МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук, управління та адміністрування
Кафедра Інформаційних технологій

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка мобільного варіанту гри BrickShooter

Виконав студент 4 курсу групи К-41
Спеціальність 122 Комп'ютерні науки
Сокуренко Олег Олександрович

Керівник ст.викл.
Рольщиков Вадим Борисович

Консультант к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Рецензент к.геогр.н., доцент
Лужбін Анатолій Михайлович

Одеса 2020

ЗМІСТ

Скорочення та умовні позначки 6
Вступ 8
1 Аналітична частина 10
Характеристика об’єкта розробки 10
Опис предметної області 10
Моделювання процесу розробки проекту 10
Аналіз існуючих аналогів 12
Опис версії гри на ПК 12
Аналіз гри на мобільний пристрій 15
Аналіз засобів розробки 19
Опис мови програмування C# 19
Опис мови програмування Java 20
Функціональні можливості бібліотеки LibGDX 21
Опис середовища розробки Unity 2020 23
Опис IDE Visual Studio 25
Функціональний опис Adobe Photoshop CC 2019 26
Вибір СУБД 28
Опис SQLite 28
Опис MySQL 29
2 Проектна частина 30
Вимоги до функціональних характеристик і сумісності гри 30
Проектування дизайну та зовнішнього вигляду 30
Діаграма діяльності можливості гри UML 33
3. Опис реалізації 36
Графіка і спрайти 36
Реалізація переходу між сценами 37
Розробка анімації 39
Заповнення ігрового поля 41

Підключення до БД 43
Ієрархія об’єктів Unity 44
4 Опис кінцевого додатку 46
Порівняння з метою проекту 46
Тестування ПП 46
Вікно збірки проекту 47
Можливе удосконалення та впровадження гри 48
Висновки 49
Перелік джерел посилання 50
Додаток А Лістинг класів 52

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

БД – База даних, впорядкований набір даних.

Геймплей – (англ. gameplay) інтерактивна взаємодія гри та гравця.

Ігровий рушій – центральна програмна частина, яка відповідає за всю технічну сторону.

ОС – операційна система.

ПК – персональний комп'ютер.

ПП – програмний продукт.

СКБД – Система керування базами даних.

Скріншот – (англ. screenshot) знімок екрану мобільного пристрою чи комп’ютера.

Спрайт – графічний об'єкт(растрове зображення) в комп'ютерній графіці.

Action – дія на діаграмі UML.

Activity Diagram – діаграма активності UML.

ADT – Android Development Tools, середовище розробки додатків для Android.

Android – операційна система, зазвичай для мобільних пристроїв.

Adobe Photoshop – графічний редактор.

API – Application Programming Interface, прикладний програмний інтерфейс.

C# – об’єктно-орієнтована мова програмування.

Google Play – магазин додатків для мобільних пристроїв.

IDE – Integrated Development Environment інтегроване середовище розробки.

JDK – Java Development Kit, набір бібліотек і інструментів для створення, компіляції та налагодження програм.

JIT-компілятор – Just in time Compilation

JVM – Java Virtual Machine, віртуальна Java-машина.

Material design – графічний стиль в іграх.

SDK – Software Development Kit, набір засобів для розробки програмного забезпечення.

UML – Unified Modeling Language, уніфікована мова моделювання.

Unity – ігровий рушій, фреймворк.

Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення.

ВСТУП

У XXI столітті одним з найпопулярніших способів саморозваги стали мобільні ігри. З приходом мобільних телефонів кількість користувачів, що воліли грати в будь-якому місці стрімко росло .Це пов’язано і з тим, що смартфони є більш доступними за ПК.

Кожного дня розробники завантажують в Google Play нові унікальні за жанрами та стилями ігри, щоб дивувати користувачів та заробляти на монетизації гроші. Необов’язково створювати нові ігри для завантаження, можна переписати відому гру під мобільний пристрій. Це буде актуально в наші дні бо за відносно рівну вартість ПК та мобільного пристрою, телефони мають певний ряд переваг – компактність, продуктивність, області використання, більш легка система комунікації з іншими людьми.

На ОС Android існує так багато ігор, що можна витратити купу часу в спробах знайти гру варту уваги без надокучливої реклами та вірусів, але якщо користувачі мають гру на ПК, то скоріш за все установлять її тільки вже на мобільний пристрій.

Читанням, різноманітними завданнями, а також розвиваючими іграми можна і потрібно тренувати мозок. Аналогічно до розвитку м’язів, людський розум буде потребувати більшої складності.

Логічні ігри на Android нізащо не перестануть бути популярними. Знищення монстрів чи різноманітні “пісочниці” з часом починають набридати. Для ОС Android сьогодні функціонують тисячі логічних ігор – відкрити 100 дверей, відгадати слово, пройти квест, лабіринти[1][1)].

Головоломки – ігри, що мають багато сукупного з іграми на логіку, допомагають в розвитку інтуїтивних здібностей та стратегічному мисленні. Іноді завдання бувають надто складними, бо саме їх вирішення потребує послідовного використання певних прийомів та навичок.

[1)][1] Лучшие игры-головоломки для Android. URL: http://android.mobile-review.com/articles/47398/. (дата звернення 28.04.2020).

Об’єктом даної розробки є мобільна гра у стилі «Three in a row», яка направлена на розвиток мислення та в водночас буде дозвіллям для дітей і дорослих[2][1)].

Жанр ігор три в ряд відомий на весь світ. В нього грали, грають і будуть грати батьки, діти та їх онуки. З часом буде змінюватися сюжет, графічний інтерфейс, дизайн, але в цілому сенс гри залишиться незмінним. В даному проекті за основу для перепису під мобільний пристрій взята гра «Brick Shooter», що деякою частиною механіки та геймплею схожа на улюблений світом жанр гри три в ряд.

Представлена пояснювальна записка до дипломної роботи містить 51 сторінку, 34 рисунки, 14 посилань та 9 листів додатків.

[1)][2] Лучшие игры «три в ряд» на Android. URL: https://cubiq.ru/luchshie-igry-tri-v-ryad-na-android/.(дата звернення 28.04.2020)

1 АНАЛІТИЧНА ЧАСТИНА

Характеристика об’єкта розробки

Об’єктом розробки є мобільна гра «Brick Shooter» в стилі «Three in a row». За основу взята комп’ютерна гра «Brick Shooter», яка має доволі цікавий та багаточасовий геймплей навіть можна сказати нескінченний.

Опис предметної області

Ігри-головоломки як правило популярні. І вже є безліч захоплюючих головоломок для користувачів які полюбляють Андроїд ігри на комбінування та роздуми.

Нестандартна графіка яка затягує і дозволяє насолоджуватися кожною хвилиною витраченою в грі. Приємний звуковий та музичний супровід, а також привабливий інтерфейс як можна краще допоможе в залученні до ігрового процесу якомога більшої кількості гравців.

Результат напряму буде залежати від креативного і творчого мислення, моделювання та бажання доповнити класичний варіант.

Взявши за основу комп’ютерну версію гри потрібно як можна більше виділити зауважень та претензій, це напряму вплине на успіх проекту.

Моделювання процесу розробки проекту

Проект «Мобільна версія гри Brick Shooter» повинен у цілому задовольняти потреби: комфортної гри з мобільного пристрою на системі Android, бути аналогом комп’ютерної гри та нести в собі зразок сучасного дизайну.

При аналізі проекту було визначено необхідні стрілки: механізмів, управління, входів та вихід розробки проекту (рис.1).

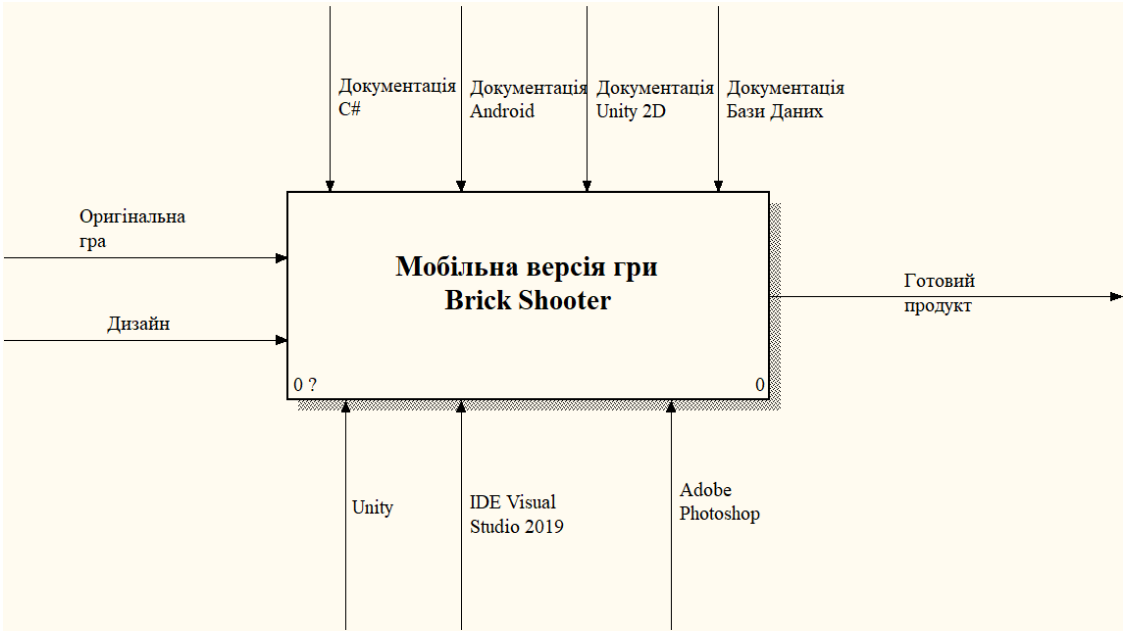

Рисунок 1 – IDEF0 загальний опис проекту

Більш детальніше розглянуто відповідальність кожного структурного елементу, необхідні ресурси та виходи кожної роботи (рис. 2)

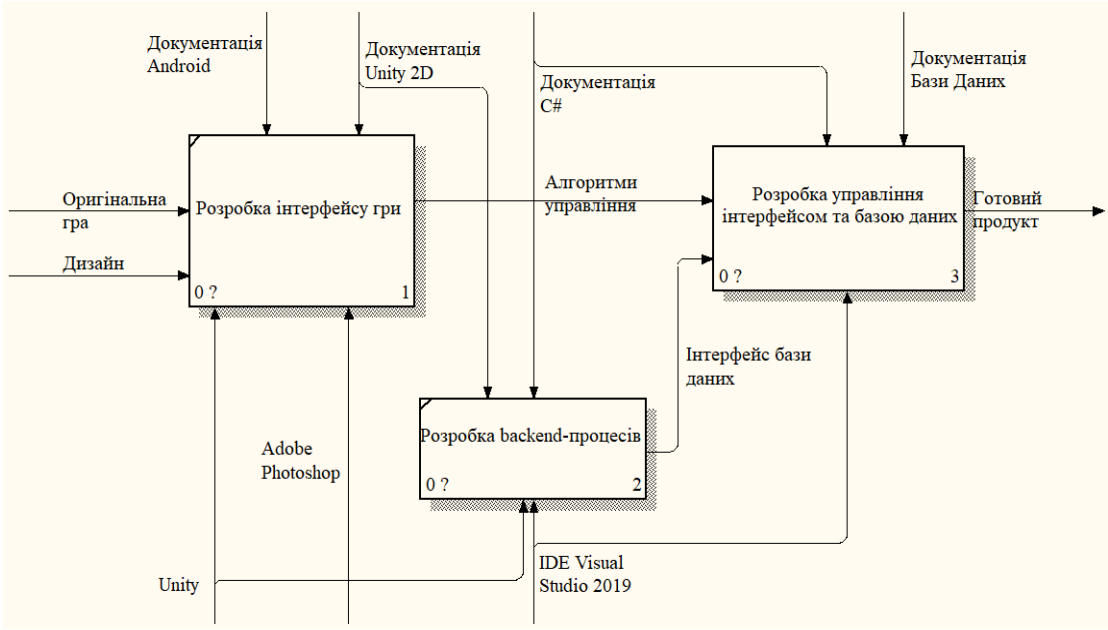

Рисунок 2 – Діаграма декомпозиції 0. Опис елементів структури «Мобільна версія гри Brick Shooter»

Для поставленої задачі необхідно мати уявлення про ключові етапи (рис. 3).

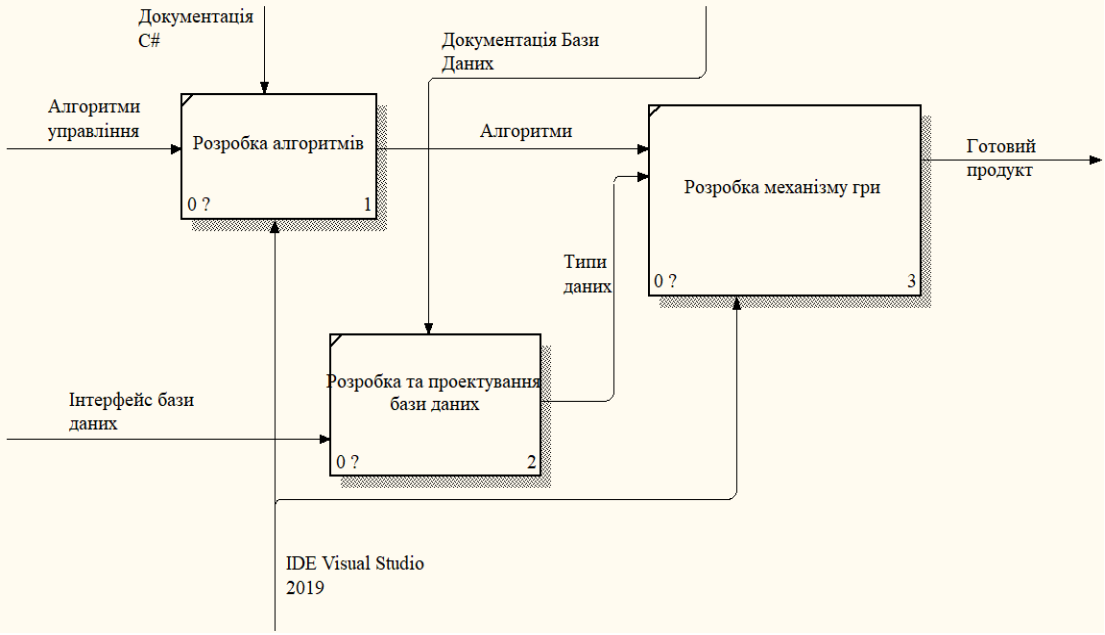

Рисунок 3 – Діаграма декомпозиції 1. Опис елементу структури «Розробка управління інтерфейсом та базою даних»

Аналіз існуючих аналогів

Опис версії гри на ПК

BrickShooter Classic – комп’ютерна гра написана в 1999-му році Дмитром Пупініним. Має прихильників по всьому світу, саме тому потрібна мобільна версія гри, яка повинна мати таку саму механіку та функціонал гри для розповсюдження на мольні пристрої.

На головному екрані гри (рис. 4) є 10 кнопок: new, scores, undo, minimize, save, quit, options, help, prev., next. По їх назвам на англійській мові зрозуміло за що вони відповідають, розглянемо більш детальніше options (рис. 6).

Перший пункт вибір кількості кольорів від 5 до 10, відповідно до їх кількості залежить складність ігрового процесу.

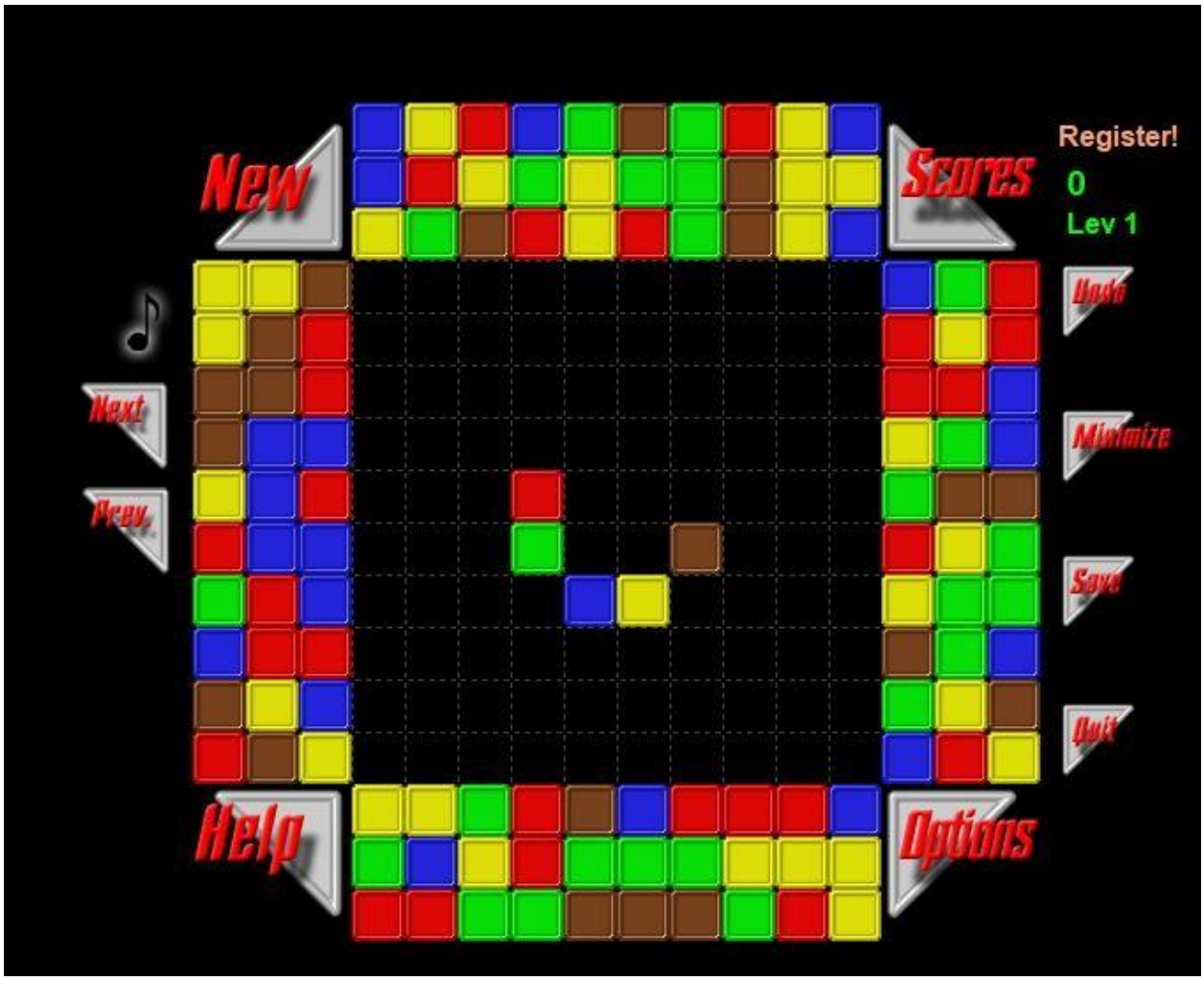

Рисунок 4 – Інтерфейс гри «BrickShooter Classic» на ПК

Наступний пункт включення чи вимкнення звуків та музики, потім кнопка «Skin», що дозволяє відкрити файл на ПК та завантажити новий стиль блоків та інших компонентів інтерфейсу. «Fullscreen» – дозволяє розгорнути гру на весь екран, що в мобільній версії буде не актуально, бо за замовчуванням більшість додатків вже мають повноекранний режим.

Останнє налаштування «Timed game» (рис. 5), що вмикає функцію відрахування часу до кінця гри, після стікання часу програш, якщо не встигли очистити ігрове поле.

До плюсів комп’ютерної версії можна віднести:

– функціональні особливості гри;

– музичний та звуковий супровід;

– анімація переміщення;
– кнопка «Undo», що дозволяє відмінити останній хід та «New», яка при поганому початку гри чи програшній ситуації дозволяє почати все з початку ;
– відображення поточних балів та рекордних;

З мінусів можна виділити:

– відсутність фону, чорний екран дуже негативно впливає на довго часову гру ;
– застарілий інтерфейс та дизайн;
– вікна “привітання” з користувачем додатку, що буде означати приготування до гри ;
– хоч в грі і присутній музичний супровід та все ж трохи застарілий;

Рисунок 5 – Таймер гри

При розробці мобільного додатку будуть враховані плюси комп’ютерної версії гри, та максимально уникнути мінусів

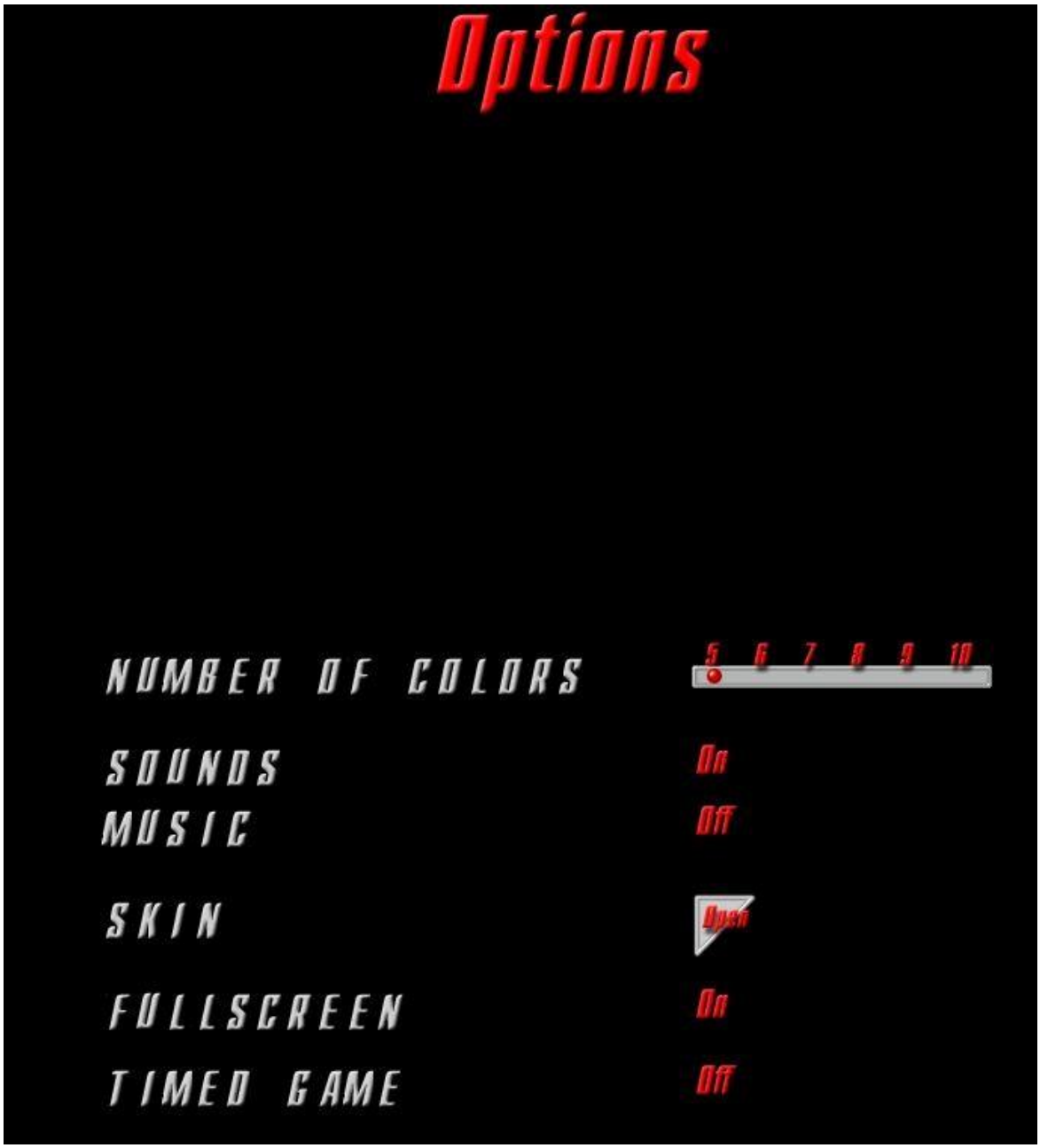

Рисунок 6 – Функціонал кнопки «Options»

Аналіз гри на мобільний пристрій

Існує декілька аналогів з Google Play (рис.7), але ми візьмемо на розгляд «Sliding Block», так як цей додаток більш за все схожий на класичну версію гри на ПК. Щоб у подальшій роботі чітко розуміти постановку задачі та вимоги до об’єкту розробки. Будемо оцінювати гру за такими критеріями: функціонал, інтерфейс і геймплей.

Гра «Sliding Block» (рис. 8) – це додаток, ідея якого набрати якомога більшу кількісті балів. Мета полягає у знищені всіх блоків на ігровому полі з'єднуючи три, чотири, п'ять, шість або сім блоків.

Це проста, нескінчена та розслаблююча гра для якої не потрібне з'єднання з мережею[3][1)].

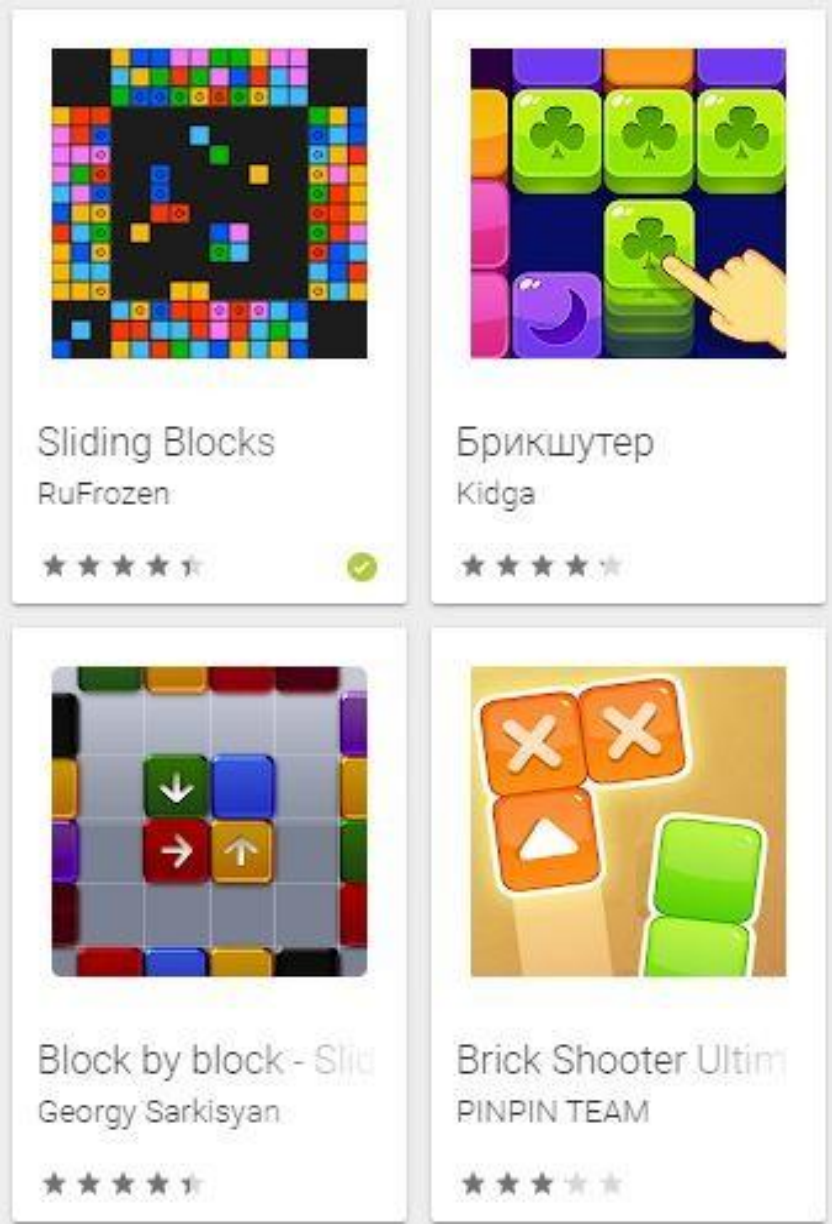

Рисунок 7 – Результату пошуку по запиту «brick shooter»

[1)][3] ТОП-10 ігор для Android, яким не потрібне з'єднання з інтернетом. URL: https://root-nation.com/ua/games-ua/game-articles-ua/ua-top-10-offline-android-games/ (дата звернення 30.04.2020)

Переваги ігрового додатку:

–анімація переміщення, створення та зникнення блоків;

–велика кількість вибору рівнів від 5 до 200;

–можливість грати при горизонтальному та вертикальному положенні телефону.

До недоліків можна віднести:

–чорний фон який давить на очі;

–відсутність будь-якої музики та звукових ефектів;

–відсутність таблиці рекордів;

–хоч гра і має 200 рівнів та через меню налаштування можна перейти на будь-який;

Після запуску гри зображено головне меню ігрового додатку (рис. 8). Також всі навігаційні кнопки.

Функціонал гри звичний, у верхній лівій частині (рис. 8) знаходяться 3 кнопки «RESTART» «MENU» «UNDO». В правій верхній частині, знаходяться текстові поля «SCORE», «LEVEL» та «HARD» для відображення поточних балів, рівня та складності. Кнопка «undo» що дозволяє зробити один крок назад.

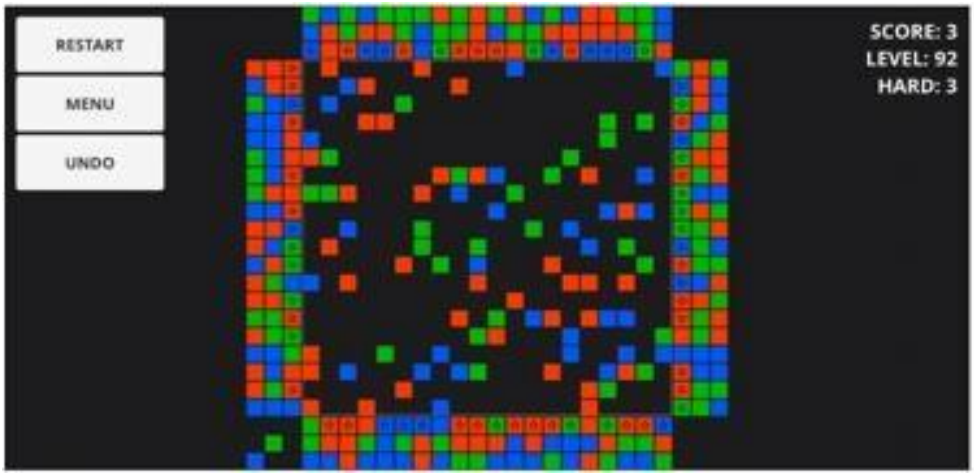

Рисунок 8 – Горизонтальне представлення гри «Sliding Block»

Рисунок 9 – Меню налаштувань гри «Sliding Block»

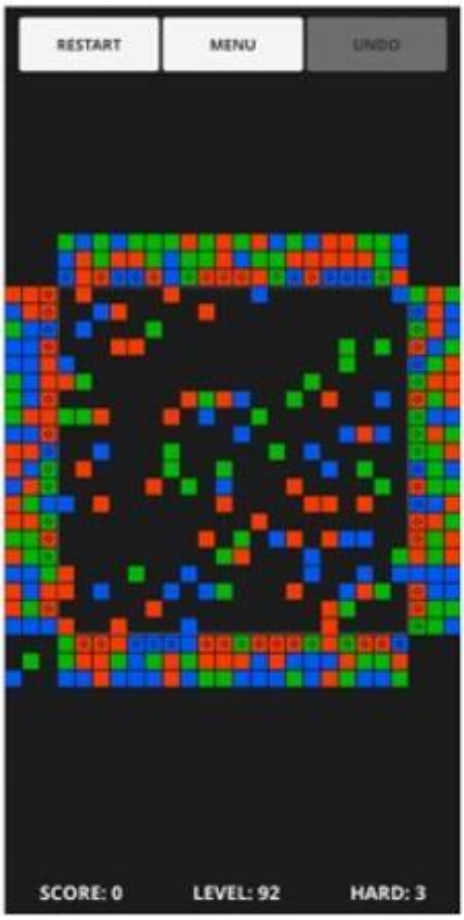

Рисунок 10 – Вертикальне представлення гри «Sliding Block»

За результатом аналізу аналогу можна зробити наступні висновки ідея

полягає у створенні проекту, який максимально має всі перераховані переваги та по можливості позбутися від всіх недоліків.

Тому гра буде в значній мірі копією аналога на ПК, оскільки він взятий за основу проекту. Увага буде приділена стилю графіки, принципу гри, легкості управління, відсутності затримок, музикальному та звуковому супроводу, та можливо введенню підказок. Гра орієнтована на дітей та дорослих.

Аналіз засобів розробки

Так як розробка відеоігор в першу чергу є основою прибутку, компанії зв'язані з нею шукають якомога більше можливостей для розповсюдження своїх додатків та збільшення аудиторії фанатів. Компанії знайшли вихід в кросплатформності розробки додатків.

Це збільшує продажі додатків та їх завантажень, та за рахунок цього компанія має більший прибуток, а також розширює бренд в сфері відео ігор[4][1)].

Опис мови програмування C#

C# є об'єктно-орієнтованою мовою програмування, але підтримує компонентно-орієнтоване. Синтаксис близький до Java і C++, підтримує поліморфізм, вказівники на функції-члени класів, перевантаження операторів, події, атрибути, винятки та властивості [5][2)].

Мова C# дуже близький родич мови програмування Java. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі). Для розосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного

[1)][4]Зароботок в Google Play. URL: https://vipidei.com/internet/mobilnye-prilozheniya/prodvizhenie-v-google-play/. (дата звернення 5.05.20)

[2)] [5] C# – Преимущества и недостатки. URL: https://shwanoff.ru/plus-minus-c-sharp/. (дата звернення 6.05.20).

байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Синтаксис C# дуже багатий, та же не заважає при вивчені даної мови. Розробники, що знають C, C++ або Java, зазвичай починають швидко та ефективно працювати в C#. Синтаксис C # спрощує багато складності C ++, але при цьому надає потужні функції, наприклад обнуляє типи значень, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті. C # підтримує універсальні методи і типи, які забезпечують більш високий рівень безпеки і продуктивності, а також ітератори, що дозволяють визначати в класах колекцій власну поведінку ітерації, яке може легко застосувати в клієнтському коді. Вирази LINQ створюють дуже зручну конструкцію мови для строго типізованих запитів.

Опис мови програмування Java

Java – об’єктно-орієнтована мова програмування, належить компанії Oracle, але була розроблена компанією Sun Microsystems. Програми написані на Java транслюються в байт-код, саме тому вони можуть працювати на комп’ютерній архітектурі будь-якого типу, за допомогою JVM [6][1)].

Важлива особливість технології Java є система безпеки, тому що віртуальна машина повністю контролює виконання програм. Будь-які операції котрі перевищують встановлені права програми, викликають негайну зупинку. Раніше програми написані мовою Java були повільніші та займали багато оперативної пам’яті, аніж написані мовою C.

В 1997–1998 роках швидкість виконання програм на мові Java була значно покращена за допомогою JIT-компілятора версії 1.1.

[1)] [6] Java – взгляд изнутри. URL: https://tproger.ru/blogs/jvm-insides/. (дата звернення 6.05.20).

Функціональні можливості бібліотеки LibGDX

LibGDX – це високопродуктивний Java фреймворк для створення кросплатформних ігор та додатків, створений, щоб за допомогою однієї бібліотеки використовувати один і той же код для мобільних пристроїв та натільних комп'ютерів (рис. 11).

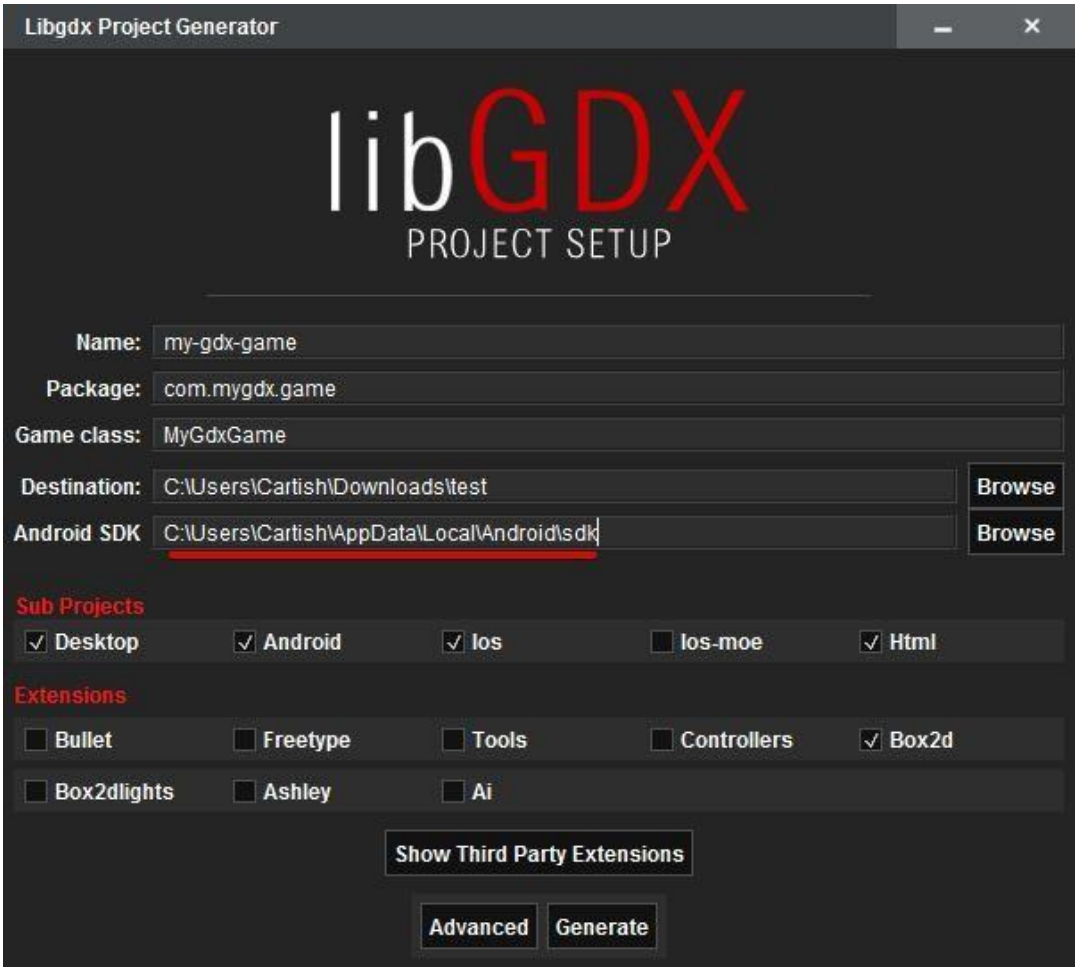

Рисунок 11 – Інтерфейс додатку «LibGDX»

LibGDX має не поганий інтерфейс, що одразу кидається в очі, з перших 5 хвилин зрозуміло, що фреймворк трохи застарілий для розробки середнього рівня ігор, але для освоєння початкових знань з розробки додатків добре підходить.

Архітектура даного рушію ігор дозволяє розробникам писати, налагоджувати та тестувати додатки на їх власному ПК та використовуючи той же код на Андроїд.

Створений проект буде розроблятися в зручному для Андроїд розробки середовіщі Android Studio (рис. 12).

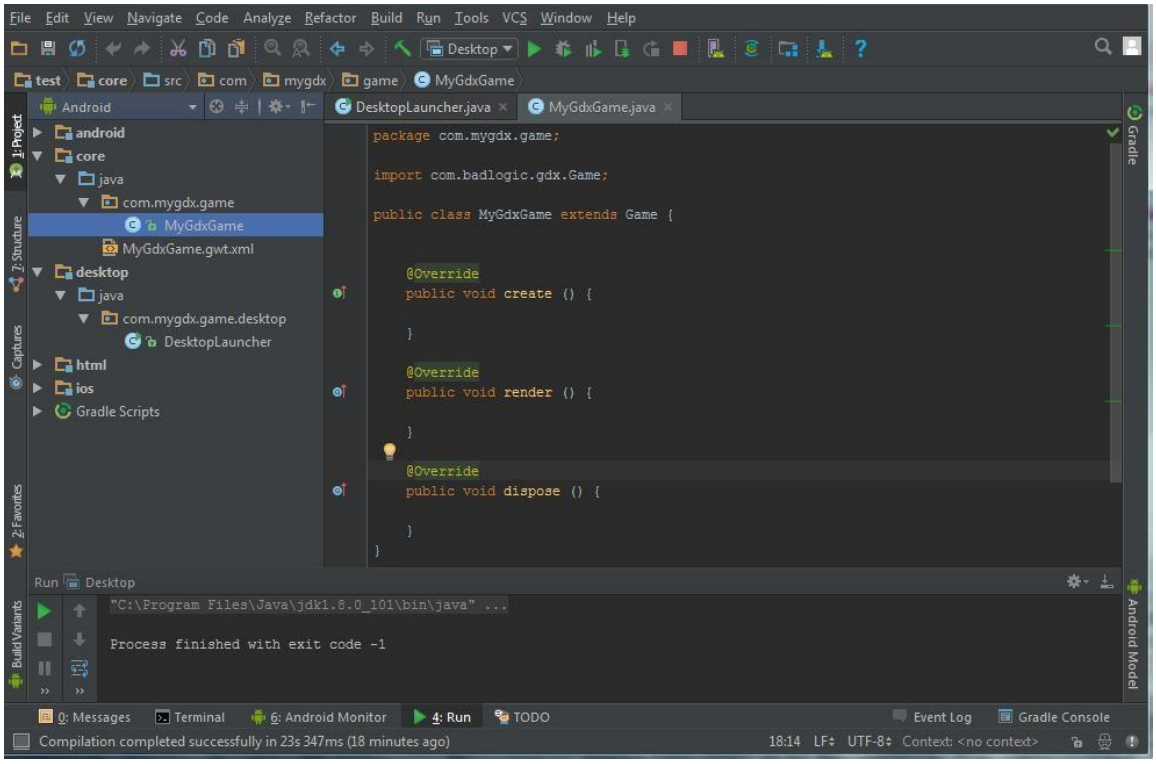

Рисунок 12 – Інтерфейс додатку «Android Studio»

Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS. Android Studio можна встановити на Windows, Mac і Linux. Обліковий запис розробника додатків в Google Play App Store коштує $ 25. Android Studio створювалася на базі IntelliJ IDEA.

IDE можна завантажити і користуватися безкоштовно. У ній присутні макети для створення UI, з чого зазвичай починається робота над додатком. В Studio містяться інструменти для розробки рішень для смартфонів і план-

шетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Середа Android Studio призначена як для невеликих команд розробників мобільних додатків (навіть в кількості однієї людини), або ж великих міжнародних організацій з GIT або іншими подібними системами управління версіями. Досвідчені розробники зможуть вибрати інструменти, які більше підходять для масштабних проектів[7][1)].

Опис середовища розробки Unity 2020

На мою думку найзручніше середовище розробки для ігор, багатоплатформовий інструмент для розробки двовимірних ігор[8][2]. Додатки створені за допомогою рушія Unity працюють на системах Linux, OS X, Windows та Android. Також можна створювати інтернет-додатки та ігри на гральні консолі (рис. 13).

Ігрова логіка пишеться на мові C#. Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі, що стане плюсом для кожного розробника, але найкраще підходить для починаючих.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація).

Проект в Unity ділиться на сцени (рівні) — окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань.

[1)] [7] Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 3.04.2020)

[2)] [8] ТОП 10 лучших игр на движке Unity. URL: https://gamedata.club/article-top/1359-top-10-luchshih-igr-na-dvizhke-unity.html (дата звернення 3.04.2020)

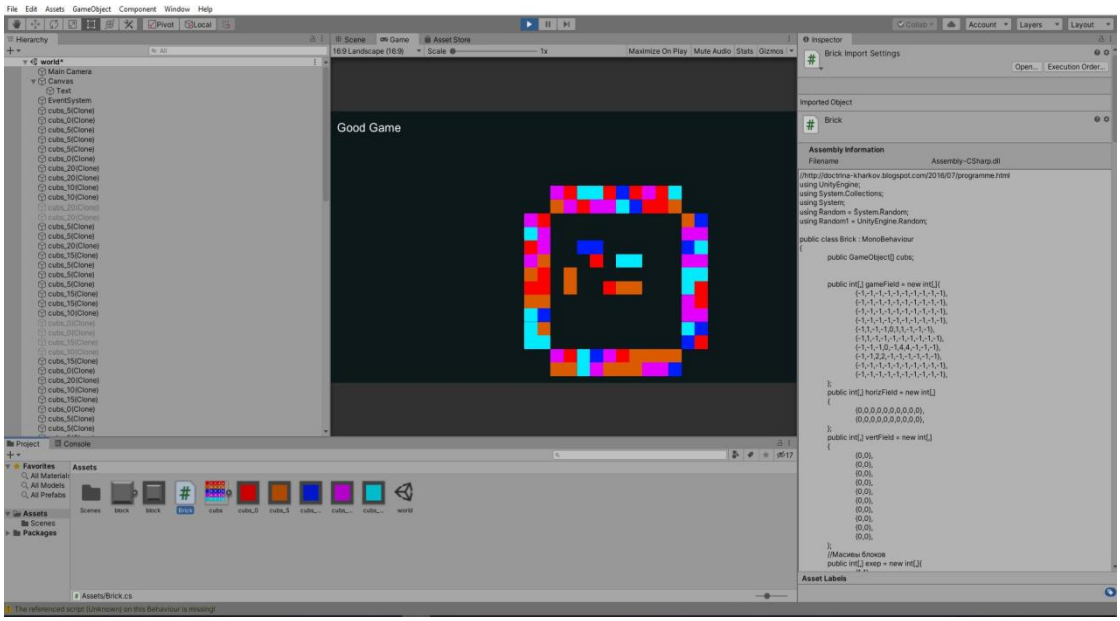

Рисунок 13 – Интерфейс програми Unity

Також в Unity зручна робота із спрайтами (рис. 14), що дозволяє з одного малюнка з розширенням .png розділити на багато одиничних об’єктів, та при роботі з анімаціями теж дуже зручно працювати.

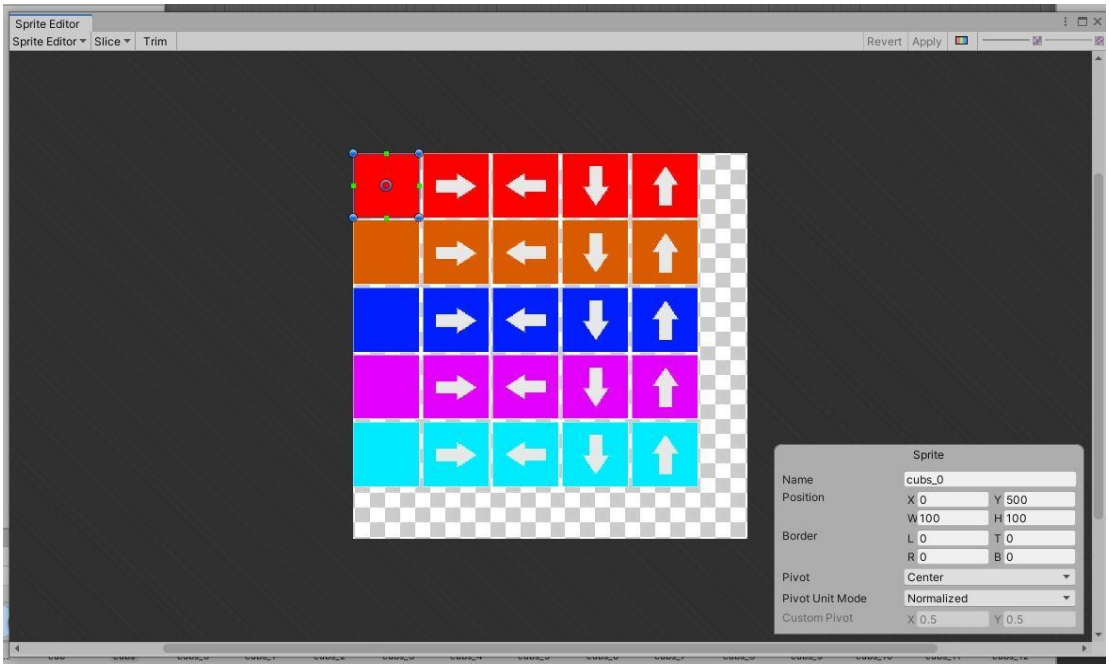

Рисунок 14 – Sprite Editor

Опис IDE Visual Studio

ПП Microsoft (рис. 15) найзручніше на мою думку IDE для написання коду, що включає підтримку останніх фреймворків і технологій. Visual Studio надає весь набір інструментів для роботи з різними видами проектів починаючи від одноосібного проекту, закінчуючи проектами десятками, а то і навіть сотнями розробників працюючих над одним проектом.

Visual Studio – інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Дані середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічнім інтерфейсом, наприклад віконні Додатки Windows Forms, а також Web-Додатки.

Середовище Visual Studio дозволяє розробляти Додатки, використовуючи Різні мови програмування: Visual C #, Visual Basic, Visual F #, Visual C++, Python. Також існує можливість розробляти Додатки не тільки під Windows, а і під інші Популярні Платформа: Android, iOS [9][1)].

В Visual Studio Installer є багато компонентів направлених для різних галузей розроби наприклад: веб-застосувань, Python, Node.js, SQL Server, C++, .NET. Був обраний через наявність інтеграції з Unity 3d, підтримки мови програмування C #. Також це найбільш вивчена розробником платформа, що дозволяє швидше розробляти додатки не витрачаючи час на вивчення нового матеріалу.

Дуже вигідним є гнучке налаштування стилізації програми, темні стилі для меншого тиску на очі розробника, темно-синя та світла.

Програма має платне розширення. Якщо не авторизуватися в системі то розробник має у своєму розпорядженні пробний період в розмірі 30 днів з моменту входу, для ознайомлення з нею.

[1)] [9] Середовище розробки Microsoft Visual Studio. URL: https://informatics.in.ua/programming_csharp/part_01.php (дата звернення 1.05.2020)

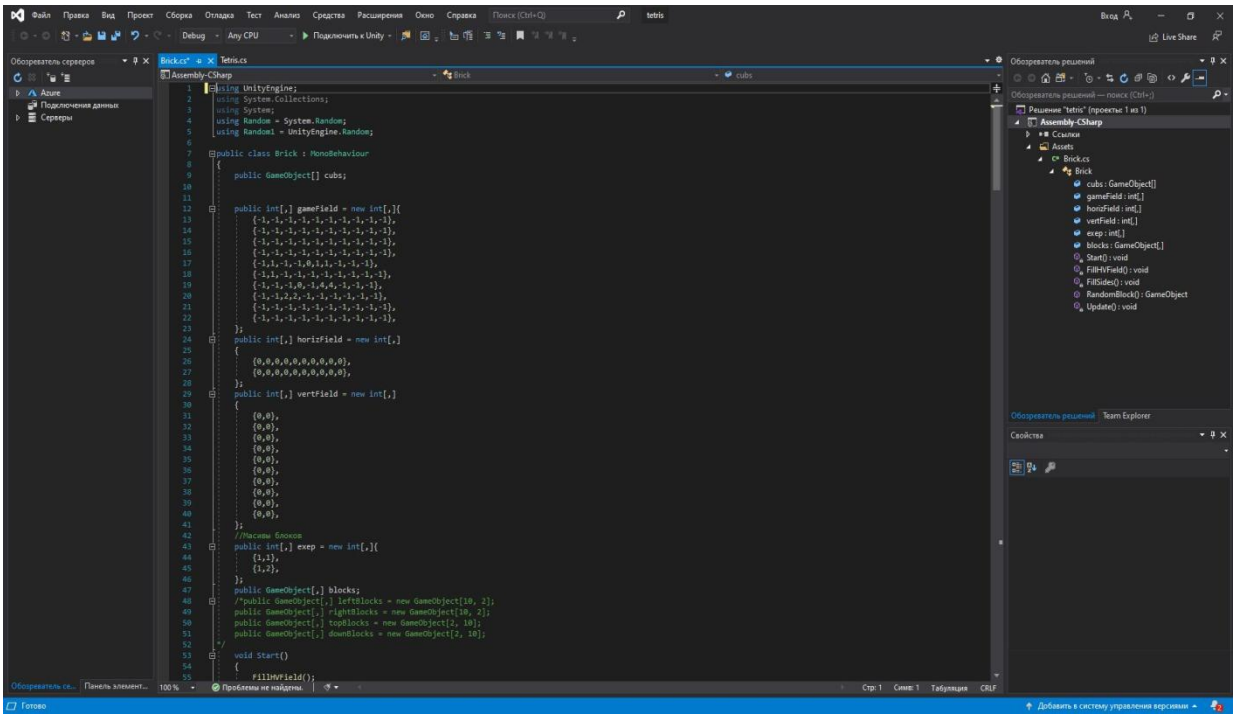

Рисунок 15 – Інтерфейс «Visual Studio IDE»

Функціональний опис Adobe Photoshop CC 2019

Графіка гри надає гравцям перші враження гри в цілому. Неяскрава, нецікава, взята готовою з інтернету може похитнути саму захоплюючу гру.

Компанія Adobe хоч і має платні ПП, проте в них при належному вміні можна створювати якісні продукти. В даний час Photoshop доступний на платформах macOS, Windows і iPadOS. Для Windows Phone і Android доступна спрощена версія програми під назвою Adobe Photoshop Touch. Також існує версія Photoshop Express для Windows Phone 8 і 8.1.

Тому розробнику важливо витратити певний час на створення ігрової графіки та хоч би найпримітивнішої анімації, яка знатна буде затримати погляд користувача.

Графічний додаток Adobe Photoshop CC 2019 (рис. 16) зберігає перші місця в ролі редактора для створення та анімації двовимірних ігор.

З переваг особливо хочу виділити:

– швидка корекція зображень;

– професійне управління кольором і тоном;
– обробка зображень за допомогою зовнішнього ведучого в своїй галузі модуля Camera Raw;
– кмітливі та поліпшені зображення;
– обробка зображень за допомогою професійних інструментів малювання;
– професійні можливості композитінга;
– ефективне та гнучке робоче середовище;
– прискорена організація робочого процесу[10][1)].

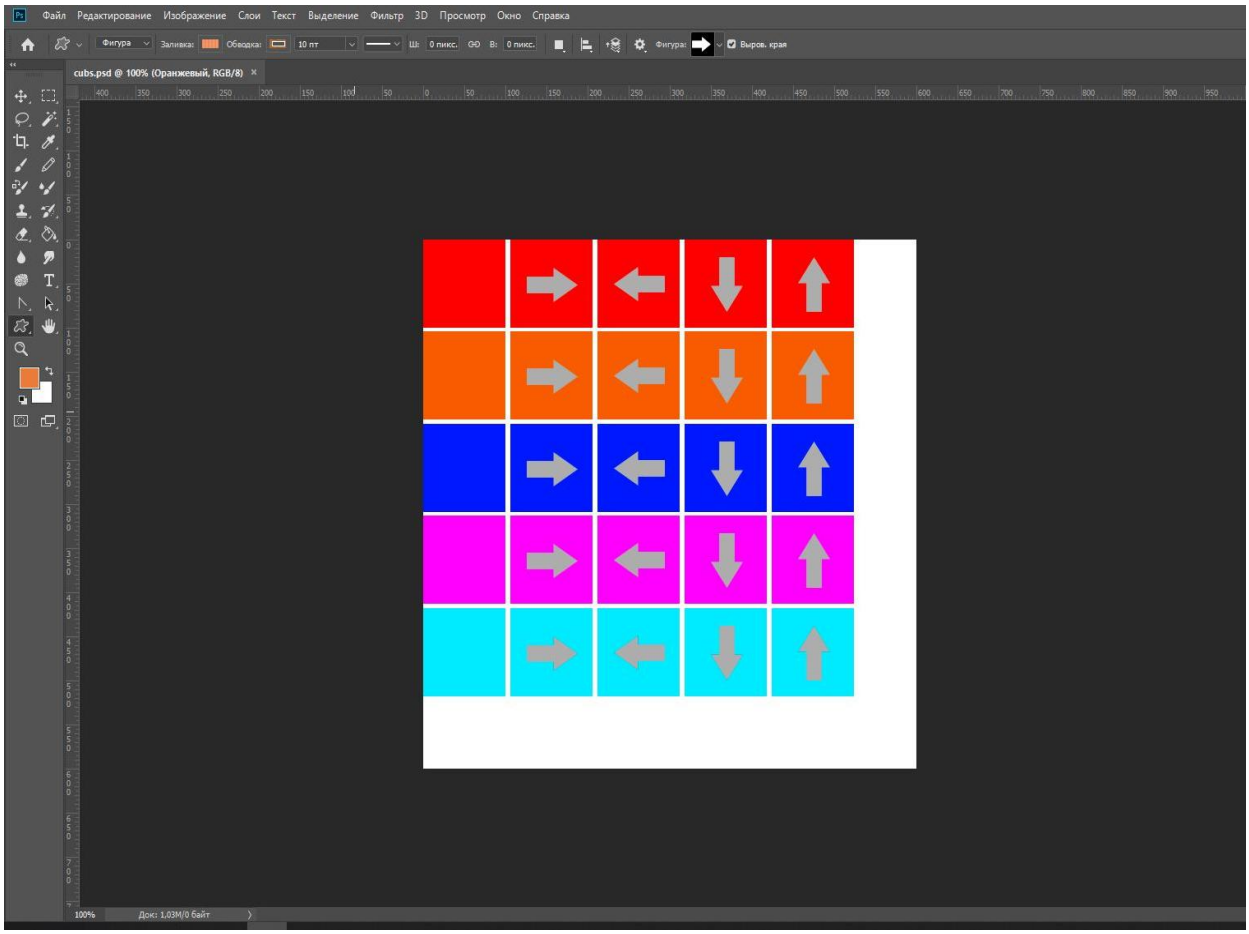

Рисунок 16 – Інтерфейс додатку «Adobe Photoshop CC 2019»

[1)] [10] Adobe Photoshop CC. URL: https://itpro.ua/product/adobe-photoshop-cc/?tab=description (дата звернення 28.04.2020)

Вибір СУБД

З можливих БД для даного проекту було вибрано декілька варіантів в яких є потреба легкого доступу до СУБД, безкоштовна робота с даним ПП та найважливішим фактором є зручність у користуванні та підключенні до вибраної середи розробки ігор.

Опис SQLite

SQLite – це безкоштовна бібліотека, полегшена реляційна СУБД. SQLite не використовує парадигму кліент-сервер це означає, що рушій не є окремим процесом SQLite, з яким взаємодіє додаток, а надає користувачам бібліотеку разом з допомогою якої програма компілюється та рушій стає складовою частиною програми. Цей метод дає змогу зменшити накладні витрати, спрощує програму та зменшує час відгуку[11][1)].

SQLite зберігає всю БД в єдиному форматі файл на комп'ютері, на якому виконується додаток.

До особливостей SQLite можна віднести:

- встановлення не потребує ні установки, ні адміністрування;
- прокоментований сирцевий код, що має 100 % текстове покриття розгалужень;
- транзакції послідовні, ізольовані, атомарні та міцні навіть після збоїв живлення, чи системи;
- БД зберігається в одному файлі(крос-платформовому) на диску;
- підтримка БД в терабайтних розмірах та гігабайтних розмірах рядків;
- маленький розмір коду: менше ніж 200 КВ з опущеними додатковими функціями та 350 КВ повністю налаштованого коду;
- легкий, простий у використанні API;

[1)] [11] SQLite. URL: https://lecturesdb.readthedocs.io/databases/sqlite.html (дата звернення 25.04.2020).

Інструменти обслуговування та створення БД здійснюються за допомогою текстового консоллю SQL-командами, чи спеціальними інструментами, в їх числі з графічним інтерфейсом для користувачів.

Опис MySQL

MySQL – це система керування базами даних з відкритим кодом, яка була розроблена компанією «ТсХ» з метою підняття швидкодії обробки масивних баз даних. Данна СКБД створена як альтернативна комерційним системам[12][1)]. На даний час MySQL одна з найпоширеніших СКБД.

У 2008 році Sun Microsystem придбала розробника СКБД MySQL за 1 мільярд доларів. Після поглинання компанією Oracle Corporation у 2009 році MySQL стала належати Oracle.

До можливостей сервера MySQL можна віднести:

– доволі висока швидкість виконання команд;

– одночасний доступ до БД необмеженої кількості користувачів;

– кількість рядків таблиці може досягати до 50 мільйонів;

– простий у користуванні та встановленні;

– ефективна та проста система безпеки.

БД має доволі високий попит у користувачів, бо система має широкий спектр підтримки кластерних серверів, використання MySQL на протязі багатьох років, дає розробникам велику кількість ресурсів, що і дозволяє розраховувати на швидку розробку ПП. Також є можливості для зміни коду у відповідності до потреб користувачів.

MySQL має гнучку ліцензійну політику, що є більш гнучкою в порівнянні з конкуренто схожими БД, її використання безкоштовне крім винятків коли вона допомагала в створені ПП для продажі послуг.

[1)] [12] MySQL — система управления базами данных. URL: https://web-creator.ru/articles/mysql . (дата звернення 25.04.2020)

2 ПРОЕКТНА ЧАСТИНА

Вимоги до функціональних характеристик і сумісності гри

Вимоги до функціональних характеристик проекту розробки наступні:

–вихідний код необхідно реалізувати на об'єктно-орієнтованій мові програмування C#;

–зручний і лаконічний інтерфейс, що не перевантажують ігровий процес;

–обов’язкове звукове впровадження у ігровий процес, так як вона спрямована на довгий часовий період;

–естетична візуалізація для залучення уваги користувачів;

–ігрова механіка повинна бути інтуїтивно зрозуміла користувачеві і мати внутрішньо ігрову інструкцію, а управління здійснюватися через сенсорні системи введення для мобільних пристроїв;

–користувачеві повинна бути надана можливість збереження ігрового прогресу для подальшого завантаження з можливістю вибору збереження.

Проектування дизайну та зовнішнього вигляду

Після запуску гри першим, що буде відображено на екрані мобільного пристрою сцена вітання «FirstLoad», на ній буде розташовано «Button» у вигляді тримірного блоку та поле «Text» з підказкою на англійській мові «Click To Start» (рис. 17). І вхідні дані з ім’ям користувача для відображення рекордних балів.

Для другої сцени нам знадобляться два поля «Text» для поточних балів та рекордних. Кнопка налаштувань можливо для зміни гучності музики та звукових ефектів, зміни важкості рівня (рис. 18).

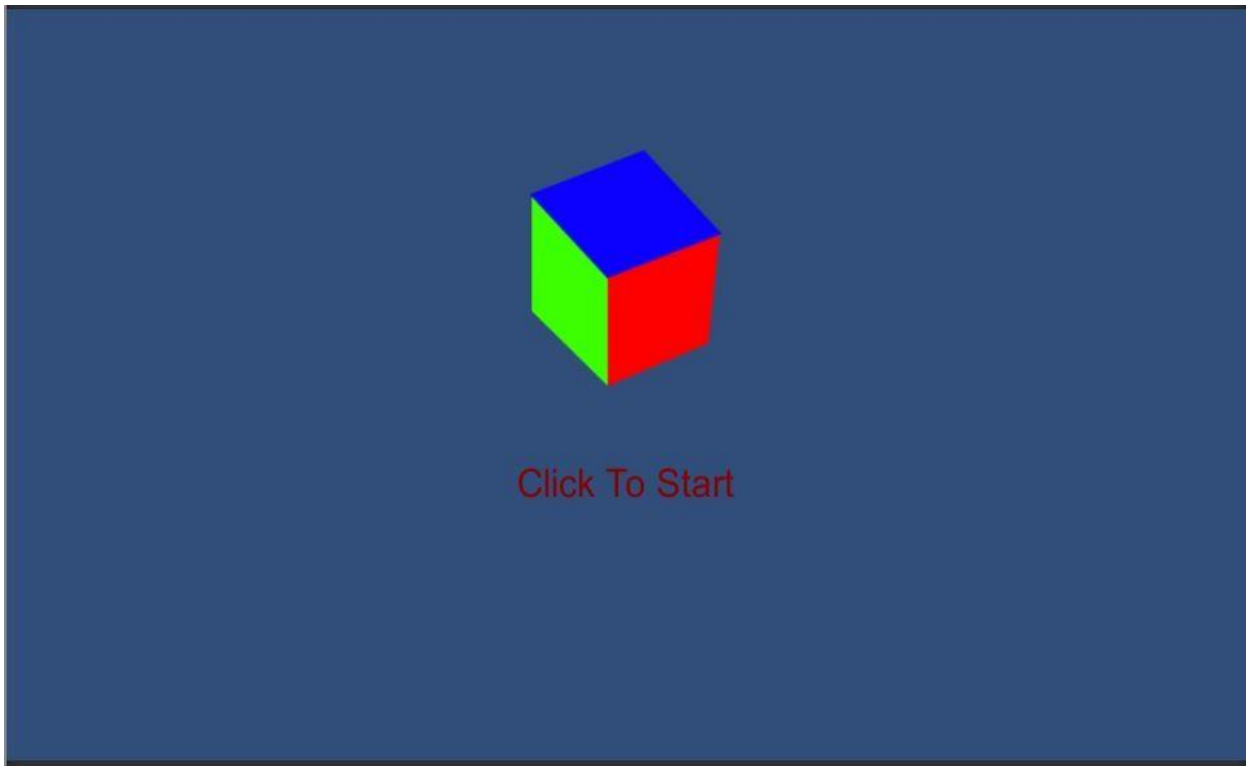

Рисунок 17 – Приклад першої сцени гри

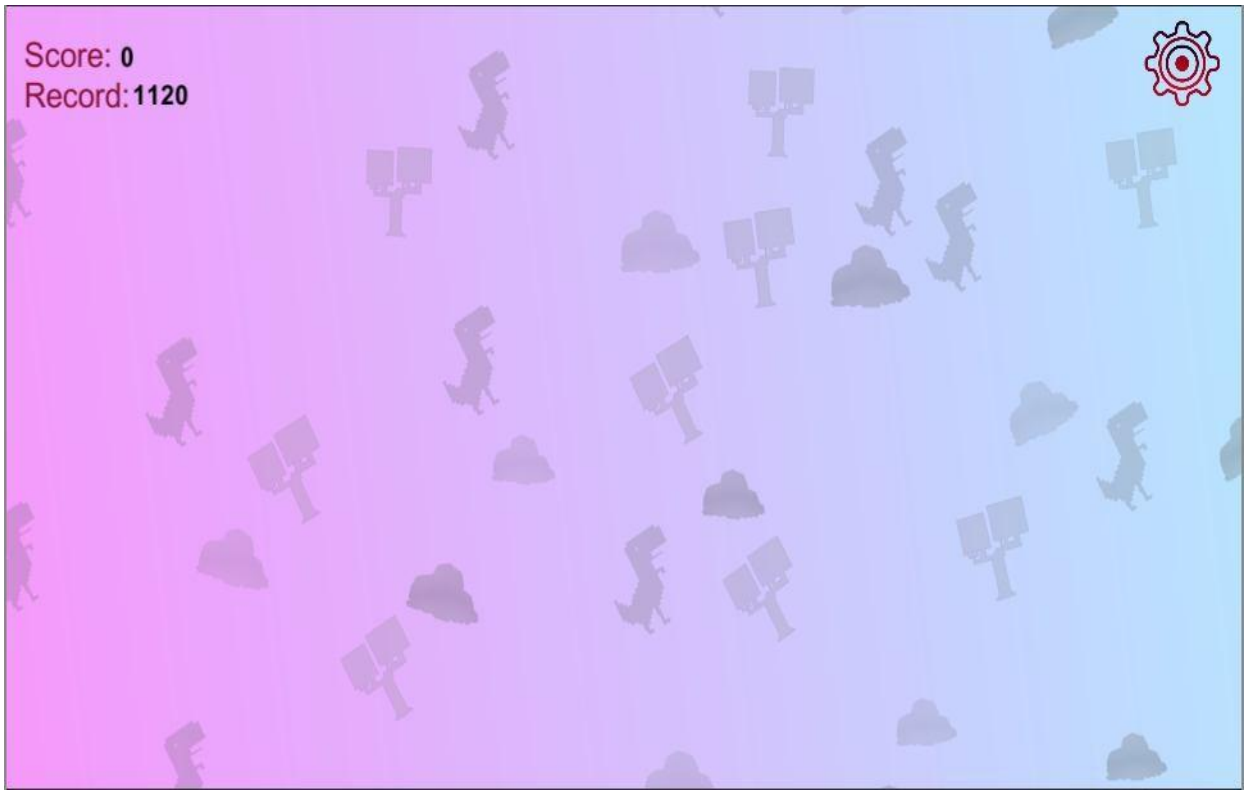

Рисунок 18 – Головна сцена гри «GameField»

За допомогою звучного інтерфейсу програми Unity можна просто перетягнути файл з музикою до «Canvas» і музика буде автоматично включена до сцени та мати широкий функціонал налаштувань(рис. 19).

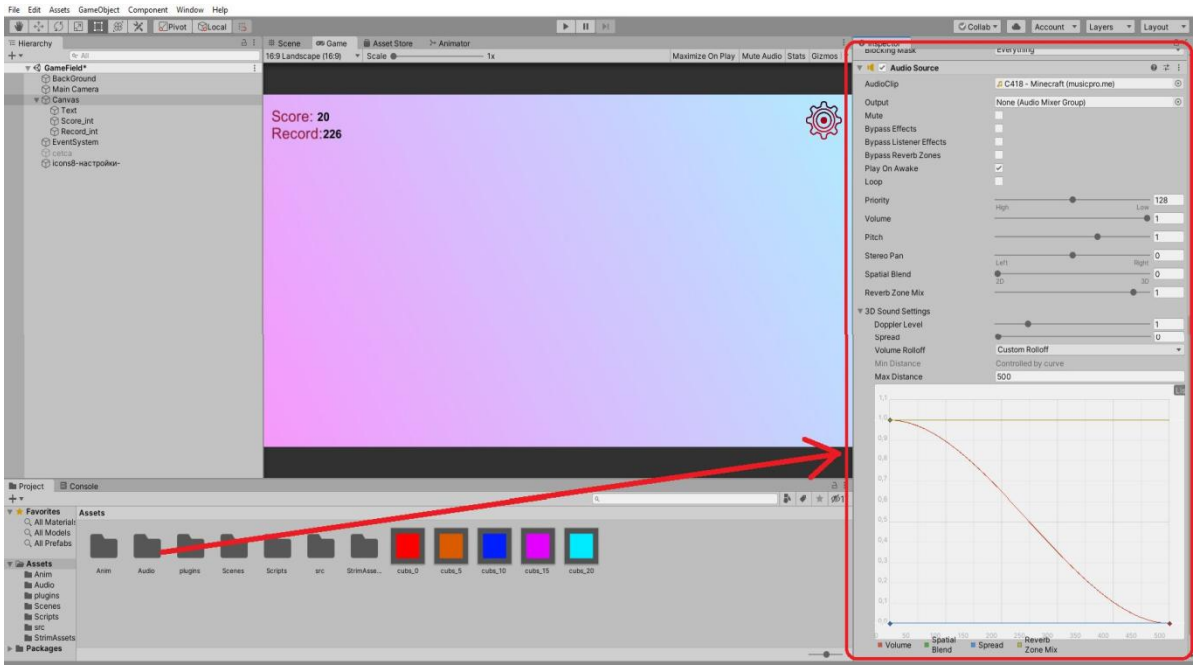

Рисунок 19 – Додавання фонової музики

Скрипт який прив’яжемо до «Slider», для налаштування гучності у грі.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Volume : MonoBehaviour
{
    private AudioSource audioSrc;
    private float musicVolume = 1f;
    void Start()
    {
        audioSrc = GetComponent<AudioSource>();
    }
    void Update()
    {
        audioSrc.volume = musicVolume;
    }
    public void SetVolume(float vol)
    {
        musicVolume = vol;
    }
}
```

Для завантаження сцени та її перезапуску використовуємо один і той же скрипт представлений нижче.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class nextEtap : MonoBehaviour
{
    public void Etap()
    {
        SceneManager.LoadScene("Game");
    }
}
```

Діаграма діяльності гри UML

UML (англ. Unified Modeling Language – уніфікована мова моделювання) – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур. UML є мовою широкого профілю – це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, так званої UML-моделі[12][1)]. Ця мова створювалася для того, щоб проектувати та документувати програмні системи, та хоч вона і не являється мовою програмування на підставі її моделей можливо генерувати код.

Перевагами UML є:

– об’єктно-орієнтований тип, за рахунок чого опис результатів аналізу та проектування семантично схожі до методів сучасних об’єктно-орієнтованих мов програмування;
– UML доповнює та дає змогу вводити власні графічні і текстові стереотипи, що в свою чергу сприяє застосуванню в різних сферах діяльності окрім програмної інженерії;

[1)] [13] Ведення в UML 2.0. URL: http://bourabai.kz/dbt/uml/index.htm. (дата звернення 29.04.2020).

– діаграми порівняно прості для розуміння після вивчення його синтаксису;
– UML може описати розглянуту систему з всіх точок зору і аспекти її поведінки;
– Мова UML динамічно розвивається та широко поширюється;

Діаграма активності і діаграма діяльності, на ній показані дії, стани яких описані на діаграмі станів. Під діяльністю розуміється специфіка поведінки у вигляді послідовного, паралельного та координованого виконання елементів – окремих дій та вкладених видів діяльності, з'єднаних потоками, які йдуть від виходів одних вузлів до кінцевого.

UML нотація передбачає п'ять видів систем:

– з точки зору процесів;
– вид з точки зору реалізації;
– з точки зору прецедентів;
– вид з точки зору проектування;
– з точки зору розгортання.

Кожен з вище перерахованих способів представлення системи, містить послідовні дії, які можна описати з допомогою алгоритмів.

Діаграма діяльності в UML – це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій[13][1)].

На вході бачимо екран вітання, де можна ввести ім'я, якщо вже є завантажуємо існуючий рекорд. Далі сам процес гри де після кожної дії йдуть перевірки на новий рекорд та наявність блоків. Вихід з програми виконується користувачем в будь-який момент часу, блок з рекордом завжди зберігається.

[1)] [14] Діаграма діяльності. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності#cite_note-rumbaught-1 (дата звернення 10.05.2020)

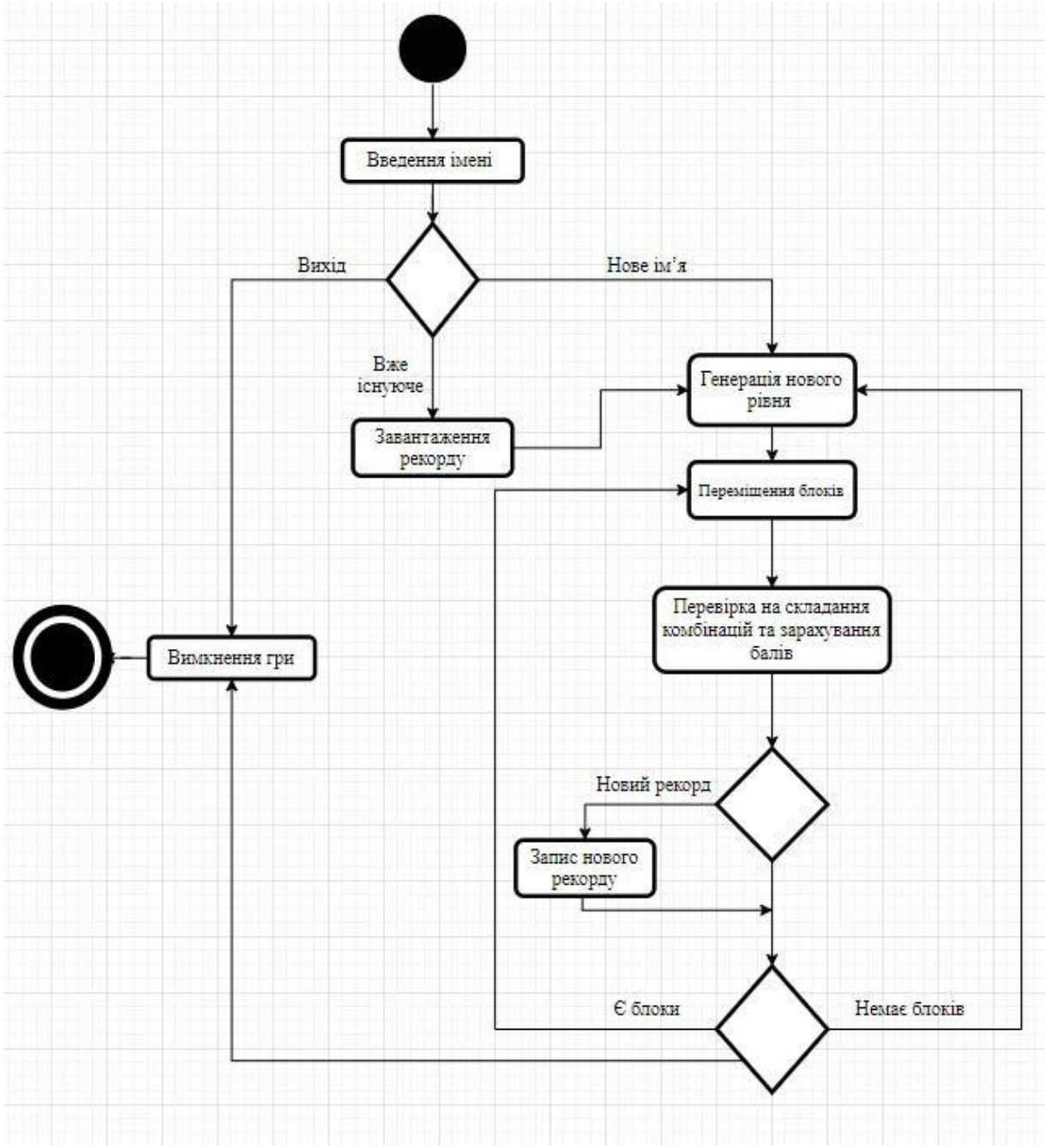

Рисунок 20 – UML діаграма діяльності

Діаграми діяльності будуються з обмеженої кількості форм, з'єднаних стрілками.

Основні типи форм:

– кружок чорного кольору означає початок процесу;

– закруглені прямокутники означають дію;

– стрілки означають порядок дій;

– ромб означає рішення;

– кружок чорного кольору в колі означає кінець процесу.

Як можна побачити з діаграми на рис. відображається варіанти розвитку дій користувача у грі.

Після запуску користувачу потрібно ввести ім’я для додавання рекорду в глобальну таблицю

3. ОПИС РЕАЛІЗАЦІЇ

Графіка і спрайти

При розробці графіки часто бувають ситуації коли потрібно велика кількість малюнків невеликого розміру, анімація, ігрові персонажі чи текстури, якщо в нас буде 5 персонажів і для кожного по 5 малюнків для анімації, то вже будемо мати 25 файлів, тож розумніше буде зробити один великий графічний файл “атлас”.

Атлас – це зазвичай файл з розширенням .png в якому зберігаються, або всі малюнки, або поділені за якимось критерієм.

Будь-яке 2D зображення можливо розділити на базові елементи, тобто створити 2D графіку як об'єднання цих елементів.

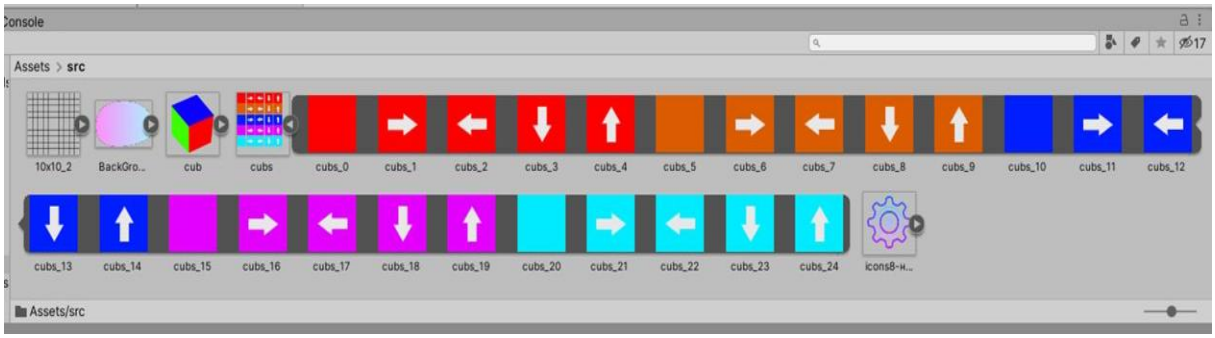

Рисунок 21 – Приклад вигляду малюнку після роботи в «Sprite Editor»

Спрайт – це, в комп'ютерній графіці, графічний об'єкт. Часто – растрове зображення, яке можна відобразити на екрані. Атлас спрайтів є растровим

зображенням, яке поєднує спрайт в набір кадрів анімацій, або іншим набором спрайтів. Для проекту розробка спрайтів була намальована в програмі Adobe Photoshop CC 2020. Приклад всіх видів блоків приведено на (рис. 21). В грі присутній 5 кольорів – червоний, оранжевий, синій, рожевий і блакитний.

Ми маємо той самий файл, але при натисканні на сіру кнопку з правої сторони нашого спрайту ми маємо окремі елементи які можемо використовувати в своїх потребах.

Спрайт куба було намальовано в Adobe Photoshop, а кнопки налаштування та назад знайдено на сайті з класичними невеликими малюнками в форматі .png.

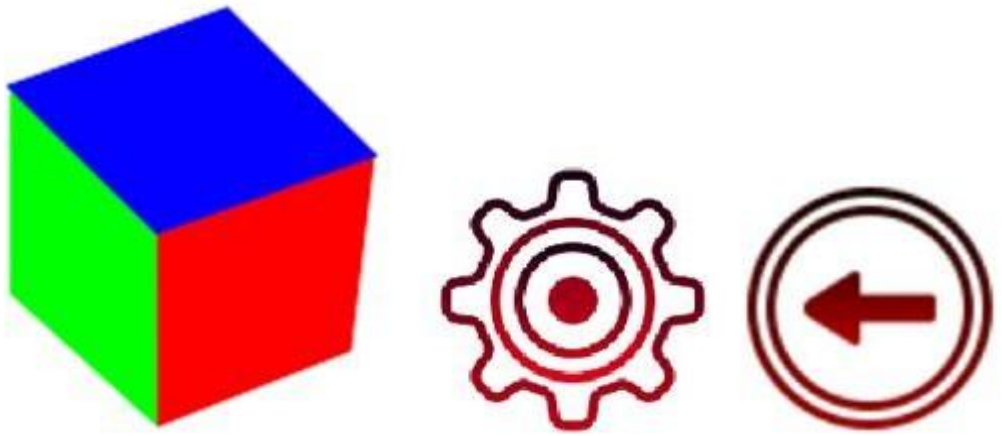

Рисунок 22 – Спрайти кнопок «Start» і «Back»

Реалізація переходу між сценами

Створюємо скрипт «nextEtap», який завантажує основну сцену. І перенесемо наш скрипт на кнопку «Start» (рис. 23). Цей же метод будемо викликати при перезавантаженні гри.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
```

```
public class nextEtap : MonoBehaviour
{
    public void Etap()
    {
        SceneManager.LoadScene("GameField");
    }
}
```

Не забуваємо додати скрипт на кнопку та вибрати параметри запуску нашого додатку

Рисунок 23 – Обробник подій при натисненні на кнопку

Також на кнопку (рис. 24) має такий самий обробник подій при натисканні на неї.

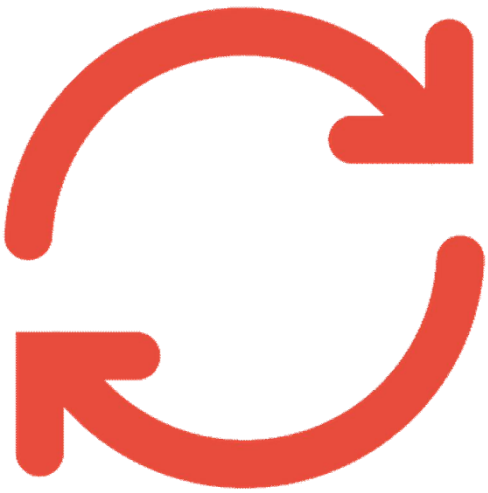

Рисунок 24 – Кнопка рестарту гри

Розробка анімації

Додаємо анімацію на кнопку початку гри вверх та вниз, для цього створюємо новий файл з розширенням .anim, в параметрах розтягуємо на 2 секунди, перша точка входу 0:00 з параметрами зміни кнопки по осі y = 70, друга у точці 1:00 рівній y = 85 і третя точка в 2:00, що дорівнює y = 70 для замкненої і плавної анімації.

Повторюємо всі маніпуляції, але для текстового поля і міняємо параметр прозорості теж на протязі 2 секунд, від 100 % до 0 % і назад до 100%.

Обов’язково додаємо в Canvas вхідний параметр ім’я, за яким буде зберігатися рекорд. Прив’язуємо наші анімації (рис. 25).

Після завершення створення анімації, вони будуть симетрично один одного відтворюватися.

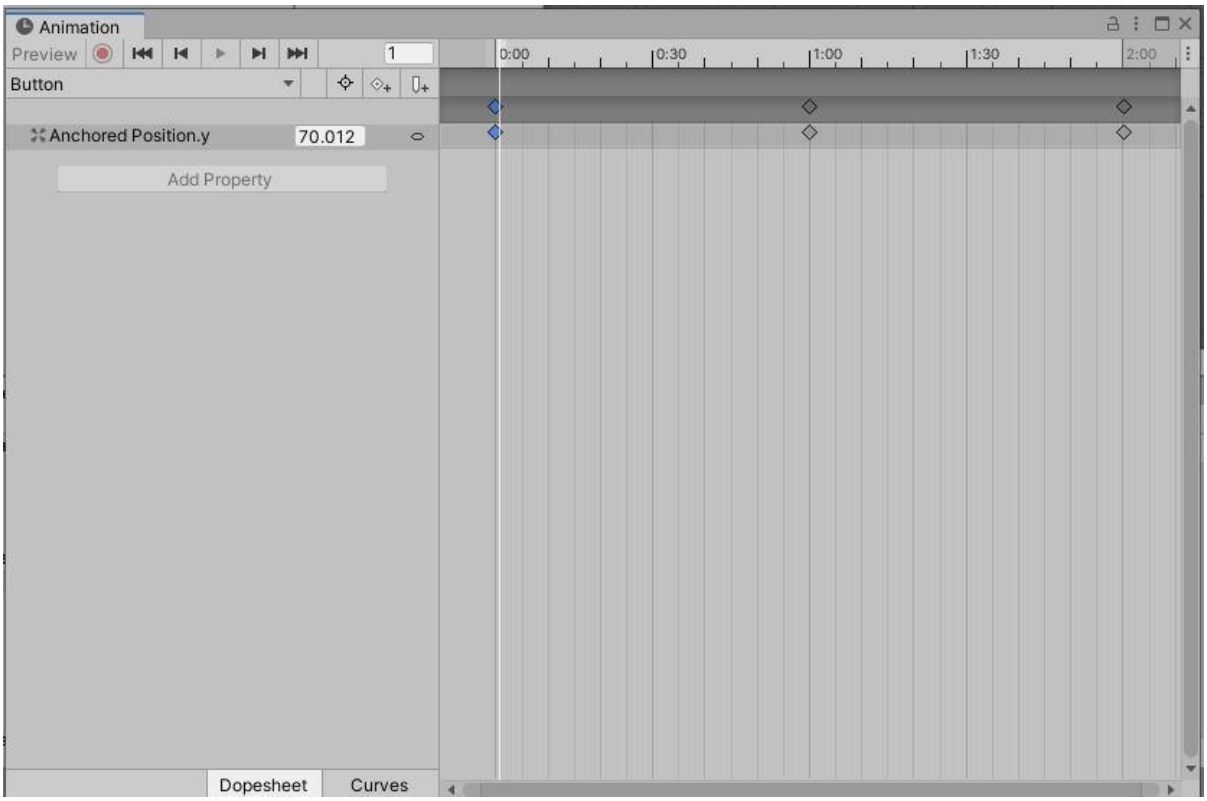

Рисунок 25 – Вікно «Animation» в Unity

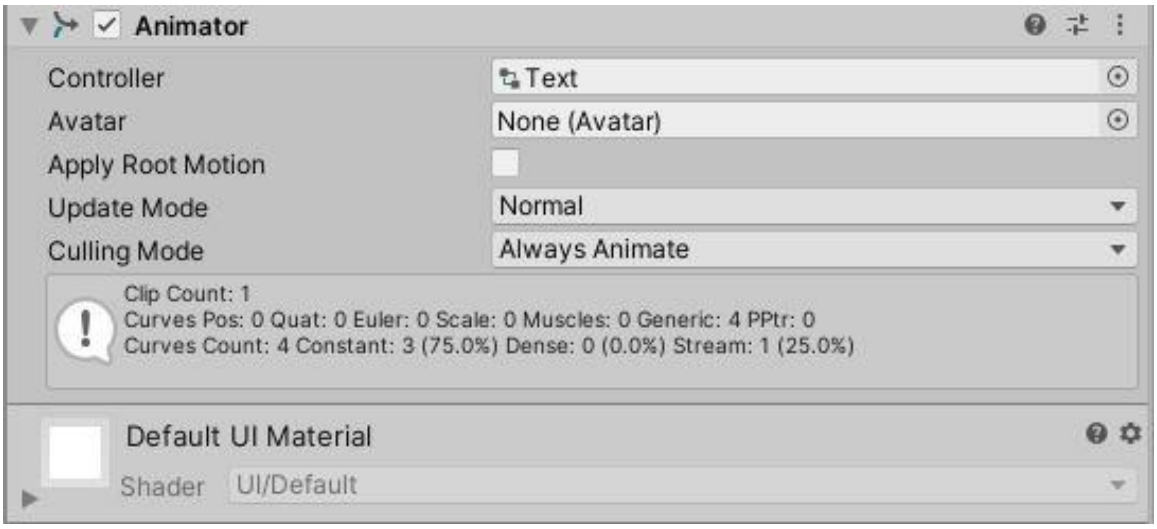

Рисунок 26 – Прив’язка анімації

Рисунок 27 – Анімація «Button» і «Text»

Також разом з анімацією створюється файл з розширенням .controller в якому за замовчуванням анімація ставиться при загрузці сцени що нас повністю задовольняє (рис. 28).

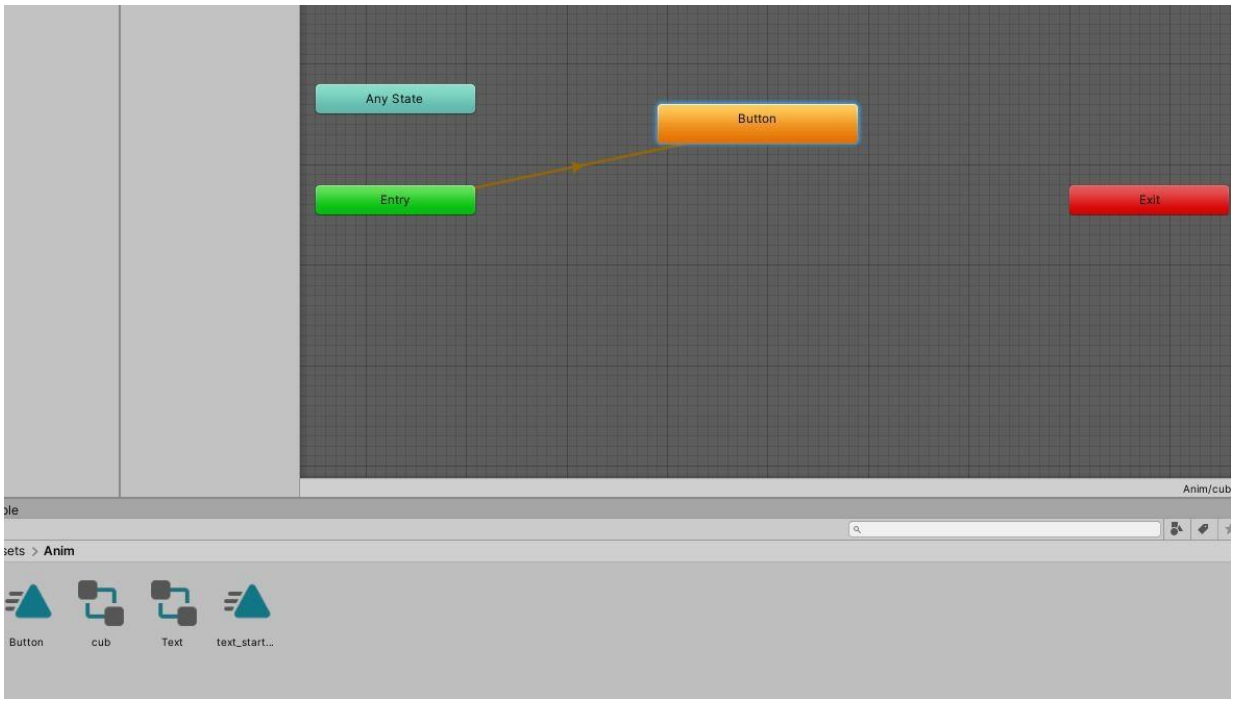

Рисунок 28 – Опис прив’язки початку анімації «cub»

Заповнення ігрового поля

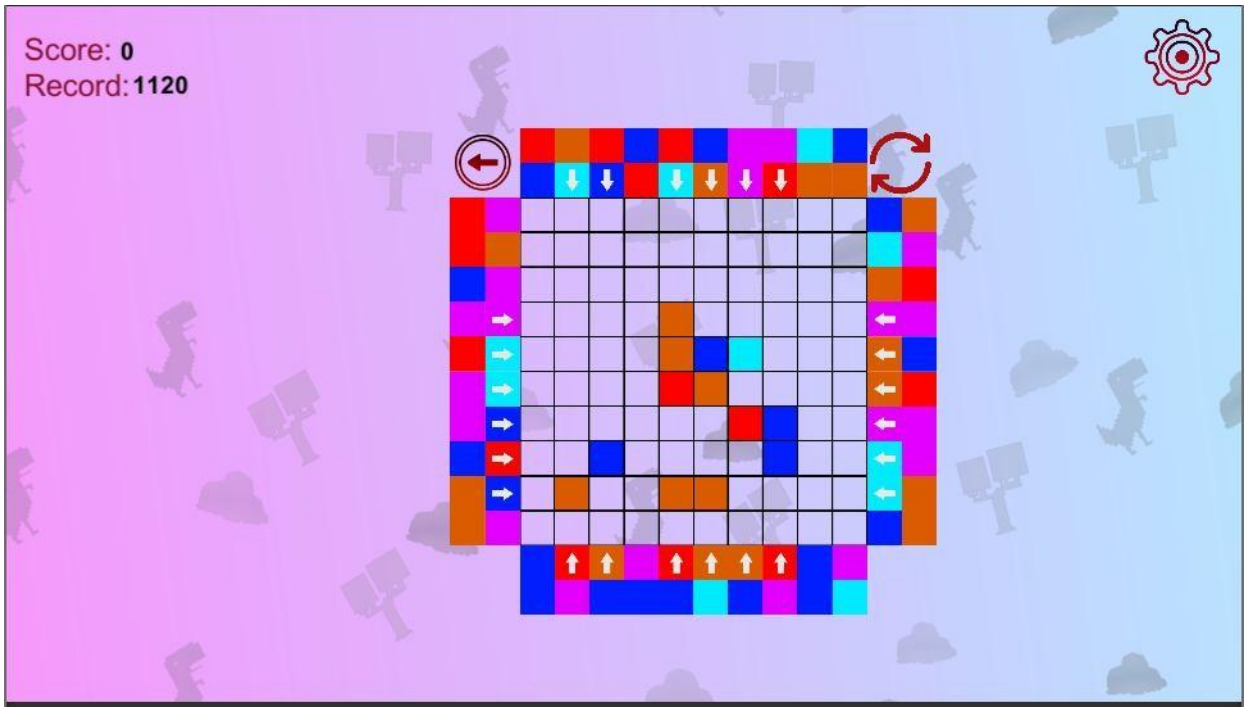

Рисунок 29 – Ігрове поле

Реалізація заповнення ігрового поля приведена нижче. Заповнення ігрового поля відбувається на старті програми, передаючи метод FillHVField() в метод void Start().

```
void FillHVField()
     {
          blocks = new GameObject[10, 10];
          for(int x = 0; x < 10; x++)
          {
               for (int y = 0; y < 10; y++)
               {
                    if (gameField[x, y] < 0) continue;

                    blocks[x          ,          y]          =
  GameObject.Instantiate(cubs[gameField[x , y]]);
                    blocks[x , y].transform.position =
                     new Vector2(x , 4 + y);
                    blocks[x , y].SetActive(true);
                   blocks[x , y].gameObject.tag =
                   "blockForClear";
               }
          }
}
```

Пересування блоків відбуваються по натисканні на об’єкт із стрілками.

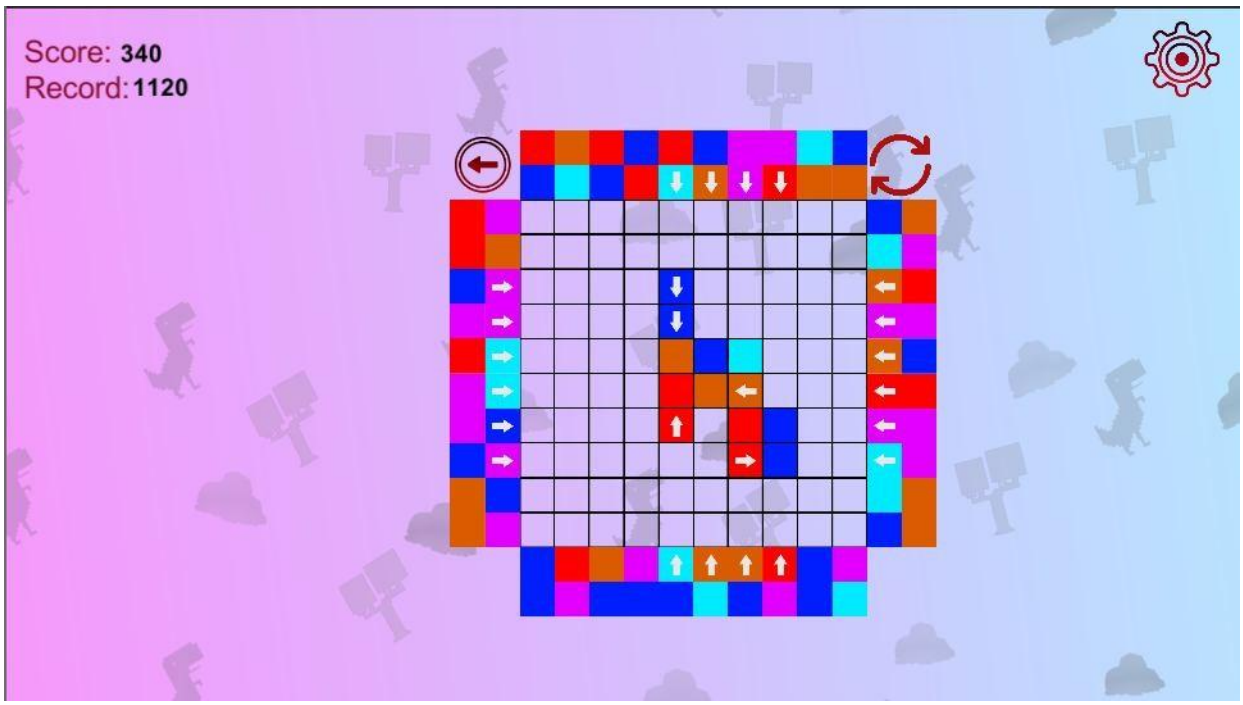

Рисунок 30 – Ігровий процес, набирання балів та натискання на об’єкти

```
private void OnMouseDown()
    {
        switch (tag)
        {
            case  "Ytop" :
                rb.MovePosition(rb.position +
                Vector2.down);
                //rb.tag = "blockForClear";
                break;
            case "Ydown":
                rb.MovePosition(rb.position + Vector2.up);
                //rb.tag = "blockForClear";
                break;
            case "Xright":
                rb.MovePosition(rb.position +
                Vector2.left);
                //rb.tag = "blockForClear";
                break;
            case "Xleft":
                rb.MovePosition(rb.position +
                Vector2.right);
                //rb.tag = "blockForClear";
                break;
            default:
                Debug.Log("Error:MoveOnClkick=!");
                break;

        }
```

Підключення до БД

Нижче наведено метод для з'єднання з БД, класу connection_db. Таблиця складається з 3 стовбців id, Name, Record.

```
    public void setConnection()
    {
        path = Application.dataPath +
        "/StrimAssets/mydb.bytes";
        dbConnection = new SqliteConnection("URI=file:" +
        path);
        dbConnection.Open();
        if(dbConnection.State == ConnectionState.Open)
        {
            SqliteCommand cmd = new SqliteCommand();
            cmd.Connection = dbConnection;
            cmd.CommandText = "SELECT * FROM Items";
            SqliteDataReader r = cmd.ExecuteReader();

            List<Item> items = new List<Item>();
            while (r.Read())
                items.Add(new Item(
                    int.Parse(String.Format("{0}", r[0])),
                    r[1].ToString(),
                    int.Parse(String.Format("0",r[2]))));
```

```
                Debug.Log(String.Format("{0} {1} {2}",
                r[0], r[1], r[2]));

            }
            else
            {
                Debug.Log("Error connection");
            }
        }
    }
```

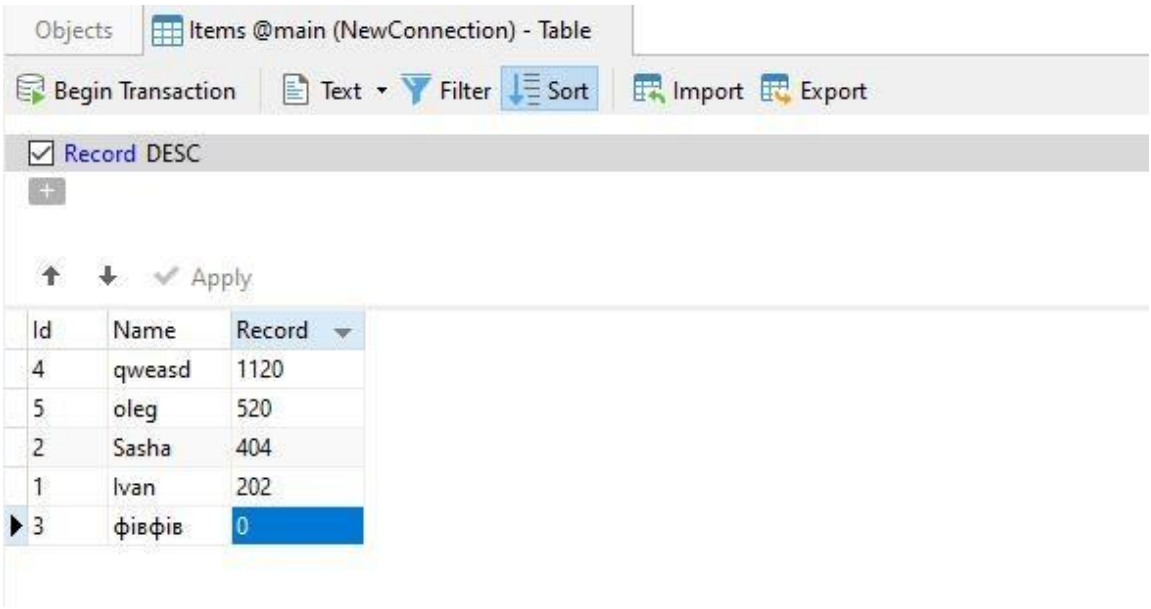

Рисунок 31 – База Даних

Ієрархія об’єктів Unity

На нашу сцену додаємо задній фон, Canvas з трьома текстовими полями, бали рекордні бали та самі слова «Score» «Record» та системні події.

В яких вже же модуль для обробки вхідних торкань, не забуваємо що це наша друга сцена, на першій сцені теж розташовуємо фон кнопку запуску гри та вхідне поле для ім’я користувача.

До нашої основної камери прив’язано головний скрипт, в якому всі ігрові блоки додаються методом перетаскування префабів та спрайтів в спеціально створені для цього місця.

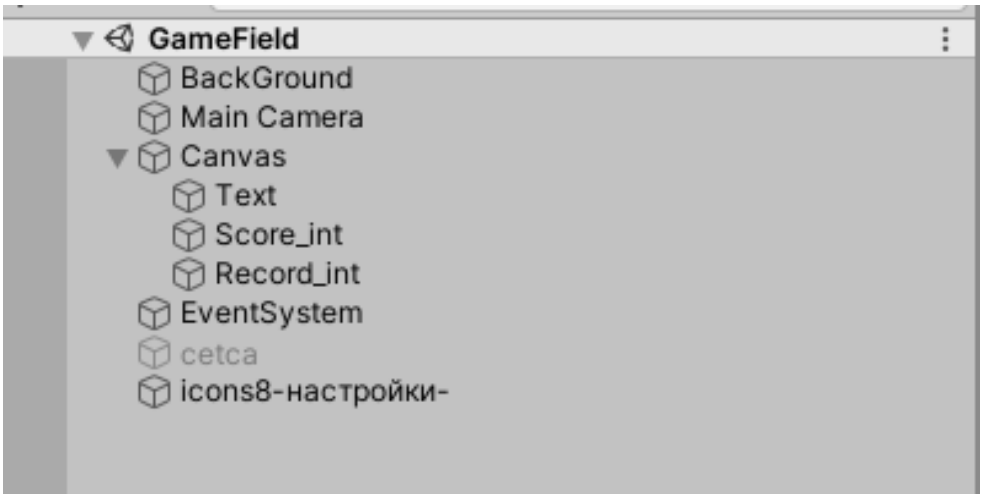

Рисунок 32 – Ігрові об’єкти ієрархія

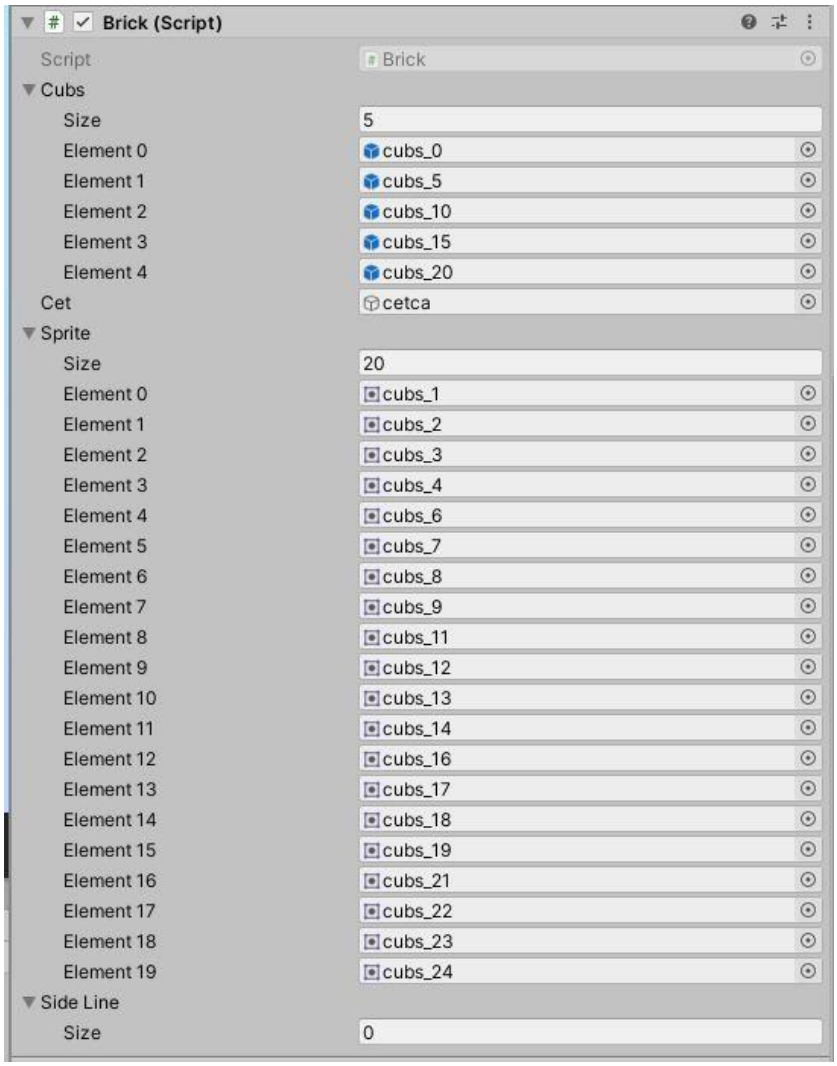

Рисунок 33 – Елементи додані до головної камери

4 ОПИС КІНЦЕВОГО ДОДАТКУ

Порівняння з метою проекту

Проект порівнюючи з метою, виконаний досить непогано і можна описати основні виконані дії. До виконаного функціоналу мобільного додатку можна віднести наступні позиції:

– присутня ідея та фунціонал гри;
– музичний та звуковий супровід з елементами нового стилю музики;
– анімація не тільки переміщення блоків, та об’єктів інтерфейсу;
– кнопки відміни останнього ходу та рестарт;
– відображення поточних балів та рекордних;
– мінімалістичний м’який фон, що дасть при багаточасовій грі вам набриднути;
– оновлений дизайн хоч і не на багато, але на пару років гра помолодшала;
– привітальна сцена, що характеризує собою початок гри та знайомство з користувачем ;
– гра в горизонтальному положенні, як найліпше впливає на ігровий процес;
– таблиця рекордів;

До можливого впровадження в гру можна виділити:

– додавання рівнів з підвищенням складності без можливості переходу до нових рівнів, якщо не пройдено поточний рівень;

Тестування ПП

В процесі розробки мобільної гри критичні системні помилки виправлялися одразу ж, середовище розробки Unity дозволяє запускати гру вже після 2 строк коду, тож одразу після збірки проекту гра була протестована на реальних пристроях і ніяких помилок не виявлено.

Була незначна помилка відображення на маленьких екранах мобільних пристроїв та вона була швидко виправлена.

Вікно збірки проекту

Після завершення проекту, проводимо деякі налаштування збірки, додавання сцен, режим відображення (портретний чи ландшафтний). Після збірки отримаємо файл с розширенням APK, який при запуску на мобільному пристрої встановить дану гру.

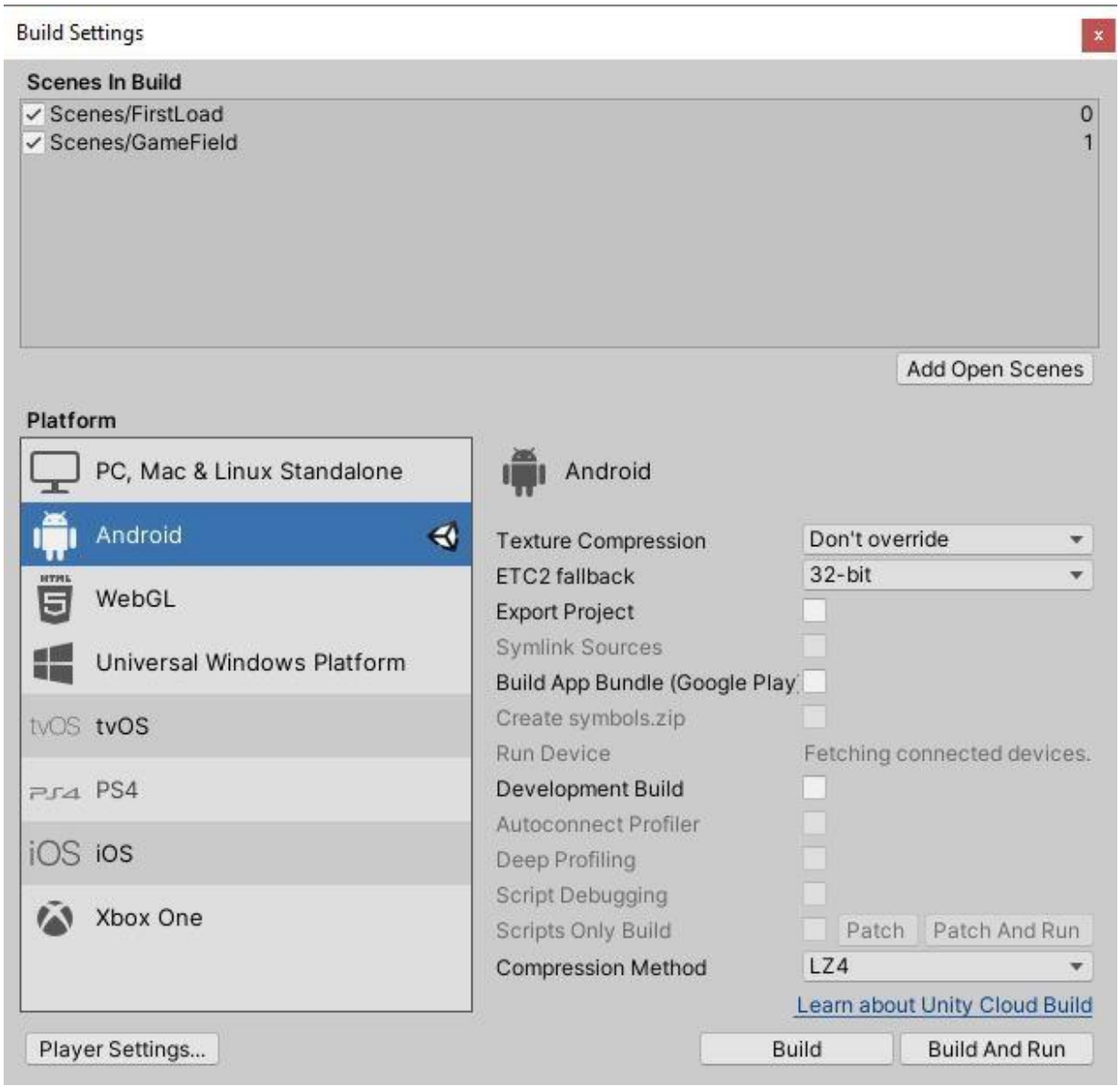

Рисунок 34 – Вибір платформи та збірка проекту

Можливе удосконалення та впровадження гри

Можливе додавання модулю котрий буде відповідати за з’єднання комп’ютерної версії з мобільною, що дасть змогу продовжити гру з іншого пристрою без обнуління поточної кількості балів, не проходити одні й ті ж рівні до декілька раз, а мати єдиний ігровий прогрес.

Подальше завантаження гри на різні андроїд платформи після додавання модулю реклами.

ВИСНОВКИ

Для початківців розробників ігрових додатків ОС Android ідеальна. Вона користується популярністю у великої аудиторії користувачів мобільних пристроїв та має у своєму розпорядженні потужні інструменти програмування. При незначних витратах розробка мобільних додатків може приносити хороший прибуток. Незважаючи навіть на високу конкуренцію, розробникам під силу досягти успіху, бо запити у користувачів швидко змінюються і ті додатки які були популярними зараз нікому не потрібні.

Результатами проектної частини диплому є:

– моделювання процесу розробки ПП;
– проведений аналіз аналогів додатку, на основі якого встановлені вимоги та сформовані цілі роботи;
– обрані технічні засоби для реалізації даного проекту;
– побудована діаграма діяльності додатку.

За результатами практичної частини виконано:

– створення графічних об’єктів для додатку, спрайти, фон, кнопки;
– оформлення сцен додатку;
– реалізовано функцію генератору блоків для випадкового формування рівнів гри;
– підключення до БД;

У подальшому планується принести до гри щось нове і незвичне та завантажити її до Google Play, також в рамках розвитку подальшого програмного продукту стане тестування та розширення функціоналу додатку для того ж комерційного використання

Отже, для створення дійсно вражаючої гри не потрібна команда із 100 чоловік, головне мати ідею, можливості реалізації та знати мову програмування для середовища розробки в якому ви бачите ваш проект. Ще не маловажливий фактор – це зусилля та час який вам потрібно віддавати для створення незвичних речей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Лучшие игры-головоломки для Android. URL: http://android.mobile-review.com/articles/47398/ (дата звернення 28.04.2020).
2. Лучшие игры «три в ряд» на Android. URL: https://cubiq.ru/luchshie-igry-tri-v-ryad-na-android/ (дата звернення 28.04.2020).
3. ТОП-10 ігор для Android, яким не потрібне з'єднання з інтернетом. URL: https://root-nation.com/ua/games-ua/game-articles-ua/ua-top-10-offline-android-games/ (дата звернення 30.04.2020).
4. Зароботок в Google Play. URL: https://vipidei.com/internet/mobilnye-prilozheniya/prodvizhenie-v-google-play/ (дата звернення 5.05.20).
5. C# — Преимущества и недостатки. URL: https://shwanoff.ru/plus-minus-c-sharp/ (дата звернення 6.05.20).
6. Java – взгляд изнутри. URL: https://tproger.ru/blogs/jvm-insides/ (дата звернення 5.05.20).
7. Android Studio IDE от Google. URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата звернення 3.04.2020)
8. ТОП 10 лучших игр на движке Unity. URL: https://gamedata.club/article-top/1359-top-10-luchshih-igr-na-dvizhke-unity.html (дата звернення 3.04.2020)
9. Середовище розробки Microsoft Visual Studio. URL: https://informatics.in.ua/programming_csharp/part_01.php (дата звернення 1.05.2020)
10. Adobe Photoshop CC. URL: https://itpro.ua/product/adobe-photoshop-cc/?tab=description (дата звернення 28.04.2020)
11. SQLite. URL: https://lecturesdb.readthedocs.io/databases/sqlite.html (дата звернення 25.04.2020).
12. MySQL — система управления базами данных. URL: https://web-creator.ru/articles/mysql (дата звернення 25.04.2020)

13. Ведення в UML 2.0. URL: http://bourabai.kz/dbt/uml/index.htm. (дата звернення 29.04.2020).

14. Діаграма діяльності. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності#cite_note-rumbaught-1 (дата звернення 10.05.2020)

ДОДАТОК А
Лістинг класів

```
using UnityEngine;
using System.Collections;
using System;
using Random = System.Random;
using Random1 = UnityEngine.Random;
using UnityEngine.UI;
using System.Linq.Expressions;
using Unity.Mathematics;
using System.Collections.Generic;

public class Brick : MonoBehaviour
{
     public GameObject[] cubs;
     public GameObject cet;
     public GameObject param;
     public GameObject reload;
     public Sprite[] sprite = new Sprite[20];
     public bool gun { set; get; }

     SpriteRenderer render;


     public int[,] gameField = new int[10,10]{
          {-1,-1,-1,-1,-1,-1,-1,-1,-2,-2},
          {-1, 1,-1,-1,-1,-1,-1,-1,-1,-1},
          {-1,-1, 2,-1,-1,-1,-1,-1,-1,-1},
          {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1},
          {-1, 1,-1,-1, 0, 1, 1,-1,-1,-1},
          {-1, 1,-1,-1, 1, 2,-1,-1,-1,-1},
          {-1,-1,-1, 0,-1, 4,-1,-1,-3,-1},
          {-1,-1, 2, 2,-1,-1,-1,-1,-2,-1},
          {-1,-1,-1,-1,-1,-1,-1,-1,-1,-1},
          {-1,-1,-1,-1,-1,-1,-1,-1,-1,-2},
     };

     public GameObject[,] blocks;
     public GameObject[] sideLine;
     void Start()
     {
          param.SetActive(true);
          reload.SetActive(true);
          cet.SetActive(true);
          FillHVField();
          FillSides();
          FillSprite();
     }

     void FillSprite()
    {
          int k = 0;
          for(int i = 0; i < 25; i++)
        {
               if (i % 5 == 0) continue;
               gameObject.GetComponent<SpriteRenderer>().sprite
                = sprite[k];
               k++;
```

```
                Debug.Log(k);
            }
        }
         void FillHVField()
         {
              blocks = new GameObject[10, 10];
              //blocks.SetValue()
              for(int x = 0; x < 10; x++)
              {
                   for (int y = 0; y < 10; y++)
                   {
                        if (gameField[x, y] < 0) continue;

                        blocks[x , y] =
                      GameObject.Instantiate(cubs[gameField[x , y]]);
//в массиве координаты по Y увеличиваются вниз, а в Unity вверх
                        blocks[x , y].transform.position =
                              new Vector2(x , 4 + y);
                        blocks[x , y].SetActive(true);
                        blocks[x , y].gameObject.tag =
                                                    "blockForClear";
                   }
              }
         }

         public GameObject RandomBlock()
         {
              Random rnd = new Random();
              int k = rnd.Next(0,4);
              return cubs[k];
         }

         private IEnumerator ShiftTilesDown(int x, int yStart,
                                            float shiftDelay = .03f)
         {
              IsShifting = true;
              List<SpriteRenderer> renders =
                                          new List<SpriteRenderer>();
              int nullCount = 0;

              for (int y = yStart; y < ySize; y++)
              {  // 1
                   SpriteRenderer render =
                         tiles[x, y].GetComponent<SpriteRenderer>();
                   if (render.sprite == null)
                   { // 2
                        nullCount++;
                   }
                   renders.Add(render);
              }

              for (int i = 0; i < nullCount; i++)
              { // 3
                   yield return new WaitForSeconds(shiftDelay);// 4
                   for (int k = 0; k < renders.Count - 1; k++)
                   { // 5
                        renders[k].sprite = renders[k + 1].sprite;
                        renders[k + 1].sprite = null; // 6
                   }
              }
```

```
     IsShifting = false;
}

void Update ()
{

}

void FillSides()
{

     sideLine = new GameObject[10];
     for (int y = 0; y < 10; y++)
     {
          //X left
          int rnd = Random1.Range(0, cubs.Length);
          int strelci = ((rnd + 1) * 4) - 4;
          sideLine[y] = GameObject.Instantiate(cubs[rnd]);
          for (int x = 0; x < 10; x++)
          {
               if (gameField[x, y] > -1)
               {

sideLine[y].GetComponent<SpriteRenderer>().sprite =
                                                  sprite[strelci];
               }

          }

          sideLine[y].transform.position = new Vector2(-1,
            y + 4);
          sideLine[y].SetActive(true);
          sideLine[y].gameObject.tag = "Xleft";
          //second X left
          rnd = Random1.Range(0, cubs.Length);
          sideLine[y] = GameObject.Instantiate(cubs[rnd]);
          sideLine[y].transform.position = new Vector2(-2,
            y + 4);
          sideLine[y].SetActive(true);
          sideLine[y].gameObject.tag = "sideX";
          //x right
          rnd = Random1.Range(0, cubs.Length);
          sideLine[y] = GameObject.Instantiate(cubs[rnd]);
          strelci = ((rnd + 1) * 4) - 3;
          for (int x = 0; x < 10; x++)
          {
               if (gameField[x, y] > -1)
               {

sideLine[y].GetComponent<SpriteRenderer>().sprite =
                                                  sprite[strelci];
               }

          }
          sideLine[y].transform.position = new Vector2(10,
            y + 4);
          sideLine[y].SetActive(true);
          sideLine[y].gameObject.tag = "Xright";
          //second x right
          rnd = Random1.Range(0, cubs.Length);
```

```
            sideLine[y] = GameObject.Instantiate(cubs[rnd]);
            sideLine[y].transform.position = new Vector2(11,
              y + 4);
            sideLine[y].SetActive(true);
            sideLine[y].gameObject.tag = "sideX";

            //Y down
            rnd = Random1.Range(0, cubs.Length);
            sideLine[y] = GameObject.Instantiate(cubs[rnd]);
            strelci = ((rnd + 1) * 4) - 1;
            for (int x = 0; x < 10; x++)
            {
                 if (gameField[y, x] > -1)
                 {

sideLine[y].GetComponent<SpriteRenderer>().sprite =
                                                  sprite[strelci];
                 }

            }
            sideLine[y].transform.position = new Vector2(y,
              3);
            sideLine[y].SetActive(true);
            sideLine[y].gameObject.tag = "Ydown";
            //second Y down
            rnd = Random1.Range(0, cubs.Length);
            sideLine[y] = GameObject.Instantiate(cubs[rnd]);
            sideLine[y].transform.position = new Vector2(y,
              2);
            sideLine[y].SetActive(true);
            sideLine[y].gameObject.tag = "sideY";

            //Y top
            rnd = Random1.Range(0, cubs.Length);
            sideLine[y] = GameObject.Instantiate(cubs[rnd]);
            strelci = ((rnd + 1) * 4) - 2;
            for (int x = 0; x < 10; x++)
            {
                 if (gameField[y, x] > -1)
                 {

sideLine[y].GetComponent<SpriteRenderer>().sprite =
                                                  sprite[strelci];
                 }

            }
            sideLine[y].transform.position = new Vector2(y,
              14);
            sideLine[y].SetActive(true);
            sideLine[y].gameObject.tag = "Ytop";
            //second Y top
            rnd = Random1.Range(0, cubs.Length);
            sideLine[y] = GameObject.Instantiate(cubs[rnd]);
            sideLine[y].transform.position = new Vector2(y,
              15);
            sideLine[y].SetActive(true);
            sideLine[y].gameObject.tag = "sideY";
        }
```

```
     }

using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UIElements;

public class moveOnClick : MonoBehaviour
{
    public static int getAxis;
    private Rigidbody2D rb;
    private Sprite sp;
    int _AxisX, _AxisY;
    moveOnClick mv;
    Renderer render;

    private GameObject GetAdjacent(Vector2 castDir)
    {
        RaycastHit2D hit = Physics2D.Raycast(transform.position,
castDir);
        if (hit.collider != null)
        {
            return hit.collider.gameObject;
        }
        return null;
    }

    private List<GameObject> GetAllAdjacentTiles()
    {
        List<GameObject> adjacentTiles = new List<GameObject>();
        for (int i = 0; i < adjDirections.Length; i++)
        {
            adjacentTiles.Add(GetAdjacent(adjDirections[i]));
        }
        return adjacentTiles;
    }
    private List<GameObject> FindMatch(Vector2 castDir)
    { // 1
        List<GameObject> matchingTiles = new List<GameObject>();
// 2
        RaycastHit2D hit = Physics2D.Raycast(transform.position,
castDir); // 3
        while (hit.collider != null &&
hit.collider.GetComponent<SpriteRenderer>().sprite ==
render.sprite)
        { // 4
            matchingTiles.Add(hit.collider.gameObject);
            hit =
Physics2D.Raycast(hit.collider.transform.position, castDir);
        }
        return matchingTiles; // 5
    }
    private void ClearMatch(Vector2[] paths) // 1
    {
        List<GameObject> matchingTiles = new List<GameObject>();
// 2
        for (int i = 0; i < paths.Length; i++) // 3
        {
            matchingTiles.AddRange(FindMatch(paths[i]));
```

```
        }
        if (matchingTiles.Count >= 2) // 4
        {
            for (int i = 0; i < matchingTiles.Count; i++) // 5
            {
matchingTiles[i].GetComponent<SpriteRenderer>().sprite = null;
            }
            matchFound = true; // 6
        }
    }
    public void ClearAllMatches()
    {
        if (render.sprite == null)
            return;

        ClearMatch(new Vector2[2] { Vector2.left, Vector2.right
});
        ClearMatch(new Vector2[2] { Vector2.up, Vector2.down });
        if (matchFound)
        {
            render.sprite = null;
            matchFound = false;
            SFXManager.instance.PlaySFX(Clip.Clear);
        }
    }



    void Start()
    {

        rb = GetComponent<Rigidbody2D>();
        sp = GetComponent<Sprite>();
    }

    private void OnMouseDown()
    {
        switch (tag)
        {
            case  "Ytop" :
                rb.MovePosition(rb.position + Vector2.down);
                rb.tag = "blockForClear";

                break;
            case "Ydown":
                rb.MovePosition(rb.position + Vector2.up);
                rb.tag = "blockForClear";
                break;
            case "Xright":
                rb.MovePosition(rb.position + Vector2.left);
                rb.tag = "blockForClear";
                break;
            case "Xleft":
                rb.MovePosition(rb.position + Vector2.right);
                rb.tag = "blockForClear";
                break;
            default:
                Debug.Log("Error:MoveOnClkick!!!");
                break;
```

```
        }


    }
    private IEnumerator ShiftTilesDown(int x, int yStart, float
shiftDelay = .03f)
    {
        IsShifting = true;
        List<SpriteRenderer> renders = new
List<SpriteRenderer>();
        int nullCount = 0;

        for (int y = yStart; y < ySize; y++)
        {  // 1
            SpriteRenderer render = tiles[x,
y].GetComponent<SpriteRenderer>();
            if (render.sprite == null)
            { // 2
                nullCount++;
            }
            renders.Add(render);
        }

        for (int i = 0; i < nullCount; i++)
        { // 3
            yield return new WaitForSeconds(shiftDelay);// 4
            for (int k = 0; k < renders.Count - 1; k++)
            { // 5
                renders[k].sprite = renders[k + 1].sprite;
                renders[k + 1].sprite = null; // 6
            }
        }
        IsShifting = false;
    }

    // Update is called once per frame
    void Update()
    {
        previousSelected.ClearAllMatches();
        ClearAllMatches();
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class nextEtap : MonoBehaviour
{
    public void Etap()
    {
        SceneManager.LoadScene("GameField");
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class Volume : MonoBehaviour
{
    private AudioSource audioSrc;
    private float musicVolume = 1f;
    void Start()
    {
        audioSrc = GetComponent<AudioSource>();
    }
    void Update()
    {
        audioSrc.volume = musicVolume;
    }

    public void SetVolume(float vol)
    {
        musicVolume = vol;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class connection_db : MonoBehaviour
{
    public SqliteConnection dbConnection;
    private string path;

    public class Item
    {
        public int Id;
        public string Name;
        public int Record;
        public Item(int _id, string _name, int _record)
        {
            Id = _id;
            Name = _name;
            Record = _record;
        }
    }
    void Start()
    {

    }

    void Update()
    {

    }

    public void setConnection()
    {
        path = Application.dataPath + "/StrimAssets/mydb.bytes";
        dbConnection = new SqliteConnection("URI=file:" + path);
        dbConnection.Open();
```

```
            if(dbConnection.State == ConnectionState.Open)
            {
                SqliteCommand cmd = new SqliteCommand();
                cmd.Connection = dbConnection;
                cmd.CommandText = "SELECT * FROM Items";
                SqliteDataReader r = cmd.ExecuteReader();

                List<Item> items = new List<Item>();
                while (r.Read())
                    items.Add(new Item(
                        int.Parse(String.Format("{0}", r[0])),
                        r[1].ToString(),
                        int.Parse(String.Format("0",r[2]))));
                    Debug.Log(String.Format("{0} {1} {2}", r[0],
                                                    r[1], r[2]));

            }
            else
            {
                Debug.Log("Error connection");
            }
        }
    }
}
```