

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему «Розробка програмного додатку розвиваючої гри»

Виконав: студент 4 курсу, групи

КНТЗ-511сп

Спеціальності 123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

ГЮЛЬНАЗАРЯН Я.В.

(прізвище та ініціали)

Керівник ІЛ'ЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(прізвище та ініціали)

Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і назва)

Освітня програма (спеціалізація) Комп'ютерні системи та мережі

(Назва)

ЗАТВЕРДЖУЮ

Зав. кафедр Кудерметов Р.К.

“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

ГЮЛЬНАЗАРЯН Ярослав Віталійовича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка програмного додатку розвиваючої гри»

керівник проекту (роботи) ІЛІЯШЕНКО М.Б., к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проекту (роботи) 26.05.2024 року

3. Вихідні дані до проекту (роботи) ігровий рушій Unity, мова програмування C# 8.0, Visual Studio Professional, Photoshop, Amazon Web Services

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

1) Аналіз програмного середовища.

2) Проєктування програмного середовища.

3) Розробка програмного середовища.

4) Тестування програмного середовища.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

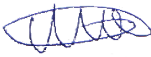
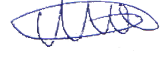
1) Контекстна діаграма іграшкового програмного продукту;

2) Схема роботи серверу для автооновлень;

3) Діаграма прецедентів ігрового програмного продукту;

4) Результат тестування ігрового програмного продукту.

6. Консультанти розділів проекту (роботи)

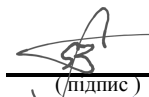
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ІЛЬЯШЕНКО М.Б. , к.т.н., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст.викладач		

7. Дата видачі завдання 15.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

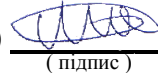
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд літератури	20.04.2024 р.	
2	Аналіз технічного завдання	25.04.2024 р.	
3	Огляд існуючих ігрових програмних продуктів	30.04.2024р.	
4	Проектування об'єктів розвиваючої гри	10.05.2024 р.	
5	Розробка програмного середовища та вимог програмного забезпечення	15.05.2024 р.	
6	Тестування ігрового програмного продукту	20.05.2024 р.	
7	Оформлення ПЗ	26.05.2024 р.	

Студент


(підпис)

Ярослав ГЮЛЬНАЗАРЯН
(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

Матвій ІЛЬЯШЕНКО
(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 58 с., 40 рис., 1 табл., 20 джерел.

ГРА, ПРОГРАМА, UNITY, РОЗВИТОК, РЕАЛІЗАЦІЯ, ПРОЕКТ, СЦЕНАРІЙ

Метою даного проекту є створення навчально-розвиваючого середовища, яке сприятиме розвитку навичок дітей дошкільного віку. Для досягнення цієї мети потрібно вирішити наступні завдання:

- провести аналіз ігрових програмних продуктів;
- провести формалізацію об'єктів предметної області для навчально-розвиваючої платформи;
- обґрунтувати вибір алгоритму для побудови лабіринтів у грі;
- розробити архітектуру та алгоритм роботи платформи;
- провести тестування ігрового програмного продукту.

У першому розділі проведено аналіз і класифікацію навчально-розвиваючих середовищ, вивчено технології, що використовуються у їх розробці, та проаналізовано особливості розвиваючих ігор.

У другому розділі запропоновано процес систематизації об'єктів у навчально-розвиваючому середовищі на основі фізичних характеристик, що дозволило наблизити віртуальні процеси до реальних. Обґрунтовано алгоритм генерації лабіринту для гри.

У третьому розділі було розроблено програмне середовище навчально-розвиваючої гри, побудовано алгоритми роботи додатку та проведено його верифікацію через тестування.

Четвертий розділ присвячений тестуванню розробленого програмного забезпечення для перевірки його коректності в різних ситуаціях.

ЗМІСТ

Вступ.....	8
1 Аналіз програмного середовища.....	10
1.1 Класифікація навчально-розвиваючих середовищ.....	10
1.2 Технології розробки	14
1.3 Аналіз особливостей розвиваючих ігор для дітей дошкільного віку	17
1.4 Висновки за розділом	20
2 Проєктування програмного середовища.....	21
2.1 Проєктування об'єктів розвиваючої гри.....	21
2.2 Систематизація процесів	25
2.3 Алгоритм генерування лабіринтів.....	29
2.4 Висновки за розділом.....	33
3 Розробка ігрового програмного середовища.....	34
3.1 Визначення вимог.....	34
3.2 Розробка архітектури	38
3.3 Алгоритми програмного середовища.....	41
3.4 Висновок за розділом.....	47
4 Тестування ігрового програмного продукту	48
4.1 Тестування інтерфейсу користувача	48
4.2 Тестування лаунчера	52
4.3 Тестування роботи сервера	53
4.4 Висновки за розділом.....	55
Висновки	56
Перелік джерел посилання	57
Додаток А Діаграма декомпозиції робочих процесів ІПП.....	59
Графічний матеріал:	
Плакат 1 - Контекстна діаграма іграшкового програмного продукту.	
Плакат 2 – Схема роботи серверу для автооновлень.	

Плакат 3 – Діаграма прецедентів ігрового програмного продукту.

Плакат 4 – Результат тестування ігрового програмного продукту.

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ІПП	ігровий програмний продукт
ПЗ	програмне забезпечення
VS	visual studio
AWS	amazon web services
ПК	персональний комп'ютер

ВСТУП

У сучасному високотехнологічному світі, де швидко розвивається інформаційне суспільство та науково-технічний прогрес, виникає велике завдання - створення високоякісного програмного забезпечення, особливо для дітей, яке сприятиме їхньому майбутньому розвитку. У цій роботі розглядається вирішення цих проблем та важливість розвитку ігрової індустрії для формування молодого покоління.

Ігрова індустрія є однією з найскладніших галузей інформаційних технологій, яка потребує міцної бази у точних дисциплінах та стрімкого розвитку. Компанії, що займаються розробкою ігор, незалежні розробники та студії, такі як Valve, Bethesda, EA, Ubisoft, RockStart Games, Markus Alexej Persson, активно звертають увагу на оновлення технологій та компонентів розробки.

Для створення ігрового програмного забезпечення використовуються професійні програмні додатки, що дозволяють прискорити процес розробки, а також для створення тривимірних об'єктів застосовуються різноманітні програмні продукти, такі як Blender, 3D Max та інші.

Розробка якісного ігрового програмного забезпечення вимагає значних часових затрат, оскільки ігровий продукт вимагає значних ресурсів, особливо велике навантаження отримує графічний процесор. Однак мінімальні вимоги до апаратної частини комп'ютера можуть бути в два рази вищими, ніж у випадку настільного програмного забезпечення.

Актуальність розробки навчально-розвиваючих платформ пояснюється фізіологічними особливостями дітей, які краще сприймають графічну інформацію з яскравими кольорами, що сприяє розвитку їх асоціативного мислення.

Завдяки ігровому програмному забезпеченню вдається розвивати навички дітей, їхні розумові здібності та навчати їх новим речам, таким як

іноземні мови та дисципліни. Тому діти навіть у ранньому віці можуть володіти знаннями, які перевершують знання дорослих, завдяки ігровому програмному забезпеченню.

Мета даної розробки є розробка програмного навчального-розвиваючого середовища, яке допоможе дітям дошкільного віку розвивати свої навички.

Завдання, які вирішуються у зв'язку із зазначеною метою є:

- аналіз публікацій і практик розробки ігрового контенту;
- проектування об'єктів предметної області при створенні навчально-розвиваючої платформи;
- обґрунтування вибору алгоритму побудови лабіринтів при реалізації гри;
- проектування архітектури та розробка алгоритму роботи платформи;
- тестування розробленого програмного забезпечення.

Практична значущість та новизна роботи пояснюється у методі та програмних засобах розвитку навичок та отримання нових знань з використанням інформаційного простору.

В ході роботи над проектом було одержано наступні результати:

- запропоновано формалізацію об'єктів та процесів у навчально-розвиваючому середовищі на основі фізичних характеристик об'єктів, що дало змогу забезпечити максимальну наближеність віртуальних процесів до процесів, які протікають в реальній предметній області при розробці гри;
- набули подальшого розвитку методи і засоби проектування навчально-розвиваючих платформ шляхом збільшення кількості та якості ігрового контенту платформи, що дозволить підвищити ефективність навчання дітей дошкільного віку.

Даний підхід до навчання дає змогу дитині краще засвоювати отримані знання та навички для подальшого їх використовування у початковій школі.

1 АНАЛІЗ ПРОГРАМНОГО СЕРЕДОВИЩА

1.1 Класифікація навчально-розвиваючих середовищ

В сучасному світі комп'ютерних ігор, майже кожен продукт розвиває певні навички у користувача або допомагає засвоїти певні знання. Проте, не всі програми акцентують увагу на цьому.

Розглянемо деякі програмні продукти які допомагають в розвитку дітей, а також дорослих.

Ігровий програмний продукт «Super Mario», рисунок 1.1.



Рисунок 1.1 - Ігровий програмний продукт «Super Mario»

Один із таких культових проєктів 20 століття - гра під назвою «Super Mario», і вона відзначається своєю здатністю розвивати реакцію людини. Її створення було пов'язано з численними випробуваннями, труднощами та переживаннями. Гравцеві потрібно витратити час на те, щоб вчасно реагувати на події, що відбуваються на екрані, і це розвиває його реакційні навички [1].

Ігровий програмний продукт «Witness», рисунок 1.2.



Рисунок 1.2 - Ігровий програмний продукт «Witness»

Witness - це незалежна гра-головоломка, що вийшла у світ 2016 року і відзначена в галузі розвиваючого програмного забезпечення. Цей ігровий продукт ілюструє, як комп'ютерні ігри сприяють розвитку різноманітних навичок у людини. Гра сприяє розвитку логічного мислення при вирішенні різноманітних головоломок та розвиває уважність, оскільки деякі завдання вимагають нестандартного підходу і вирішення проблеми від користувач. Гра була створена на такі платформи як Windows, PlayStation 4 та Xbox One [2].

Ігровий програмний продукт «Europa Universalis 4» (рис.1.3).

Цікавим продуктом є «Europa Universalis 4», яка допомагає підвищити рівень знань зі світової історії. Гра дозволяє користувачам провести час з користю, покращуючи свої знання. Вона також допомагає вивчити іноземні мови, оскільки підтримує англійську, німецьку, французьку та іспанську мови [3].



Рисунок 1.3 - Ігровий програмний продукт «Europa Universalis 4»

Ігровий програмний продукт «Spore», рисунок 1.4.

«Spore» - це ігровий продукт, який дозволяє дитині самостійно керувати еволюцією живого організму, починаючи з одноклітинного організму і закінчуючи цивілізацією, що підкорює космос [4]. У грі велика кількість елементів гри в режимі реального часу, що дозволяє внести великий діапазон змін і вести нескінченну гру.

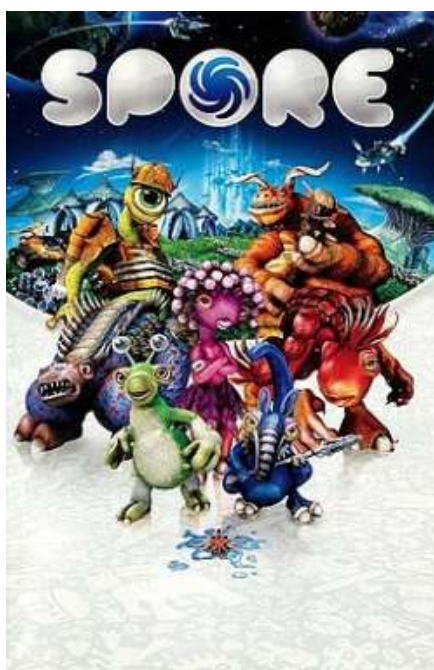


Рисунок 1.4 - Ігровий програмний продукт «Spore»

Серія ігрових продуктів «Just Dance» (рис. 1.5).

«Just Dance» - це ігровий продукт, що спрямований на відтворення рухів, які показує гра на екрані. Цей вид ігор корисно впливає на моторно-рухову систему організму, змушуючи гравців активно рухатися [5].



Рисунок 1.5 - Ігровий програмний продукт «Just Dance»

Таким чином, представлені ігрові продукти допомагають розвивати різні навички та знання, і їх можна класифікувати за характеристиками, що наведені у таблиці 1.1.

Таблиця 1.1 - Класифікація ігрових продуктів за розвиваючими навиками

Назва	Реакція	Логіка	Моторика	Уважність	Предмети
Super Mario	+	+	-	+	-
Witness	-	+	-	+	-
Europa Universalis 4	-	+	-	+	+
Spore	-	-		+	+
Just Dance	+	-	+	+	-

1.2 Технології розробки

В умовах активної інформатизації та постійного розвитку технологій, розробникам доводиться ставити перед собою непрості завдання щодо створення високоякісного програмного забезпечення, спрямованого на дітей. Основна мета такого програмного забезпечення полягає в тому, щоб надати молодому поколінню навички та вміння, які будуть корисні у майбутньому та допоможуть їм бути конкурентоспроможними на ринку праці.

Галузь ігрової індустрії виявляється однією з найбільш складних у сфері інформаційних технологій, оскільки для успішного розвитку вимагає не лише високий технічний рівень, а й глибоких знань у галузях, таких як психологія, ергономіка та фізіологія, оскільки ці аспекти важливі для розробки ігор для різних вікових груп. Один з ефективних методів розробки гри - використання відповідних ігрових движків. Створення ігрового продукту для дошкільнят в даний час набуває особливої актуальності і потребує глибокого аналізу функціональних можливостей та наборів готових шаблонів, що наявні у сучасних ігрових движках.

Важливим аспектом при виборі ігрового движка є його підтримка та розвиток. Тому компанії в галузі IT, такі як Valve, Bethesda, EA, Ubisoft, RockStart games, Markus Alexej Persson, постійно вдосконалюють існуючі компоненти, оновлюють ядро движка та розширюють набір компонентів для подальшого використання. Останнім часом компанії також почали видавати безкоштовні версії ігрових движків, що позитивно вплинуло на галузь розробки ігор, зробивши доступ до навчання різних дисциплін більш простим та доступним.

Незважаючи на те, що більшість ігрових движків стали доступними безкоштовно, вимоги до системи для їх роботи залишаються високими. Серед найпопулярніших ігрових движків на сьогоднішній день варто відзначити Unity, Unreal Engine та Godot Engine [6]. Кожен з них має свої особливості та

переваги, а вибір між ними визначатиметься завданнями розробки та потребами в конкретних функціях та можливостях.

Механізм Unreal Engine, є найбільш досвідченим, насиченим та результативним інструментом для створення відеоігор. Управління грою, при використанні цієї платформи, може бути реалізоване за допомогою мови C++, або за допомогою візуального програмування Blueprint, що спрощує вхід для розробників ігор. Крім цього, є можливість розробки різноманітних процесів, інтеграції та автоматизації потоків роботи за допомогою стандартизованих сценаріїв мови програмування Python.

Godot Engine представляє собою наймолодший гральний інструмент, основною його особливістю є те, що він має відкритий вихідний код і підтримує мови програмування C# 8.0, C++, Python, Rust, Nim, D та інші.

Вибір технології та інструментів для розробки інформаційних систем є складним завданням, оскільки цей вибір визначатиме складність розробки та якість готового програмного продукту.

Для реалізації було обрано такі засоби:

- ігровий рушій Unity;
- мова програмування C# 8.0;
- Visual Studio Professional 2019;
- систему контролю версій Git;
- Photoshop CC 2017;
- Blender;
- Windows Communication Foundation;
- Windows Presentation Foundation;
- Amazon Web Services.

Visual Studio є одним із передових середовищ розробки, яке об'єднує різноманітні технології програмування, такі як C#, F#, Python та інші. Воно надає розробникам широкий спектр інструментів, які спрощують процес розробки, включаючи засоби відлагодження, спостерігачі, візуальний пакетний менеджер і багато іншого. Крім того, Visual Studio дозволяє створювати

різноманітні види програм, включаючи десктопні додатки, веб-сервіси, мобільні застосунки та ігрові продукти.

Adobe Photoshop є потужним графічним редактором, який дозволяє створювати будь-яку 2D графіку будь-якого рівня складності.

Blender - це програма для створення 3D графіки, яка також надає можливість анімації створених об'єктів та експортує матеріали у відповідні ігрові движки. Всі популярні ігрові рушії підтримують формат файлів цього програмного продукту.

WPF (Windows Presentation Foundation) - це технологія для створення програм з інтерфейсом користувача для ОС Windows. Вона ідеально підходить для розробки лаунчера для ігрових продуктів.

Amazon Web Services (AWS) - це набір технологій, розташованих у хмарі, що спрощує розробку сервісів, які будуть доступні цілодобово.

Для ефективної розробки програмного забезпечення, а також для вчасного виконання кваліфікаційних завдань часто використовується методологія розробки програмного забезпечення «Канбан» [9]. Приклад організації даного підходу наведено на рис. 1.6.

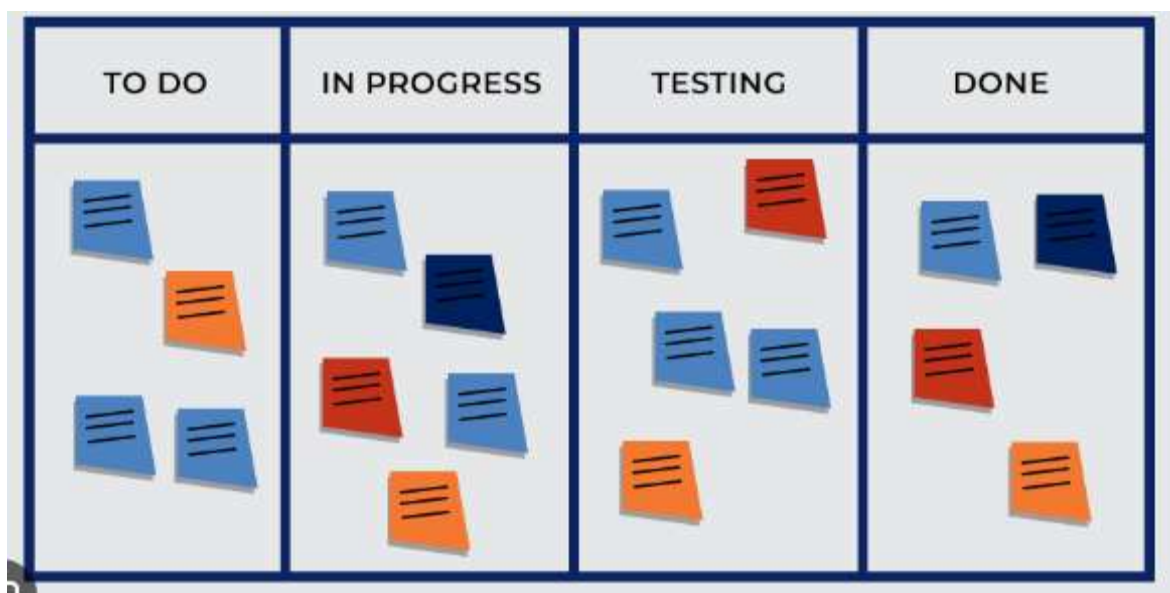


Рисунок 1.6 - Методологія розробки програмного забезпечення «Канбан»

Канбан- це метод поліпшення робочого процесу у роботі із командою. Завдяки цьому методу кожен учасник команди може бачити всі процеси роботи та відслідковувати виконання завдань. Зазвичай для цього методу використовують спеціальну дошку і набір карток або стикерів.

Для того щоб користуватися методом необхідно створити план роботи над проєктом, проаналізувати послідовність дій. Далі розділити дошку на стопці, в яких зазвичай записують послідовні етапи процесу. Карти або стікери поступово переходять із стовпчика до стовпчика в залежності від послідовності виконання необхідних задач:

- to do – «на черзі»;
- in progress – «в процесі»;
- testing – «тестування»;
- done – «зроблено».

Виходячи з поставлених задач, для їх виконання вибрані інструменти пропонують всі необхідні технології для створення якісного та продуктивного ігрового програмного забезпечення, зменшивши витрати часу на розробку.

1.3 Аналіз особливостей розвиваючих ігор для дітей дошкільного віку

Розвиваючі ігри представляють собою форму розваг, яка сприяє розвитку та удосконаленню різних навичок. Навчальні ігри призначені для того, щоб допомогти людям освоїти певні теми, розвинути концепції, зрозуміти історичні та культурні події, а також розвивати нові навички під час гри[10].

Ці ігри можна класифікувати за спрямованістю на настільні, карткові та відеоігри. Батьки та вихователі враховують психологічні потреби у навчанні та позитивний вплив гри на освітні процеси. Концепція ігрового моделювання стала одним з ключових засобів освіти. Симуляція створює інтерактивне

оточення, що спрямовує учасників до встановлення цілей, формування правил, адаптації до різних ситуацій та розв'язання проблем.

Відзначається, що зі зростанням доступності технічного обладнання змінюються й типи ігор, що сподобалися користувачам. Відеоігри все частіше використовуються порівняно з традиційними іграми. Подано поняття концептуальних ігор як ігор, які передбачають проектування ролей персонажів, та відзначено рольові ігри, які включають уявну проблему. Описано важливість глибокого занурення у гру для набуття когнітивних та соціальних навичок.

Ігрове навчання - це спосіб навчання, що має певні освітні цілі. Цей підхід ставить за мету поєднання теоретичного розуміння теми з ігровим процесом, щоб розвинути вміння застосовувати знання в реальному житті.

Науково доведено, що люди найкраще запам'ятовують те, що вони роблять. Тому досвід, отриманий у процесі практики, є важливим для працевлаштування, оскільки роботодавці цінують практичні навички. Ілюстрація градації ефективності запам'ятовування предметів представлена на рисунку 1.7.

Ігрові продукти стимулюють користувача використовувати всі свої органи чуття: зір, слух і моторику. Коли людина занурюється у гру, вона спостерігає, чує та виконує певні дії, використовуючи отриману інформацію, що відповідає першому та третьому пунктам інтерактивного електронного навчання».

У навчальних закладах недостатньо уваги приділяється розвитку розумових і фізичних навичок у дітей. Хоча в дитини може бути книга з формулами та законами для розв'язання завдань, у реальному житті не завжди є можливість просто знайти відповідь, як у підручнику.

Використовуючи ігрові додатки, користувачеві потрібно розв'язувати завдання, використовуючи доступні ресурси та свої власні зусилля, без допомоги вчителя.

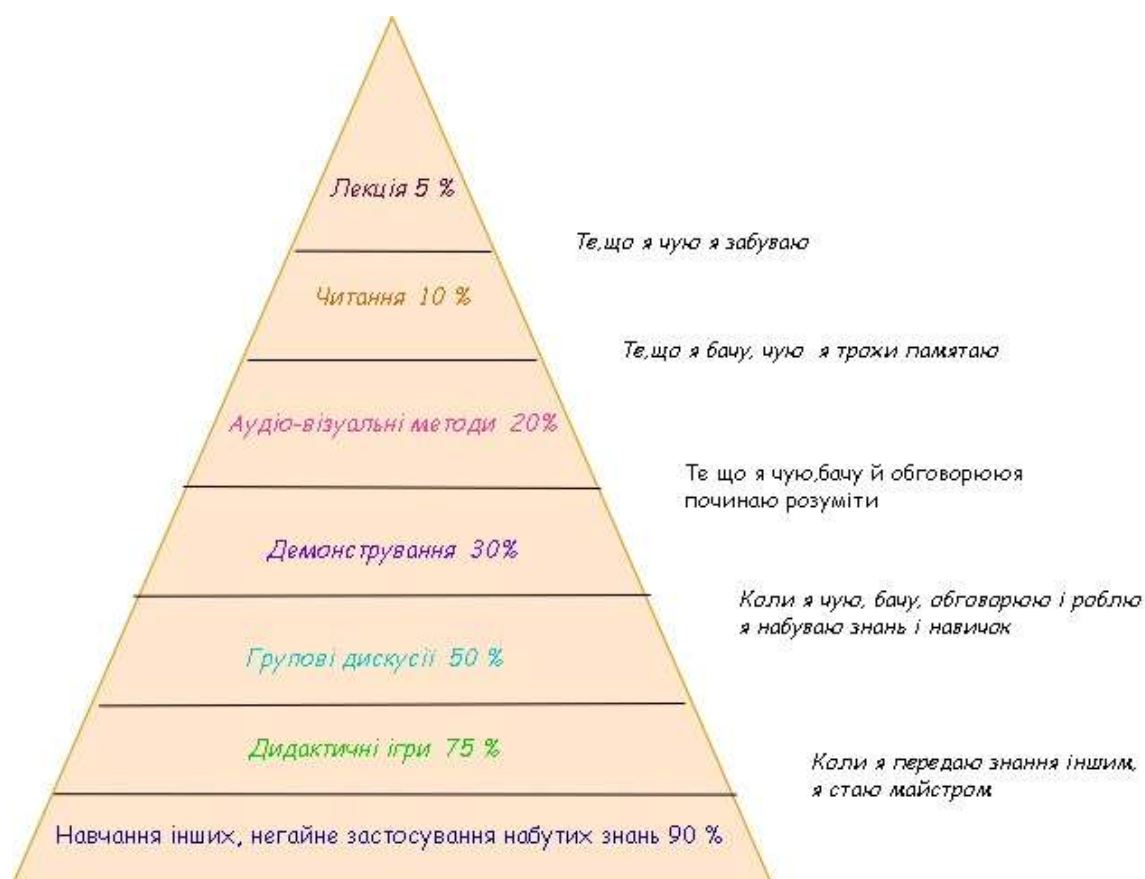


Рисунок 1.7 – Інтерактивні методи електронного навчання

Отже, цей тип навчання має велике значення в сучасному світі, оскільки допомагає отримувати навички та знання швидше та ефективніше, що є важливим у реальному житті.

На основі проведеного аналізу отримано наступні результати:

- проведено аналіз існуючих ігрових програмних продуктів, у результаті якого встановлено типи навичок і знання, які здобувають діти при використанні навчально-розвиваючих середовищ;
- на основі дослідження технологій розробки програмного додатку навчальних середовищ визначено оптимальні технології програмного забезпечення для їх реалізації;
- проведено аналіз особливостей ІПП, у результаті якого визначено функціональні вимоги до них та необхідність врахування особливостей розвитку дітей дошкільного віку.

1.4 Висновки за розділом

Сучасні комп'ютерні ігри не лише забезпечують розвагу, але й можуть слугувати інструментом для розвитку різноманітних навичок та здобуття знань у користувача. У розділі наведено низку ігрових продуктів, таких як Super Mario, Witness, Europa Universalis 4, Spore та Just Dance, які сприяють розвитку реакції, логічного мислення, уваги, моторики та допомагають вивчати історію і мови.

Для розробки якісних ігрових продуктів використовуються потужні інструменти та технології, зокрема ігрові рушії Unity, Unreal Engine та Godot Engine, мови програмування C#, C++, Python, середовище розробки Visual Studio, графічні редактори Photoshop і Blender, хмарні технології Amazon Web Services тощо.

Для ефективної організації процесу розробки програмного забезпечення застосовується методологія "Канбан", яка передбачає візуалізацію робочого процесу та відстеження виконання завдань командою.

Розвиваючі ігри відіграють важливу роль у навчанні дітей, оскільки поєднують теоретичні знання з практичним досвідом і сприяють кращому засвоєнню інформації через активну участь та занурення в ігровий процес. Під час розробки ігрових продуктів для дітей дошкільного віку необхідно враховувати особливості їхнього розвитку та психологічні потреби, а також забезпечити відповідність функціональним вимогам до навчально-розвиваючих середовищ.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО СЕРЕДОВИЩА

2.1 Проєктування об'єктів розвиваючої гри

Під проєктуванням розуміють процес перетворення певної області у формальну систему, в якій структура та форма стають об'єктом аналізу незважаючи від змісту.

Завдяки проєктуванню можна виявити загальні думки, сформулювати принципи та правила мислення. Саме цей процес дозволяє перетворити будь-яке конкретне мислення або текстовий документ чи текст у відповідну систему формул.

Через те, що інформаційно-програмний продукт (ІПП) містить багато компонентів, які потрібно спілкуватися один з одним, необхідно описати всі ці компоненти та їх характеристики для кращого розуміння системи під час розробки додатку.

На першому рівні реакції знаходяться два основних об'єкта:

- «Пташка» – головний об'єкт на сцені, користувач повинен керувати ним;
- «Краплі дощу» – один з головних об'єктів, який має свій життєвий цикл, даний об'єкт виступає перешкодою для пташки.

Властивості об'єкта «Пташка»:

- компонент відповідальний за фізику об'єкта;
- компонент відповідальний за переміщення об'єкта;
- швидкість переміщення об'єкта;
- життя об'єкта.

Властивості об'єкта «Краплі дощу»:

- компонент відповідальний за фізику об'єкта;
- компонент поведінки.

На другому рівні реакції знаходяться наступні об'єкти:

- квадрат – головний об'єкт, який знаходиться на сцені, задача

користувача здійснювати керування даним об'єктом;

- сніговики – статичні об'єкти, які перешкоджають об'єкту Квадрат;
- платформи – статичні об'єкти, які використовуються для переміщення головного персонажа.

Властивості квадрата:

- компонент відповідальний за фізику об'єкта;
- компонент керування;
- компонент анімації.

На першому рівні пам'яті присутні наступні об'єкти:

- шарик - головний об'єкт сцени, яким користувач може керувати;
- ключ – об'єкт для позначення закінчення рівня;
- стіни лабіринту – статичні об'єкти, які виконують функцію обмеження користувача в переміщенні.

Властивості шарика:

- компонент відповідальний за фізику об'єкта;
- компонент керування.

Другий рівень пам'яті складається з статичних об'єктів. Головними об'єктами сцени є «Зірки» які містять стан – колір. На сцені присутні шість «Зірок», а кожна має певний колір, є три комбінації кольорів, тобто дві «Зірки» мають один колір.

На першому рівні логіки знаходиться відома головоломка «П'ятнашки», створена Ноєм Чепменом у 1878 році [12]. Цей рівень містить в собі вісім об'єктів, кожен з яких має свій номер, а також одне порожнє місце для переміщення об'єктів. Основна мета гри полягає в тому, щоб перемістити об'єкти по полю так, щоб цифри від одного до восьми були розташовані в зростаючому порядку, після чого гра вважається завершеною. Приклад партії в грі «П'ятнашки» наведено на рис. 2.1.

5	10	14	7
8	3	6	1
15		12	9
2	11	4	13

Рисунок. 2.1 - Гра «П'ятнашки»

Другий рівень логіки включає реалізацію відомої гри «Хрестики-нулики» [13]. Для гри потрібно два гравці, яких можна позначити як «Х» та «О» відповідно. Гравці по черзі роблять свої ходи на полі, яке складається з дев'яти комірок. Кожен гравець використовує своє позначення, яке було визначено до початку гри. Мета гри полягає в тому, щоб утворити рядок, стовпчик або діагональ з своїх позначень. Приклад гри наведено на рисунку 2.2.

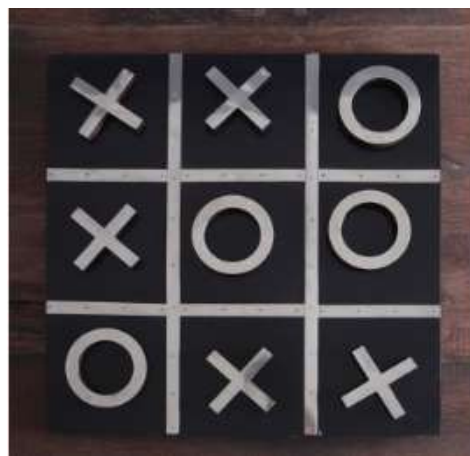


Рисунок 2.2 - Гра «Хрестики-нулика»

Партії можуть бути як переможні так і коли жоден з гравців не виграв гру, тоді дана партія вважається нічийною. Етапи такої партії показано на рисунку 2.3.

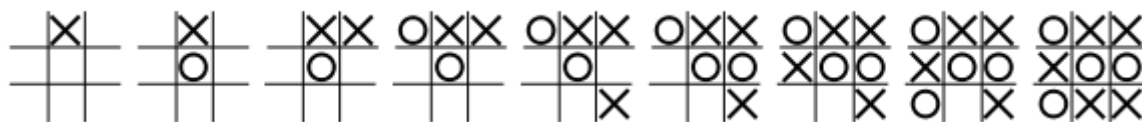


Рисунок 2.3 - Нічийна партія в грі «Хрестики-нулики»

Хрестики-нулики вважається одним з ідеальних варіантів для отримання комбінаторних навиків, мислення на декілька ходів – важливий атрибут цієї гри, оскільки нічийний результат гри можна передбачити після перших ходів в партії.

Заключним рівнем в даній розвиваючій платформі є «Підготовка до школи». Головними об'єктами на даній сцені є:

- два числа;
- арифметична операція (додавання, віднімання, множення, ділення);
- кількість правильних відповідей.

Суть даного рівня полягає в отриманні початкових знань при роботі з арифметичними операціями: додавання, віднімання, множення, ділення. Даний рівень поділяється на п'ять підрівнів:

- 1-й етап - додавання – на даному рівні користувач оперує тільки додаванням двох чисел;
- 2-й етап - віднімання – в даному підрівні користувач використовує тільки операцію віднімання;
- 3-й етап - множення – на даному етапі для виконання завдання, користувач оперує оператором множення;
- 4-й етап - ділення – в даному підрівні, для здійснення операції, користувач використовує тільки оператор ділення;
- 5-й етап - змішаний – заключний підрівень, який комбінує в собі всі чотири оператори для роботи з числами.

2.2 Систематизація процесів

Для розуміння того, як об'єкти взаємодіють та які дії вони виконують, потрібно описати процеси, що відбуваються в Інтегрованій Платформі Програмування (ІПП).

В першому рівні реакції є два найважливіших процеси:

- переміщення пташки;
- процес падіння крапель дощу.

У першому процесі, компонент «Rigidbody2D» в пташки відповідає за її фізику. Для того, щоб змусити пташку рухатись горизонтально, потрібно натискати клавіші «стрілка вправо» або «стрілка вліво», після чого ігровий рушій Unity обробляє ці дії [14].

В компоненті «Rigidbody2D» присутній параметр «Velocity», який використовується для руху пташки. При натисканні клавіші «пробіл», щоб зробити стрибок до куба, додається сила. Це дозволяє кубу перестрибнути перешкоду або застрибнути на платформу, а також для анімації. Даний параметр вираховується за формулою:

$$V=H*S, \quad (2.1)$$

де H – стан клавіші (натиснуто, відпущено);

S – базова швидкість.

Падіння крапель – це процес вільного падіння. Вільне падіння – це переміщення фізичного тіла, при умовах, коли єдина сила, що діє на предмет це гравітаційна сила:

$$z = h + v \cdot t - \frac{g \cdot t^2}{2}, \quad (2.2)$$

де h – початкова висота;
 v – швидкість тіла в початковий момент; t – час;
 g – прискорення вільного падіння.

Для переміщення головного персонажа за вертикаллю, відбувається зміна позиції об'єкта в просторі.

В другому рівні реакції головний процес це переміщення куба до кінця рівня. Рух вперед відбувається без задіяння користувача, на куб діє сила:

$$f = v \cdot g, \quad (2.3)$$

де v – вектор напрямку;
 g – базова сила.

Аналогічно, коли необхідно зробити стрибок до куба додається сили, в момент натискання на клавішу пробіл, для того щоб куб зміг перестрибнути перешкоду на його шляху або ж застрибнути на платформу, також для анімації.

В першому рівні пам'яті основний процес, який протікає на даному етапі – це переміщення шарика в трьохвимірному просторі. Для його переміщення необхідно додати момент сили, який вираховується за формулою:

$$T = v \cdot H \cdot c, \quad (2.4)$$

де v – вектор напрямку;
 H – стан клавіші (натиснуто, відпущено);
 c – базовий момент.

Крім того, ігровий рушій Unity підтримує виявлення зіткнень, якщо використовується внутрішній фізичний рушій. Виявлення зіткнень - це математичний алгоритм, який описує, яким чином перевіряється зіткнення двох фізичних тіл. Реакція на результат цього алгоритму - це фізична симуляція, що описує поведінку об'єктів після їх зіткнення.

Ці два поняття є головними в фізичних рушіях, оскільки на них

базується будь-яка поведінка об'єктів між собою.

Розрізняють два типи перевірки на зіткнення:

- виявлення зіткнення, після того як воно відбулось;
- виявлення зіткнення, до того як воно відбулось.

У першому сценарії, фізичний рушій отримує список усіх фізичних тіл на сцені та перевіряє, чи відбувається зіткнення або чи знаходяться об'єкти надто близько один до одного. Це досягається шляхом аналізу траєкторій руху, щоб визначити, чи перетинаються об'єкти.

У другому випадку, перевірка зіткнень базується на передбаченнях. Вона отримує список всіх фізичних тіл, перевіряє їх траєкторії, проводить симуляції та виявляє момент зіткнення перед тим, як воно відбудеться насправді.

Переваги першого методу полягають у тому, що він не вимагає додаткової інформації про об'єкти, такої як тертя, пружність тощо, і працює зі зміною часу, що робить його реалізацію простішою. Натомість, другий метод відрізняється більшою точністю та стабільністю, але вимагає складнішої реалізації. Як простий приклад, для відображення зіткнення двох об'єктів на основі квадрата та круга наведено на рис. 2.4.

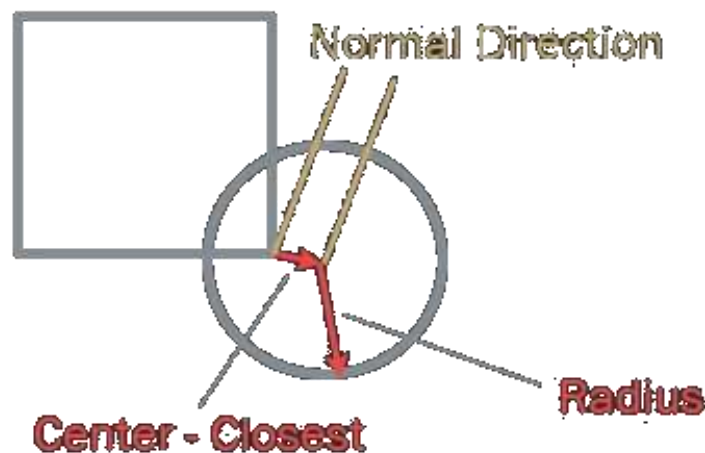


Рисунок 2.4 - Момент зіткнення двох тіл

Для розрахунку даного зіткнення можна використати формулу:

$$P=R-(C \cdot V), \quad (2.5)$$

де R – радіус кола;
 C – центр кола;
 V – найближча точка квадрата.

Якщо два об'єкти зіткнулись, відповідно на них діяла якась сила, після їх зіткнення вони повинні відштовхнутись одне від одного, як це продемонстровано на рис. 2.5.

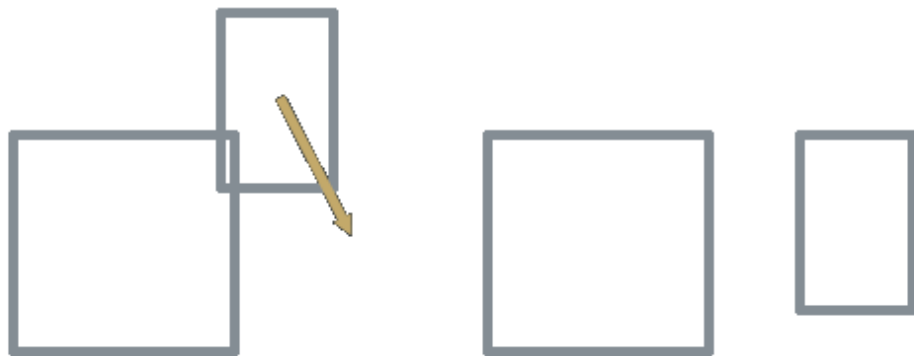


Рисунок 2.5 - Фізичний процес взаємодії двох об'єктів

Даний процес можна описати за допомогою формули:

$$I = \frac{-1 \cdot (1 + e) \cdot v}{1/m_1 / m_2} \cdot n, \quad (2.6)$$

де e – коефіцієнт повернення;
 v – відносна швидкість;
 m_1 – маса першого тіла;
 m_2 – маса другого тіла;
 n – нормаль зіткнення.

2.3 Алгоритм генерування лабіринтів

Лабіринт - це система, що включає в себе переплутані шляхи, дороги, кімнати або інші елементи, які зазвичай мають один вхід та один вихід[15].

Основні алгоритми генерації лабіринтів:

- алгоритм Еллера;
- алгоритм Крускала;
- алгоритм Прима;
- алгоритм Вільсона;
- пошук у глибину.

В даній розробці було використано алгоритм останній тип із вищенаведених алгоритмів.

Пошук у глибину - це метод створення лабіринту, що математично описаний із лише одним входом та виходом.

Зберігаючи інформацію про порядок переходу у самому лабіринті, цей алгоритм може бути представлений у вигляді петель. Крім того, він дозволяє швидко знайти рішення: ви можете починати з будь-якої точки та досягти виходу.

Лабіринт, що створений за допомогою алгоритму пошуку у глибину, має малу кількість розгалужень і може включати довгі коридори, оскільки алгоритм перевіряє кожен гілку на максимальну глибину перед тим, як повернутися назад.

Є два способи для генерування даним алгоритмом:

- рекурсивний;
- ітеративний.

Рекурсивний метод створення є досить небезпечний, оскільки може викликати переповнення стеку, що призведе до припинення виконання [16]. Для того, щоб уникнути даної проблеми можна використовувати ітеративний спосіб, використовуючи цикл.

Для створення лабіринту з використанням методу рекурсії необхідно виконати наступні дії:

- отримати комірку, як параметр методу;
- позначити отриману комірку, як відвідану;
- відвідати всіх сусідів обраної комірки, якщо вони ще не були відвідані;
- вибрати не відвіданого сусіда;
- зняти стінку між комірками;
- викликати рекурсивний метод для обраної комірки.

Необхідні кроки для створення лабіринту використовуючи ітеративний метод:

- вибрати стартову комірку;
- позначити вибрану комірку, як відвідану;
- додати комірку до списку;
- взяти нову комірку з стеку;
- зробити вибрану комірку поточною;
- якщо в даній комірці є не відвідані сусіди, додати її назад в стек;
- вибрати першого не відвіданого сусіда;
- забрати перешкоду між комірками;
- позначити вибрану комірку, як відвідану;
- додати дану комірку до списку.

В результаті роботи ми повинні отримати лабіринт з одним входом та одним виходом, приклад наведено на рисунку 2.6.

Для того, щоб розпочати генерацію, необхідно ініціалізувати всі поля, які в подальшому будуть використовуватись алгоритмом (рис. 2.7).

Після того, як було створено всі необхідні компоненти для роботи, можна приступати до алгоритму вибору комірки, виконання операції наведено на рис. 2.8.

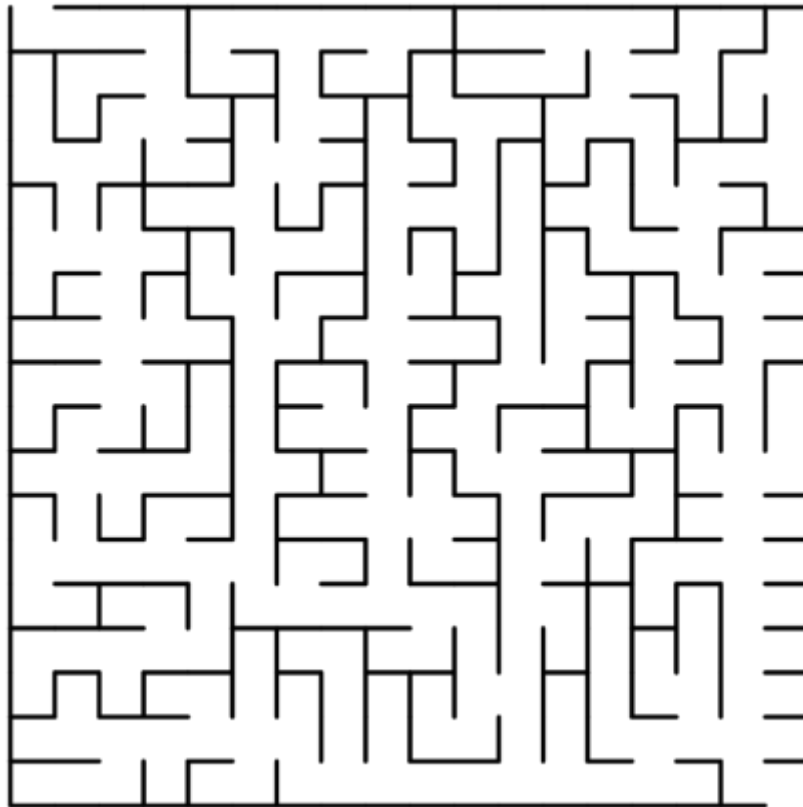


Рисунок 2.6 - Результат створення лабіринту

```

mMazeRows = Mathf.Abs(rows);
mMazeColumns = Mathf.Abs(columns);
if (mMazeRows == 0)
{
    mMazeRows = 1;
}
if (mMazeColumns == 0)
{
    mMazeColumns = 1;
}
mMaze = new MazeCell[rows,columns];
for (int row = 0; row < rows; row++)
{
    for(int column = 0; column < columns; column++){
        mMaze[row,column] = new MazeCell();
    }
}

```

Рисунок 2.7 – Лістинг ініціалізації полів для алгоритму

```

if(movesAvailableCount > 0){
    switch(movesAvailable[Random.Range(0,movesAvailableCount)]){
        case Direction.Start:
            break;
        case Direction.Right:
            mCellsToVisit.Add(new CellToVisit(ctv.Row,ctv.Column+1,Direction.Right));
            break;
        case Direction.Front:
            mCellsToVisit.Add(new CellToVisit(ctv.Row+1,ctv.Column,Direction.Front));
            break;
        case Direction.Left:
            mCellsToVisit.Add(new CellToVisit(ctv.Row,ctv.Column-1,Direction.Left));
            break;
        case Direction.Back:
            mCellsToVisit.Add(new CellToVisit(ctv.Row-1,ctv.Column,Direction.Back));
            break;
    }
}
else{
    mCellsToVisit.Remove(ctv);
}

```

Рисунок 2.8 - Лістинг вибору комірки

Основою даного алгоритму є перевірка сусідів комірки для подальшої генерації. Як відбувається перевірка сусідів, в даному випадку правого сусіда наведено на рисунку 2.9.

```

if(ctv.Column+1 < ColumnCount
    && !GetMazeCell(ctv.Row,ctv.Column+1).IsVisited
    && !IsCellInList(ctv.Row,ctv.Column+1))
{
    movesAvailable[movesAvailableCount] = Direction.Right;
    movesAvailableCount++;
}
else if(!GetMazeCell(ctv.Row,ctv.Column).IsVisited && ctv.MoveMade != Direction.Left)
{
    GetMazeCell(ctv.Row,ctv.Column).WallRight = true;
    if(ctv.Column+1 < ColumnCount)
    {
        GetMazeCell(ctv.Row,ctv.Column+1).WallLeft = true;
    }
}

```

Рисунок 2.9 - Лістинг перевірки правого сусіда комірки

Для отримання комірки з списку використовується метод GetMazeCell (рис.2.10). Цей метод `представляє собою функцію або метод у програмуванні, який призначений для отримання інформації про певну комірку (cell) у лабіринті (maze). Він може повертати різноманітну інформацію про комірку, таку як її стан, координати, наявність стін і інші властивості. Фактичний зміст і функціональність методу `GetMazeCell` можуть змінюватися

в залежності від контексту програми або мови програмування, в якій він використовується.

```
public MazeCell GetMazeCell(int row, int column)
{
    if (row >= 0 && column >= 0 && row < mMazeRows && column < mMazeColumns)
    {
        return mMaze[row,column];
    }
    else
    {
        Debug.Log(row+" "+column);
        throw new System.ArgumentOutOfRangeException();
    }
}
```

Рисунок 2.10 - Лістинг витягування комірки

На підставі аналізу було отримано такі результати:

- шляхом математичного представлення фізичних властивостей об'єктів, було максимально точно відтворено зовнішній вигляд, що нагадує об'єкти реального світу;
- було формалізовано процес взаємодії та реакції віртуальних об'єктів на дії користувачів, забезпечивши інтерактивність, враховуючи фізіологічні особливості дітей дошкільного віку;
- обґрунтовано алгоритм генерації лабіринту за допомогою методу пошуку у глибину для реалізації розвиваючої гри, що сприяє розвитку практичних навичок та набуттю знань дітьми.

2.4 Висновки за розділом

У цьому розділі було проведено проєктування основних об'єктів, процесів та алгоритмів, які використовуються в розробленій інтегрованій платформі програмування для дітей дошкільного віку.

Були детально описані об'єкти та їхні властивості для кожного рівня гри:

реакції, пам'яті, логіки та підготовки до школи. Визначено найважливіші процеси, такі як переміщення об'єктів, їх взаємодія, виявлення зіткнень, а також наведено відповідні математичні формули та рівняння для їх опису.

Окремо було розглянуто алгоритм генерації лабіринтів з одним входом та виходом за допомогою методу пошуку в глибину. Детально описано рекурсивний та ітеративний способи реалізації даного алгоритму з наведенням відповідного коду та прикладів.

Проектування дозволило чітко визначити структуру та взаємодію компонентів розвиваючої платформи, а також забезпечити точність відтворення фізичних властивостей віртуальних об'єктів і процесів, що імітують реальний світ. Це сприяє створенню інтерактивного та захопливого ігрового середовища для дітей, враховуючи їхні фізіологічні особливості.

Отримані результати проектування слугують основою для подальшої розробки інтегрованої платформи програмування, гарантуючи коректну роботу всіх її складових частин та забезпечуючи ефективний розвиток практичних навичок і здобуття нових знань дітьми дошкільного віку.

3 РОЗРОБКА ІГРОВОГО ПРОГРАМНОГО СЕРЕДОВИЩА

3.1 Визначення вимог

Ігрове програмне середовище (ІПС) представляє собою програму, що сприяє розвитку та отриманню нових навичок у дітей, таких як реакція, інтелект, та пам'ять, шляхом виконання різних завдань, які включають додавання, множення, та ділення чисел. Програма має інтуїтивний інтерфейс, можливість отримання оновлень із сервера, та можливість їхньої установки[17].

Функціональна структура даного ігрового програмного середовища зображена на рис. 3.1.

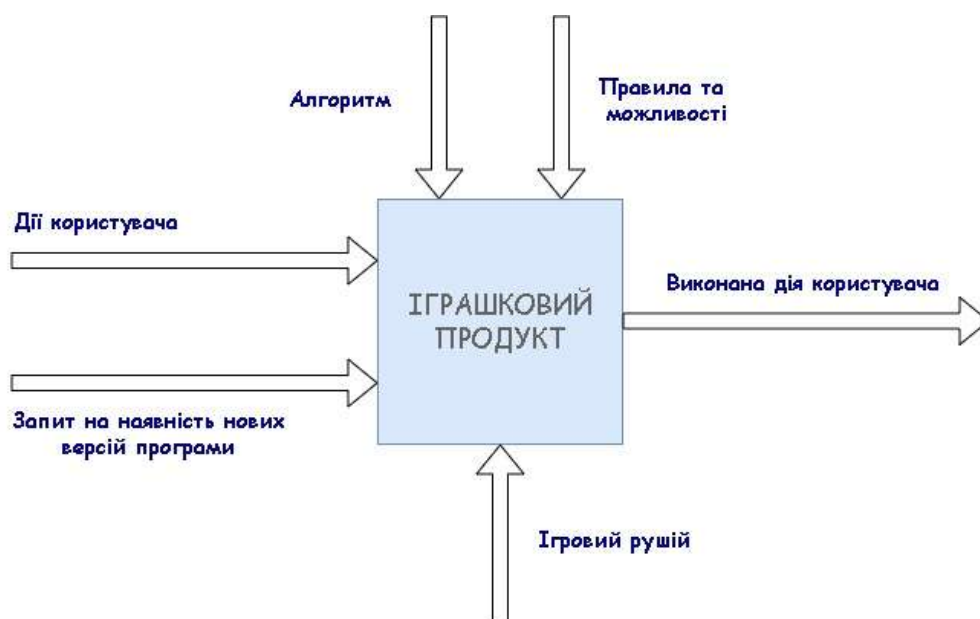


Рисунок 3.1 - Контекстна діаграма ІПП

Детальний процес роботи додатку зображено на рисунку 3.2 , додатку А.

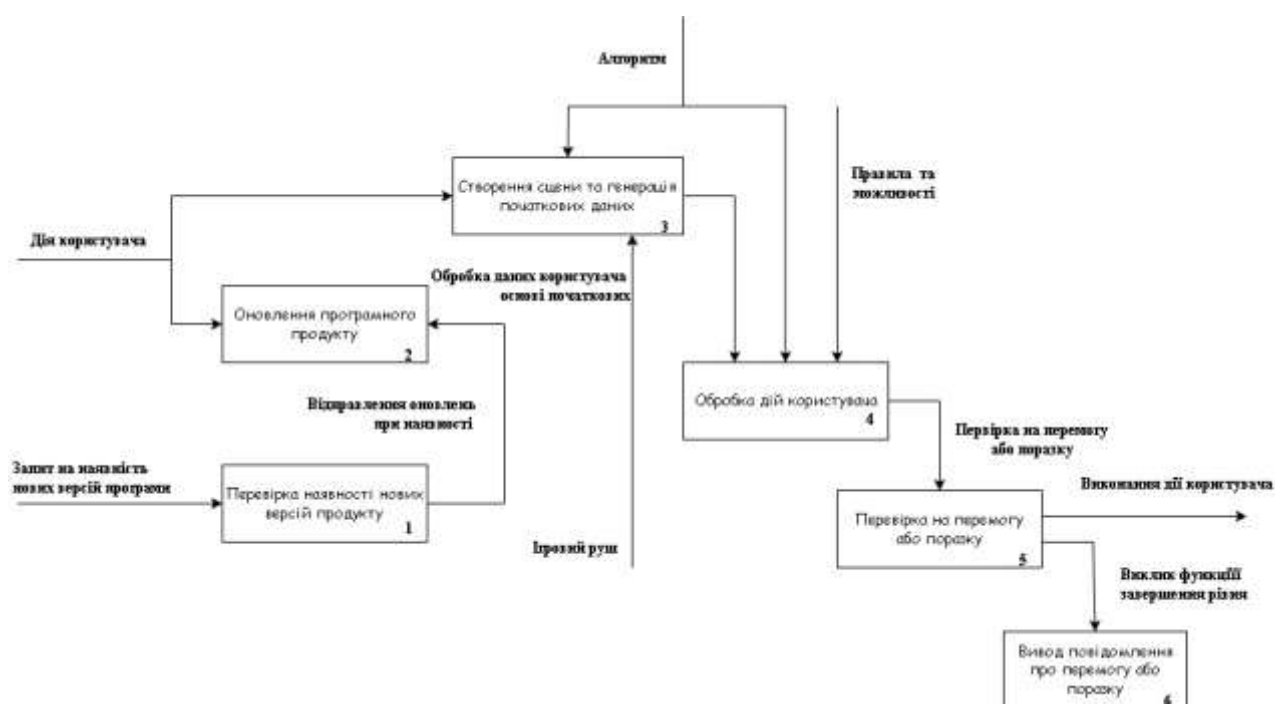


Рисунок 3.2 - Діаграма декомпозиції робочих процесів ІПП

Основна мета ІПП полягає у розвитку розумових навичок дітей. Використовуючи власні алгоритми та правила, програмний продукт відтворює дії користувача на основі введених даних. Для досягнення цієї мети, користувач

виконує певні дії, що сприяють розвитку його навичок.

- ПК з використанням операційних систем: Windows, Linux, MacOS;
- смартфон/Планшет під керуванням операційних систем: Android, IOS.

Ігрове програмне середовище має підтримувати різні операційні системи та працювати на комп'ютерах з низькими технічними характеристиками.

Додатковими функціями ІПС є можливість перевірки оновлень, використання лаунчера для доступу до оновлень та сайту виробника, та запуску самої програми. Цей лаунчер має бути розміщений в хмарі для доступу користувачів у будь-який час.

Мінімальні апаратні вимоги до хмарного сервера:

- два ядра центрального процесора;
- мінімум вісім гігабайт оперативної пам'яті;
- мінімум сто гігабайт твердотільного накопичувача;
- мережевий канал з підтримкою швидкостей більше одного гігабайта.

В ігровому програмному забезпеченні на сучасному етапі реалізовано графічний інтерфейс користувача. Інтерфейс має бути простим та зрозумілим, особливо для дітей. Компанії можуть використовувати різні підходи до дизайну інтерфейсу, враховуючи потреби своїх користувачів.

Інтерфейс користувача виконує роль посередника між користувачем і програмним забезпеченням, включаючи елементи та компоненти програми.

Оскільки цей ігровий продукт призначений для дітей дошкільного та шкільного віку, інтерфейс повинен бути розроблений так, щоб не перевантажувати програму зайвими або складними елементами, забезпечуючи максимальну простоту та зрозумілість для будь-якого користувача. Тому для зручного користування програмним продуктом було прийнято рішення реалізувати графічний інтерфейс за допомогою вбудованих технологій Unity.

Компанії з великою базою користувачів можуть використовувати нестандартні підходи у розробці інтерфейсу та впроваджувати різноманітні специфічні елементи. У той час як інді-розробники або малі компанії, які не мають такої широкої аудиторії, повинні намагатися створити добре продуманий

інтерфейс, оскільки якість і зручність інтерфейсу може суттєво вплинути на бажання користувача використовувати програмний продукт.

Важливою частиною дизайну інтерфейсу є його адаптивність до різних розширень дисплеїв. Така адаптивність допомагає забезпечити комфортне користування програмним продуктом для всіх користувачів [17].

Після визначення функціоналу ігрового програмного продукту було розроблено макет користувацького інтерфейсу.

Після того, як користувач вибрав категорії, інтерфейс повинен оновитись і дати можливість вибрати рівень в вибраній категорії (рис. 3.3, 3.4).



Рисунок 3.3 - Макет ІПП при виборі категорії



Рисунок 3.4 - Макети ІПП при виборі рівня

3.2 Розробка архітектури

Для забезпечення ефективної роботи ігрового програмного продукту та оптимального використання ресурсів комп'ютера, важливо належним чином розробити структуру алгоритмів програмного продукту.

Вихідними даними є:

- дані, що надходять від периферійних пристроїв користувача;
- інформація, що отримана під час попередньої ітерації обробки.

Згідно з поставленими завданнями, функції ігрового програмного продукту можна класифікувати наступним чином:

- алгоритми обробки вхідних даних користувача;
- допоміжні алгоритми, такі як перевірка на перемогу або поразку, оновлення статистики за сесію та інші;
- алгоритм функціонування лаунчера, який здійснює відправлення запитів на сервер, обробку відповідей від сервера, оновлення програмного додатку та запуск програми.

Схему роботи сервера для автооновлень наведено на рисунку 3.5.

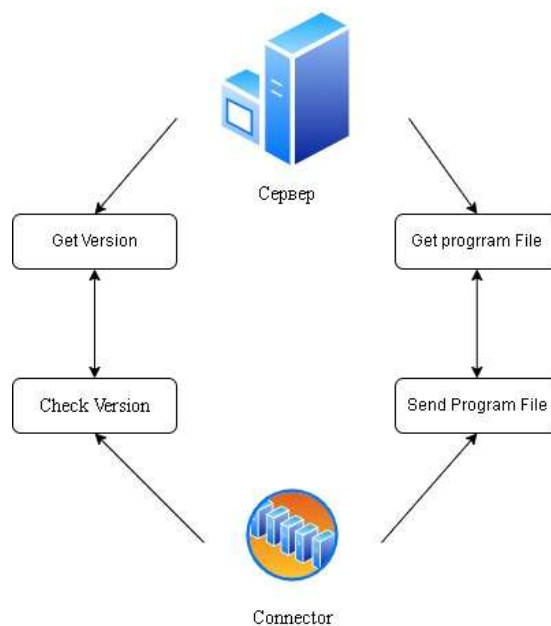


Рисунок 3.5 – Схема роботи сервера для автооновлення

Модуль «Connector» відповідає за прийом запитів від клієнта та обробку їх відповідно до структури запиту, після чого він передає необхідні дані на сервер і повідомляє користувача про результати. Компонент «Server» виконує інструкції, надані «Connector», та повертає результати обробки назад до «Connector».

На рисунку 3.6 наведено діаграму прецедентів сервера для автооновлення.



Рисунок 3.6 - Діаграма прецедентів сервера для автооновлень

На зображенні представлена схема, яка відображає процес роботи з між адміністратором та користувачем. В центрі розташоване хмаринка з написом "Запит на оновлення". Цей центральний елемент з'єднаний стрілками з чотирма іншими елементами.

Зліва вгорі розміщений "Адміністратор", який через стрілку з'єднаний з центральною хмаринкою. Це означає, що адміністратор генерує запит на оновлення.

Зліва внизу знаходиться "Завантаження ПЗ на сервер", що указує на процес завантаження інформаційної системи на сервер.

Справа вгорі є "Підтвердження змін наявності ПЗ", яка також з'єднана з центральною хмаринкою. Це вказує на процес підтвердження змін, що вносяться в інформаційну систему.

Справа внизу користувач, із стрілкою, що веде до центральної хмаринки, що може означати участь користувача в процесі оновлення ПЗ.

Кожен з цих елементів відіграє роль у процесі оновлення інформаційної системи.

Діаграма прецедентів ігрового програмного продукту наведено на рисунку 3.7.

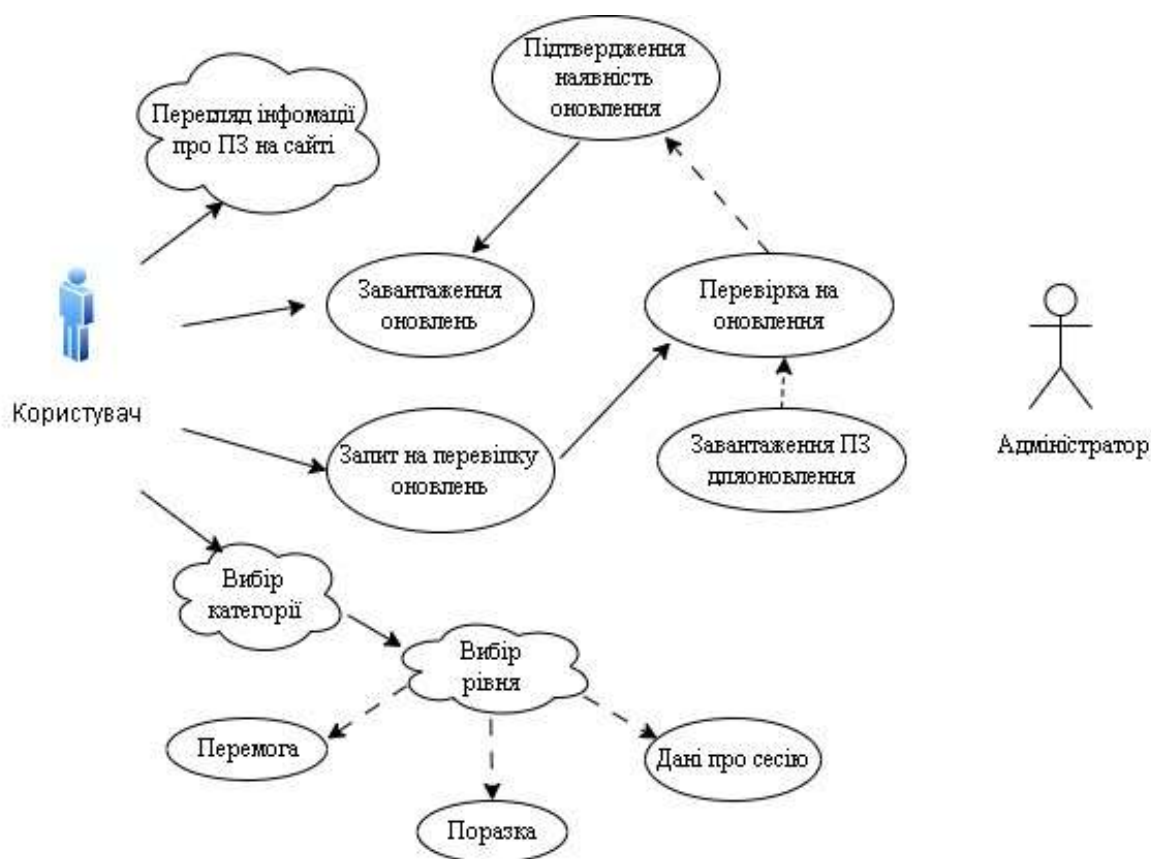


Рисунок 3.7 - Діаграма прецедентів ІПП

3.3 Алгоритми програмного середовища

Для початку, треба створити окремий контролер для кожного рівня гри, який відповідатиме за основні події, специфічні для кожного рівня. Наприклад, для кожного рівня потрібно реалізувати свій власний контролер. На діаграмі у рисунку 3.8 показаний компонент «GameManager», який керує основними аспектами в категорії підготовки до школи[18].

Для створення головного контролера було обрано паттерн проектування «Одинак».

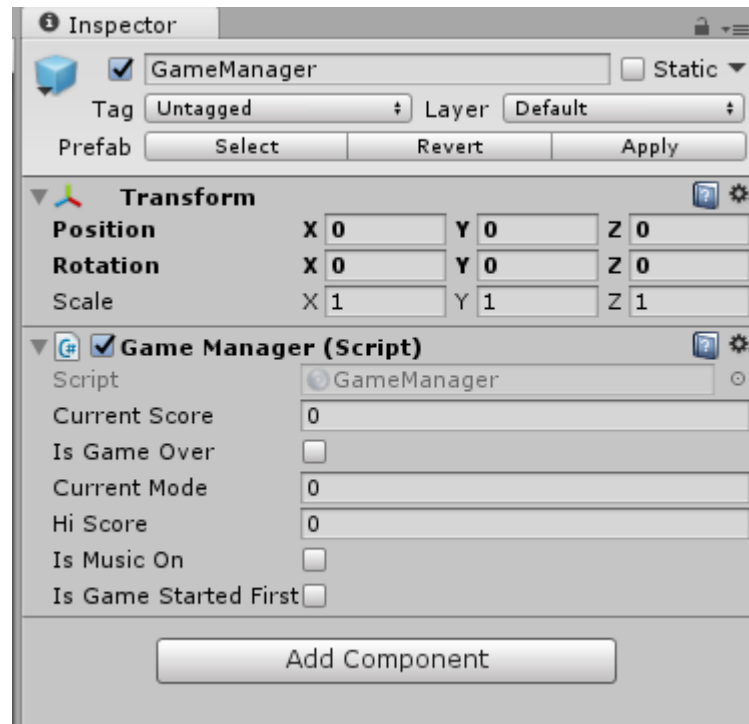


Рисунок 3.8 - Компонент «GameManager»

Шаблон проектування «Одинак» належить до категорії породжуючих шаблонів. Цей шаблон забезпечує, що у класі буде лише один екземпляр і надає глобальну точку доступу до цього екземпляра.. Крім основної частини ігрового програмного продукту, такого роду забезпечення має додаткові компоненти, такі як звуковий менеджер [18]. Цей контролер відповідає за управління всіма звуками у грі: відтворення музики, звукові ефекти, а також включення або виключення музичного супроводу.

Реалізацію звукового менеджера в Unity представлено на рисунку 3.9.

Розробка ігрового програмного продукту охоплює не лише написання програмного коду для опису бізнес-логіки, але також розробку користувацького інтерфейсу згідно з попередньо сконструйованим макетом.

Першим етапом є розміщення функціональних елементів на формах відповідно до заздалегідь спроектованих схем. Є безліч об'єктів, які можна розмістити на формі.

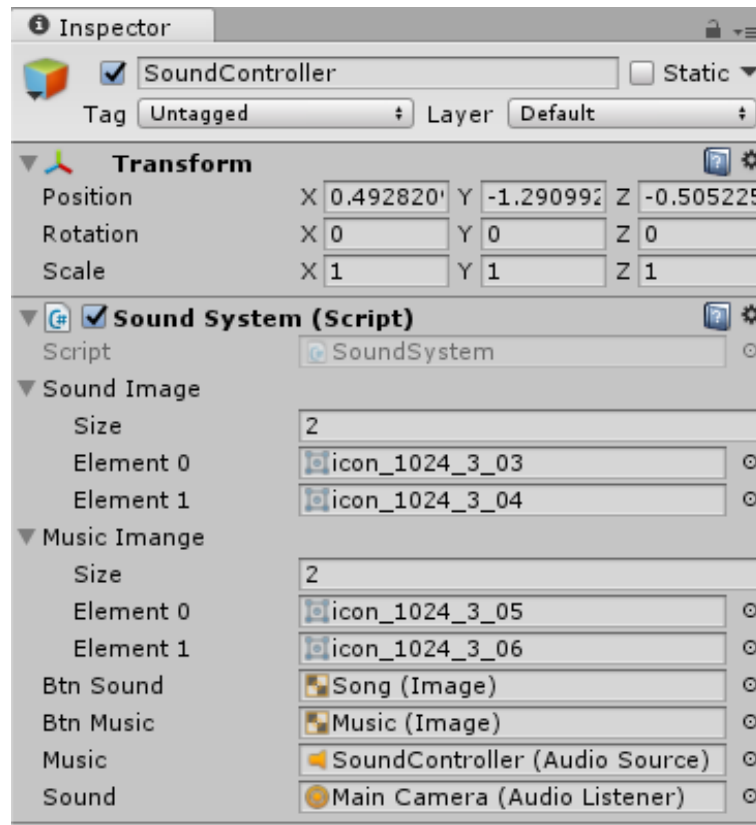


Рисунок 3.9 - Звуковий менеджер

Розглянемо лише ті, які необхідні для реалізації додатку:

- «Text» використовується для відображення текстової інформації;
- «Image» призначений для відображення графічних зображень;
- «Button» виконує різноманітні функції та важливий для зв'язку бізнес-логіки з графічним інтерфейсом;
- «Canvas» забезпечує адаптивність розміщення елементів на різних дисплеях;
- «Panel» аналогічний до Canvas, але не підтримує можливості адаптивного дизайну.

Після формування екранних форм, наступним кроком буде програмування кнопок та основного функціоналу. Для створення обробника подій потрібно:

- створити метод, який буде викликатися при взаємодії з графічним елементом;
- реалізувати функціонал цього методу;

- зв'язати елемент з методом.

Для прикладу наведемо реалізацію обробки події натискання на кнопки, що відповідають за включення та виключення музики (рис. 3.10).

```
public void SetMusic()
{
    Settings.OnMusic = !Settings.OnMusic;
    PartMusic();
}
```

Рисунок 3.10 - Лістинг обробки події натискання клавіші музики

Вважається правильним, коли метод для обробки події графічного елемента не містить в собі жодної реалізації бізнес-логіки. Він лише викликає функцію, яка відповідає за подальший функціонал. Таким чином, логіка обробки включення та виключення музики представлена на рисунку 3.11.

Як доповнення до ігрового програмного продукту необхідно реалізувати сервер і лаунчер, який буде здійснювати зв'язок з сервером, завантажувати оновлення та оновлювати програмний продукт.

```
private void PartMusic()
{
    _music.enabled = Settings.OnMusic;

    if (Settings.OnMusic)
    {
        _btnMusic.sprite = _musicImage[0];
    }
    else
    {
        _btnMusic.sprite = _musicImage[1];
    }
}
```

Рисунок 3.11 - Лістинг алгоритму роботи включення або виключення музики

Схему роботи серверу автоматичного оновлення наведена на вищенаведеному рисунку 3.5. Розглянувши схему роботи цього механізму, можна зробити висновок, що цей додаток складається з двох частин: перевірка версії та отримання нової версії.

Робота функції отримання версії зображена на рисунку 3.12.

```
public static string GetVersion(string path)
{
    var version = "Error";
    var versionPath = string.Empty;
    try
    {
        versionPath = Path.Combine(path,
"Version.txt");
    }
    catch
    {
        Console.WriteLine("Path combine error.");
        return null;
    }
    if (!Directory.Exists(path) ||
!File.Exists(versionPath))
    {
        Console.WriteLine("Some file not found.");
        return null;
    }
    try
    {
        version =
File.ReadAllText(versionPath).Trim();
    }
    catch
    {
        Console.WriteLine("Read file error.");
        return null;
    }
    return version;
}
```

.Рисунок 3.12 - Лістинг отримання поточної версії додатку

Якщо знайдено нову версію ігрового програмного продукту, необхідно отримати файли для оновлення, алгоритм роботи цієї частини показаний на рисунку 3.13.

Нарешті, останнім компонентом, який необхідно розробити, є лаунчер для запуску ігрового програмного продукту та отримання нових версій забезпечення, а також його оновлення.

```

public static byte[] GetProgramFile(string path)
{
    byte[] file = null;
    var filePath = string.Empty;
    try
    {
        filePath = Path.Combine(path, "Installer.zip");
    }
    catch
    {
        Console.WriteLine("Path combine error.");
        return null;
    }
    if (!Directory.Exists(path) ||
!File.Exists(filePath))
    {
        Console.WriteLine("Some file not found.");
        return null;
    }
    try
    {
        file = File.ReadAllBytes(filePath);
    }
    catch
    {
        Console.WriteLine("Read file error.");
        return null;
    }
    return file;
}

```

Рисунок 3.13 - Лістинг отримання оновлення

Для його розробки використано технологію створення графічних додатків WPF [19]. Підключення лаунчера до сервера відбувається за допомогою технології WCF, яку розглянуто під час вибору технологій для розв'язання поставленого завдання [20]. Крім того, за допомогою цього лаунчера можна переглянути сторінку ігрового програмного продукту на веб-сайті розробника, ознайомитись з детальним описом, переглянути скріншоти та ознайомитись із системними вимогами.

3.4 Висновок за розділом

У даному розділі було визначено основні вимоги до розробленого ігрового програмного продукту, зокрема його функціональність, цільову аудиторію, підтримувані операційні системи та апаратні вимоги. Також було розроблено макети користувацького інтерфейсу для головного меню та вибору рівнів гри.

Був проведений аналіз архітектури програмного забезпечення, визначені основні модулі та їх функції. Розглянуто схему роботи механізму автооновлення, а також наведено відповідні діаграми прецедентів для серверної частини та самого ігрового додатку.

Детально описано основні алгоритми програмного середовища, такі як реалізація ігрового менеджера з використанням патерну "Одинак", звукового менеджера для управління музикою та звуковими ефектами, а також розробка користувацького інтерфейсу з прив'язкою елементів управління до відповідних обробників подій.

Окремо розглянуто алгоритми роботи механізму автооновлення, включаючи перевірку наявності нової версії програмного забезпечення на сервері та завантаження оновлень у разі їх наявності. Також наведено приклади відповідного програмного коду.

Описано розробку лаунчера для запуску ігрового додатку та отримання оновлень з використанням технологій WPF та WCF для взаємодії з серверною частиною.

Отримані результати забезпечують створення повнофункціонального програмного середовища для ігрового програмного продукту з можливістю зручного користувацького інтерфейсу, керування звуковими ефектами та музикою, а також автоматичного оновлення програми через лаунчер.

4 ТЕСТУВАННЯ ІГРОВОГО ПРОГРАМНОГО ПРОДУКТУ

4.1 Тестування інтерфейсу користувача

Процес тестування програмного забезпечення полягає у зборі інформації та дослідженні для визначення відповідності функціоналу заданим вимогам і встановленому рівню якості. Цей процес також має на меті виявлення помилок або дефектів, що можуть виникати під час роботи програмного продукту.

Головною метою тестування є виявлення будь-яких невідповідностей у роботі програмного продукту відносно встановлених вимог.

Почнемо тестування розробленої системи. Перш за все, потрібно запустити ігровий додаток. Перевіримо можливість вибору категорій та рівнів. Для цього вимкнемо музику та звуки в додатку, як представлено на рисунку 4.1.

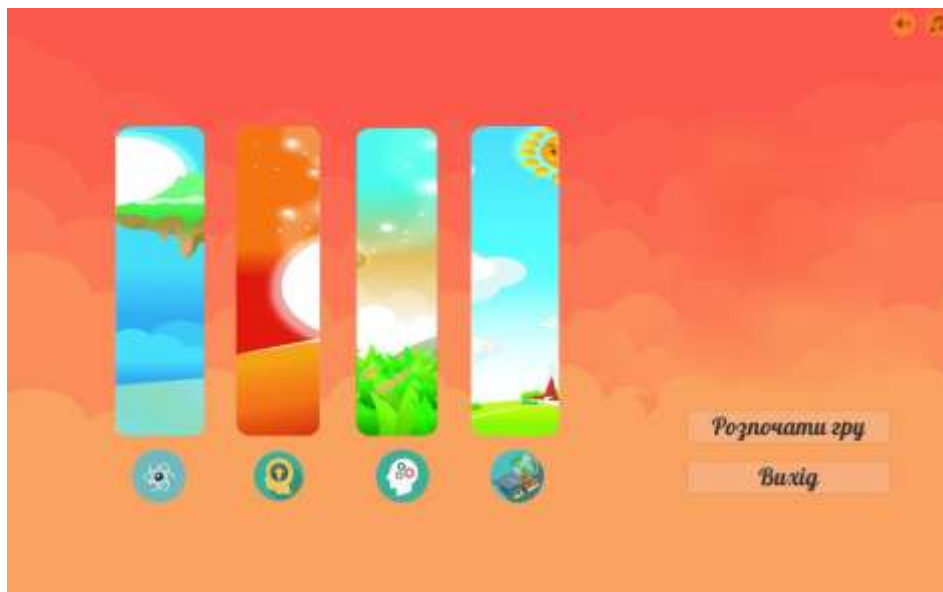


Рисунок 4.1 - Вимкнення музичного супроводу та звуків в додатку

Наступним етапом буде перевірка контролера рівня. Спробуємо зламати всі доступні можливості для проведення тестування програшу. Ігровий додаток успішно відображає повідомлення про поразку, рис. 4.2.



Рисунок 4.2 - Результат тестування першого з категорії «Реакція»

Подібно проведемо тест, щоб перевірити, чи правильно відображається повідомлення про перемогу. Натиснемо кнопку «Повторна спроба», щоб перезапустити рівень і зачекати відведену кількість часу. Якщо всі етапи виконані правильно, ми отримаємо повідомлення про перемогу (рис. 4.3).



Рисунок 4.3 - Повідомлення про поразку

Повторимо ці тести для всіх рівнів, які мають відображення повідомлень про перемогу або поразку. Після натискання кнопки «Повторна спроба» і перезапуску рівня нам потрібно буде витримати необхідний час, який вказаний

у верхньому лівому куті екрану. Якщо всі попередні етапи успішно пройдені і всі рівні пройшли тестування, ми отримаємо повідомлення про перемогу.

Подібно протестуємо всі рівні, які підтримують відображення повідомлень про перемогу або поразку. Якщо цей етап тестування буде успішно завершений, то всі рівні будуть перевірені і помилок в роботі не буде виявлено (рис. 4.4).



Рисунок 4.4 - Повідомлення про перемогу

Наступною задачею буде перевірка рівнів, де немає необхідності виконувати певні завдання для перемоги. У таких рівнях гра триває до тих пір, поки користувач не змінить рівень або не закриє додаток.

Для проведення цього етапу тестування, оберемо категорію «Пам'ять» і перейдемо на другий рівень. Завантаження успішно завершено, і результат цього процесу показано на рисунку 4.5.

Тепер перевіримо, чи збільшиться кількість пройдених спроб після завершення ігрового циклу. Результат тестування зображено на рисунку 4.6.



Рисунок 4.5 - Результат завантаження рівня з категорії «Пам'ять»



Рисунок 4.6 - Результат тестування статистики за сесію

Подібно цьому етапу тестування ми проведемо тест для всіх інших рівнів, які підтримують відображення статистики за сесію. Цей етап успішно завершений, і всі рівні успішно пройшли тестування.

Тестування ігрового програмного продукту завершено; під час його проведення не було виявлено жодних помилок у роботі. Наступним кроком буде перевірка роботи лаунчера та сервера для автооновлень.

4.2 Тестування лаунчера

Для перевірки лаунчера ми запустимо його і перевіримо наявність повідомлення про наявність нових версій продукту (рис. 4.7).

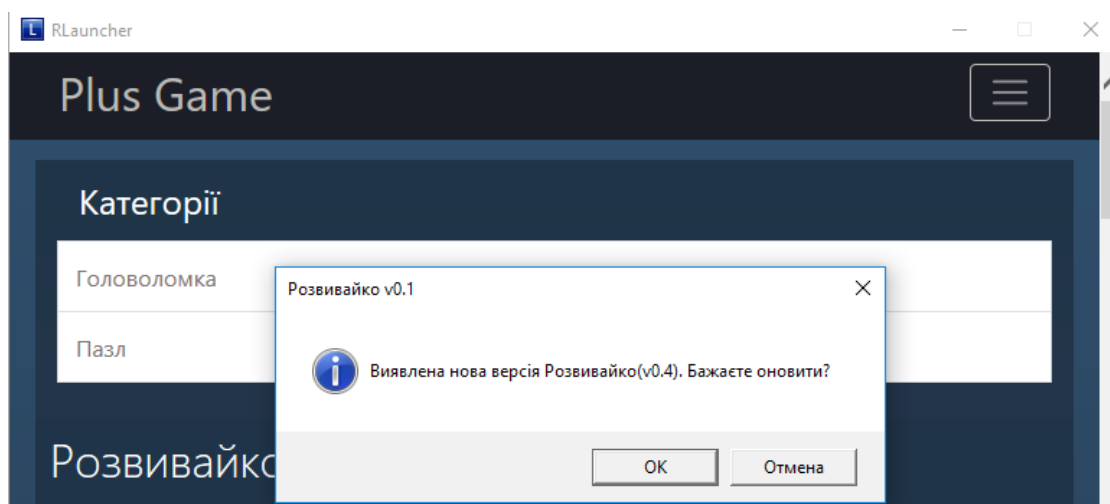


Рисунок 4.7 - Повідомлення про наявність оновлень

Наявність оновлень підтверджено, програма запропонувала нам завантажити їх натискаємо «Так», після чого відбувається завантаження оновлень (рис. 4.8).

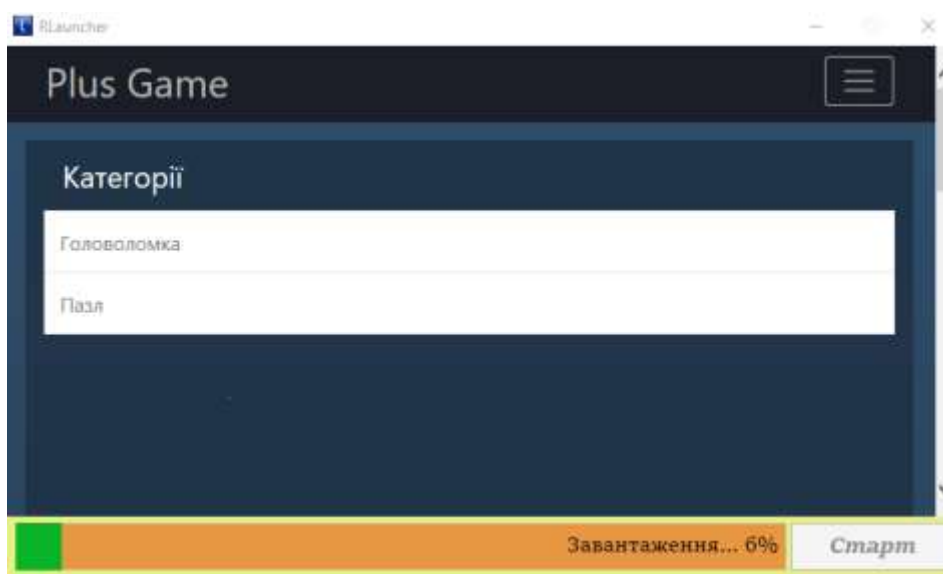


Рисунок 4.8 - Завантаження оновлень

Після завантаження, лаунчер сам оновить всі необхідні файли, користувачу залишається лише очікувати (рис. 4.9).

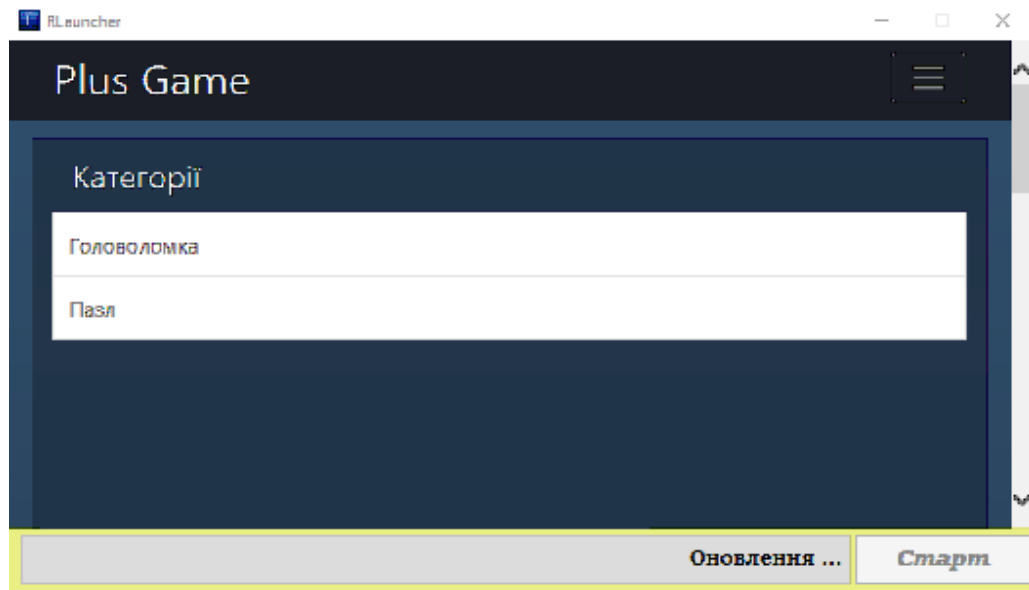
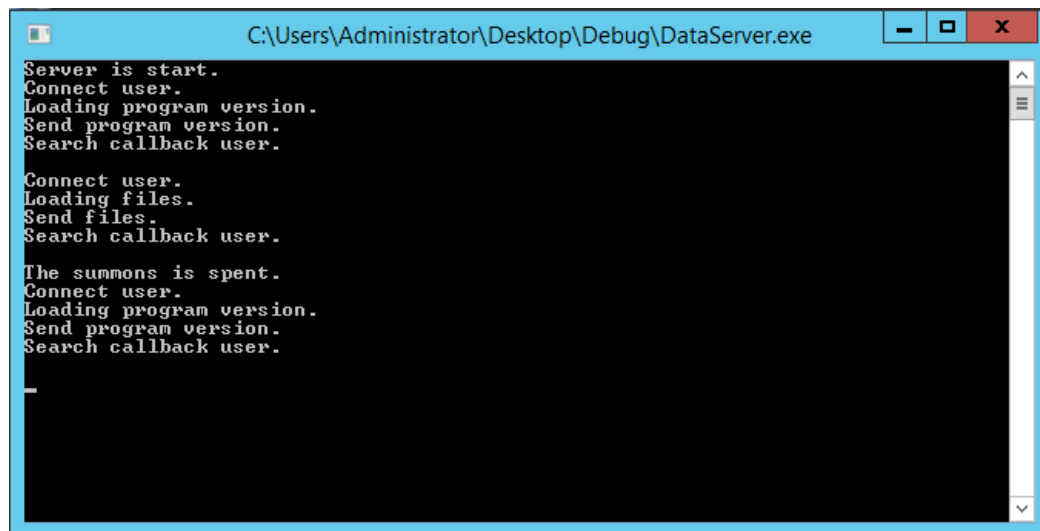


Рисунок 4.9 - Оновлення ігрового програмного продукту

Проведення перевірки системи оновлень завершилося успішно; функціональні можливості оновлення були вивчені та перевірені, і ніяких проблем не виявлено.

4.3 Тестування роботи сервера

Останнім етапом тестування буде аналіз роботи сервера. Оскільки ми вже завантажували оновлення з сервера, залишається лише скористатися віддаленим доступом для з'єднання з сервером та перевірки інформації, яку він відображає у консолі (рис. 4.10).



```

C:\Users\Administrator\Desktop\Debug\DataServer.exe
Server is start.
Connect user.
Loading program version.
Send program version.
Search callback user.

Connect user.
Loading files.
Send files.
Search callback user.

The summons is spent.
Connect user.
Loading program version.
Send program version.
Search callback user.

```

Рисунок 4.10 - Результат тестування сервера для автооновлення

Таким чином, система працює бездоганно і не потребує подальших вдосконалень. Даний додаток успішно пройшов всі етапи тестування, весь його функціонал працює правильно, а ніяких збоїв у роботі програми не виявлено.

Переваги цього продукту включають:

- простий і зрозумілий інтерфейс;
- можливість розвивати розумові навички;
- стабільна робота додатку та сервера;
- API (прикладний програмний інтерфейс) сервера можна використовувати в будь-якому проекті;
- можливість сервера обробляти велику кількість користувачів.

На основі цієї роботи отримано наступні результати:

- визначено основні вимоги до навчально-розвиваючого середовища за допомогою контекстуальної діаграми та діаграми декомпозиції, що дозволило побудувати архітектуру інтегрованого програмного продукту (ІПП);
- створено макети інтерфейсу користувача, на основі яких було реалізовано інтерфейс в ІПП;
- описано основні алгоритми, які були розроблені у програмному додатку з використанням мови програмування C# та ігрового рушія Unity;
- проведено ручне тестування навчально-розвиваючого середовища.

4.4 Висновки за розділом

У цьому розділі було детально описано процес тестування розробленого ігрового додатку, лаунчера та серверу для автооновлень. Тестування інтерфейсу користувача дозволило перевірити всі функціональні можливості програми, такі як вибір категорій і рівнів, відображення повідомлень про перемогу або поразку, збереження статистики ігрових сесій. Усі компоненти системи успішно пройшли тестування, і жодних помилок чи збоїв у роботі виявлено не було.

Окремо було протестовано роботу лаунчера та серверу для автооновлення програмного забезпечення. Перевірка підтвердила коректну роботу системи завантаження та встановлення оновлень.

Таким чином, весь функціонал розробленого програмного середовища було ретельно протестовано, і система готова до використання. Додаток має простий та зрозумілий інтерфейс, можливість розвивати розумові навички, стабільну роботу усіх компонентів. Сервер здатний обробляти велику кількість користувачів та надає API для інтеграції з іншими проектами.

ВИСНОВКИ

В результаті розробки даного дипломного проєкта було виконано всі вимоги технічного завдання.

На етапі аналізу електронних джерел було проведено аналіз предметної області, сформульована та обґрунтована задача, функції системи та її структура, визначені вимоги до системи.

Було обрано просте та недороге програмне забезпечення, яке доступне широкому колу користувачів.

Для відображення властивостей у вигляді реального часу на основі фізичних характеристик було проведено математичне представлення властивостей.

За допомоги проєктування було створено процес взаємодії та реакції віртуальних об'єктів на дії користувачів, з урахуванням особливостей дітей дошкільного віку.

Проаналізовано існуючий фізичний рушій в Unity. Наведено приклади роботи простих фізичних механізмів в фізичних рушіях.

Даний ігровий програмний продукт, позиціонує себе, як додаток для розвитку розумових навичок, а також отримання нових знань дитиною.

Були розроблені концепції інтерфейсу користувача, за допомогою яких було створено інтерфейс у вигляді ігрового програмного продукту. Також було проведено тестування навчально-розвиваючого середовища.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Super Mario: Історія найвідомішої в світі відеогри [Електронний ресурс] – Режим доступу: <https://drukarnia.com.ua/articles/super-mario-istoriya-naividomishoyi-v-sviti> (дата звернення 18.04.2024).
2. The Witness [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/The_Witness (дата звернення 18.03.2024).
3. Europa Universalis IV — відеогра [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Europa_Universalis_IV (дата звернення 21.04.2024).
4. Spore — відеогра жанрів симулятора життя істот і симулятор бога [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/Spore> (дата звернення 18.04.2024).
5. Just Dance [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Just_Dance (дата звернення 05.04.2024).
6. Основи програмування в Unity [Електронний ресурс] – Режим доступу: <https://gamedev.dou.ua/forums/topic/38526/> (дата звернення 18.02.2024).
7. Розкриття потенціалу Unreal Engine 5. Реальне застосування (тutoriал для дизайнерів) [Електронний ресурс] – Режим доступу: <https://ux.pub/sergeykozlov/rozkrittia-potentsialu-unreal-engine-5-riialnie-zastosuvannia-tutorial-dlia-dizainieriv-2g57> (дата звернення 10.03.2024).
8. Розробка ігор на unreal engine 5 [Електронний ресурс] – Режим доступу: <https://avada-media.ua/ua/services/razrabotka-igr-na-unreal-engine-5/> (дата звернення 18.03.2024).
9. Методологія Kanban: можливості та принципи роботи [Електронний ресурс] – Режим доступу: <https://para.school/blog/management/metodologija-kanban-vozmozhnosti-i-printsipu-rabotu> (дата звернення 17.03.2024).
10. Роль гри в розвитку творчості дітей дошкільного віку [Електронний

ресурс] – Режим доступу: <https://vseosvita.ua/library/embed/01008l2y-fd42.doc.html> (дата звернення 21.03.2024).

11. Континіум електронного навчання [Електронний ресурс] – Режим доступу: <https://elr.tnpu.edu.ua/mod/forum/discuss.php?d=322> (дата звернення 26.03.2024).

12. П'ятнашки. [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/П'ятнашки> (дата звернення: 16.04.2024).

13. Хрестики-нулики. [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/Хрестики-нулики> (дата звернення: 20.04.2024).

14. Виявлення зіткнень. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Виявлення_зіткнень (дата звернення: 30.03.2024).

15. Алгоритм створення лабіринту. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм_створення_лабіринту (дата звернення: 01.02.2024).

16. Алгорі́тм пошуку́ в глибині́ з ітеративним заглибленням по графу. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Пошук_в_глибину_з_ітеративним_заглибленням (дата звернення 29.02.2024).

17. Розробка дизайну інтерфейсу гри [Електронний ресурс] – Режим доступу: <https://avada-media.ua/ua/services/razrabotka-dizayna-interfeysa-igry/> (дата звернення 16.03.2024).

18. Sound Manager для невеликих ігор та прототипів на Unity [Електронний ресурс] – Режим доступу: <https://habr-com.translate.google/articles/274529/> (дата звернення 02.04.2024).

19. Windows Presentation Foundation. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation (дата звернення: 07.12.2023).

20. Windows Communication Foundation. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Windows_Communication_Foundation (дата звернення 30.01.2024).

Рисунок А.1 - Діаграма декомпозиції робочих процесів ІПІ

Контекстна діаграма іграшкового програмного продукту



					13.02070849.00061 ПЛІ											
					Розробка програмного додатку розвиваючої гри						Літ.		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата												
Розробив	Гюльназарян Я.В.															
Перевірів	Ільяшенко М.І.															
Т.контр.											Аркуш 1		Аркушів 1			
Н.контр.	Польська О.В.				Контекстна діаграма іграшкового програмного продукту						НУ «Запорізька політехніка», гр. КНТЗ-511сп					
Затвердив	Кудерметов Р.Є.															

Схема роботи сервера для автооновлень

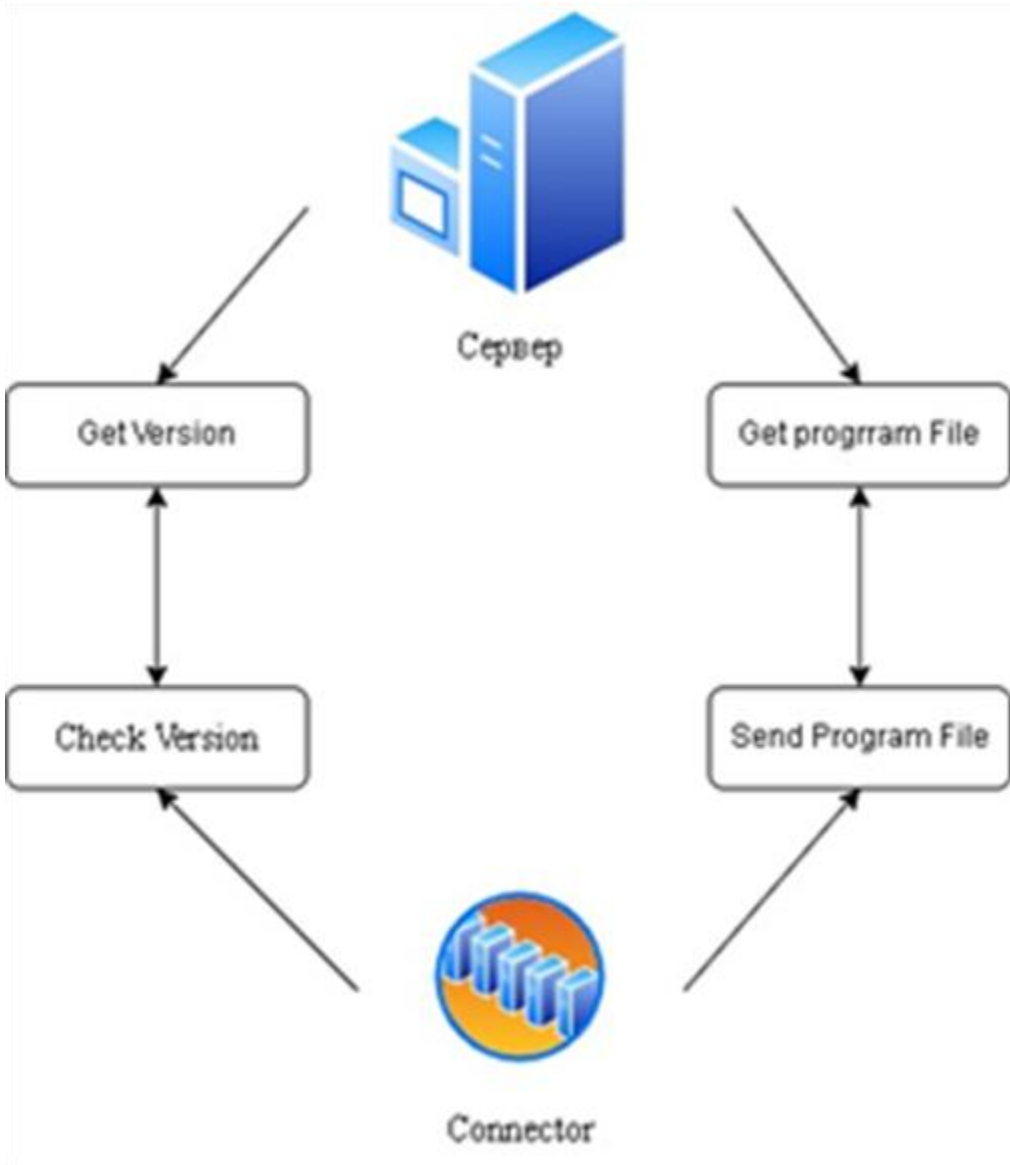
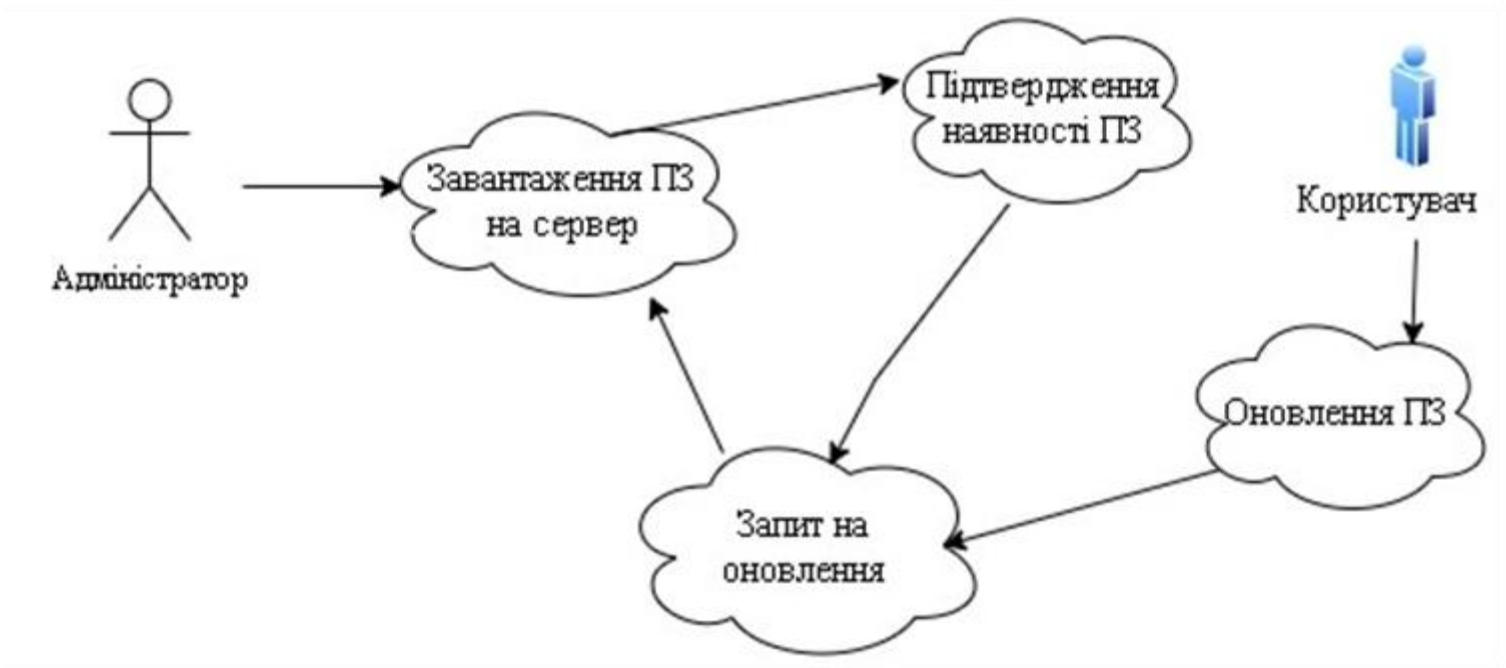


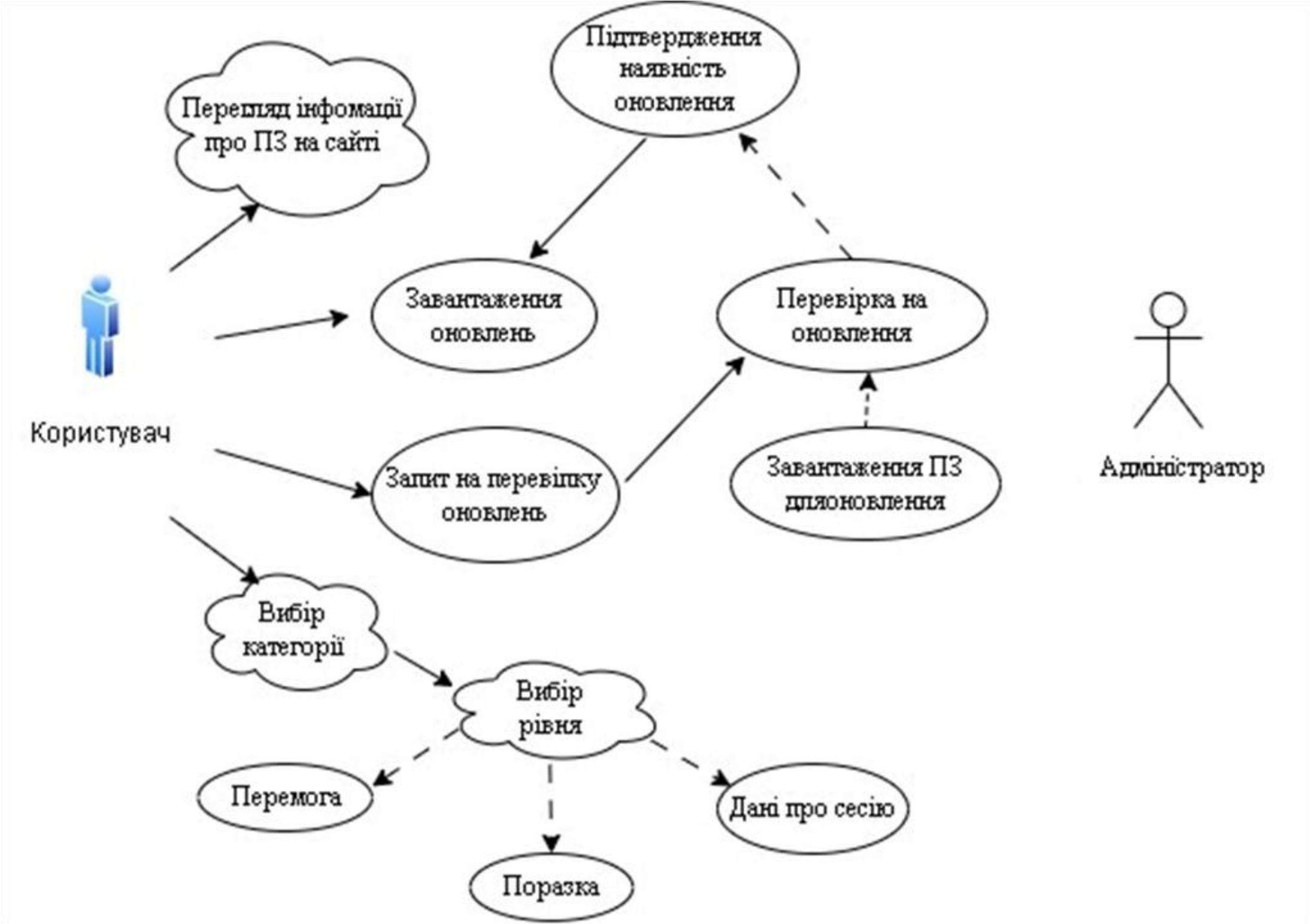
Схема роботи серверу автооновлення



Діаграма прецедентів сервера для автооновлень

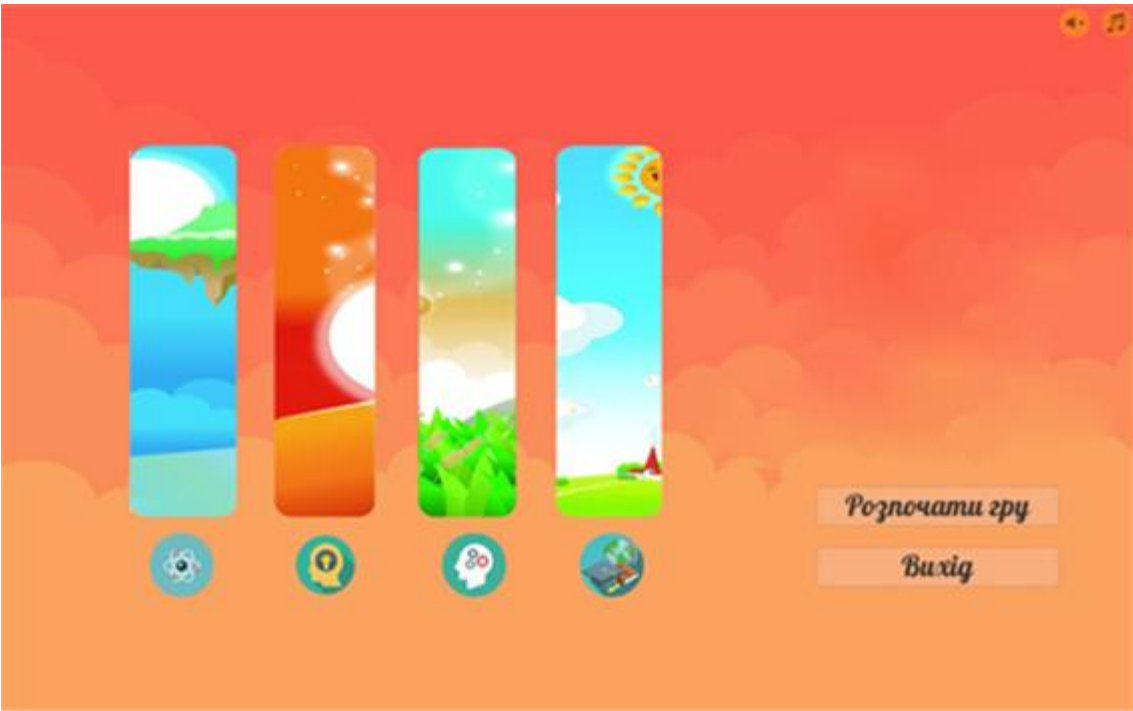
					13.02070849.00061 ПЛІ2				
					Розробка програмного додатку розвиваючої гри		Літ.	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Гюльназарян Я.В.							
Перевірів		Ільяшенко М.Б.							
Т.контр.					Схема роботи сервера для автооновлень		Аркуш 1		Аркушів 1
Н.контр.		Польська О.В.					НУ «Запорізька політехніка», гр. КНТз-511сп		
Затвердив		Кудерметов Р.К.							

Діаграма прецедентів ігрового програмного продукту



					13.02070849.00061 ПЛЗ									
					Розробка програмного додатку розвиваючої гри	Літ.			Маса			Масштаб		
Зм.	Арк.	№ документа	Підпис	Дата										
Розробив		Гюльназарян Я.В.												
Перевірів		Ільяшенко М.Б.												
Т.контр.						Аркуш 1			Аркушів 1					
Н.контр.		Польська О.В.			Діаграма прецедентів ІПП	НУ «Запорізька політехніка», гр. КНТЗ-511 сп								
Затвердив		Кудерметов Р.С.												

Результат тестування ігрового програмного додатку розвиваючої гри



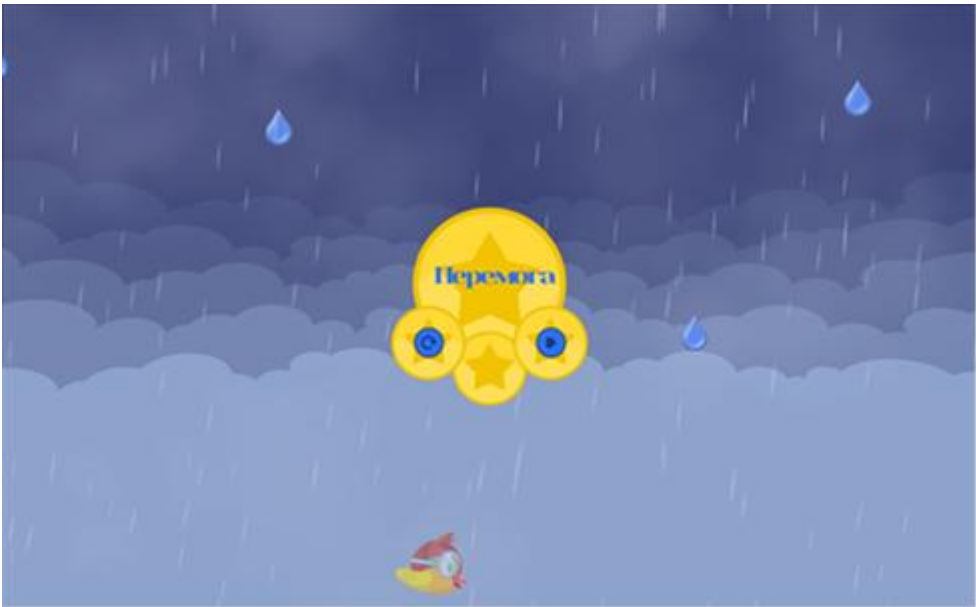
Вимкнення музичного супроводу та звуків в додатку



Повідомлення про поразку



Результат тестування першого з категорії «Реакція»



Повідомлення про перемогу

					13.02070849.00061 ПЛ4										
					Розробка програмного додатку розвиваючої гри					Літ.		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата						Аркуш 1		Аркушів 1			
Розробив		Гюльназарян Я.В.													
Перевірив		Ільяшенко М.Б.													
Т.контр.															
Н.контр.		Польська О.В.			Результати тестування ігрового програмного додатку розвиваючої гри					НУ «Запорізька політехніка», гр. КНТз-511сп					
Затвердив		Кудерметов Р.К.													