*. URL: https://github.com/TylerBrinks/SqlParser-cs (дата звернення: 27.05.2024).

9. Методичні рекомендації до викон. випускної кваліфікаційної роботи на здобуття освітнього ступеня «Бакалавр» спеціальності 122 «Комп’ютерні науки» освітньо-професійної програми «Інформаційні системи та штучний інтелект» ден. форми навчання [Електрон. ресурс] / уклад. О. М. М’якшило, М. П. Костіков. – К.: НУХТ, 2022. – 34 с.

10. Конспект лекцій з дисципліни [електронний ресурс] "Проектування інформаційних систем" для студентів спеціальності 122 "Комп'ютерні науки" /укл. М’якшило О.М., Харкянен О.В.- К.:НУХТ, 2018 – 48 с.

11. М'якшило, О. М. CASE-технології у проектуванні інформаційних систем [Електронний ресурс] [Текст] : навч. посіб. / О. М. М'якшило, Л. Г. Загоровська. — Київ : НУХТ, 2017. — 190 с. — каф. інформаційних систем.

12. Проектування та розробка програмного забезпечення [Електронний ресурс] : лабораторний практикум для здобувачів освітнього ступеня «Бакалавр» спеціальності 122 «Комп’ютерні науки» освітньо-професійних програм «Комп’ютерні науки» та «Інформаційні системи та штучний інтелект» денної та заочної форм навчання / укладачі : О. М. М’якшило, О. В. Харкянен ; Національний університет харчових технологій. – Київ : НУХТ, 2022. – 102 с.

13. Управління IT проектами [Електронний ресурс]: методичні рекомендації до самостійної роботи для здобувачів освітнього ступеня «Бакалавр» спеціальності 122 «Комп’ютерні науки» освітньо-професійних програм «Комп’ютерні науки» та «Інформаційні системи та штучний інтелект» денної та заочної форм навч. / уклад. С. В. Грибков, О. Л. Сєдих – К.: НУХТ, 2022 – 27 с.

14. NYU Game Center. NYU Game Center Lecture Series Presents Nicky Case, 2019. *YouTube*. URL: https://www.youtube.com/watch?v=WUI6X9YJiGE (дата звернення: 28.05.2024).

ДОДАТКИ

Додаток А. Модель бази даних

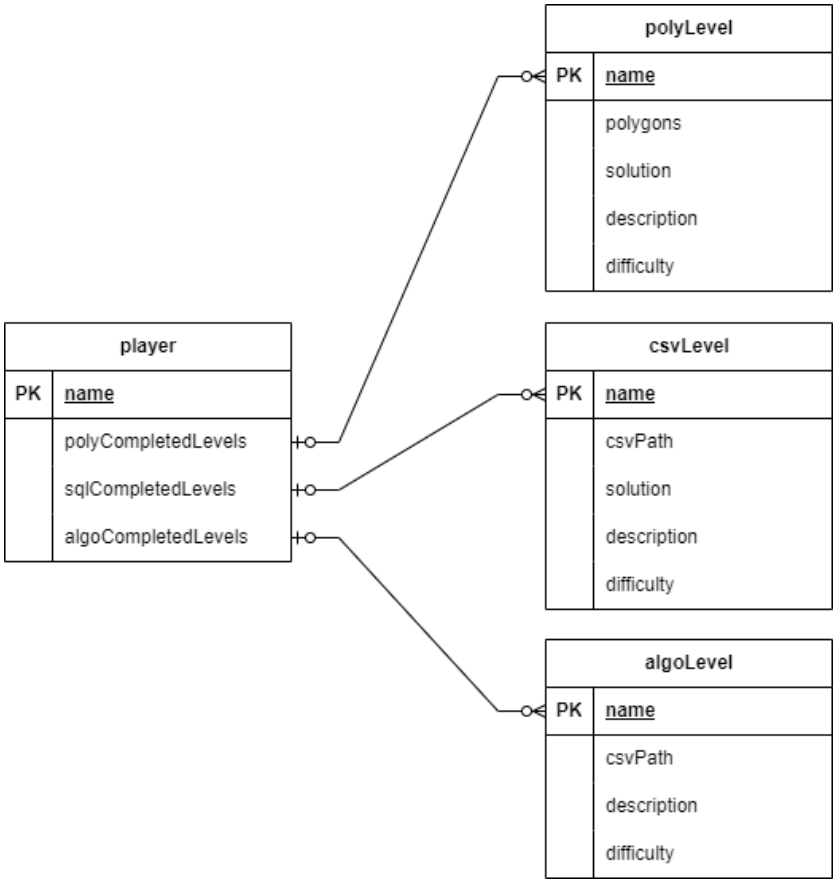

Рисунок А.1 — Модель даних

Додаток Б. Інтерфейс користувача

Рисунок Б.1 — Головне меню, реєстрація

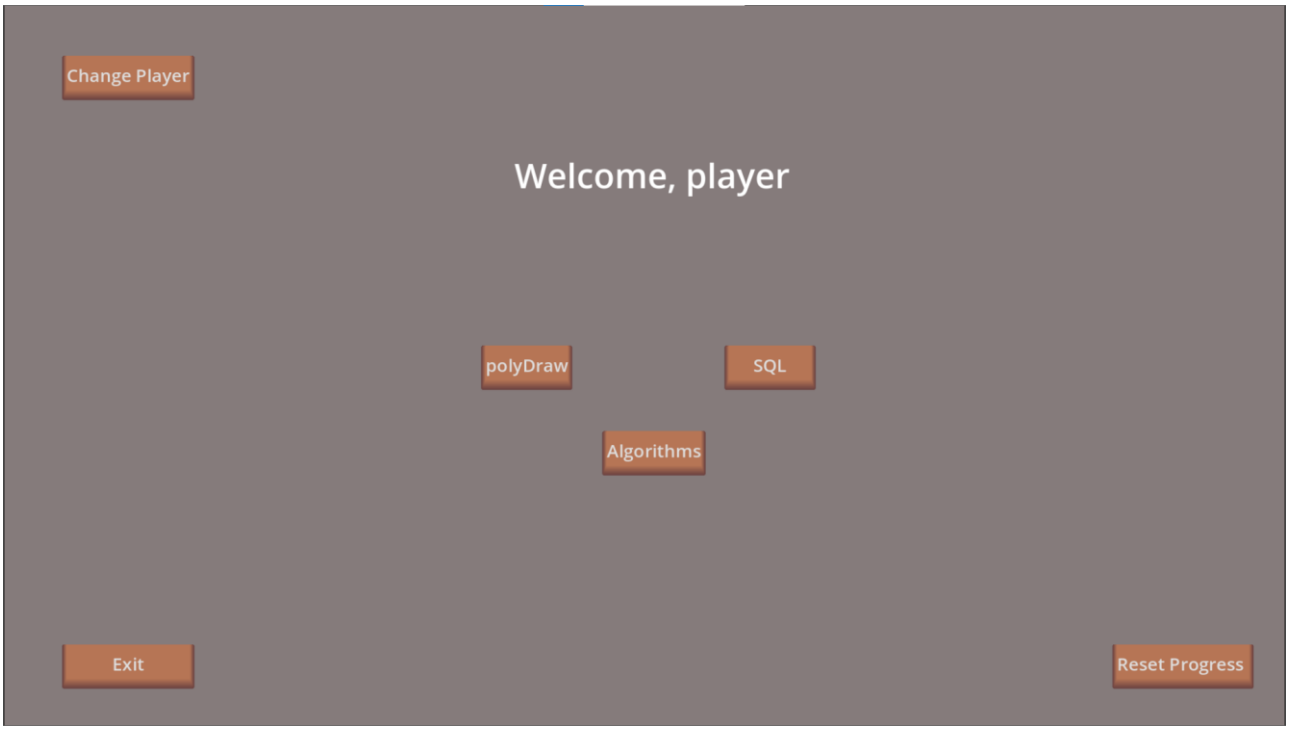

Рисунок Б.2 — Головне меню, вибір режиму, без реєстрації

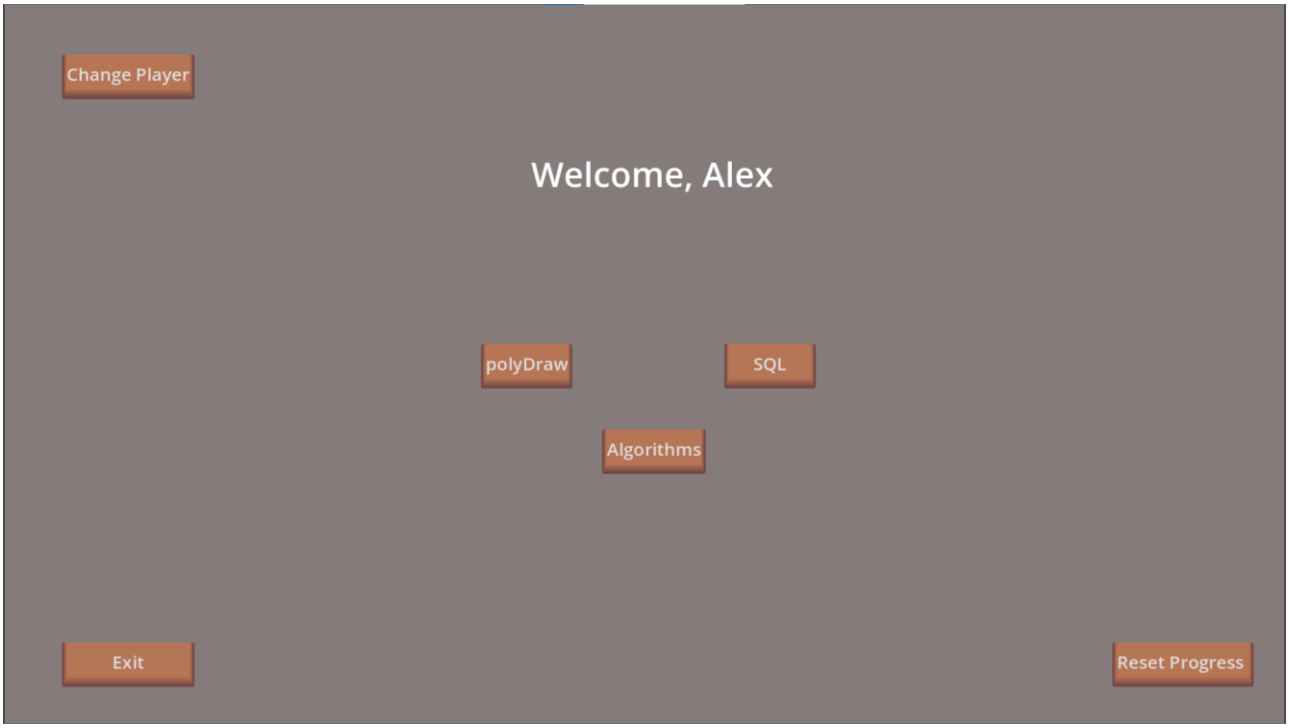

Рисунок Б.3 — Головне меню, вибір режиму, з реєстрацією

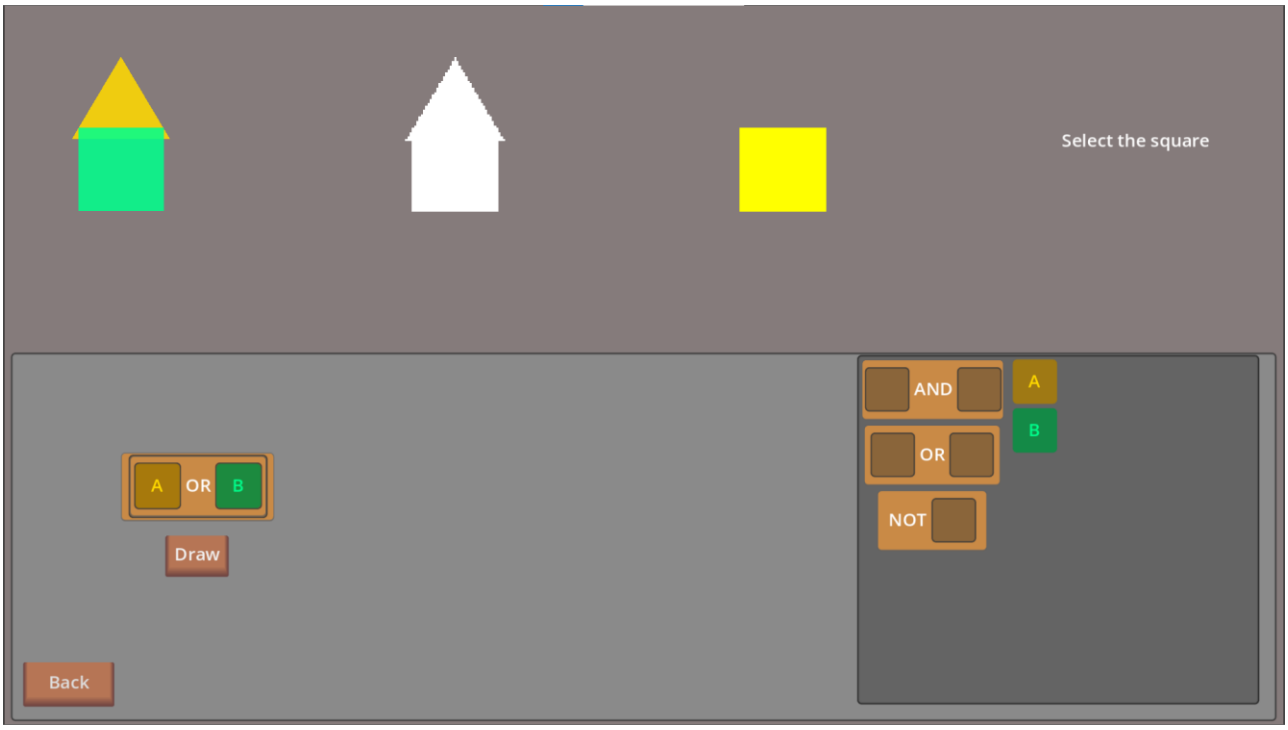

Рисунок Б.4 — Режим PolyDraw

Рисунок Б.5 — Режим SQL

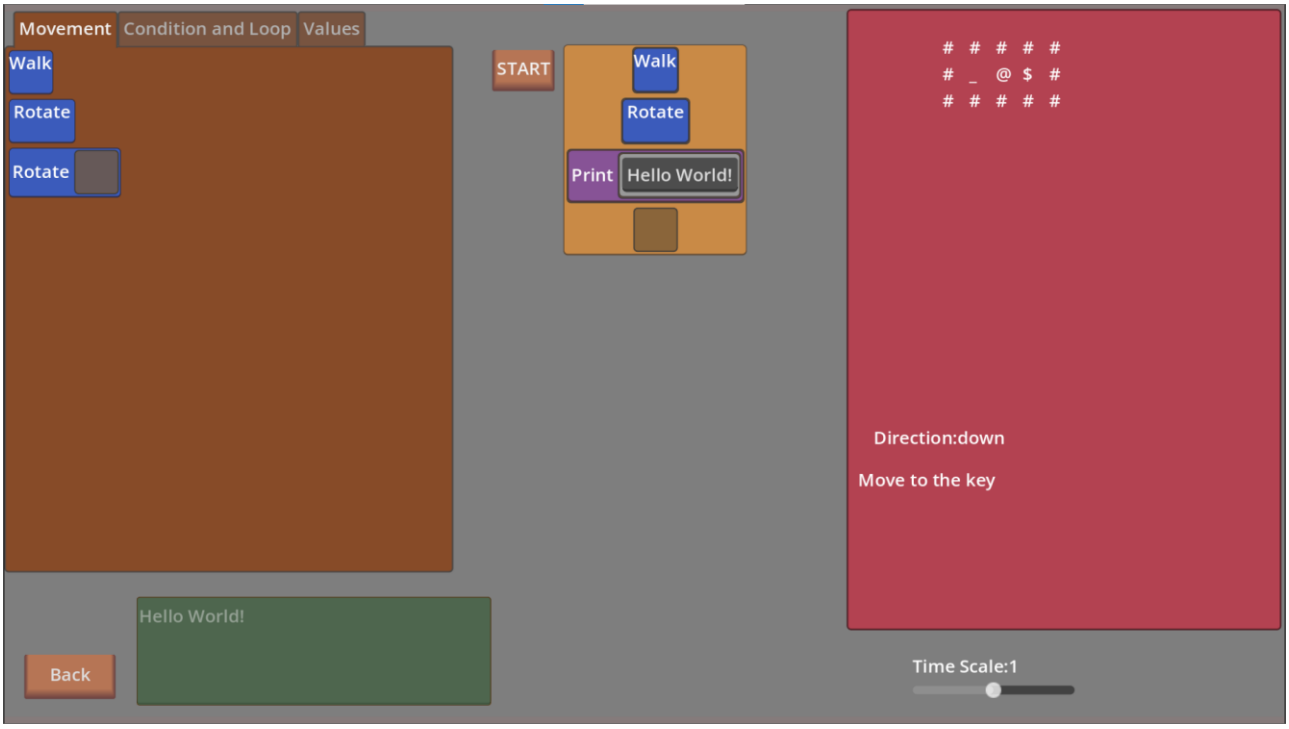

Рисунок Б.6 — Режим Algorithms, вкладка Movement

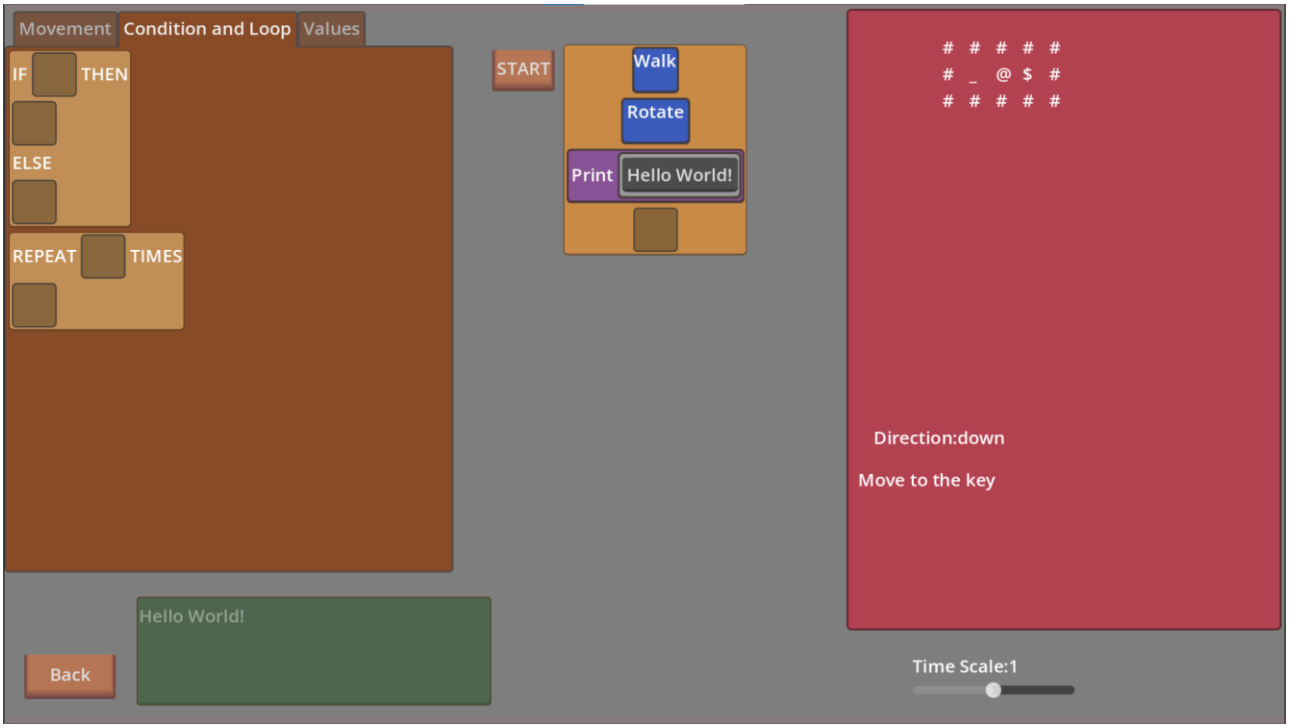

Рисунок Б.7 — Режим Algorithms, вкладка Condition and Loop

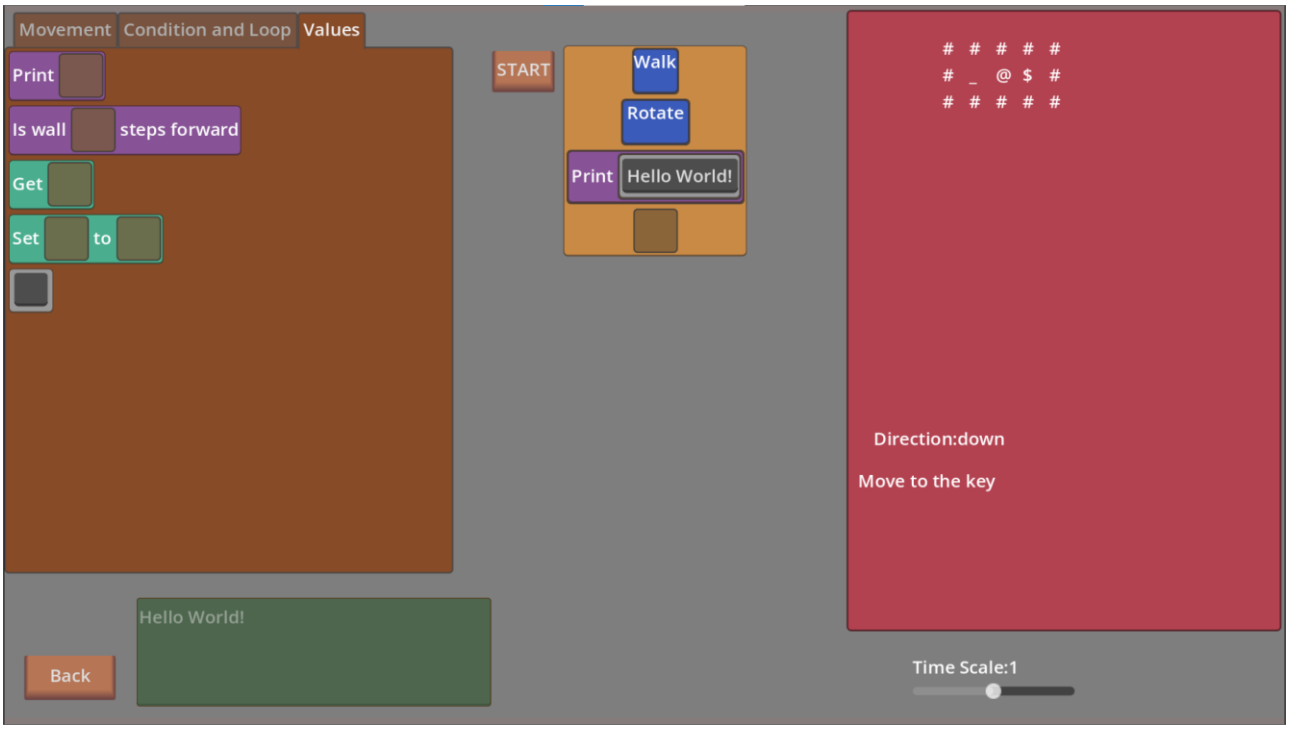

Рисунок Б.8 — Режим Algorithms, вкладка Values

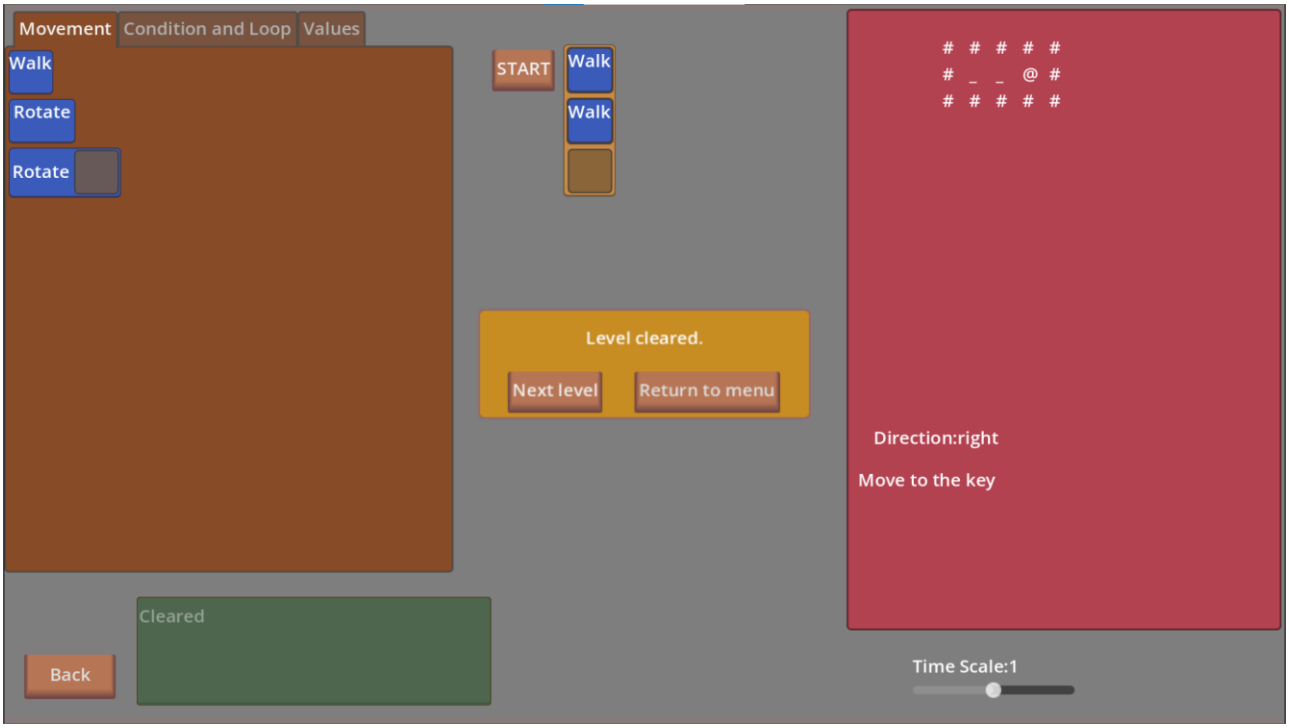

Рисунок Б.9 — Вікно повідомлення, не всі рівні пройдені

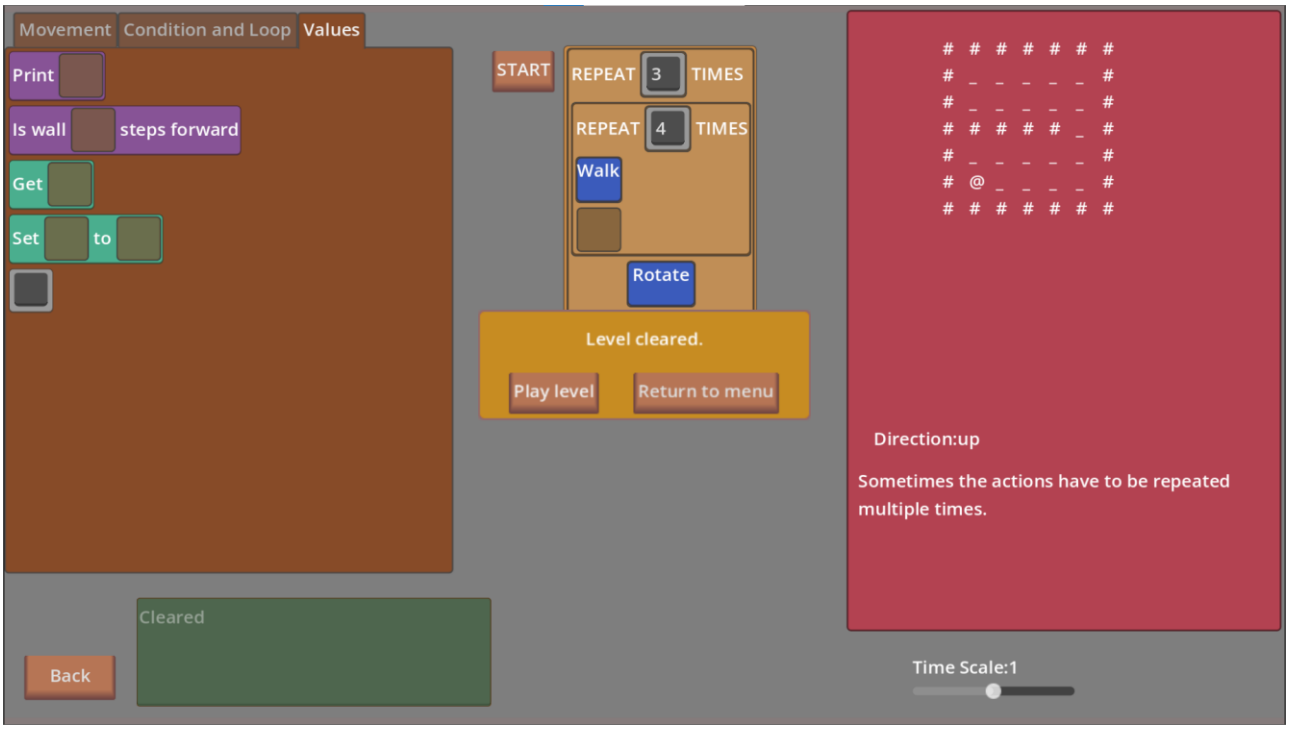

Рисунок Б.10 — Вікно повідомлення, не всі рівні пройдені

Додаток В. Приклади даних рівнів

Приклад даних рівня для режиму PolyDraw:

{
"description":"Select the area that belongs to A OR to B",
"difficulty":"1",
"polygons":{"A":{"color":"(1, 0.851, 0, 1)","contour":"[(62, 120), (106, 46), (150, 120)]","holes":["[]"]},"B":{"color":"(0, 1, 0.549, 1)","contour":"[(68, 110), (68, 185), (145, 185), (145, 110)]","holes":["[]"]}},
"solution":{ "Data": [{ "Data": "A", "Type": "dataHolder" }, { "Data": "B", "Type": "dataHolder" }], "Operator": "OR", "Type": "dataOperator" }
}

Приклад даних рівня для режиму SQL:

{
"csvPath":"l1.csv",
"description":"Select column a",
"difficulty":"0",
"solution":"SELECT a"
}

Приклад даних рівня для режиму Algortithms:

{
"csvPath":"l1.csv",
"description":"Move to the key",
"difficulty":"1"
}

Приклад таблиці даних для режиму SQL:

a,b,c

1,2,3

6,5,4

9,7,8

Приклад мапи рівня для режиму Algorithms:

#,#,#,#,#

#,@,,$,#

#,#,#,#,#

Додаток Г. Приклади коду

Код Draggable:

```
using Godot;
using System;

public partial class Draggable : Control
{
	[Export] string[] suffix;
	bool droppedOnTarget=false;
	public override void _Ready()
	{
		AddToGroup("DRAGGABLE");
		if (suffix!=null && suffix.Length!=0)
		{
			foreach (string s in suffix)
			{
				AddToGroup("DRAGGABLE_"+s);
			}
		}
	}

   public override Variant _GetDragData(Vector2 atPosition)
   {
      if (!droppedOnTarget)
		{
			SetDragPreview(GetPreviewControl());
			GD.Print(GetGroups());
		}
		return this;
   }

   public Control GetPreviewControl()
   {
		var Preview=new Control();
		Preview=(Control)this.Duplicate();
```

```
                Preview.Size=this.Size;
                Color newColor=this.Modulate;
                newColor.A/=2;
                Preview.Modulate=newColor;
                return Preview;
}

        public override bool _CanDropData(Vector2 atPosition, Variant data)
        {
                bool canDrop=false;
                if ((Node)data!=null)
                {
                        canDrop=((Node)data).IsInGroup("DRAGGABLE");
                }
                if (((Node)data).GetType()==Type.GetType("DragCreator")) return false;
                return canDrop;
        }

public override void _DropData(Vector2 atPosition, Variant data)
{
   Control node=(Control)data;
                if (this.IsAncestorOf(node)) return;
                if (node.IsAncestorOf(this)) return;
                Node thisParent=this.GetParent();
                Vector2 thisPosition=this.Position;
                Node nodeParent=node.GetParent();
                Vector2 nodePosition=((Control)node).Position;
                if (nodeParent==null && thisParent!=null) thisParent.AddChild(node);
                if (nodeParent!=null) node.Reparent(thisParent);

                if (thisParent==null && nodeParent!=null) nodeParent.AddChild(this);
                if (thisParent!=null) this.Reparent(nodeParent);

                this.Position=nodePosition;
                node.Position=thisPosition;
```

```
                {if (thisParent is DropZone zone) zone.draggable=(Draggable)node;}
                {if (nodeParent is DropZone zone) zone.draggable=(Draggable)this;}
    }
}
```

Код DragCreator:

```
using Godot;
using System;

public partial class DragCreator : Draggable
{

        bool droppedOnTarget=false;

        public override Variant _GetDragData(Vector2 atPosition)
    {
       if (!droppedOnTarget)
                {
                        SetDragPreview(GetPreviewControl());
                        GD.Print(GetGroups());
                }
                Node duplicate=this.Duplicate();
                return duplicate;
    }

    public override bool _CanDropData(Vector2 atPosition, Variant data)
    {
                return false;
    }
}
```

Код DropZone:

```
using Godot;
using System;
```

```
public partial class DropZone : Control
{
	public Draggable draggable;
	[Export] string[] suffix;

	public override bool _CanDropData(Vector2 atPosition, Variant data)
	{
		bool canDrop=false;
		if ((Node)data!=null)
		{
			if (((Node)data).IsInGroup("DRAGGABLE"))
			if (suffix!=null && suffix.Length!=0)
			{
				foreach (string s in suffix)
				{

canDrop=canDrop||((Node)data).IsInGroup("DRAGGABLE_"+s);
				}
			}
			else
			{
				canDrop=true;
			}
		}
		return canDrop;
	}

   public override void _DropData(Vector2 atPosition, Variant data)
   {
      Control node=(Control)data;
		if (this.IsAncestorOf(node)) return;
		if (node.IsAncestorOf(this)) return;

		if (node.GetParent()==null)
		AddChild(node);
```

```
                else {
                if (node.GetParent() is DropZone zone) zone.draggable=null;
                node.Reparent(this);
                }
                node.Position=Vector2.Zero;
                if (node.GetType()==Type.GetType("DragCreator"))
                {
                        string oldPath = node.GetPath();
        node.SetScript(GD.Load<Script>("res://Draggable.cs"));
        node = (Control)GetNode(oldPath);
                }
                if (draggable==null) draggable=(Draggable)node;
                else
                {
                        draggable=(Draggable)node;
                }

    }
}
```

Код DragZone:

```
using Godot;
using System;

public partial class DragZone : Control
{

        public override bool _CanDropData(Vector2 atPosition, Variant data)
        {
                bool canDrop=false;
                if ((Node)data!=null)
                {
                        canDrop=((Node)data).IsInGroup("DRAGGABLE");
                }
                return canDrop;
        }
```

```
            public override void _DropData(Vector2 atPosition, Variant data)
        {
           Control node=(Control)data;
                    if (node.GetParent()==null)
                    AddChild(node);
                    else {
                    if (node.GetParent() is DropZone zone) zone.draggable=null;
                    node.Reparent(this);
                    }
                    if (node.GetType()==Type.GetType("DragCreator"))
                    {
                            GD.Print("type");
                            string oldPath = node.GetPath();
            node.SetScript(GD.Load<Script>("res://Draggable.cs"));
            node = (Control)GetNode(oldPath);
                    }
                    (node).Position=atPosition;
        }
    }
```

При натискані на кнопку у режимі PolyDraw виконується даний код:

```
    public void FigDraw()
            {
                    map.Clear();
                    Vector2I step=new Vector2I(map.TileSet.TileSize.X,
map.TileSet.TileSize.Y);
                    Array<Vector2> points=figSummator.Summate(polygons, AreaSize, step);
                    prevAtt=points;
                    GD.Print(figSummator.ReturnJSON());
                    Array<Vector2> goalPoints=figSummator.Summate(polygons, AreaSize,
step, goalJSON);
                    if (goalPoints!=null && points!=null)
GD.Print(goalPoints.RecursiveEqual(points));
                    if (points!=null)
                    foreach (Vector2 point in points)
```

```
                {
                        map.SetCell(0, map.LocalToMap(point),1, new Vector2I(0,0) );
                }

                if (importLevel)
                {
                        if (goalPoints!=null && points!=null &&
goalPoints.RecursiveEqual(points)) Solved();
                        else if (goalPoints==null && points==null) Solved();
                }
                else
                {
                        AutoLoad autoLoad=(AutoLoad)GetNode("/root/AutoLoad");
                        Dictionary dict=
                autoLoad.PolygonsAndGoals(PolygonsToDictionary(),
figSummator.ReturnDictionary(), description, "0");
                        GD.Print(dict);
                }
        }
```

При натисканні на кнопку у режимі SQL виконується даний код:

```
public virtual void ButtonPressed()
        {
                var sql=Text;
                Array<int[]> parsed=ParseSQL(sql, sceneControl);
                sceneControl.ShowTable(parsed[0], parsed[1], sceneControl.tablePlacement,
sceneControl.defaultColor);
                Array<int[]> goalParsed=ParseSQL(sceneControl.goalSQL, sceneControl);
                GD.Print(parsed[0].SequenceEqual(goalParsed[0]) &&
parsed[1].SequenceEqual(goalParsed[1]));
                if (parsed[0].SequenceEqual(goalParsed[0]) &&
parsed[1].SequenceEqual(goalParsed[1]))
                {
                        sceneControl.Solved();
                }
        }
```

Код функції ParseSQL:

```
public static Array<int[]> ParseSQL(string sqlStatement, SceneControl sceneControl)
        {
                if (sqlStatement==null) return new Array<int[]>{ null,null };
                var sql=sqlStatement;

                var ast = new Parser().ParseSql(sql);

                var select = ast[0].AsQuery()!.Body.AsSelect();
                var having = select.Having?.AsBinaryOp();

                var top=select?.Top;

                string[] headers=sceneControl.headers;
                Array<string[]> rows=sceneControl.rows;

                Array<int> selectedIndices = new Array<int>();

                if (select.Projection is [SelectItem.Wildcard])
                {
                        Array<int> indices=new Array<int>();
                        for (int i=0; i<headers.Length; i++)
                        {
                                indices.Add(i);
                        }
                        selectedIndices.AddRange(indices);
                }
                else
                {
                        Array<int> indices=new Array<int>();
                        var columnNames=select.Projection.Select(col =>
col.AsUnnamed().Expression.AsIdentifier().Ident.Value).ToArray();
                        for (int i=0; i<headers.Length; i++)
                        {
                                if (columnNames.Contains(headers[i]))
```

```
                    indices.Add(i);
                }
                selectedIndices.AddRange(indices);
            }

            Array<int> selectedRows=new Array<int>();

            for (int i=0; i<rows.Count; i++)
            {
                if (top!=null)
                {
                    var topVar = (TopQuantity.Constant)select.Top.Quantity!;
                    if (selectedRows.Count>=(int)topVar.Quantity) break;
                }
                string[] row=rows[i];
                if (having!=null)
                {
                    bool binOp=false;;
                    try{
                        binOp=BinOp(having, row, selectedIndices, headers);}
                    catch(System.Exception e)
                    {
                        GD.PrintErr(e.Message);
                    }
                    if (!binOp) continue;
                }
                selectedRows.Add(i);
            }
            return new Array<int[]>{selectedIndices.ToArray(),
selectedRows.ToArray()};
        }
```

Код функції BinOp:

```
        static bool BinOp(SqlParser.Ast.Expression.BinaryOp op, string[] row, Array<int> indeces,
string[] header)
        {
```

```
                var left = op.Left;
                var right = op.Right;
            var opType=op.Op.ToString();
                switch(opType)
                {
                    case("And"):case("Or"):
                    {
                        bool leftBool = BinOp(left.AsBinaryOp(), row, indeces,
header);
                bool rightBool = BinOp(right.AsBinaryOp(), row, indeces, header);
                bool result=new bool();
                if (opType=="And")
                    result=leftBool&& rightBool;
                if (opType == "Or")
                  result = leftBool || rightBool;
                return result;
                    }
                    case ("Eq"):case ("NotEq"):case ("Lt"):case ("LtEq"):case ("Gt"):case
("GtEq"):
                    {
                        var leftIdent = left.AsIdentifier().Ident;
                        int leftId=-1;
                        for (int i=0; i<header.Length; i++)
                        {
                            if (header[i]==leftIdent) leftId=i;
                        }
                        string rightVal="";
                if (right.GetType() == typeof(SqlParser.Ast.Expression.Identifier))
                        {
                            string rightIdent = right.AsIdentifier().Ident;
                            for (int i=0; i<header.Length; i++)
                            {
                                if (header[i]==rightIdent) rightVal=row[i];
                            }
                        }
```

```
                if (right.GetType() == typeof(SqlParser.Ast.Expression.LiteralValue)) rightVal =
right.AsLiteral().Value.AsNumber().Value;
                                switch(opType)
                                {
                                        case("Eq"):return row[leftId]==rightVal;
                                        case("NotEq"):return row[leftId]!=rightVal;
                                        case("Lt"):return
float.Parse(row[leftId])<float.Parse(rightVal);
                                        case("LtEq"):return
float.Parse(row[leftId])<=float.Parse(rightVal);
                                        case("Gt"):return
float.Parse(row[leftId])>float.Parse(rightVal);
                                        case("GtEq"):return
float.Parse(row[leftId])>=float.Parse(rightVal);
                                }
                                break;
                        }
                }
                throw new System.Exception("Oops!");
        }
```

При натисканні на кнопку у режимі Algorithms виконується даний код:

```
public async void PressButton()
        {
                if (_isPlaying==true)
                {_abortPlaying=true;
                return;}
                ClearText();
                InitField();
                Array<DropZone> dropZones = sizableDropZones.dropZones;
                Array<AlgoActionBasic> actions=new Array<AlgoActionBasic>();

                foreach (var zone in dropZones) if (zone.draggable!=null)
                {
                        AlgoActionBasic
algoAction=zone.draggable.GetChild<AlgoActionBasic>(0);
```

```
                if (algoAction!=null) actions.Add(algoAction);
            }
            AlgoToken[] algoTokens=new AlgoToken[actions.Count];
            for (int i=0; i<actions.Count; i++)
            {
                algoTokens[i]=actions[i].ReturnAction();
            }

            _isPlaying=true;
            for (int i=0; i<actions.Count; i++)
            {
                var token=algoTokens[i];
                try
                {
                Task task=new Task(() => token.Action(this));
                task.Start();
                await task;
                GD.Print(token);
                }
                catch(Exception e)
                {
                    Print(e.Message);
                }
            }
            _isPlaying=false;
            _abortPlaying=false;
            if (keyPositions.Count==0)
            {
                AutoLoad autoLoad=(AutoLoad)GetNode("/root/AutoLoad");
                Print("Cleared");
                if (currentLevel!=null)
                {
                    if  (!autoLoad.algoCompletedLevels.Contains(currentLevel))
autoLoad.algoCompletedLevels.Add(currentLevel);
                    GD.Print(autoLoad.CheckAlgoCompleteness());
```

```
                }
                if (PopUp!=null)
                {
                    if (autoLoad.CheckAlgoCompleteness() && PopUp is
ConfirmationDialog confirmationDialog) confirmationDialog.OkButtonText="Play level";
                    PopUp.Show();
                }
            }
        }
```

Елементи AST у режимі Algorithms усі наслідують клас абстрактний AlgoToken:

```
public abstract class AlgoToken
    {
        public abstract void Action(AlgorithmReader algorithmReader);
    }
```