Міністерство освіти і науки України

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра економічної кібернетики та інформатики

Головатий Андрій Петрович

Система генерації квестових завдань для стимулювання засвоєння навичок програмування

спеціальність 124 Системний аналіз

освітньо-професійна (наукова) програма Системний аналіз

випускна кваліфікаційна робота за освітнім ступенем «магістр»

Виконав студент

групи Сам-21

Головатий А.П.

_____

підпис

Науковий керівник:

д.т.н., професор

Пасічник Р.М.

_____

підпис

Тернопіль – 2024

ВСТУП

Актуальність теми. Зростання ролі інформаційних технологій в сучасному суспільстві обумовлює необхідність підготовки кваліфікованих фахівців в галузі програмування. Однак, традиційні методи навчання часто не здатні забезпечити достатній рівень мотивації та залученості учнів у навчальний процес.

В даній роботі пропонується інноваційний підхід до вирішення цієї проблеми шляхом створення системи генерації квестових завдань для стимулювання засвоєння навичок програмування. Гейміфікація навчання дозволяє перетворити складний процес опанування програмування на захоплюючу гру, що сприяє підвищенню інтересу учнів, розвитку їх творчого потенціалу та формуванню практичних навичок.

Ключовою особливістю цієї системи є її орієнтація на створення захопливого навчального середовища. Замість традиційних уроків та вправ, учні занурюються у світ пригод виконуючи різноманітні квестові завдання, де кожне правильно виконане завдання наближає гравців до їхньої мети. Вона також інтегрує різні механізми заохочення, такі як нарахування балів чи відкриття досягнень. Здобуття винагород позитивно впливає та мотивацію, що стимулює активну участь в навчальному процесі. Завдяки цьому, учні з захопленням освоюють навички програмування, перетворюючи навчання на цікаву та продуктивну гру.

Впровадження розробленої системи в освітній процес має значний потенціал для підвищення якості навчання програмуванню та підготовки конкурентоспроможних фахівців в галузі інформаційних технологій.

Головні аспекти новизни є:

- Розробка системи квестових завдань з програмування, орієнтованої на захопливе навчання та індивідуальний підхід до кожного учня.
- Інтеграція різноманітних ігрових механік (нарахування балів, відкриття досягнень, рейтинги тощо) для підвищення мотивації та залученості учнів у навчальний процес.
- Створення інтуїтивно зрозумілого інтерфейсу системи, що сприяє легкому та приємному використанню.

Вперше запропоновано систему гейміфікованої підтримки процесу вивчення основ програмування на мові C#, яка дозволяє підтримувати та розвивати інтерес користувача до вивчення основ використання формалізованої алгоритмічної мови за допомогою інтегруючої ігрової місії курсу, створення ігрових перешкод, блокування ефективного використання ігрових ресурсів. Різноманітна реалізація вказаних ігрових засобів створює мотивацію для послідовного розв’язання навчальних задач курсу із контрольованим рівнем успішності для виконання ігрової місії. Засвоєння матеріалу курсу виступає як побічний ефект виконання згаданої місії, усуваючи втому від засвоєння формалізованого предмета.

Практичне значення, теоретичне значення та апробація залишаються в силі, але з урахуванням вищезазначених коректив.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Генерація квестів та інфраструктура для їх виконання та оцінювання

Система генерації квестових завдань для стимулювання засвоєння навичок програмування потребує продуманої інфраструктури яка б забезпечила не лише створення різних завдань, а давала змогу ефективно оцінити виконання завдання користувачем. Інфраструктура повинна врахувати різні потреби викладачів, потреби різних користувачів, від учня до студента й вище. Вона має мати змогу адаптуватись до навчального процесу.

Важливою складовою є система подання завдання, як було сказано вище, вона має бути гнучкою та адаптивною для внесення змін під певні потреби. Вона повинна дозволяти створювати завдання різних складностей, різної тематики та формату, повинна врахувати індивідуальні навички студентів та рівні їх підготовки. Це може бути реалізоване через загальні налаштування квестів, вибір мови для вивчення, створення різних типів завдань також в подальшому можливі інтеграції з різними навчальними ресурсами для вдосконалення.

Типи квестів. Застосунок має декілька різних квестів для виконання деяких квестів, користувачу потрібно проявити не лише навички програмування, а також вміння приймати не звичайно рішення та адаптуватись до отриманих умов. Одним з квестів є класичне “знайди та принеси”, де користувач має з допомогою різних фрагментів коду створити об’єкт щоб подолати різні перешкоди, досягти потрібної цілі, та мати можливість повернутись назад для отримання винагороди за виконання завдання. Ще одним типом є пошук та виправлення помилок в коді, коли наприклад користувачеві не прогружається світ чи в нього не правильно налаштоване керування. Тоді гравець як спеціаліст повинний знайти та усунути поломку, це може бути видалення шкідливого об’єкта, додавання чи усунення корисних чи шкідливих елементів. Також модифікація програмного середовища, користувачу потрібно буде придумати функцію, чи метод для подолання дивної чи не звичної перешкоди яку класичними методами здолати не вдасться.

Середовищем виконання буде додаток для мобільних пристроїв, що працюють на операційній системі Android також при потребі можна завантажити додаток для операційних систем Windows проте оптимізація націлена на роботу з мобільними девайсами. Залежно від вибору платформи буде змінена взаємодія з додатком, якщо користувач обирає комп'ютерну версію, то взаємодія з середовищем буде відбуватись через маніпулятори, а саме для роботи потрібна буде клавіатура та миша, якщо у гравця є сенсорний дисплей, він може завантажити версію спеціально для персональних комп'ютерів з підтримкою сенсорного дисплею проте в користувача залишиться змога керувати середовищем з клавіатури, але на екрані будуть відображатись інші елементи керування. Версія для мобільних пристроїв на базі android, передбачає собою зручний інтерфейс, можливість відображати заряд батареї та годину для планування часу, що на жаль може погано вплинути на уважність користувачів, але дозволить ефективніше розподіляти час, оскільки виконувати завдання можна буде в різних місцях та моментах доби.

Оцінювання відбуватиметься комбінованим підходом, додаток перевірятиме час виконання, після чого отриманий час відображатиметься як краще проходження рівня, також може бути взаємодія з викладачем, для отримання більш точної оцінки,

1.2. Існуючі мобільні додатки

Навчання за допомогою мобільних девайсів вже давно не рідкість, зараз існує безліч різних додатків для навчання іноземних мов, одним з самих цікавих та популярних застосунків є : "Duolingo: уроки іноземної мови". На рисунку 1.1 зображено одне з завдань програми.

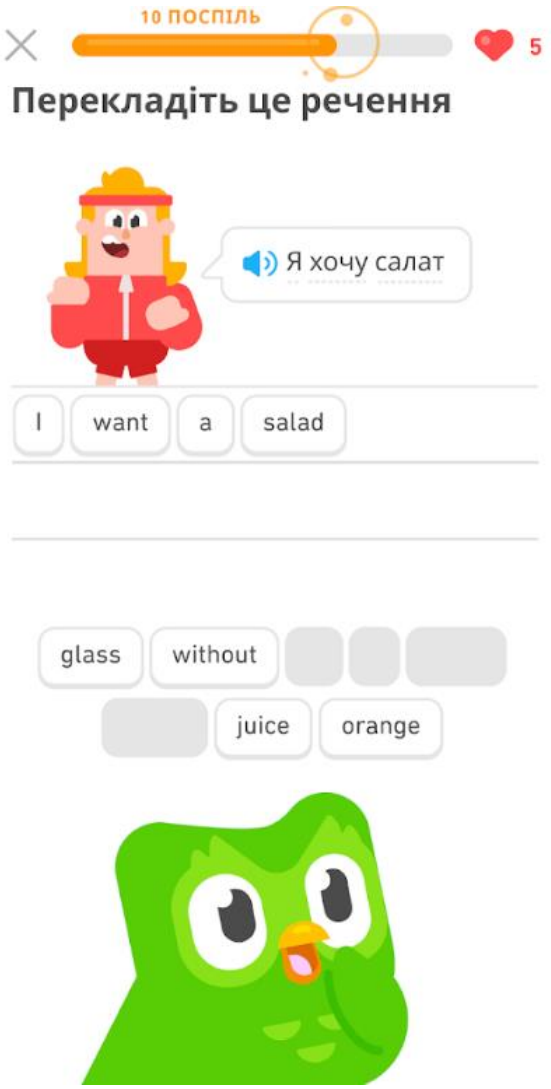

Рисунок 1.1 – Завдання “Duolingo”

Програми для навчання програмуванню можна розділити на три рівні, додатки, що підходять початківцям, для більш досвідчених та програми для практики.

Encode: Learn to Code – програма розроблена Upskew Pty. Ltd. та пропонує користувачеві короткі інтерактивні уроки з JavaScript, Python чи з веб-розробкою. Вона добре підходить для вивчення основ у вільний час. Перевагами даної програм є офлайн доступ до завдань, чіткі пояснення та більш зосереджений фокус на практиці. На рисунку 1.2 зображено завдання Encode.

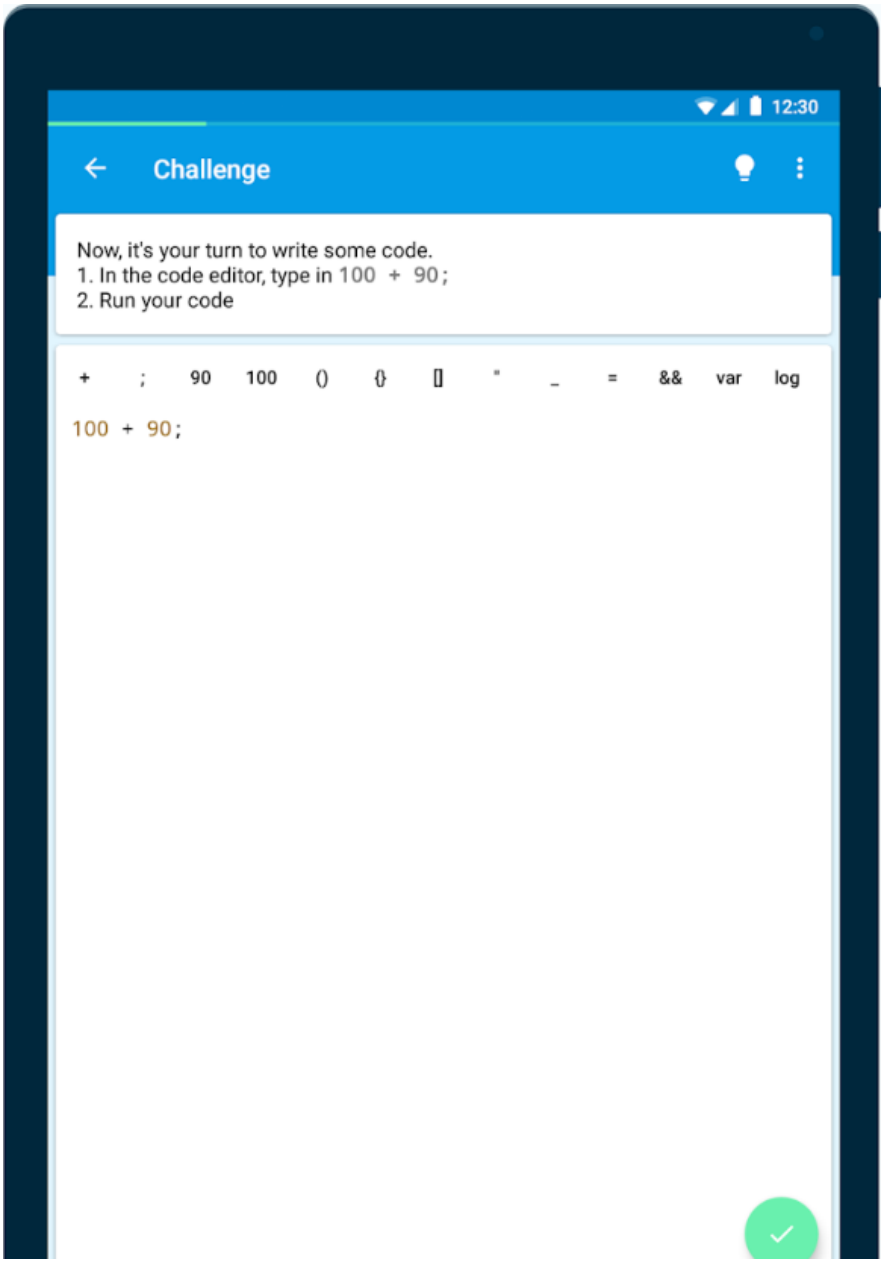

Рисунок 1.2 – Завдання Encode

Одним з рішень для більш досвідчених користувачів є Sololearn.

Sololearn: Навчіться кодити – додаток має велику базу курсів різних мов програмування (Java, C++, Python, Swift та інші). Програма включає в себе можливість спілкуватись з іншими студентами, інформативні статті та різні завдання. Одне з завдань по редагуванню коду, зображено на рисунку 1.3.

Рисунок 1.3 - Редагування коду в Sololearn

Ще одним з рішень для досвідчених користувачів є Enki: Learn to code. Додаток допомагає покращити навички програмування через персоналізовані плани навчання, короткі щоденні завдання. До переваг можна віднести: Персоналізацію, адаптивність, фокус на регулярній практиці. Програма пропонує різні завдання для навчання, одне з таких завдань зображено на рисунку 1.4.

Рисунок 1.4 – Тестове завдання Enki

З програм для практики, покращення знань архітектури, тощо можна розглянути наступні додатки LeetCode, HackerRank та CodeGym.

LeetCode – має велику кількість завдань з програмування, велика кількість даних завдань часто використовується для підготування до співбесід. Інша програма HackerRank пропонує дещо інші завдання, програмування алгоритмів, структур даних та AI. Вагомим плюсом програми є можливість змагання з іншими користувачами для кращого засвоєння навичок. CodeGym додаток, що фокусується на Java, має базу з більше 1200 різних завдань, редактор коду та віртуальний ментор, який надає корисні підказки та перевіряє код. Даний додаток потроху перетворює навчання в захоплюючу гру, з сюжетом та різними персонажами. Меню завдань CodeGym зображено на рисунку 1.5.

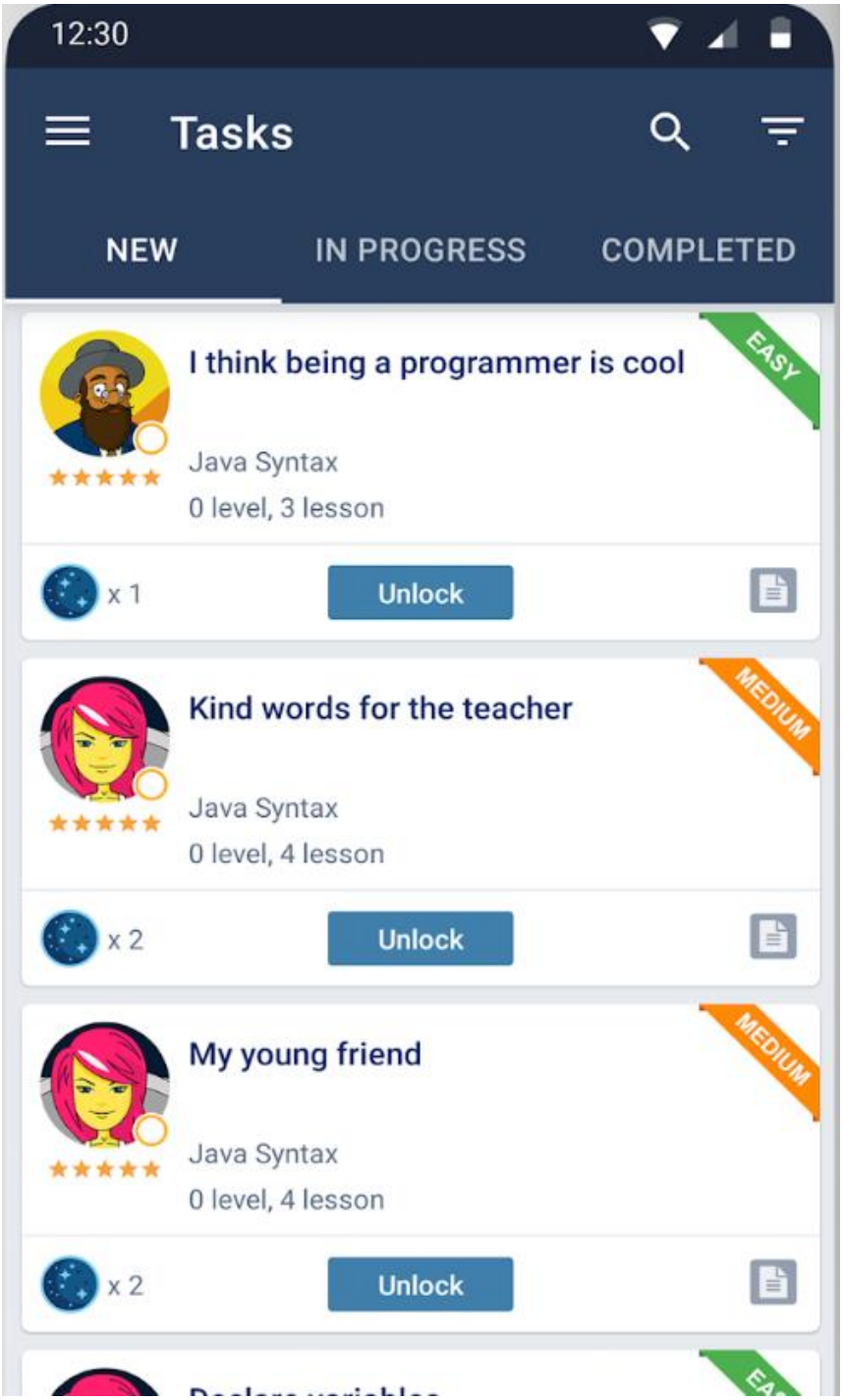

Рисунок 1.5 – Меню завдань CodeGym

1.3. Технологічна основа проекту

Розробка інтерактивних додатків, сьогодні спирається на потужні інструменти. Завдяки цьому створюються нові можливості реалізувати в життя найрізноманітніші ідеї. Основні технологічні компоненти стають фундаментом проекту. На сьогоднішній день вони допомагають взаємодіяти з формою об’єктів, дозволяють краще налаштувати взаємодію у віртуальному просторі налаштовуючи їх фізичні властивості. Деякі механізми дозволяють ефективно створювати множинні копії об’єктів, наповнювати віртуальний світ динамікою. Розглянемо декілька ключових компонентів:

- Collider: Компонент який надає об’єктам фізичну форму, завдяки чому об’єкти можуть взяємодіяти між собою. В основному вони реалізовуються для виявлення зіткнень, створення подій та реалізації фізичних взаємодій, підняття, ковзання чи падіння, тощо..
- Rigidbody: Даний компонент надає об’єкту фізичні властивості, масу, гравітацію, тощо.. Завдяки даному компоненту, об’єкти можуть реагувати на імпульси, силу, та рухатись відповідно законів фізики. Компонент особливо корисний, для створення взаємодії з оточенням.
- Prefab: Створюється для використання множинних копій об’єкта, це шаблон об’єкта. Завдяки ним можна економити час, ресурси. Їх можна налаштовувати окремо один від одного. Використовуючи дані заготовки, можна швидко створювати та налаштовувати рівні чи інші елементи.
- Animator: Компонент який дає змогу створювати анімації. Аніматор використовує анімаційні кліпи та переходи між ними для створення плавних та реалістичних рухів. Можна створювати такі анімації як: біг, ходьба чи атака. Аніматор також надає можливість взаємодіяти з подіями в додатку, наприклад до аніматора можна підключити запуск функції, тощо.
- AudioSource: Компонент завдяки якому можна додати звукові ефекти, музику, тощо в додаток. Він підтримує дуже багато різних форматів, та дозволяє налаштувати звук під потреби.

Це лише невелика частина основних інструментів різних середовищ. Кожен з даних компонентів виконує сою унікальну функцію, дозволяє створювати різноманітні механіки чи візуальні ефекти. Комбінуючи різні компоненти, комбінуючи різні компоненти, можна створювати унікальні та захоплюючи механіки.

1.4. Постановка задачі

Розглянемо функціонал що пропонуються для мобільного додатку.

Зручне середовище для засвоєння навичок кодування, система сповіщення при виявленні помилок. Також пропонується варіювання параметрів квестів, а саме мова програмування, кількість завдань чи обмеження за часом. Використання різноманітних типів завдань, створення коду, редагування коду, пошук елементів коду, тощо.

Цільова аудиторія: Додаток має бути доступний для різних груп користувачів, в основному націлений на молоде покоління, дітей шкільного віку.

Основні функції: Застосунок повинний виконувати наступні моменти розбивати складні концепції програмування на невеликі легко засвоювані кроки, можливість одразу приступити до засвоєння отриманих знань виконуючи практичні завдання, видавати спеціальні винагороди та за допомогою візуальнх елементів занурити користувача в світ.

Безпека: Додаток має забезпечити захист користувачів, обов’язкова модерація контенту для щоб запобігти публікації неприйнятного чи образливого контенту, фільтрація контенту, гравець має бути захищеним від такого як ненормативна лексика, спам або фішинг. Політика конфіденційності повинна чітко пояснювати користувачам, як додаток збирає використовує та захищає дані.

РОЗДІЛ 2. РОЗРОБЛЕННЯ КОНЦЕПЦІЇ МОБІЛЬНОГО ДОДАТКУ

2.1. Аналіз ринку і потреб користувачів

Ринок мобільного навчання демонструє динамічний ріст, який був утворений через значне поширення смартфонів, планшетів та швидко якісного мобільного інтернету. Як повідомляє ресурс Lynda.com, таке навчання підвищує продуктивність на 45%. Схиляючись на статистику від Learning Guild Research яка вказує на те, що для учнів є важливим доступ до навчальний матеріалів у смартфонах, тощо. За допомогою доступу до навчальних матеріалів, користувачі можуть власноруч складати для себе графік навчання, чи виконувати його у вільний час будь-де та будь-коли: у кафе, під час перерв, в парку на прогулянці, в транспорті, тощо. Більше 6,2 млрд людей отримують доступ до мережі інтернет завдяки мобільним пристроям. Такі оцінки опублікували Huawei, за її словами саме така кількість людей зможе щоденно користуватись мобільним інтернетом. Дані цифри не можуть не вплинути на традиційні методи навчання, адже технології стрімко розвиваються.

Ключовими тенденціями є:

Зростання ринку: Мобільне навчання стрімко зростає. Фактори, що сприяють зростанню мобільного навчання насамперед є ріст користувачів смартфонів чи інших гаджетів. На графіку, що зображений на рисунку 2.1 зображена динаміка ринку мобільного навчання.

Рисунок 2.1 – Динаміка мобільного навчання

Можна зробити наступні висновки, а саме, що зростання мобільно навчання є очевидним. Дана тенденція, ймовірно збережеться на найближчі роки оскільки зараз значне поширення смартфонів на ринку, розвиток технологій через що росте й попит на гнучкі рішення.

Мікронавчання: це насамперед короткі але цілеспрямовані модулі, що легко засвоїти та не потребують значного витрачання часу.

Персоналізоване навчання: є адаптацією до потреб кожного учня, завдяки чому отримується підвищення ефективності, адаптація навчально контенту під певні побажання користувачів.

Гейміфікація: ігрові елементи в навчанні, підвищують зацікавлення та віддачу користувачів.

Останнім часом в українській дистанційній освіті відбувся стрімкий ріст попиту. У міністерстві освіти і науки повідомили, що кінець та початок 2022 та 2023 років в Україні навчалась більша кількість дітей дистанційно, а саме 36,4% що орієнтовно дорівнює 1,5 млн дітей. В очному форматі навчалось 33,3% приблизно 1,3 млн дітей та змішаному форматі було 30,3% що дорівнює приблизній кількості в 1,2 млн дітей. Даний розподіл зображено на рисунку 2.2 – розподіл школярів.

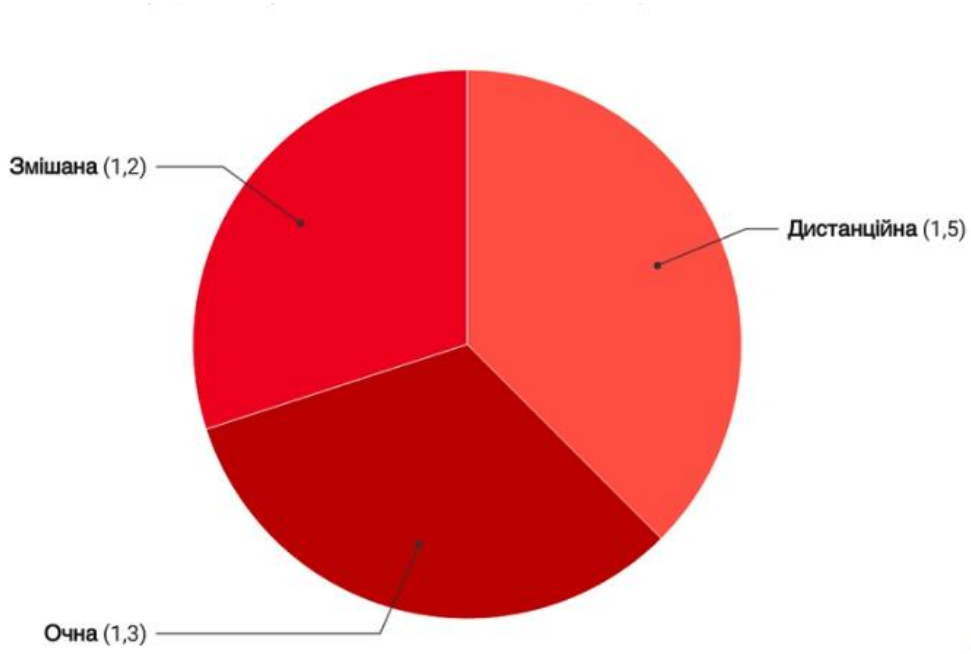

Рис. 2.2 - Розподіл школярів України у 2023 році за формами навчання (млн. осіб)

Не можна не зазначити, що не велика кількість школярів отримує середню освіту в приватних школах орієнтовно десята частина загальної кількості.

Також варто зазначити, що до ковіду та війни, в дистанційному форматі навчалось 2% школярів, але як вже помічалось раніше, з 2020 року відбувся вагомий приріст дистанційного навчання. Дистанційне навчання розвивається не лише в Україні, у різних розвинених країнах, до ковіду вже навчалось близько 10-12% учнів, що є в п’ять разів більше за значення того часу в Україні. Після ковіду даний показник збільшився до 20-25%, показник залишається актуальним і на сьогоднішній день.

Враховуючи використання онлайн навчання в Україні вже не перший рік, ринок онлайн освіти ще на стадії формування. Однією з перших проявів такої освіти стали курси по вивченню іноземних мов, вони досить швидко набули популярності оскільки пропонували за невеликий проміжок часу отримати базу для подальшого навчання, такі додатки можна було знайти та завантажити вже близько 10 років тому. На рисунку 2.3 зображений графік попиту на навчання онлайн за формами.

Рисунок 2.3 - графік попиту на навчання онлайн за формами.

Найбільшу частку ринку отримує вивчення іноземних мов – 43,0%. Це звісно не дивно оскільки вивчення іноземних мов таким методом було доступно вже близько 10 років тому. Значної популярності набуло й вивчення чи

підготовка до екзаменів за допомогою мобільних пристроїв – 24,0%. Таким методом вже давно користуються медичні заклади освіти для підготовки своїх студентів до екзаменів, а саме до КРОК – єдиний державний кваліфікаційний іспит для студентів. Також великим попитом користуються різні автошколи оскільки за допомогою таких методів навчання, учні можуть швидко підготуватись, повторити для засвоєння вже пройдений матеріал, даною програмою також користується й вже досвідчені водії.

Ринок мобільних додатків почав стрімко зростати ще з 2010 року, проте епоха ковіду та останні роки ізоляції позитивно позначились на тенденціях розвитку даної сфери. Ринок вимагав термінових змін та адаптації, багато потреб споживачів покривались за рахунок додатків на смартфон. Ситуація яка отворилась можна охарактеризувати наступними пунктами:

- Нестабільність ринку:
  - Постійні зміни на ринку змушують розробників приймати складні рішення щодо розробки.
  - Непередбачуваність ринку також ускладнює досягнення поставлених цілей адже це призводиить до невиизначеності результатів.
- Користувачі стали більш вимогливими:
  - Люди почали обережніше ставитись до витрат, фінансових, часових, тощо..
  - Користувачі очікують швидких результатів за короткий проміжок часу, хочуть оперативно використовувати свої ресурси.
- Потрібні нові підходи до аналізу:
  - Старі методи аналізу ринку вже застарілі, вони не можуть врахувати швидкі зміни.
  - Виникла необхідність використовувати дані в реальному часі для отримання результату, а не прогнозу.

Отже, ситуація на ринку стала динамічною та складною, якщо порівнювати її з минулими роками. Для успіху потрібно вміти швидко

адаптуватись до змін, ефективно використовувати ресурс на орієнтуватись на потреби користувачів.

Під час огляду минулих років, можемо виділити наступне, що кількість нових додатків стрімко зросла хоча й не рівномірно. Рисунок 2.4 – Кількість нових додатків

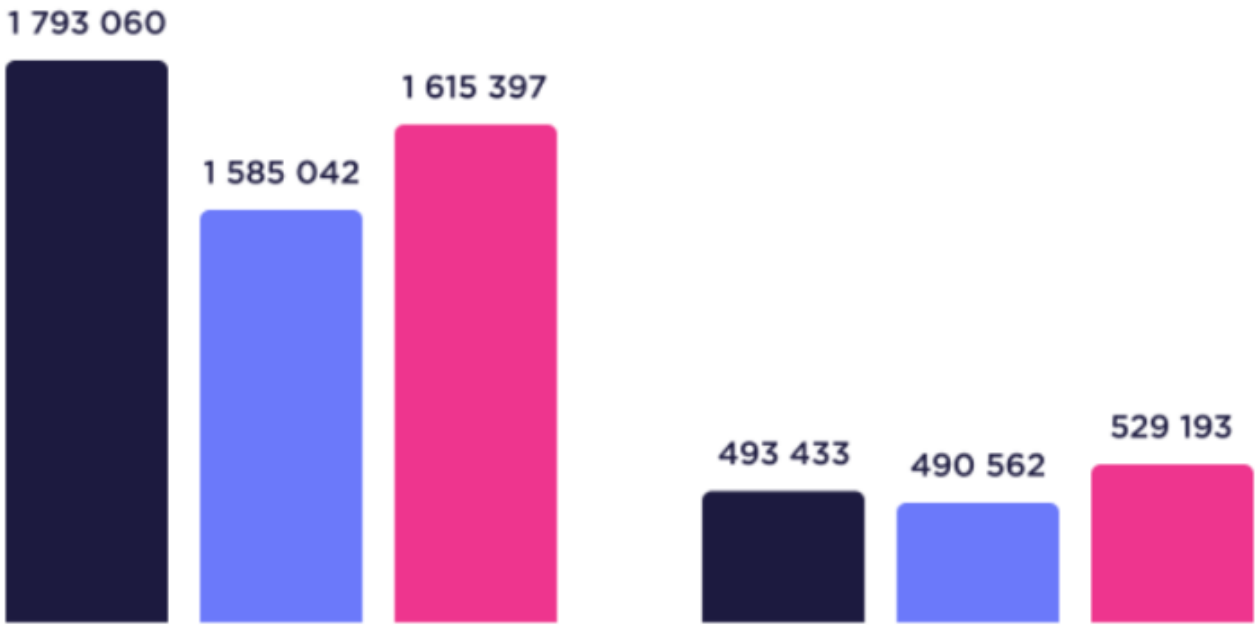

Рисунок 2.4 – Кількість нових додатків

Варто також розглянути показники щодо завантажень, дані показники показали стрімкий ріст протягом усіх виділених трьох років, можна побачити на рисунку 2.5.

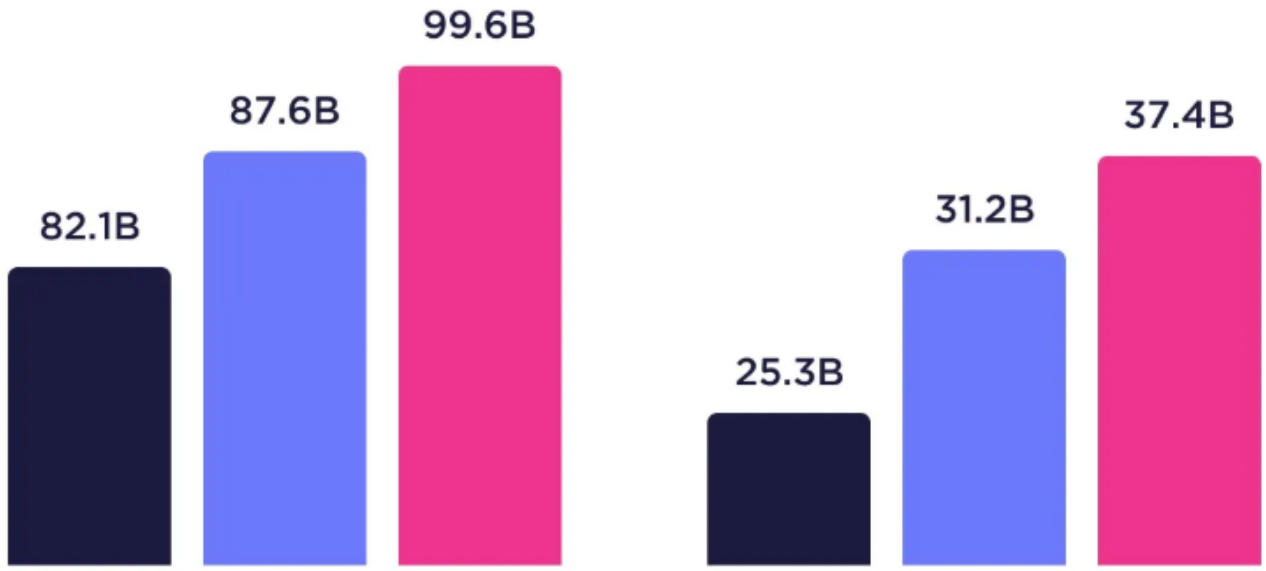

Рисунок 2.5 – Показники завантажень

Отже поточний рік є стабілізаційним до нормальної поведінки користувачів. Такі категорії як бізнес чи подорожі та кіно на разі здобули стабільність та свою аудиторію, проте сфера розваг, навчання та саморозвитку демонструє ріст аудиторії.

Мобільні пристрої доступніші доступ до інтернету відкриває безліч можливостей. Школярі та студенти більшість свого часу проводять за мобільним телефонами враховуючи це потрібно забезпечити корисне цікаве та якісне використання цього часу.

На рисунку 2.6 – зображена спрощена схема взаємодії між застосунком та спеціалістами.

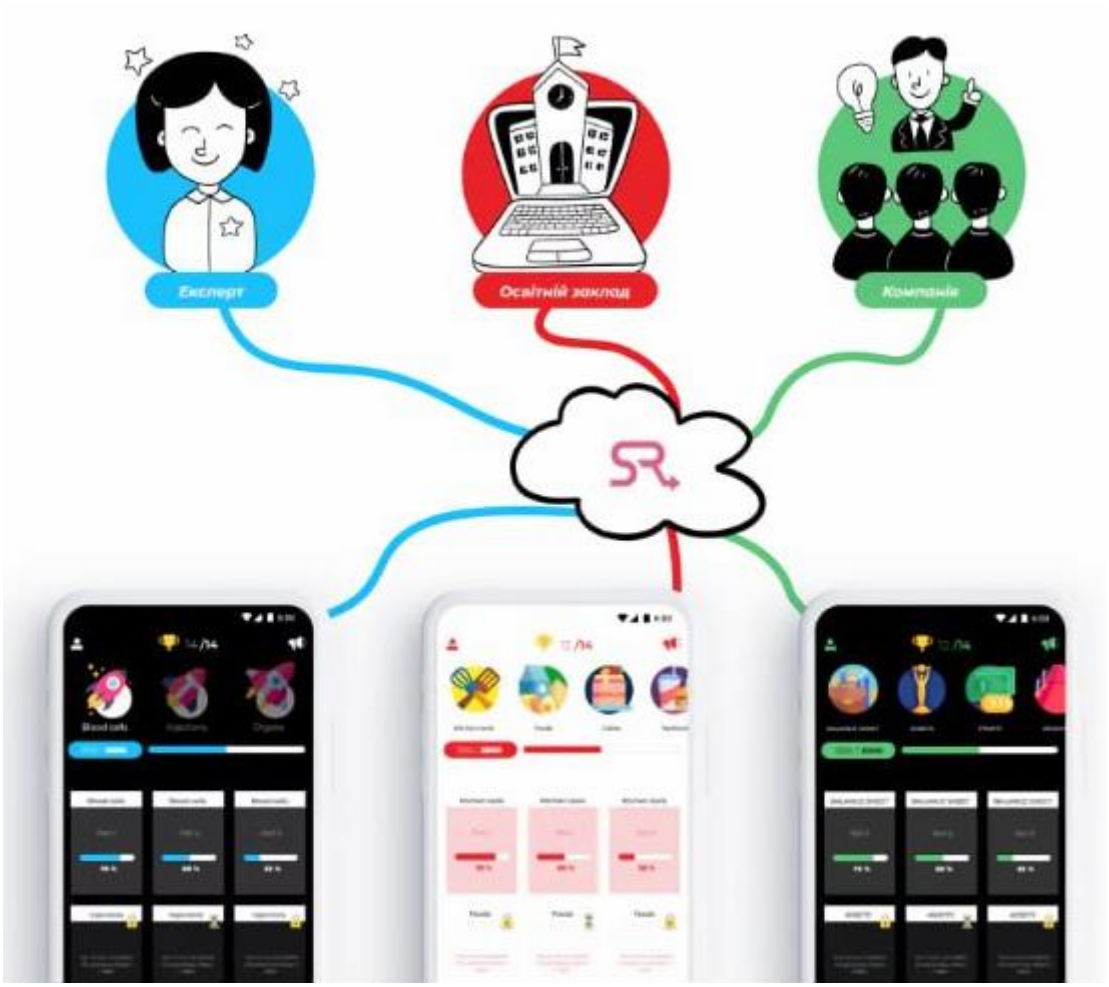

Рисунок 2.6 – Схема взаємодії спеціалістів та застосунку

До мобільного навчання потрібно долучати кваліфікованих працівників вищої освіт, для взаємодії з розробниками програмного забезпечення, оскільки реалізація мобільного навчання вимагає тісної співпраці між педагогами та IT спеціалістам. Це зумовлено тим, що викладачі знають як привернути увагу, вміють працювати з дітьми та більш винахідливо пояснювати навчальний матеріал. Завдяки такій співпраці, рівень навчання може вийти на новий рівень, де сам користувач буде надзвичайно зацікавлений у вивченні навчального матеріалу.

2.2. Проектування функціоналу мобільного додатку

Метою створення додатку є навчання програмуванню, зробити вивчення більш доступним, кодування цікавішим та зрозумілим для широкої аудиторії. Розглянемо ключові цілі:

- Знизити бар’єр для входу: Багато людей вважають програмування досить складним та недоступним.
- Забезпечити зручність: Навчання через мобільний додаток, має надати змогу користувачам навчатись, засвоювати знання у власному темпі.
- Покращення мотивації: Завдяки гейміфікації додатку, школярі, студенти та інші користувачі поринають у більш захопливий курс навчання.
- Розвинути практичні навички: Завдяки практичним завданням, користувач має краще засвоїти отриманні знання.

Отже мобільний додаток повинний бути доступним для користувачів як вже з досвідом програмування так і для нових користувачів, проте варто зауважити, що додаток позиціонується як закріплення вже пройденого матеріалу в навчальному закладі. Завдання мають бути спроектовані відповідно до знань програмуванню користувачів. Складні концепції мають бути розкладені на прості та доступні кроки з окремими поясненнями. Завдяки віртуалізації, анімації та діаграми для наученого пояснення матеріалу, створенням діалогів із обговоренню певної проблеми, гравець буде мати краще занурення у світ програмування.

Відкриття нових рівнів під час виконання завдань. Протягом користуванням додатком, користувач відкриватиме нові рівні з більшою складністю разом із складністю рівнів з сторони коду, складнішою буде кожна наступна локація в якій користувачеві потрібно буде розгадувати головоломки та паралельно проявляти вже отримані навички програмування для подолання певних перешкод чи розгадування головоломки.

Також мобільний додаток дозволить зв’язатись з викладачем або розробником в розі виникнення труднощів з розумінням, від самої суті завдання до елементів коду, тощо. Взаємодія з персонажами. Гравець може взаємодіяти з середовищем, з різноманітними персонажами, які будуть давати підказки, пояснення чи корисні ідеї для проходження перешкод, також завдяки ним можна буде поцікавитись відмінністю різних мов програмування, для чого вони використовуються, тощо.

Алгоритм розробки наступний:

1. Створення основного персонажа:
    - Створення персонажа, налаштування руху, створення та налаштування анімацій, зіткнення з об’єктами.
    - Налаштування звукових ефектів персонажа.
2. Побудова рівнів:
    - Створення рівнів використовуючи різні платформи, перешкоди, фони.
    - Налаштування візуального стилю та музичного супровіду.
3. Додавання ворогів:
    - Створення противників, налаштування AI для противників.
    - Налаштування звукових ефектів противників, анімації та взаємодію з гравцем.
4. Додавання головоломок:
    - Створення завдань для взаємодії з середовищем, продумування геймдизайну.
5. Впровадження механік:
    - Реалізація основних механік, таких як пересування, взаємодія з об’єктами, збір предметів, крафт коду.
6. Тестування
    - Ретельно протестований додаток на різних пристроях, виправлення помилок.

Інтерфейс програми повинний бути інтуїтивно зрозумілим. Класичного головного меню в додатку не буде, натомість головне меню буде реалізоване як відкритий світ в якому можна буде переглянути трофеї здобуті протягом навчання, зв’язатись з розробником чи викладачем. Дизайн навколишнього середовища буде в стилі середньовіччя, в 2D.

2.3. Вибір платформи і середовища розробки

Вибір платформи та середовища розробки є важливим етапом, який вплине на процес розробки додатку. Розберемо декілька популярних рішень для розробки які будуть хорошим рішенням для початківцям та досвідченим розробникам. Розглянемо наступні середовища Godot, GameMaker Studio 2, Unity та Unreal Engine.

Godot – це потужний движок, він є досить зручним, має відкритий вихідний код. Середовище ідеально підходить для створення 2D та 3D відеоігор. Пропонує великий набір інструментів розробки. Програма має зручний редакторр рівнів, анімації та вбудовану мову програмування GDScript. Дана мова є доступною та не складною для вивчення. Godot має зручний інтерфейс (Рисунок 2.7).

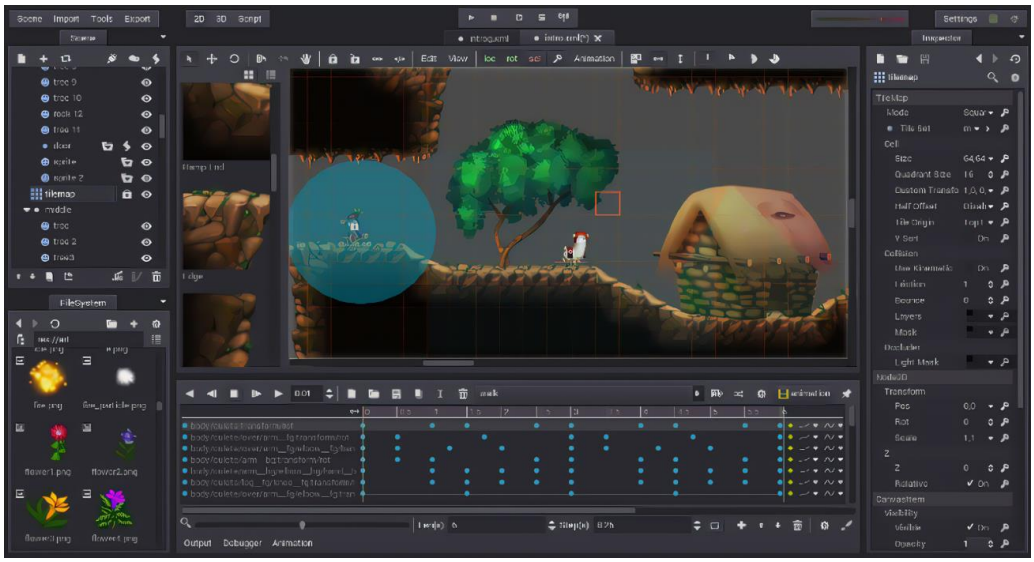

Рисунок 2.7 – Інтерфейс Godot

Основними мінусами середовища є як не дивно вбудована мова розробки GDScript. Дана мова програмування має досить багато прихльнииків та є простою у вивчені, проте вона може мати обмежені можливості в порівнянні з

такими мовами програмування як C# чи C++. Потрібно відмітити, що середовище підтримує й інші мови програмування, проте GDScript є основною мовою движка, і велика кількість ресурсів саме на ній. Невелика кількість ресурсів та недоліки відкритого коду. Спільнота Godot з кожним днем зростає, проте вона все є досить маленькою при порівнянні з Unity чи Unreal Engine. Це значить, що при потребі буде дуже складно знайти потрібну підтримку чи інформацію для реалізації задуманих напрямків. Відкритий код програми є вагомим плюсом, проте він має й багато недоліків, наприклад при виникненні технічних помилок буде досить складно знайти рішення від офіційної команди розробника, стабільність окремих функцій можуть бути не такими передбачуваними.

GameMaker Studio 2 – Це досить популярне середовище розробки, в основному націлене на 2D додатки, також воно хвалиться своїм простим та зрозумілим інтерфейсом (Рисунок 2.8).

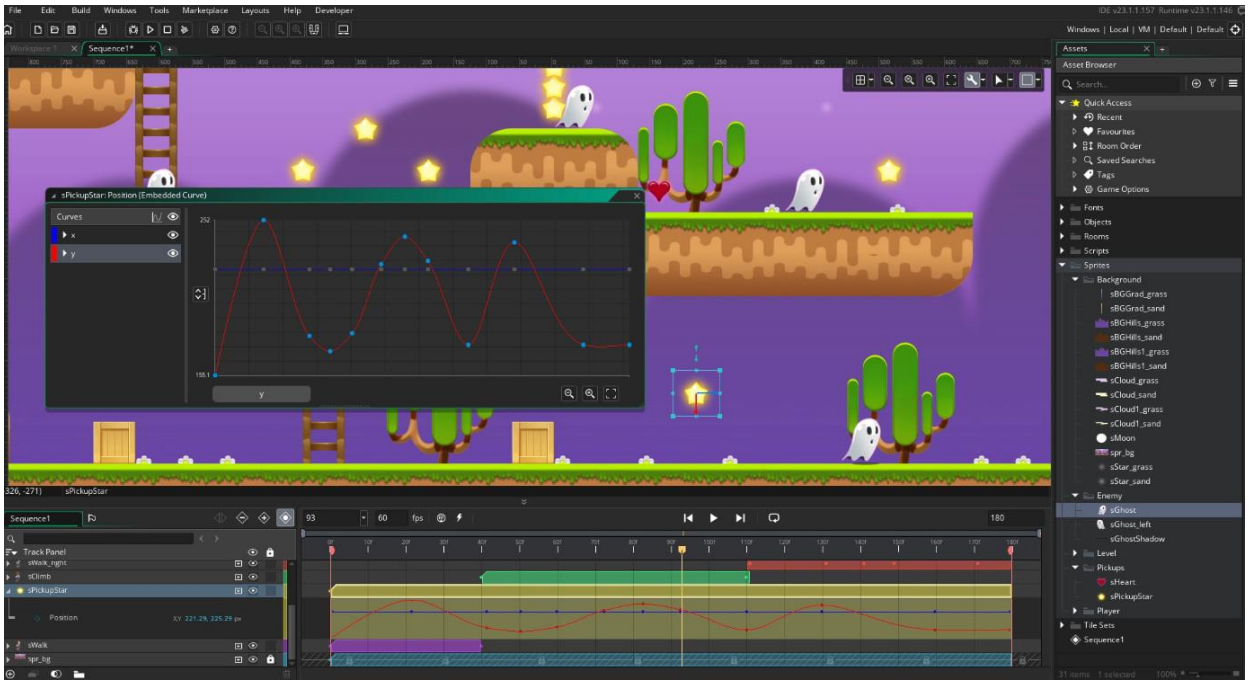

Рисунок 2.8 – інтерфейс GameMaker 2

Для розробки в середовищі використовується візуальне програмування та власна мова програмування GML (Game Maker Language).

Вагомими перевагами середовища є легкість у вивченні завдяки інтуїтивно зрозумілому інтерфейсу та сстемі “drag-and-drop”, що робить середовище досить

комфортним та легким для початківців. Створювати алгоритми програми за допомогою візуального програмування доступно користувачам без глибоких знань в програмуванні, що є хорошим рішенням для вивченню початківцям. Також до переваг варто віднести кросплатформенність, завдяки якій можна реалізувати проект на різних платформах: Windows, macOS, Android та інші.. Середовище також має вбудовані інструменти як от редактор рівнів, анімацій, спрайтів, тощо.

Недоліками GameMaker Studio 2 є GML (Game Maker Language), зважаючи на те, що вона є досить простою вона також має обмежені можливості порівняно з іншими мовами програмування, через що протягом роботи можуть виникти труднощі при створенні важких застосунків. Також середовище може мати проблеми з продуктивністю в складних або великих проектах. Для комерційної розробки GameMaker Studio 2 вимагає платну цифрову ліцензію.

Unreal Engine – надзвичайно потужний ігровий движок, розробниками якого є Epic Games. Середовище використовується для створення різноманітних ігор, від мобільних 2D аркад до надзвичайно складних симуляторів, шутерів від першої особи чи для пригоднецьких ігор які будуть ще довго вражати своєю графікою.

Ключовими перевагами Unreal Engine є фотореалістична графіка яка створюється завдяки просунутому рендеру. Середовище має великий набір інструментів, завдяки чому розробник отримує широкий спектр можливостей для креативності. Велика підтримка різних платформ, додатки легко можна підлаштувати під консолі, Windows, macOS тощо… Для розробки використовується мова програмування C++, завдяки чому розробники отримують більшу гнучкість додатку.

Також Unreal Engine використовує C++ та Blueprint (Рисунок 2.9), візуальне

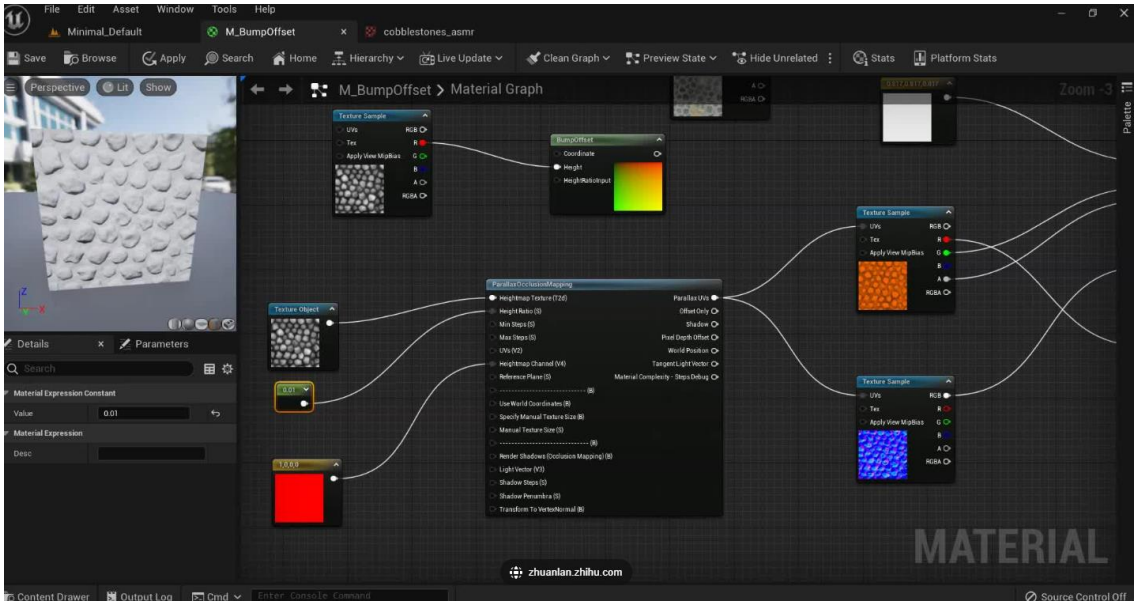

Рисунок 2.9 – Використання Blueprint

програмування, що дозволяє створювати алгоритми відеоігор без програмування. Велике товариство, спільнота та безліч ресурсів які допоможуть при вивчені чи створенні нових проектів.

Щодо недоліків варто віднести складність та високі системні вимоги. Середовище може бути складним у вивченні для початківців та вимагає досить потужного комп’ютера для розробки.

Unity – одне з найпопулярніший платформ розробки у світі. Дозволяє створювати 3D, 2D, VR та AR додатки чи інтерактивний контент. Ключовими особливостями є кросплатформенність, інструменти для розробки які надають широкий спектр можливостей таких як: редактор рівнів, систему частинок, анімації, хороший фізичний движок, тощо. Вагомими перевагами є легкість у вивчені, Unity використовує мову програмування C# це є популярна та потужна мова. Велика спільнота та ресурси де можна знайти безліч корисної інформації, допомоги чи активів.

Зручний інтерфейс (Рисунок 2.10), який можна налаштувати персонально під користувача забрати все лишнє або додати щось своє. Asset Store – місце де можна знайти багато корисних безкоштовних та платних плагінів, корисні актив, текстури чи звукові ефекти, також там можна публікувати свої роботи.

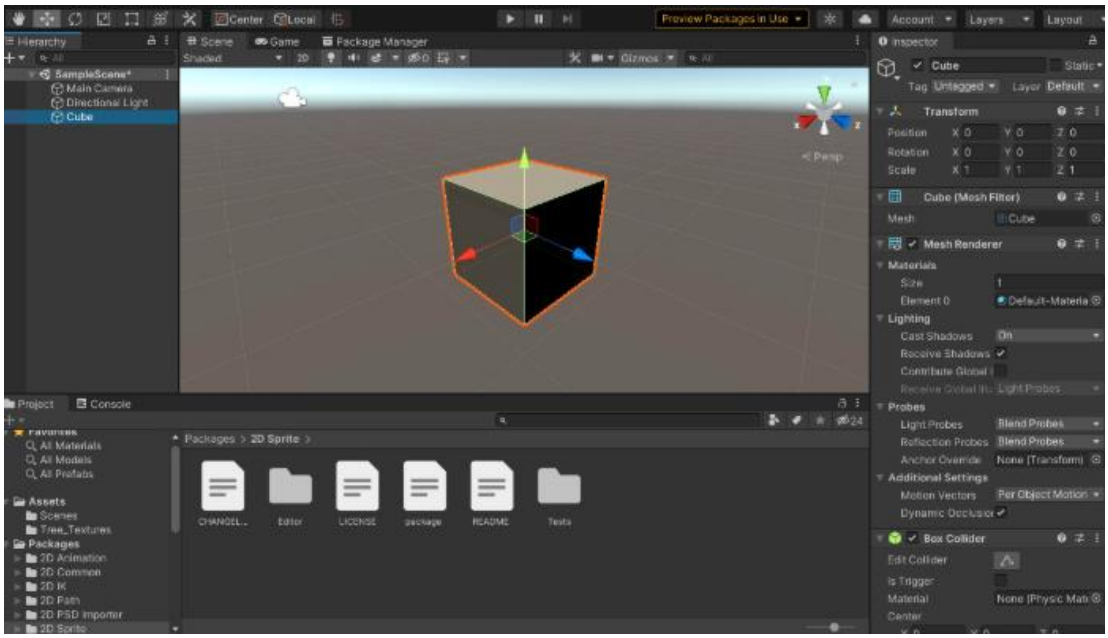

Рисунок 2.10 – Інтерфейс програми

До недоліків середовища варто віднести наступні аспекти, такі як продуктивність, при неправильній оптимізації, у великих проектах можуть бути проблеми з продуктивністю.

Вибір движка це завжди компроміс, проте для різних розробників, в тому числі початківців Unity має великий ряд переваг, які роблять його чудовим вибором. Простота в вивченні движка, інтуїтивно зрозумілий інтерфейс, потужна мова програмування яка забезпечить великий контроль над додатком та Asset Store який допоможе в пошуках готової бази, тощо. Хочеться відмітити велику спільноту розробників через що будь які труднощі можна буде вирішити не витрачаючи дні на пошуки інформації.

РОЗДІЛ 3. ТЕХНІЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ЗАСВОЄННЯ НАВИЧОК З ПРОГРАМУВАННЯ

3.1. Архітектура і структура додатку

Гейміфікація навчання дозволяє реалізовувати безліч можливостей, створення захоплюючих та інтерактивних навчальних середовищ. Одним з ключових елементів є реалізація різних завдань, які не лише допомагають учневі, не лише перевіряють вже набуті знання, а й виступають стимулятором для активного дослідження. Одна з невід'ємних частин – тести, вони також можуть бути гейміфікованими. Для тестів було виділено декілька тем, таких як: змінні, функції, цикли, масиви та умовні оператори, тощо.. В першому рівні гравець проходить та засвоює базові поняття. Прикладом одного з тестів є : "Що таке змінна в програмуванні", тест має декілька різних відповідей, одна з яких є правильною: "місце для зберігання даних", "математична формула", "тип помилки". Ще одним прикладом тестів є : "Який тип змінної використовується для зберігання цілих чисел", тест має наступні відповіді: "string", "int", "223". Тести запропоновані в різних форматах, також вони є потрібними для подолання рівня, перешкоди, тощо. Завдяки такій системі, проходження тестів ігровий процес стає процесом оцінювання. Проте тести хоч є важливим інструментом оцінювання, вони часто мають асоціацію з нудьгою та стресом, це відбувається з декількох причин:

- Одноманітність: Тести переважно мають однаковий формат, постійте повторення може призвести до втрати інтересу
- Націленість на запам'ятовування: Безліч тестів перевіряють лише здатність запам'ятовувати факти, а не розуміння та застосовування інформації на практиці
- Відсутність зворотного зв'язку та високий рівень стресу: користувачі часто отримують лише кінцевий результат тесту, без детального аналізу, помилок та рекомендацій. Також тести сприймаються як випробування, від результатів яких часто залежить успіх учня. Страх помилки та

отримування поганої оцінки створює напружену атмосферу і погано впливає на емоційний стан.

Для уникнення цих проблем, були реалізовані й інші системи навчання, закріплення матеріалу. Наприклад система квестів в навчанні. Система квестів – це захоплюючий підхід, завдяки системі процес навчання перетворюється на захопливу пригоду. Базується на виконанні учнями послідовності завдань, завдяки який гравець підбирається до своєї мети. Кожне завдання – це крок до вирішення проблеми, отримання нових знань чи розгадки таємниці. Одним з таких квестів є пошуки значення, отримати які можна розгадуючи головоломки чи пошуками чи боротьбою з противниками, в результаті правильного проходження гравець зможе побудувати шлях над перешкодою, тощо. Також в результаті виконання таких квестів гравець отримує змогу активувати зброю, отримати рідкісні обладунки, тощо. Перевагами таких завдань є

- Підвищення мотивації: квести роблять навчання захопливим, завдяки чому стимулює учнів до активної участі.
- Розвиток навичок: такі завдання позитивно сприяють розвитку критичного мислення, співпраці, комунікації та творчості.
- Індивідуальне навчання: завдання можна адаптувати до різних рівнів знань та інтересів учня.

Виконуючи квестові завдання, гравець отримує корисні навички, наприклад в розділі вивчення умовних операторів та циклів, завданням є виконання дій в залежності від умови “Відкрити двері якщо є ключ”, це допоможе зрозуміти принципи роботи умовних операторів (if/else), повторити дію кілька разів (for, while) допоможе пояснити на більш цікаих прикладах принципи роботи таких операторів.

Такі дії гравець має виконати для того, щоб отримати додаткові характеристики персонажа, завдяки яким він зможе здолати лиходіїв, пройти рівні чи розв’язати якусь головоломку.

В проекті використовується процедурний підхід до програмування. Процедурне програмування фокусується на написанні послідовних функцій, які виконуються одна за іншою. Функціональність додатку розподілена по різних скриптах, кожен з яких має набір даних та функцій, для виконання певних задач. Взаємодія між скриптами реалізована за допомогою передачі параметрів. Даний підхід дозволяє розділити код на логічні блоки завдяки чому покращується взаємодія чи його читабельність.

Відмінність від класичних шаблонів програмування: класичні шаблони такі як MVC, MVVM чи Observer, пропонують більш організаційний підхід до структури коду. Дані, інтерфейс та логіка взаємодіють певними правилами.

Наведемо приклад, в нас є скрипт “TakeDamagePlayer.cs” в якому є змінні для відстеження пошкоджень персонажа:

Лістинг 3.1

```
public class TakeDamagePlayer : MonoBehaviour
{
    public static bool takedDamage = false;
    public float refreshHP = 10f;
    public byte damageAlpha = 150;
```

В даному коді є три різних типів даних, а саме bool, float та byte. Змінна для відстеження пошкоджень, для встановлення часу регенерації здоров’я та остання змінна для редагування прозорості персонажа. Вони усі мають тип доступу public, проте сама перша змінна має ключове слово “static”, воно оголошує цю змінну не для окремих об’єктів, а для самого класу, це полегшує взаємодію з нею, але при неправильному використанні можуть відбутись баги, тощо. Наприклад при присвоєнні цієї змінної противникам, при вбивстві одного – помиратимуть всі інші, оскільки змінна буде змінюватись у всіх одночасно. В нашому випадку статичні змінні використовуються лише для персонажа, наприклад для відтворення звуку пошкоджень:

Лістинг 3.2

```
if (takedDamage && restoreColorCoroutine == null)
    {
```

```
        takedDamagePlayerSound = true;
        restoreColorCoroutine = StartCoroutine(RestoreColorOverTime());

    }
```

В цьому коді відбувається перевірка чи отримав гравець пошкодження і чи отримав він його перший раз, після чого в іншій статичній змінній “takedDamagePlayer” присвоюється значення “true”, після цього в коді запускається процес регенерації здоров’я для гравця. Таким чином працює процедурне програмування в додатку.

Розглянемо основні компоненти додатку.

Гравець: керування реалізоване за допомогою сенсорного екрану на екрані зображені клавіші елементи керування (рисунок 3.1), ввід гравця обробляється скритом “MoveCharacter”.

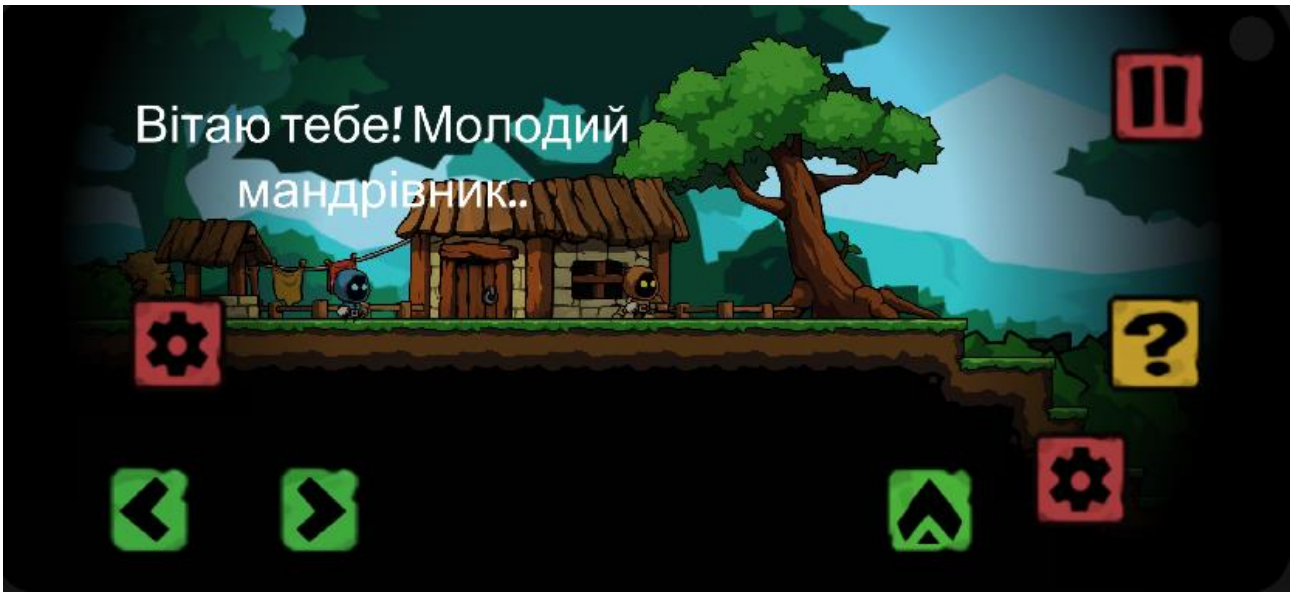

Рисунок 3.1 – Елементи керування

Коли гравець торкається клавіші “ліворуч”, скрипт в додатку це відстежує за допомогою спеціальної функції “Update”. Функція постійно перевіряє стан клавіш та оновлює гру відповідно до дій гравця, дана функція здійснює кількість оновлень які відповідають фактичній частоті оновлення екрану гравця. Після

чого запускається функція “OnPointerDownLeft” для яка перевіряє чи може персонаж рухатись, якщо так то гравець починає рухатись, змінює погляд персонажа залежно від сторони в яку потрібно пересуватись та візуально зменшує клавішу на 20%.

Лістинг 3.3

```
void Update(){
   public void OnPointerDownLeft()
   {
      if(CanMove)
      {
         moveLeft = true;
         character.GetComponent<SpriteRenderer>().flipX = true;
      }
      GameObject.Find("Left").transform.localScale = new Vector3(0.8f, 0.8f, 0.8f);
   }

   public void OnPointerUpLeft()
   {
      moveLeft = false;
      GameObject.Find("Left").transform.localScale = new Vector3(1f, 1f, 1f);
   }
```

Коли гравець відпускає клавішу, персонаж зупиняється та кнопка повертається до своїх попередніх розмірів.

Персонаж має декілька різних анімацій, анімацію спокою, бігу стрибку та три різні анімації атаки (Рисунок 3.2 – Аніматор персонажа).

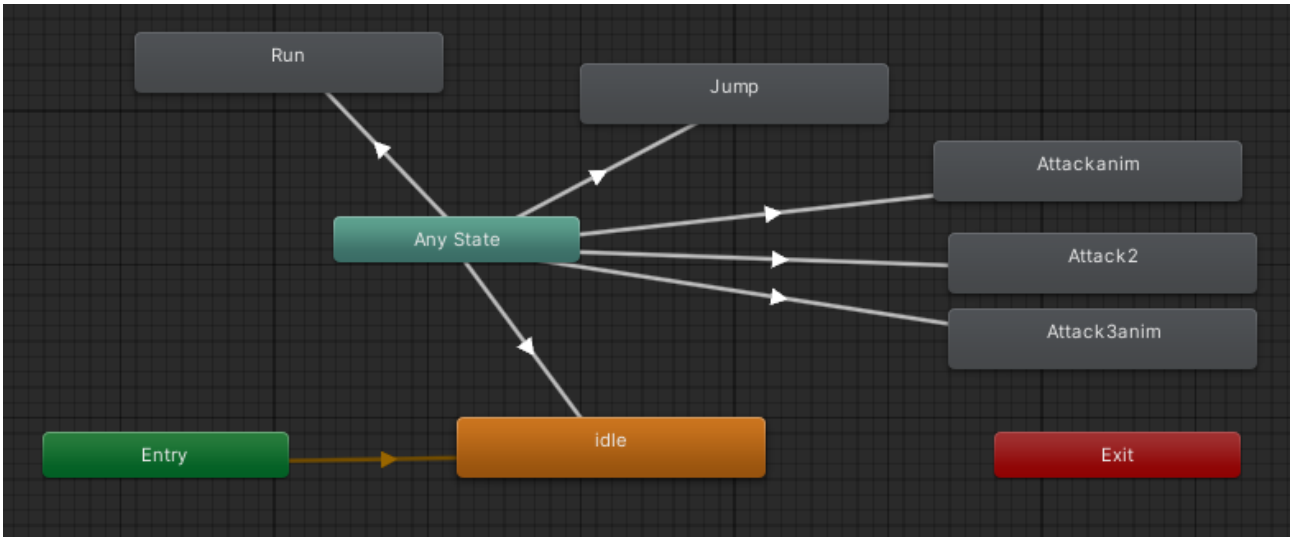

Рисунок 3.2 – Аніматор персонажа

Анімація спокою програється зациклена і якщо гравець стоїть в стані спокою ця анімація працює, усі інші анімації запускаються залежно від керування користувачем, коли гравець тисне кнопку стрибка, запускається потрібна анімація після успішного виконання стрибка.

Лістинг 3.4

```
if (groundedddScript.isGrounded & CanMove )
    {
       if(gravityDown)
       {
           character.GetComponent<Rigidbody2D>().AddForce(new Vector2(0f, jumpForce), ForceMode2D.Impulse);
       State = States.Jump;
       groundedddScript.isGrounded = true;
       PlaySoundJump = true;
       }

    }
    if(!gravityDown){rb.gravityScale = -1;}
```

Коли стрибок відбувся, і гравець торкається землі гравець переходить назад в стан спокою.

Лістинг 3.5

```
if(!moveRight & !moveLeft & groundedddScript.isGrounded ){ State = States.idle;}
else if(!groundedddScript.isGrounded){State = States.Jump;}
```

Також здійснюється перевірка чи гравець здійснює пересування ліворуч або праворуч, якщо такого не відбувається – гравець переходить назад в стан спокою.

Для взаємодії з оточенням в гравця є окремий скрипт “PlayerUse.cs”. Даний скрипт дозволяє гравцеві підіймати певні запрограмовані об’єкти як зброя чи частинки коду для виконання завдань.

Лістинг 3.6

```
public void OnUseePointerDown()
   {
      if (!isObjectPicked)
      {
         if (objectsInRange.Count > 0)
         {
            pickedObject = objectsInRange[0];
            pickedObject.GetComponent<Collider2D>().enabled = false;
            pickedObject.GetComponent<Rigidbody2D>().gravityScale = 0;
            isObjectPicked = true;
            if (pickedObject.tag == "QuestCode")
            {
               pickupDistance = QuestCodeDistance;
               pickupHeight = QuestCodeHeight;
               PlaySoundPickup = true;
```

В даному коді підчас натискання клавіші “Використати” перевіряється чи гравець вже має піднятий об’єкт, якщо такого об’єкта немає відбувається наступна перевірка чи гравець достає до об’єкта, якщо так то для піднятого об’єкта вимикається колайдер та гравітація встановлюється на нуль, також в відповідному полі встановлюється, що об’єкт піднятий дальше визначається що це за об’єкт і яке положення відповідно гравця він має зайняти.

Лістинг 3.7

```
if (overlapCount <= 1)
        {
            pickedObject.GetComponent<Collider2D>().enabled = true;
            pickedObject.GetComponent<Rigidbody2D>().gravityScale = 1;
            pickedObject = null;
            isObjectPicked = false;
        }
        else
        {
            StartCoroutine(ShowPickupError());
        }
```

Проте якщо об’єкт вже піднятий то гравець залишає поточнй об’єкт перевіряючи при тому чи об’єкт не буде перекривати інші колайдерии.

Лістинг 3.8

```
IEnumerator ShowPickupError()
   {
      Text errorText = GameObject.Find("Texty").GetComponent<Text>();
      errorText.CrossFadeAlpha(0f, 0f, false);
      errorText.text = "Я не можу це тут залишити";
      errorText.CrossFadeAlpha(1f, 0.5f, false);
      yield return new WaitForSeconds(3f);
      errorText.CrossFadeAlpha(0f, 0.5f, false); }
```

Якщо таке відбудиться і гравець не зможе залишити об’єкт він отримує відповідне повідомлення, що залишити об’єкт не можливо.

Для виявлення гравця противники використовують сферу (Рисунок 3.3), яка перевіряє чи знаходиться гравець в зоні чи поза зоною.

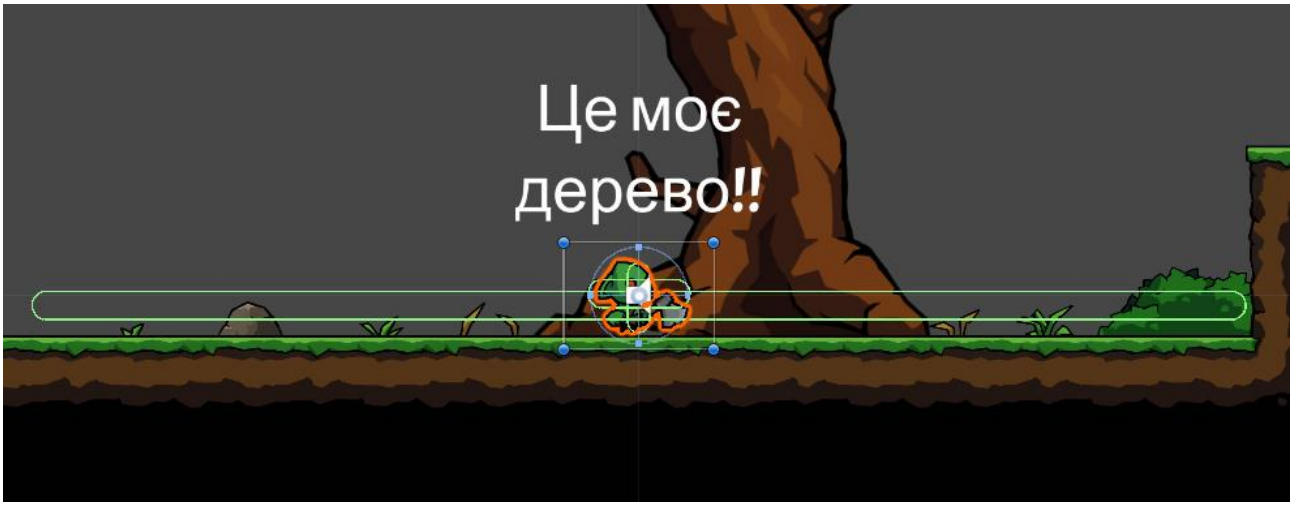

Рисунок 3.3 – Виявлення гравця

Лістинг 3.9

```
private void OnTriggerStay2D(Collider2D other)
{
    if(PlayerDetectAttack.GetComponent<Collider2D>().IsTouchingLayers(Layer
Mask.GetMask("Player")))
    {
      ColliderAttackPlayer = true;
    }
     if(!PlayerDetectAttack.GetComponent<Collider2D>().IsTouchingLayers(Layer
Mask.GetMask("Player")))
    {
      ColliderAttackPlayer = false;
    }}
```

Коли гравець в зоні – противник починає рух до гравця з метою нанесення шкоди. Програмою передбачено можливість гравцем відбивання атаки.

Лістинг 3.10

```
if(CanBlockAttack && swordAttack.isAttacking )
```

```
{
    Debug.Log("Blocked :)");
    State = StatesGoblinAxe.GoblinAxeIdle;
    isAttacking = false;
                    Vector2 knockbackDirection = (transform.position - player.position).normalized;

    knockbackDirection.y += upwardForce / knockbackForce;

    rb.AddForce(knockbackDirection * knockbackForce, ForceMode2D.Impulse);

    EndCanBlock();
}
yield return new WaitForSeconds(swordAttack.attackCooldown);
```

Коли гравець в потрібний час натисне на клавішу атаки, противник відлетить назад та спробує атакувати ще раз якщо в нього буде достатньо здоров’я для даної дії. Вразі якщо гравець не зможе відбити атаку або ухилитись від неї, відбудеться наступна подія:

Лістинг 3.11

```
if(TakeDamagePlayer.takedDamage && ColliderAttackPlayer)
{
    PlayerDead.PlayerDeadh = true;
    Debug.Log("DeadPlayer");
}
if(!TakeDamagePlayer.takedDamage && ColliderAttackPlayer)
{
    TakeDamagePlayer.takedDamage = true;
}
```

Програма перевірить чи отримував гравець пошкодження, якщо так, то гравець загине після чого буде змушений розпочати рівень з початку.

Побудова рівнів відбувалась за допомогою інструмента “Tilemap”. Це зручний інструмент для побудови рівнів, він простий в налаштуванні, для створення рівня потрібно обрати потрібний об’єкт та місце де ми бажаємо його розташувати, після чого простими кліками ми будемо розміщувати об’єкти (Рисунок 3.4 - Tilemap).

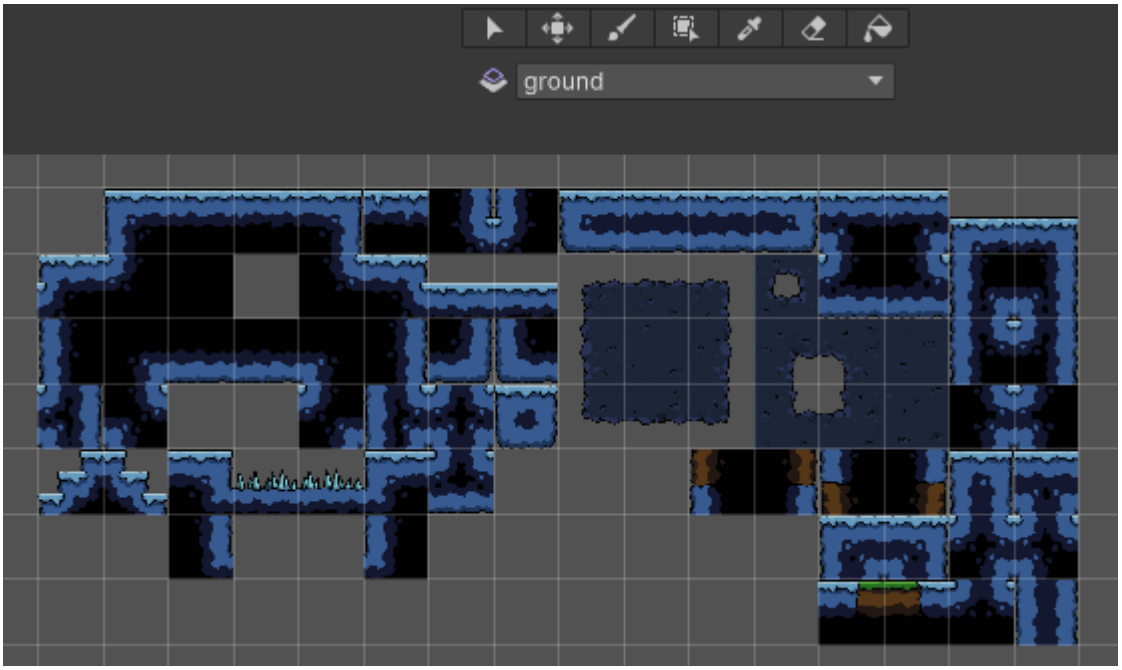

Рисунок 3.4 - Tilemap

3.2. Реалізація ключового функціоналу

Запропонований функціонал навчання має декілька типів квестів. Пошук елементів коду для створення об’єктів, створення нових функцій для подолання перешкод як пошуком об’єктів так і традиційним кодуванням розв’язування головоломок та проходження тестів у різних дружніх чи не дружніх персонажів.

Для початку розглянемо завдання в якому гравцеві потрібно дати відповідь на запитання, щоб отримати шматочок коду для крафту. Для реалізації було створено просте запитання з винагородою або штрафом у розмірі 1 монети, для того, щоб спробувати дати відповідь гравець має мати при собі потрібну кількість монет, кількість монет залежить від запитання (Рисунок 3.5).

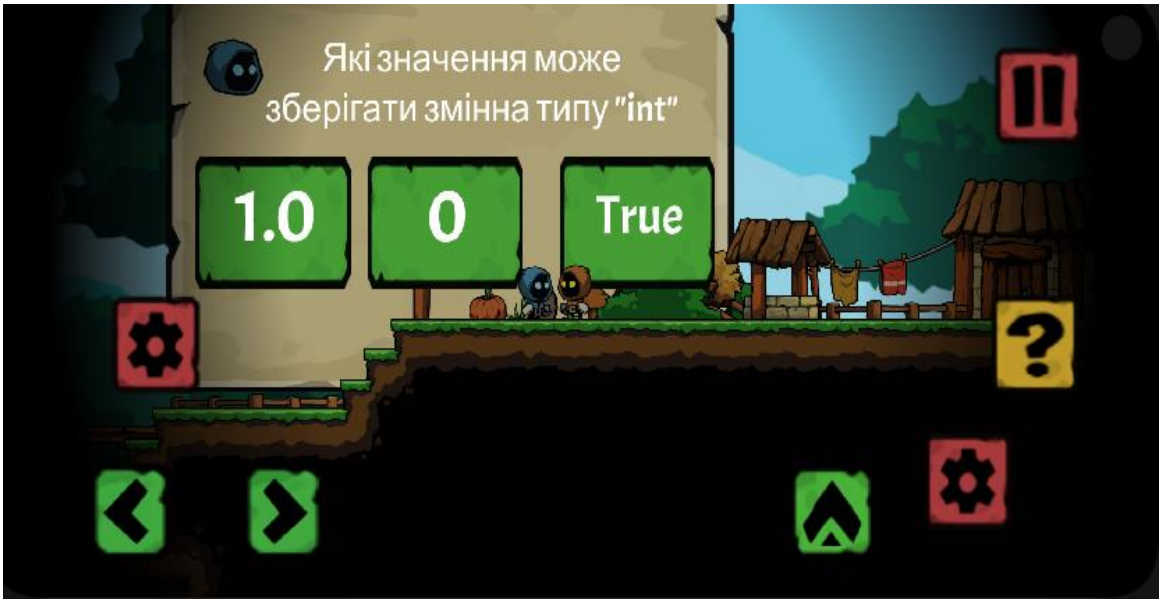

Рисунок 3.5- Тестове запитання

Якщо гравець відповідає вірно, монета залишається та він отримує потрібний йому елемент для проходження рівня

Лістинг 3.12

```
public void OnButtonClick1()
    {
        if (currentCoin > 0)
        {
            Instantiate(prefab, player.position, Quaternion.identity);
        }
        else
        {
           StartCoroutine(NeedCoin());
        }
    }
    public void OnButtonClick2()
    {
        if (currentCoin > 0)
        {
            currentCoin -= valueAsk;
```

```
    }
    else
    {
        StartCoroutine(NeedCoin());
    }
}
public void OnButtonClick3()
{
    if (currentCoin > 0)
    {
        currentCoin -= valueAsk;
    }
    else
    {
       StartCoroutine(NeedCoin());
    }
}
```

В даному скрипті перевіряється 3 кнопки якщо гравцеві хватає монет, він може дати відповідь на запитання в іншому випадку – гравець отримає повідомлення про нестачу і буде змушений піти виконувати інші завдання чи знищувати противників, щоб отримати монети. Якщо гравець дасть правильну відповідь він отримає потрібний об’єкт або шматочок коду з налаштуваннями для створення нового об’єкту.

Давши правильну відповідь на запитання, гравець отримав координату, яку він залишив біля об’єкта (Рисунок 3.6). Після успішної перевірки об’єкт переміщається на задані координати.

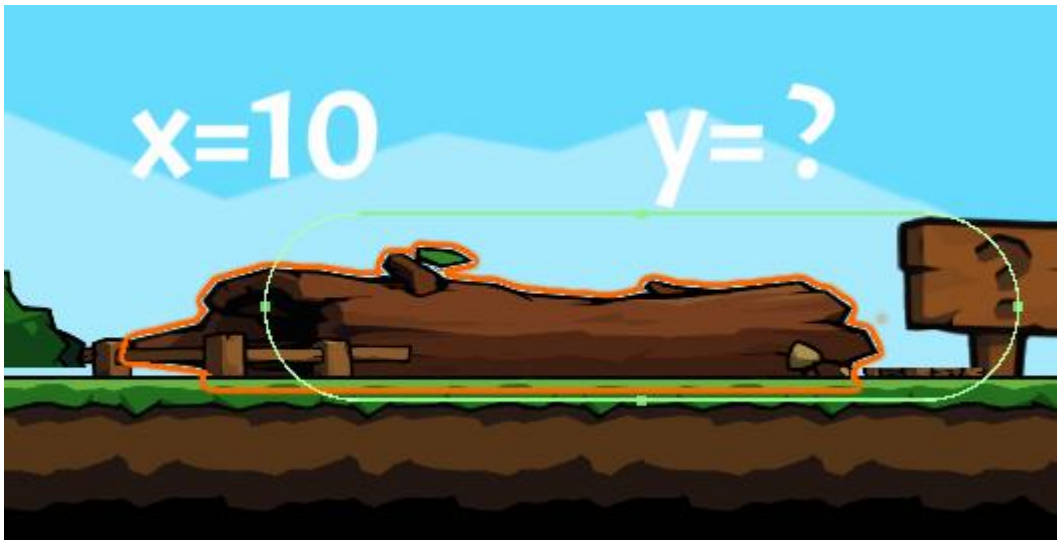

Рисунок 3.6 – Квестове завдання

Лістинг 3.13

```
if (collision.gameObject.CompareTag(«Vidpovid2»)){

Destroy(collision.gameObject);
StartCoroutine(MoveObject());}}
private System.Collections.IEnumerator MoveObject(){

while (Vector2.Distance(transform.position, targetPosition) > 0.1f) {
transform.position = Vector2.MoveTowards(transform.position, targetPosition,
speed * Time.deltaTime);
GetComponent<Collider2D>().enabled = true;
yield return null;  }}
```

В даному коді після перевірки з’єднання 2х об’єктів з відповідними тегами, завдання вважається виконаним, після чого об’єкт з тегом “Vidpovid2”

видаляється та запускається корутина для плавного переміщення об’єкта на задані координати. Після чого в об’єкта вмикається компонент колайдер, і гравець може взаємодіяти з даним об’єктом, в нашму випадку подолати перешкоду.

В завданні знайтии елементии коду для зброї. В іншому випадку зброя не наносить шкоду потрібним об’єктам. На рисунку 3.7 зображена квестова зброя, яку гравець може підняти, пересуватись, проте можлиивість знищувати об’єкти чи наносити шкоду буде відсутня.

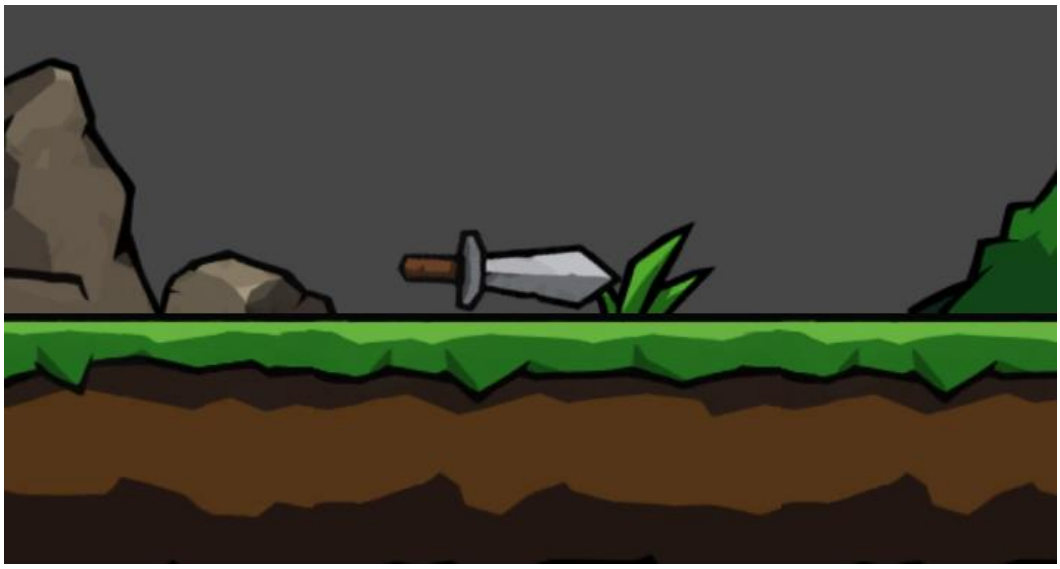

Рисунок 3.7 – Квестова зброя

Для того, щоб зброя наносила шкоду, потрібно змінити їй параметр, гравець має зайти потрібниий параметр і помістити його в предмет. Рисунок 3.8

Рисунок 3.8 – Квестовий елемент

Лістинг 3.14

```
public string targetTag = "Vidpovid3";
private void OnTriggerEnter(Collider other)
   {
      if (other.CompareTag(targetTag))
      {
         attackOn = true;
      }
   }
```

В даному коді відбувається перевірка, чи помістив гравець потрібний елемент в об’єкт, якщо так, то змінна перемикається на “true”, після чого гравець може активно нею користуватись.

Отже даними методами гравець зможе закріпити вже отримані знання з програмування, деякі з цих завдань можуть потребували сторонньої допомоги з боку викладача, проте їх можна виконувати з друзями. На зображеннях наведені лише основні елементи, за допомогою різних частинок гравці також зможуть змінювати основну логіку об’єктів чи створювати свою самотужки чи якщо цього потребуватиме завдання.

3.3. Тестування додатку

Протестуємо мобільний додаток, для засвоєння навичок з програмування який був розроблений в середовищі Unity.

Після запуску програми гравець опиняється в головному меню (Рисунок 3.9).

Рисунок 3.9 – Головне меню

Для гравця доступні декілька кнопок, а саме кнопка старту, перегляд винагород, де будуть розміщена інформація про його успіхи, кнопка поставити запитання після чого відбудеться переадресація до відповідної сторінки та кнопка інформації про розробника.

Після того як гравець тисне клавішу старт, він опиняється в іншій локації. Рисунок 3.10.

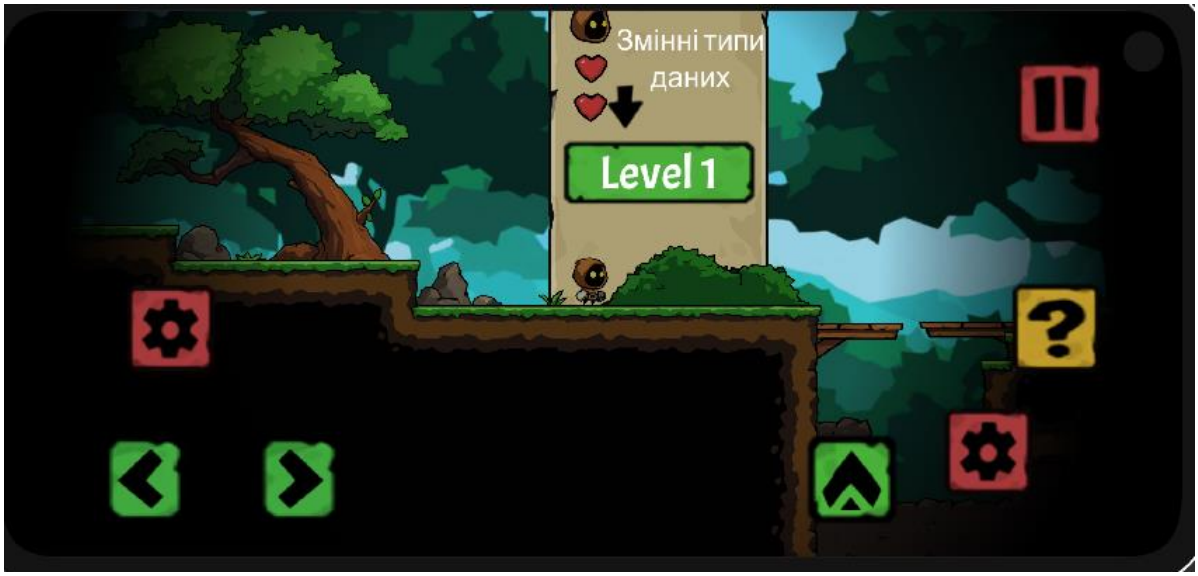

Рисунок 3.10 – Початкова локація

В даній локації гравець може обрати доступний рівень для проходження, проте якщо гравець захоче запустити одразу другий рівень чи інший, він буде

йому не доступним, адже для відкритя йому потрібно буде пройти попередній. Рисунок 3.11.

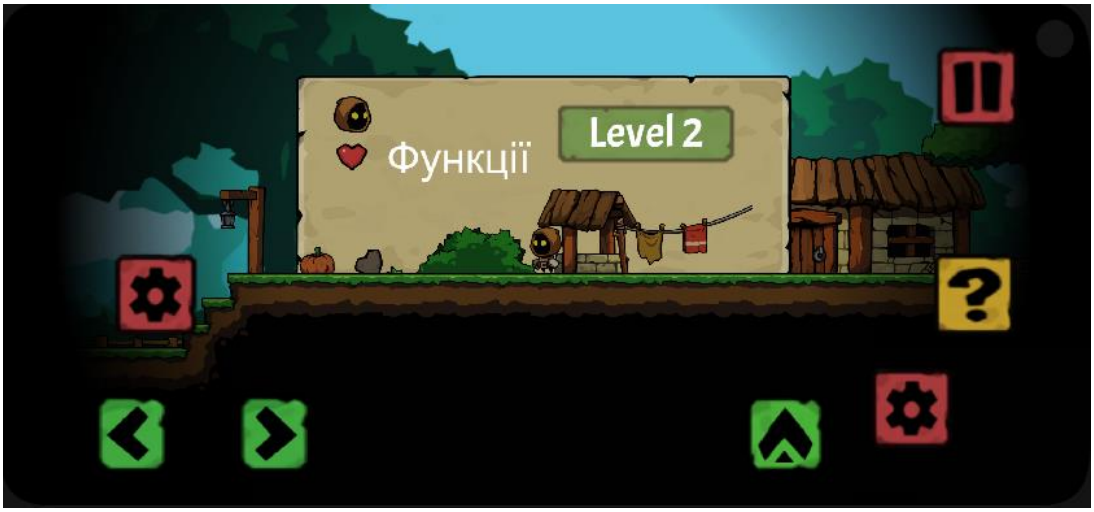

Рисунок 3.11 – Недоступний рівень

Окрім класичних клавіш керування, гравцеві доступна клавіша паузи, за допомогою якої він може зупинити додаток, проте таймер гри не зупинятиметься. Рисунок 3.12

Рисунок 3.12 – Меню паузи

В даному меню гравець може повернутись на головне меню, перезавантажити рівень або продовжити його.

Розглянемо виконання одного з завдань, для того щоб подолати перешкоду у вигляді прірви, гравцеві потрібно побудувати міст. Коли гравець знаходить чи

отримує в винагороду координату, за допомогою клавіші використати він може залишити об’єкт, після чого об’єкт переміститься в потрібне положенння та дасть змогу гравцеві пройти далі. Рисунки 3.12 та 3.13.

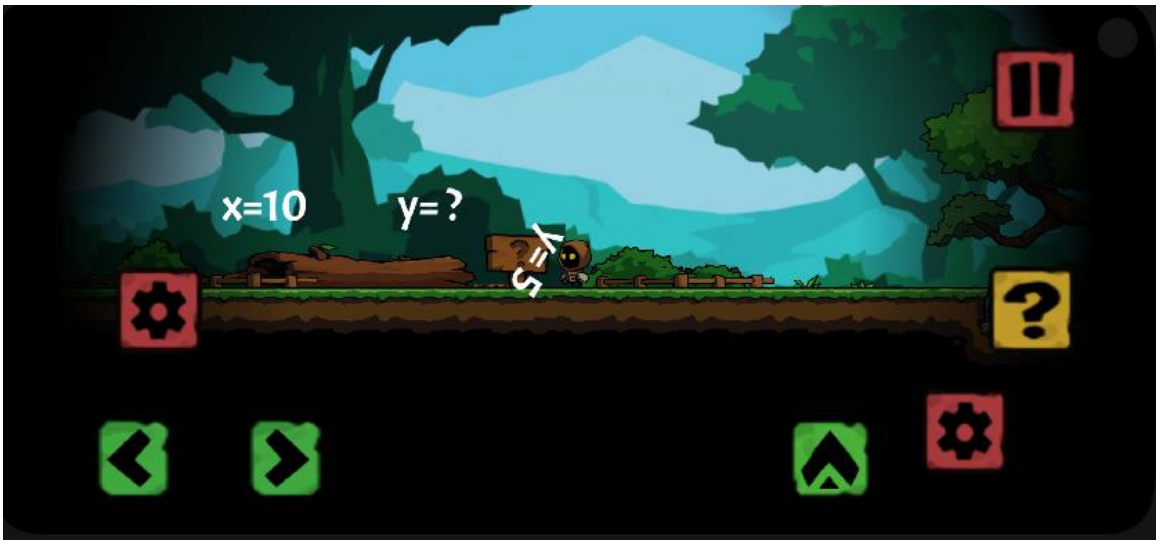

Рисунок 3.12 – Переміщення координати

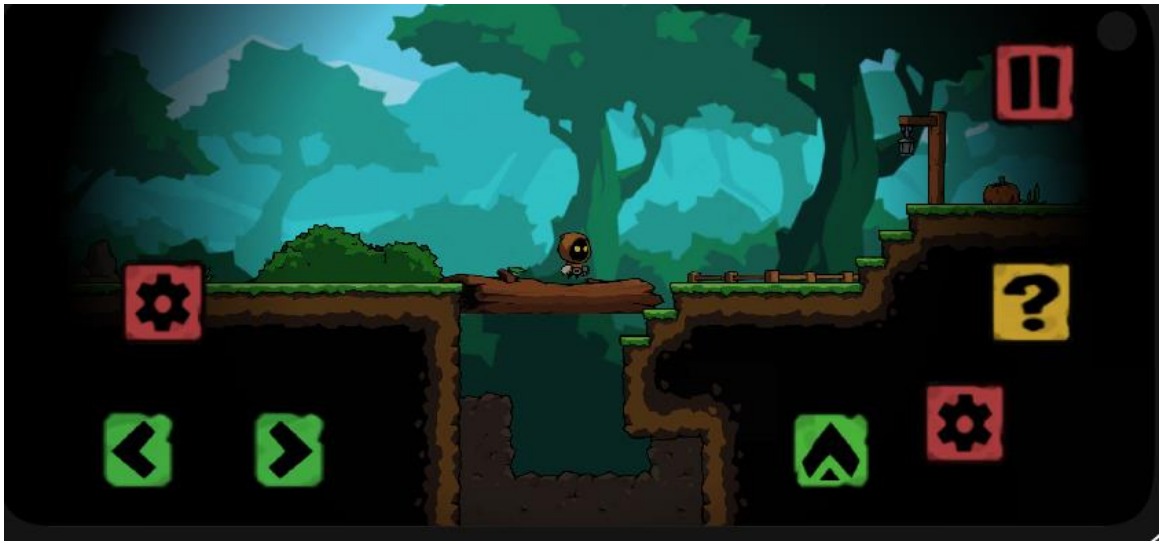

Рисунок 3.13 Подолання перешкоди

Попередні завдання, демонструють лише базові можливості коду, використання додатку. Обробка зіткнень, зміни стану об’єктів чи зміну характеристик. Для реалізації більш складних завдань, потрібна співпраця з відповідним фахівцем, для кращого розуміння психології гравців.

ВИСНОВКИ

Навчання програмуванню – це комплексне завдання, яке виходить за рамки простих прикладів коду. Наведені принципи вище демонструють базову взаємодію об’єктів, обробку даних та зміни об’єктів та їх властивостей.

Було проведено дослідження та розробку системи генерації квестових завдань для стимулювання засвоєння навичок програмування. В результаті виконаної роботи було вирішено наступні завдання:

- Проаналізовано предметну область та існуючі рішення в сфері гейміфікації навчання програмуванню.
- Створено мобільний додаток з різноманітними квестовими завданнями, інтегрованими в ігровий процес, що сприяє підвищенню мотивації та зацікавленості учнів у навчанні програмування.
- Впроваджено механізми заохочення, такі як трофеї, модернізація характеристик персонажа, для стимулювання активної участі учнів у навчальному процесі.

Практичне значення роботи полягає у створенні ефективного інструменту для навчання програмуванню, який може бути використаний в освітніх закладах та для самостійного навчання, сприяючи підвищенню зацікавленості учнів та ефективності засвоєння матеріалу.

Розроблена система має перспективи подальшого розвитку, зокрема, розширення функціоналу додатку, впровадження нових типів квестових завдань та інтеграція з іншими освітніми платформами.

Результати дослідження можуть бути використані в освітній практиці для покращення якості навчання програмуванню та підвищення мотивації учнів.