МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____В.П. Гамаюн

«_____»_____2021 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Тема: «Тривимірна мобільна відеогра "Paint Cube" на платформі Unity 3D»

Виконавець: Фридрих Павло Петрович

Керівник: Зудов Олег Миколайович

Нормоконтролер: Боровик Володимир Миколайович

Київ 2021

ВСТУП

Історія розвитку відеоігор починається з січня 1962 року, коли декілька програмістів з Массачусетського технологічного інституту написали програму, яка через місяць перетворилася в першу цифрову гру SpaceWar і мала шалений успіх серед студентів інституту. В ті роки думки про компактні ЕОМ були на рівні мрій, тому що комп'ютери фізично були дуже великими.

Пройшло не так багато часу і на початку 10-х років 21-го століття в життя простих людей ввійшли смартфони з достойним рівнем продуктивності апаратного та програмного забезпечення. Масове поширення смартфонів не могло пройти повз індустрію відеоігор і ці роки справедливо вважаються ренесансом мобільних ігор, тому що раніше кнопкові телефони зі слабою продуктивністю сильно обмежували розробників і не давали змоги робити щось по-справжньому цікаве та якісне.

Велику роль в розвитку мобільних відеоігор зіграла діджиталізація та глобалізація: розробники отримали можливість продавати свої продукти в цифровому вигляді у всьому світі. Головними платформами для реалізації мобільних ігор стали Play Market від Google та App Store від Apple.

Сьогодні важко знайти смартфон, на якому не установлено хоча б однієї гри і цей факт дає вичерпну відповідь на питання щодо актуальності теми розробки мобільних відеоігор.

Завданням дипломного проекту є створення тривимірної мобільної відеогри.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ РОЗРОБКИ МОБІЛЬНИХ ІГОР

1.1. Концептуальний дизайн

Відеогра, як і будь-який творчий продукт, починається з ідеї. Коли розробник має достатній багаж знань, хороше розуміння своєї предметної області та практичний досвід, він спроможний генерувати перспективні комплексні ідеї, які можуть перерости в готовий продукт.

Хоча відеогри – це один з жанрів творчості, але це все ще комерційні технічні продукти, на розробку та реалізацію яких необхідні значні ресурси. Тому перед тим, як приступити до безпосередньої розробки гри, автор (найчастіше це гейм-дизайнер) пише короткий і лаконічний концепт-документ (займається концептуальним проектуванням), в якому будуть описані всі основні особливості майбутньої гри. Концепт-документ презентується всій команді розробників і якщо об’єм, перспективи та рівень складності підходить для команди, розпочинається розробка.

1.2. Ігровий дизайн

Ігровий дизайн або гейм-дизайн (англ. game design) – процес створення форми та змісту ігрового процесу, який частіше називається геймплей (англ. геймплей) [1]. Гейм-дизайн – це проектування гри. Головним інструментом гейм-дизайну є дизайн-документ (англ. design document), в якому описані всі необхідні для подальшої розробки правила та принципи гри.

Ігровий дизайн визначає набір можливих варіантів розвитку подій, умови перемоги та поразки, правила контролю гравцем ігрового світу, складність тощо.

Гейм-дизайн можна умовно поділити на:

- дизайн ігрового світу – глобальний дизайн ігрового світу, його правила, принципи, атмосфера, настрій. Якщо в грі є інші, крім протагоніста, персонажі, то мають бути описані їхня поведінка, соціальні взаємодії та приблизний характер (агресивний, дружелюбний, нейтральний тощо).
- системний дизайн – процес створення математичних ігрових систем, які мають чіткий набір правил і змінюють вихідні значення в залежності від вхідних параметрів, які вносить гравець.
- дизайн рівнів – дизайн ігрового світу на більш низькому, локальному рівні. Гейм-дизайнер в даному випадку описує загальні характеристики рівнів, безпосередньо не займаючись моделюванням.
- дизайн контенту – проектування правил та характеристик для ігрових предметів, місій, квестів, звукового оформлення, освітлення.

Головною задачею гейм-дизайнера є розробка дизайнерської документації та використання правильних підходів, щоб донести свою думки, ідеї та дизайн до команди розробників. Дизайн-документ – це технічна документація, яка пишеться розробниками для розробників. Хоча цей факт мав би стандартизувати документацію такого типу, але на даний момент немає єдиного еталону для дизайн-документів і кожна студія розробляє документацію так, як їй зручно. Розробка дизайн-документу – це другий, після генерації ідеї та постановки задач, етап розробки відеоігор. Гейм-дизайнер готує документацію, майже завжди це відбувається в декілька етапів, щоб передати її розробникам-програмістам і почати процес створення прототипу гри.

Невеликі команди, на відміну від великих бюрократичних студій, можуть дозволити собі зібрати в одній кімнаті всіх членів команди і провести зустріч, на якій гейм-дизайнер чи продюсер проекту детально розповість про майбутню гру, тим самим донесе до кожного глобальне бачення проекту. Такий метод дуже ефективний і дозволяє всім учасникам прийняти участь в

обговоренні. В 99% випадків такі обговорення приводять до змін і покращень в оригінальному дизайні.

Дизайнерська документація, в тому числі, допомагає оцінити фінансові витрати на проект, визначити терміни розробки та зрозуміти, чи команда взагалі в змозі осилити проект заявленого масштабу.

Дизайн-документ не є статичним: він коригується залежно від актуального стану розробки, вимог продюсера чи директора, пропозицій інших розробників й результатів альфа- та бета-тестування.

Гейм-дизайнер в обов'язковому порядку контролює процес розробки на всіх етапах, тестує реалізований функціонал та стежить за якістю продукту.

Прикладом хорошого гейм-дизайну може слугувати гра Gardenscapes від компанії Playrix (рис.1.1.). В ній поєднана вже звична для казуальних гравців механіка «три в ряд», цікаві побічні головоломки, персонажі, які запам'ятовуються, та нетривіальний сюжет. Gardenscapes – гра з чудовим комплексним гейм-дизайном, який створює позитивний унікальний досвід гравця.

Рис.1.1. Мобільна відеогра «Gardenscapes»

1.3. Наративний дизайн

Наративний дизайн (англ. narrative design) – це проектування і написання історії ігрового світу [2]. Глобальна історія ігрового світу включає в себе локальні історії місцевості, ігрових рівнів, характери персонажів, діалоги та наповнення сюжетних завдань. Спеціаліст, який займається такою діяльністю, називається наративним дизайнером.

В мобільних іграх сюжет часто або дуже простий, або взагалі відсутній. Це пояснюється максимальним спрощенням ігрового процесу, щоб гравець міг сконцентруватися на одній механіці і не відволікатися.

Хоча мобільні казуальні ігри з хорошим наративним дизайном серед гравців цінуються більше, адже в такому випадку гра розказує певну історію і має багатошарову модель подачі нової інформації та ігрового досвіду. Також сюжетні ігри частіше поширюються за преміум моделлю (одноразова покупка), а не є безкоштовними продуктами з купою реклами.

Окремим жанром відеоігор, в тому числі мобільних, є візуальна новела. Візуальна новела – це жанр інтерактивного мистецтва, який знаходиться на стику відеоігор та літератури. Це інтерактивна історія з нелінійними діалогами, як правило двовимірною графікою, звуковим та музичним оформленням. Саме жанр візуальних новел потребує якісної кропіткої роботи наративного дизайнера.

Прикладом хорошого наративного дизайну може слугувати гра Beholder 2 Lite (рис.1.2.) – мобільна гра популярної франшизи Beholder. Відеогра має глибокий наративний дизайн світу, персонажів, яким хочеться співчувати, низку моральних виборів та захопливий сюжет.

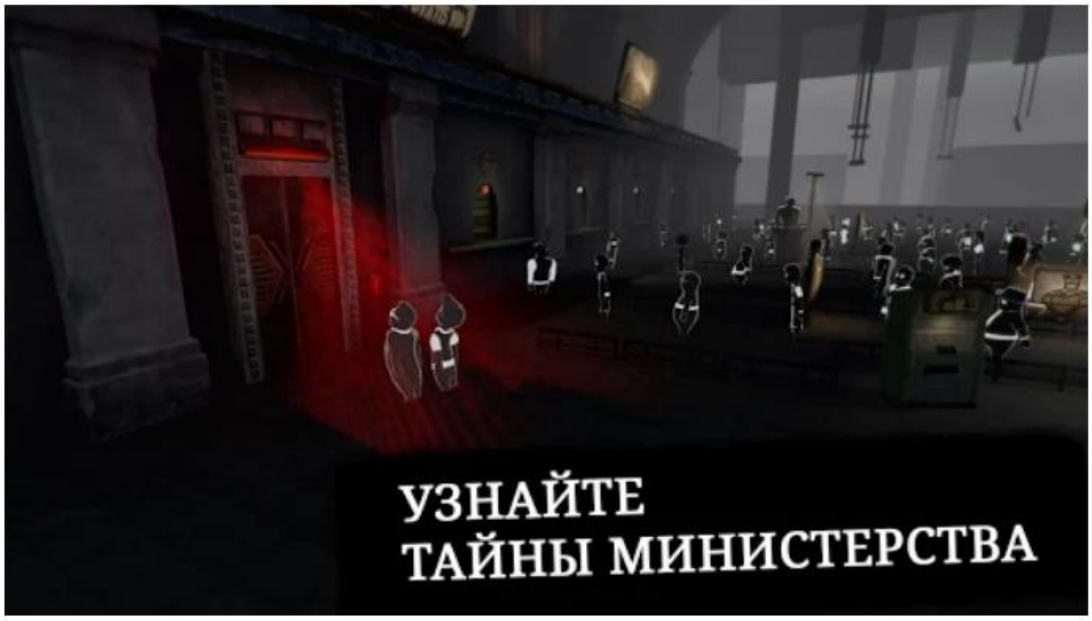

Рис. 1.2. Мобільна гра Beholder 2 Lite

1.4. Програмування та робота з платформою розробки

Після того, як програміст отримав дизайн-документ, він готовий до початку розробки. Спочатку він розробляє архітектуру майбутньої гри, враховуючи задані параметри. У випадку з мобільними іграми обов’язково вибирається операційна система чи системи та моделі смартфонів, на яких буде працювати продукт.

На етапі проектування розробляються макети ігрових сцен, кожна з яких пізніше перетвориться в ігровий рівень, меню, сторінку статистики гравця тощо [4]. В даному етапі програміст зазвичай ще не має фінальних графічних об’єктів та дизайну ігрових рівнів, тому використовує «заглушки» - прості геометричні фігури, яких достатньо для створення макетів сцен.

Після створення та першого тестування прототипу програміст приступає до наповнення гри контентом: графічним, наративним, звуковим.

Робота програміста не закінчується, поки не будуть протестовані та виправлені всі баги, помилки та неточності гри.

1.5. Графічний дизайн

Графічний дизайн (англ. graphic design) – це художньо-проектна діяльність, яка спрямована на створення гармонійного та ефективного візуально-комунікативного середовища [3].

До задач графічного дизайну відносяться:

- розробка інтерфейсу;
- ілюстрування та дизайн персонажів;
- створення ігрового світу з його унікальною атмосферою;
- проектування навігації;
- створення графічний ефектів.

Візуальний контент грає важливу роль в правильному сприйнятті ігрового світу гравцем, тому що більшість інформації людина сприймає за допомогою зору. Стилістично правильний графічний дизайн створює особливу атмосферу гри, її унікальний характер. Чудовим прикладом для ілюстрації цієї тези є гра Ninilumbra (рис.1.3.), графіка якої виконана з використанням блідих приглушених кольорів, що задає відповідний настрій.

Рис.1.3. Мобільна гра Ninilumbra

Робота графічного дизайнера починається майже одночасно з роботою гейм-дизайнера, виражається це в створенні концептуальних малюнків, на основі яких пізніше формується візуальний стиль гри.

1.5. Звуковий дизайн та музичне оформлення

Для мобільних ігор властивий простий і лаконічний звуковий дизайн, який не буде відволікати гравця від ігрового процесу. Основними звуковими ефектами при такому підході є звуки натискання на екран в різних варіаціях, звуки інтерфейсу та звуки реакцій ігрових об’єктів на взаємодію зі сторони гравця.

Музичне оформлення в більшості мобільних ігор відсутнє або дуже просте, хоча серйозні комплексні ігри, особливо ігри-сервіси, завжди прискіпливо підходять до вибору саундтреку, тому що розуміють, що хороший інтерактивний аудіо-візуальний досвід гравця неможливий без підходящої якісної музики.

Прикладом хорошого звукового дизайну та музичного оформлення може слугувати гра Don’t Starve: Pocket Edition – досить складний симулятор виживання в дикій природі, розрахований на аудиторію досвідчених гравців. Як зрозуміло з вище описаного, гра має похмуру і далеко не мажорну атмосферу, яку потрібно підтримати хорошим звуком та саундтреком. Звукові дизайнери та композитори команди розробників справились з цією задачі прекрасно – звук в Don’t Starve: Pocket Edition з перших секунд налаштовує на відповідний лад.

Рис.1.4. Мобільна гра Don’t Starve: Pocket Edition

1.6. Маркетинг та монетизація

Доля доходу мобільних відеоігор від загального доходу ігрової індустрії вже з 2018-го року становить більше 50% відсотків та існує тенденція до подальшого росту (рис. 1.5.), що говорить про великі перспективи даного виду інтерактивних розваг.

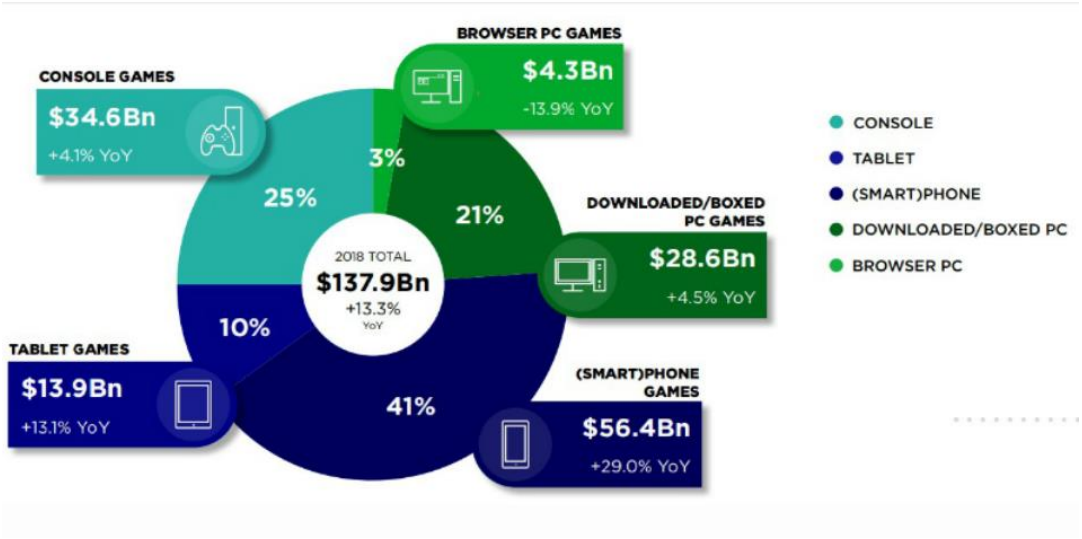

Рис.1.5. Розподіл доходу ігрової індустрії за 2018 рік за платформами

На даний момент на ринку мобільних пристроїв лідирують дві операційні системи: iOS та Android. Вже традиційно iOS користується більшою популярністю в Європі та США, в той час як Android вибирають жителі Азії, в тому числі Китаю. Китай є абсолютним лідером за кількістю мобільних гравців, особливо якщо мова йде про free-to-play ігри. Користувачі iOS також в більшості грають в free-to-play ігри, але для них нормальною практикою є покупка преміум ігор.

В чому різниця між free-to-play та преміум-іграми? Монетизація, тобто отримання прибутку, безкоштовних ігор будується, в основному, на інтеграції реклами в ігровий процес. Це можуть бути рекламні ролики між рівнями, банерна реклама, яка розміщується зверху або знизу екрану, нативна реклама.

Часто ігровий процес побудований таким чином, що навіть якщо гравець програє, то він має змогу продовжити рівень шляхом перегляду рекламного ролику.

Розробники дають змогу гравцям повністю відключити рекламу за фіксовану плату. В цей момент free-to-play гра умовно переходить в категорію преміум.

Середні та великі мобільні ігри, тобто з тривалістю ігрового процесу більше 5 годин, зазвичай інтегрують в свій продукт ігровий магазин, де за ігрову валюту можна купити додаткові бонуси, скіни (англ. skin), тобто зміну зовнішнього вигляду певних ігрових об'єктів чи придбати додатковий контент.

Цікавим феноменом сучасного мобільного геймінгу є зменшення часу для отримання нагороди, що є необхідним для частого й постійного викиду в кров дофаміну, який, в свою чергу, приносить гравцю задоволення [5]. Тому розробники мобільних ігор зменшують ігрові цикли, адаптують складність гри під конкретного гравця, завалюють його різноманітними бонусами та намагаються зробити все для того, що гравець ні в якому разі не сумував і не мав часу подумати про закриття гри.

Хорошим способом збільшити відсоток постійних гравців є реалізація щоденних бонусів. Наприклад, кожного дня, після першого запуску гри, людина отримує бонус. А якщо запускає гру три дні підряд – отримує супер-бонус. При тому, що існує тенденція до зменшення часу, який потрібен для отримання регулярних бонусів, інколи такий відрізок часу займає всього декілька хвилин.

Сучасні мобільні ігри за принципами роботи з увагою та часом гравця наслідують вже майже забуті браузерні та соціальні ігри, які були популярні в нульових та на початку десятих років. На початку нульових інтернет ставав все більш доступним для населення, що привело до стрімкого росту та розвитку браузерних ігор, ключовою особливістю яких стала соціальна взаємодія з реальними гравцями.

В середині нульових набирають популярність соціальні мережі, в яких дуже швидко знайшлося місце для відеоігор, які максимально використовували соціальну складову мережі і старалися перетворити якомога

більшу кількість користувачів в гравців. Пізніше ці ігри назвуть «соціальними».

З вище написаного слідує, що хороша мобільна гра має бути не тільки цікавою та якісною, а й використовувати сучасні підходи до маркетингу і монетизації, щоб бути комерційно успішною.

1.7. Тестування

Тестування мобільних відеоігор має одну характерну відмінність, яка відрізняє цей сектор індустрії від класичних консольних ігор та ігор для персональних комп'ютерів: тестувати продукт доводиться на великій кількості різних моделей смартфонів. Тому що гра, яка чудово працює на бюджетному пристрої однієї компанії, може мати серйозні проблеми з продуктивністю роботи навіть на флагманській моделі іншого виробника. Хоча останні декілька років компанії-виробники апаратного забезпечення для смартфонів і рухаються в сторону глобальної стандартизації своєї продукції, але кожен хоче створювати продуктивні системи з мінімальними витратами ресурсів, що призводить до постійного розвитку технологій і розходжень в підходах між різними компаніями.

Тестування мобільних відеоігор може відбуватися як на реальних пристроях різних моделей, так і за допомогою програмних емуляторів (рис.1.6.). Звичайно, перший варіант більш надійний. Також перевагою тестування ігор на реальних пристроях є можливість виміряти витрати електроенергії, що є критичним фактором для мобільного геймінгу.

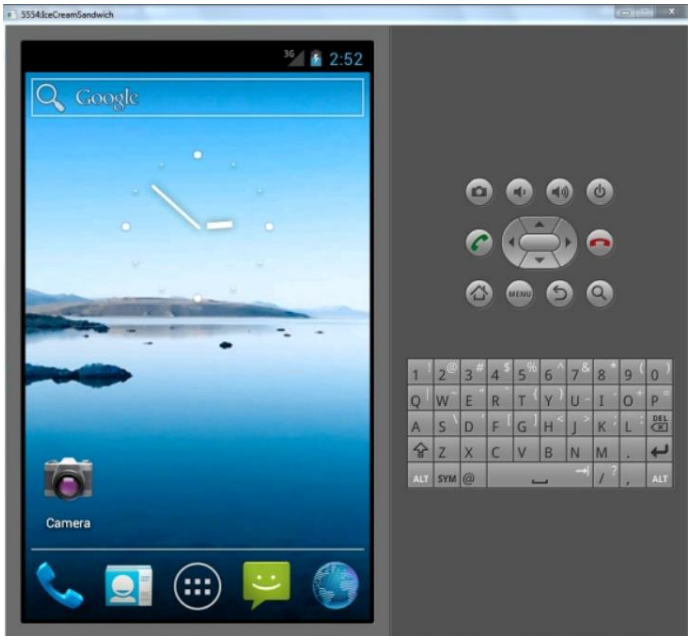

Рис1.6. Приклад програмного емулятора

Спочатку проводиться внутрішнє тестування силами команди розробників, в тому числі професійними тестувальниками, результатом якого є створення внутрішнього списку помилок (англ. bag report) для подальшого доопрацювання гри. Далі проводиться закрите або відкрите альфа-тестування із залученням обмеженої кількості гравців, знайдені неточності виправляються. Перед безпосереднім випуском гри проводиться відкрите бета-тестування, в якому беруть участь багато гравців. Поширеною практикою є нагородження бета-тестувальників додатковим ігровим контентом чи рідкісними предметами, це робиться для того, щоб залучити більше тестувальників та, результаті, випустити гру в мінімальною кількістю помилок та неточностей.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ВИБІР ТЕХНОЛОГІЙ

2.1. Аналіз платформ для розробки відеоігор

Платформа для розробки або ігровий рушій (англ. game engine) представляє собою середовище для розробки відеоігор, яке включає в себе всі необхідні програмні модулі, які можуть знадобитися в процесі роботи.

В першу чергу, платформа дає змогу працювати з програмним кодом. Для розробки сучасних відеоігор використовуються в більшості використовуються обєктно-орієнтовані мови програмування, які можна використовувати для написання скриптів.

Скрипт – цілісний фрагмент програмного коду, який виконує певну задачу. Наприклад, для тривимірних ігор вважається правильним реалізовувати управління камерою окремим скриптом, а не прописувати відповідні функції в коді управління персонажем. Скрипти дають змогу декомпозувати програмний код.

Найпопулярнішими мовами програмування відеоігор є C++ та C#. Інколи використовуються JavaScript, Lua, Boo. Для реалізації серверної частини підходять PHP та Python (рис.2.1).

Рис.2.1. Найпопулярніші мови програмування відеоігор

Ігрові студії світового рівня в більшості випадків розробляють свою власну платформу, яку буде максимально відповідати вимогам та

особливостям конкретного проекту. Для маленьких студій чи соло-розробників не вигідно витрачати час і ресурси на розробку власної платформи, тому вони вибирають одну з уже існуючих універсальних платформ.

Існує дві універсальні платформи для розробки ігор, які підходять для більшості проектів: Unreal Engine та Unity 3D.

2.2. Платформа розробки Unreal Engine

Unreal Engine (рис.2.2.) працює на мові програмування C++, має можливість дуже тонкого й професійного налаштування освітлення ігрових сцен, вміє працювати з високополігональними об’єктами та текстурами дуже високої якості. Платформа розрахована на великі проекти для персональних комп’ютерів та ігрових консолей. Unreal Engine має зручний інтуїтивно зрозумілий інструмент Blueprint, який створений для програмування ігрових об’єктів та подій без використання програмного коду, що дозволяє навіть не програмістам реалізовувати деякі нескладні макети гри та тестувати механіки.

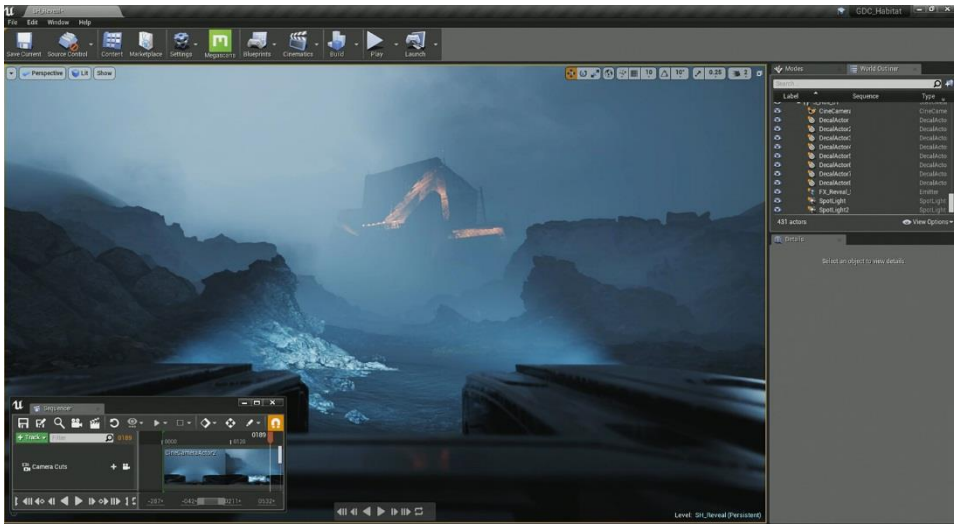

Рис.2.2. Графічний інтерфейс Unreal Engine

Unreal Engine рідко використовують для роботи над мобільними іграми, тому що платформа занадто комплексна і для компактних швидких проектів підходить не дуже. Навпаки, деякі ігри AAA класу були створені саме на Unreal Engine. Яскравим прикладом цього є гра жанру шутер (англ.

shooter) Stalker 2 від українських розробників GDC Game World, яка на даний момент розробляється на платформі Unreal Engine.

Особливістю монетизації платформи є те, що вона безкоштовна для використання, але розробники мають віддавати частину свого прибутку після випуску гри в продаж.

Платформа має власний магазин асетів (англ. asset) UE Marketplace, де спеціалісти, які пов’язані з розробкою ігор, можуть виставляти на продаж власні асети: це можуть бути тривимірні моделі, двовимірні спрайти, звукові ефекти, музика, програмні плагіни, шейдери та багато іншого. Розробники всіх масштабів користуються сторонніми асетами, тому що часто набагато дешевше купити вже готову підходящу тривимірну модель чи музичну композицію, аніж шукати людину для її створення чи учитися створювати щось подібне власноруч.

Отже, Unreal Engine не зовсім підходяща платформа для розробки мобільної гри, так як вона забирає частину прибутку після виходу гри на ринок та спроектована для розробки більш масштабних проектів.

2.3. Платформа розробки Unity 3D

Unity 3D (рис.2.3.) підтримує C#, Javascript та Boo, але переважна більшість розробників використовують C#, через його широкий спектр можливостей, гнучкість та актуальність. Unity 3D має велику, більшу ніж в Unreal Engine, спільноту розробників, а отже й більший магазин асетів, який називається Unity Asset Store.

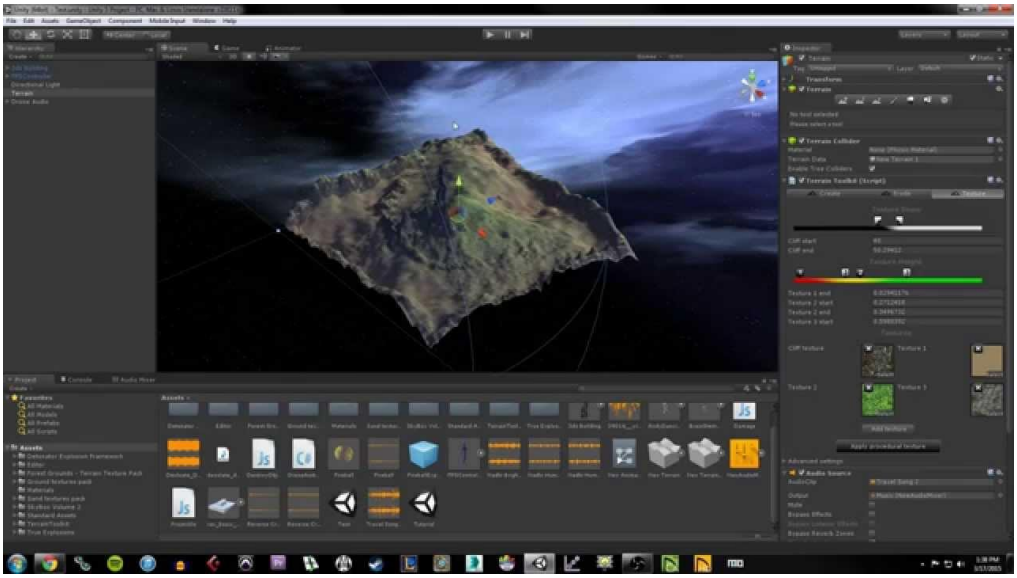

Рис.2.3. Графічний інтерфейс Unity 3D

Дана платформа відносно легка у вивченні, інтуїтивно зрозуміла та розрахована невеликі та середні за масштабом проекти. Саме тому переважна більшість мобільних ігор, та інді-ігор класу А та АА розроблялися та розробляються саме на Unity 3D.

Завдяки Unity Asset Store та великій спільноті розробників, програміст-початківець може зібрати свою власну гру самотужки, використовуючи виключно свій програмний код та безкоштовні асети з магазину. Звичайно, якість, стиль та загальна атмосфера такої гри буде не на найвищому рівні, але це вже готовий продукт, який можна розмістити в портфоліо і рухатися далі, а при високому рівні розуміння відеоігор можна розглядати варіант з випуском гри на ринок.

Отже, Unity 3D – оптимальна платформа для розробки тривимірної мобільної гри.

2.4. Редактор коду та система контролю версій

Платформа розробки Unity3D пропонує для роботи з кодом стандартний інтегрований редактор коду, але в більшості випадків його функціоналу не достатньо для організації комфортної роботи. Тому потрібно обрати більш просунутий варіант, а саме Visual Studio Code.

Visual Studio Code (рис.2.4.) - це редактор вихідного коду, розроблений компанією Microsoft для Windows, Linux та macOS. Головними особливостями редактора є підтримка налагодження, підсвічування синтаксису, інтелектуальне завершення коду та фрагментів коду, рефакторинг коду та вбудований Git. Користувачі можуть змінювати візуальну тему, комбінації клавіш, налаштування та встановлювати розширення, що додають додаткову функціональність. Вихідний код - вільний та відкритий, випущений згідно з ліцензією MIT. У опитуванні розробників Stack Overflow 2019 Visual Studio Code потрапив до найпопулярнішого інструменту середовища для розробників, 50,7% із 87 337 респондентів заявили, що використовують його.

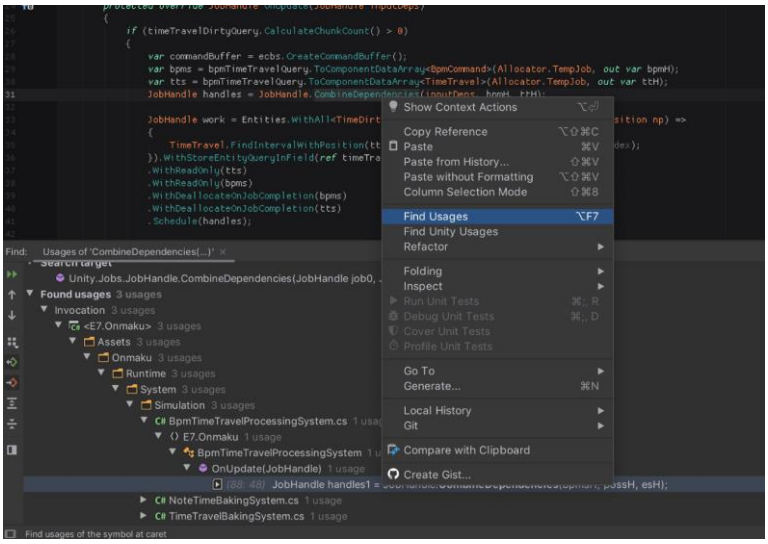

Рис.2.4. Інтерфейс Visual Studio Code

Процес розробки гри планується мовою програмування C#, на даний момент це єдиний адекватний варіант для платформи Unity.

З недавнього часу розробники Unity інтегрували в архітектуру своєї платформи найпопулярнішу в світі систему контролю версій Git, що, без сумніву, значно полегшило роботу.

Git – це програмне забезпечення для відстеження змін у будь-якому наборі файлів, яке зазвичай використовується для координації роботи між програмістами, які спільно розробляють вихідний код під час розробки програмного забезпечення (рис.2.5.). Цілі даної системи включають в себе швидкість, цілісність даних та підтримку розподілених нелінійних робочих

процесів (можливе створення та паралельна робота над тисячами гілок розробки).

Рис.2.5. Інтерфейс Git

2.5. Графічний редактор та підходи до створення графічного контенту

Обраний стиль графічного контенту для даної відеогри – піксель-арт.

Піксель-арт – це вид цифрового мистецтва, яке створюється за допомогою програмного забезпечення, найчастіше графічних редакторів, де зображення редагується на рівні пікселів. Естетика цього виду графіки походить від 8-розрядних та 16-розрядних комп'ютерів, ігрових консолей та відеоігор, а також від інших систем, які мають графічні обмеження. Більшість піксельних зображень мають дуже обмежену палітру кольорів, в деяких випадках кольорів може бути всього два. Яскравим прикладом мінімалістичного піксель-арту є графіка відеоігор для портативних консолей Game Boy (рис.2.6.).

Рис.2.6. Гра Shadow Warriors

Існує два основних підходи в створенні піксель-арту: векторний та растровий. Як зрозуміло з назви, векторний підхід передбачає використання графічних редакторів векторної графіки, а растровий – растрової. Також існує програмний спосіб створення піксель-арту, коли замість графічного редактору використовуються графічні можливості середовища розробки, наприклад графічний модуль в Dev C++, але це занадто складний спосіб, яким в більшості користуються програмісти, щоб показати рівень володіння мовою програмування. Слід розглянути два основних підходи, щоб вибрати найбільш оптимальний.

Серед графічних редакторів векторної графіки слід виділити два: CorelDraw та Adobe Illustrator. Кожен з них має свої переваги та недоліки.

CorelDRAW – це графічний редактор векторної графіки, розроблений канадською корпорацією Corel. Великим плюсом CorelDRAW є можливість відкривати файли проектів інших графічних редакторів: Photoshop та Illustrator. Редактор має відносно простий, інтуїтивно зрозумілий інтерфейс та не потребує використання великих обчислювальних потужностей. До переваг можна також віднести великий набір можливостей роботи з графікою, хоча з рядом обмежень.

CorelDRAW – простий, зрозумілий редактор векторної графіки для початківців, особливо коли потрібно швидко створити декілька графічних файлів з мінімальними витратами часу (рис.2.7.).

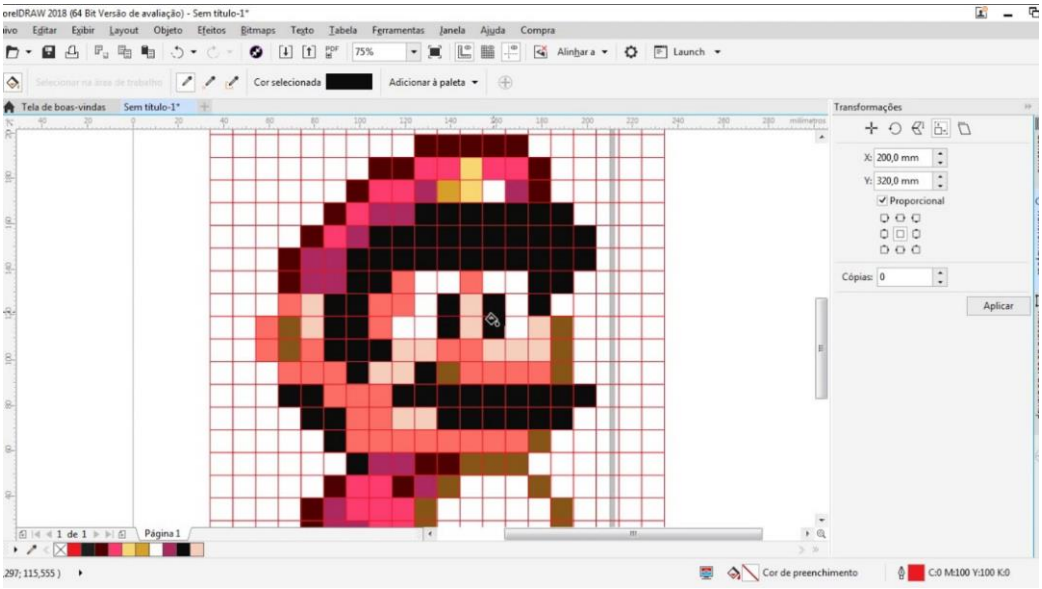

Рис.2.7. Піксель-арт в графічному редакторі CorelDRAW

Альтернативний варіант – це Adobe Illustrator, який має хоч і більш складний інтерфейс і потребує серйозних обчислювальних потужностей, але при цьому пропонує набір можливостей роботи з векторною графікою, який при правильному підході майже не має функціональних обмежень. Також Adobe Illustrator дає можливість створювати шаблони та використовувати сторонні пресети на будь-який смак (рис.2.8.).

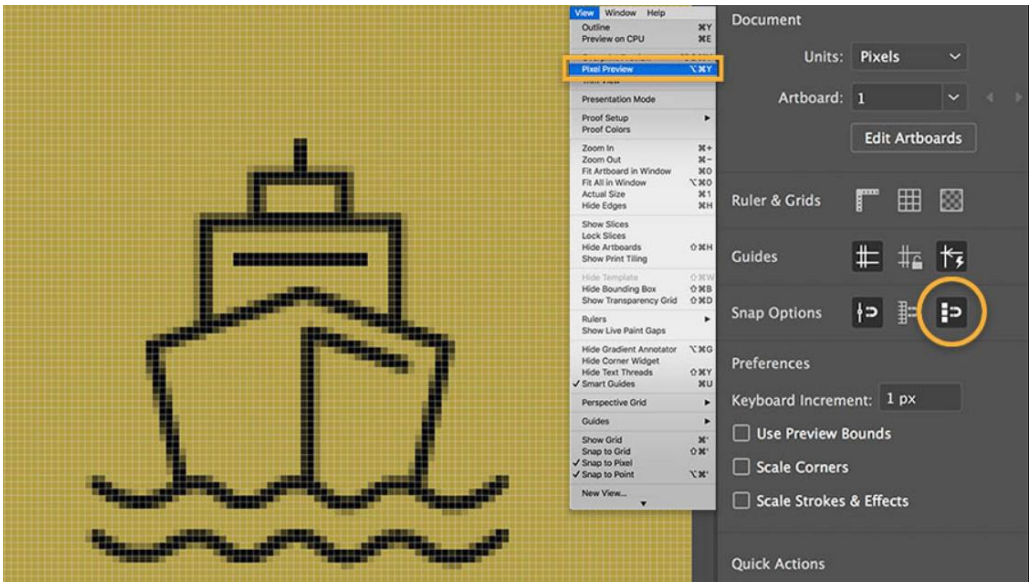

Рис.2.8. Піксель-арт в графічному редакторі Adobe Illustrator

Абсолютним лідером серед графічних редакторів растрової графіки є Adobe Photoshop (рис.2.9.), який пропонує інструменти для малювання, роботи з фотографіями, освітленням та всім можливим растровим графічним контентом. Photoshop потребує сучасної відео-карти та великого об’єму

оперативної пам’яті, взамін пропонуючи високу швидкість роботи й можливість працювати над дуже складними й комплексними проектами.

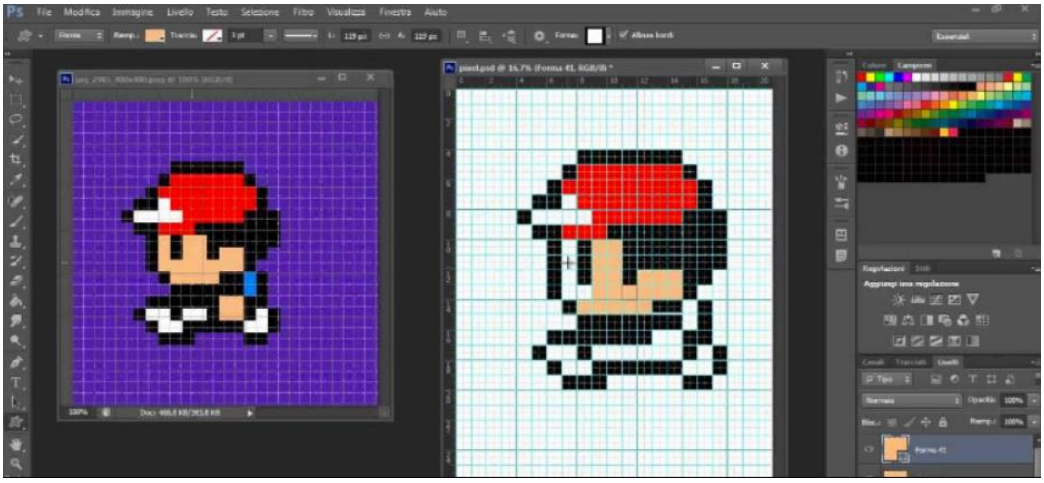

Рис.2.9. Піксель-арт в графічному редакторі Adobe Photoshop

Отже, різниця між векторним та растровим підходами до створення піксель-арту заключається, в першу чергу, в технологіях розробки.

Векторний підхід пропонує менші за об’ємом пам’яті на жорсткому диску файли, адже графіка створюється за допомогою математичних формул, але процес створення цих файлів більш трудомісткий, так як необхідно складати зображення по типу мозаїки.

В той же час, растровий підхід пропонує роботу безпосередньо з пікселями, що значно економить час. Тобто для роботи використовується полотно відповідного розміру (32*32 пікселя, 64*64 пікселя і так далі) і пензлик товщиною 1 піксель. Цей підхід особливо зручний для людей, які вміють малювати реальним пензликом, олівцем тощо.

Отже, в даному випадку найкращим варіантом для створення графічного контенту в стилі піксель-арт є використання растрового підходу та графічного редактору Adobe Photoshop.

2.6. Вимоги до проектування відеогри

Приступаючи до розробки мобільної гри, потрібно точно визначити декілька ключових моментів.

По-перше, потрібно обрати операційну систему. В даному випадку гра буде розроблятися для мобільної операційної системи Android, яка більш

поширена на місцевому ринку (рис.2.10.), має більш дружелюбний для нових розробників магазин та не вимагає додаткового апаратного та програмного забезпечення, крім ноутбука з середовищем розробки та смартфона на Android.

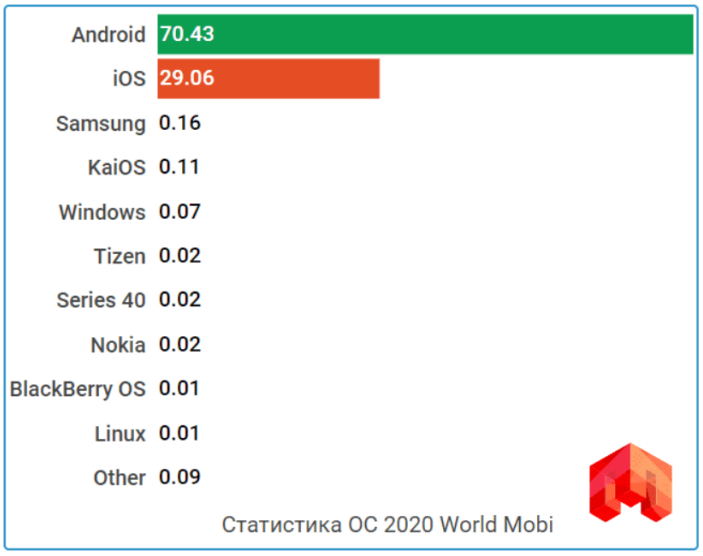

Рис.2.10. Статистика використання мобільних ОС на території України

По-друге, необхідно чітко визначити жанрові рамки гри. В даному випадку гра буде розроблятися за принципами гіперказуальних ігор, тобто буде мати одну максимально просту, але обов'язково «свіжу» механіку. Ігрова механіка – це самодостатня система, яка реагує на дії гравця чи дії ігрового світу (вхідні дані) та виконує певні операції (вихідні дані). Наприклад, якщо максимально спростити гру Тетріс (рис.2.11.), то залишиться лише одна механіка: гравець рухає блоки по осі X, щоб вони складалися в суцільні горизонтальні лінії (вхідні дані). Якщо горизонтальні лінії складаються, то вони руйнуються й гравець продовжує далі, а якщо верхній кубик досягає верхнього краю екрану – гра закінчується (вихідні дані). В оригінальній версії Тетрісу можна також вибирати варіацію форми блока, поки він спускається. Така особливість не є окремою механікою, а лиш доповнює основну.

Рис.2.11. Ігрова механіка Тетрісу

По-третє, необхідно обрати модель монетизації. Для ринку гіперказуальних ігор властива модель free-to-play, тобто модель безкоштовного поширення, коли гра заробляє за рахунок показів реклами гравцям. Також чудово себе зарекомендувала інтеграція внутрішніх необов'язкових покупок. Сумарно ці два підходи складають переважну більшість доходів індустрії мобільних ігор (рис.2.12.).

Важливо знати міру в тому, скільки реклами показувати кожному гравцю. Ідеально, якщо перегляд реклами не є обов'язковим, а дає великі ігрові бонуси чи дозволяє продовжити рівень після програшу. В такому випадку працює правило «взаємного обміну» і гравець точно розуміє, для чого він дивиться рекламу. Інтеграція рекламних блоків в програмний код та випуск гри на ринок буде відбуватися після захисту дипломного проекту, головна ціль якого – безпосередня розробка мобільної гри.

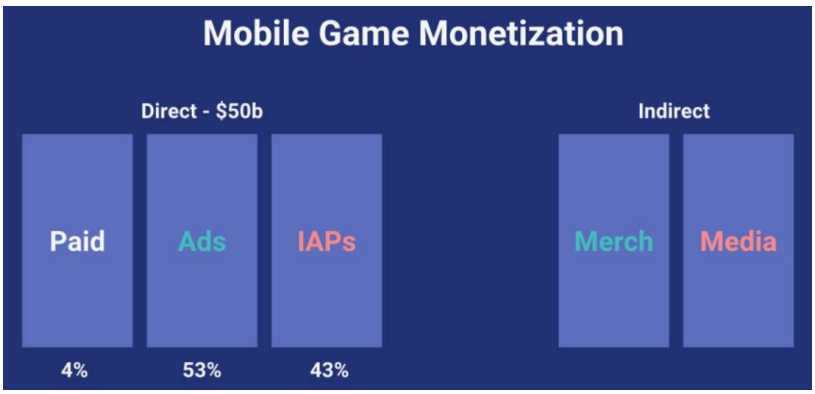

Рис.2.12. Доходи мобільних відеоігор за методом монетизації

2.7. Проектування відеогри

Обрана ідея для мобільної відеогри – це тактильна маніпуляція маленькими тривимірними геометричними фігурами кубічної форми з метою формування завчасно заданого двовимірного зображення. Тобто, це одна комплексна ігрова механіка, яка складається з двох підсистем: розбивання стартової фігури на менші об’єкти та формування з цих об’єктів певного зображення.

Окремою механікою, яка працює на доповнення ігрового досвіду, можна виділити оцінювання якості проходження рівня гравцем, яке, при найкращому проходженні, виражається у вигляді трьох зірочок. Таким чином, гравець отримує, хоч і символічну, нагороду за свої старання. Таке оцінювання та нагороду можна об’єднати в один термін: «зворотній зв'язок».

Найкраще ігровий процес можна пояснити за допомогою Use Case діаграми (рис.2.13.). Діаграма включає в себе дві дійові особи: «Гравець» та «Гра» та наступні варіанти використання:

- почати гру;
- взаємодіяти з інструментом;
- розбити стартову тривимірну фігуру;
- створити контролер та задати правила;
- взаємодіяти з кубиками;
- сформувати необхідну двовимірну фігуру;
- перейти до наступного рівня;
- отримати оцінку;
- отримати зворотній зв’язок.

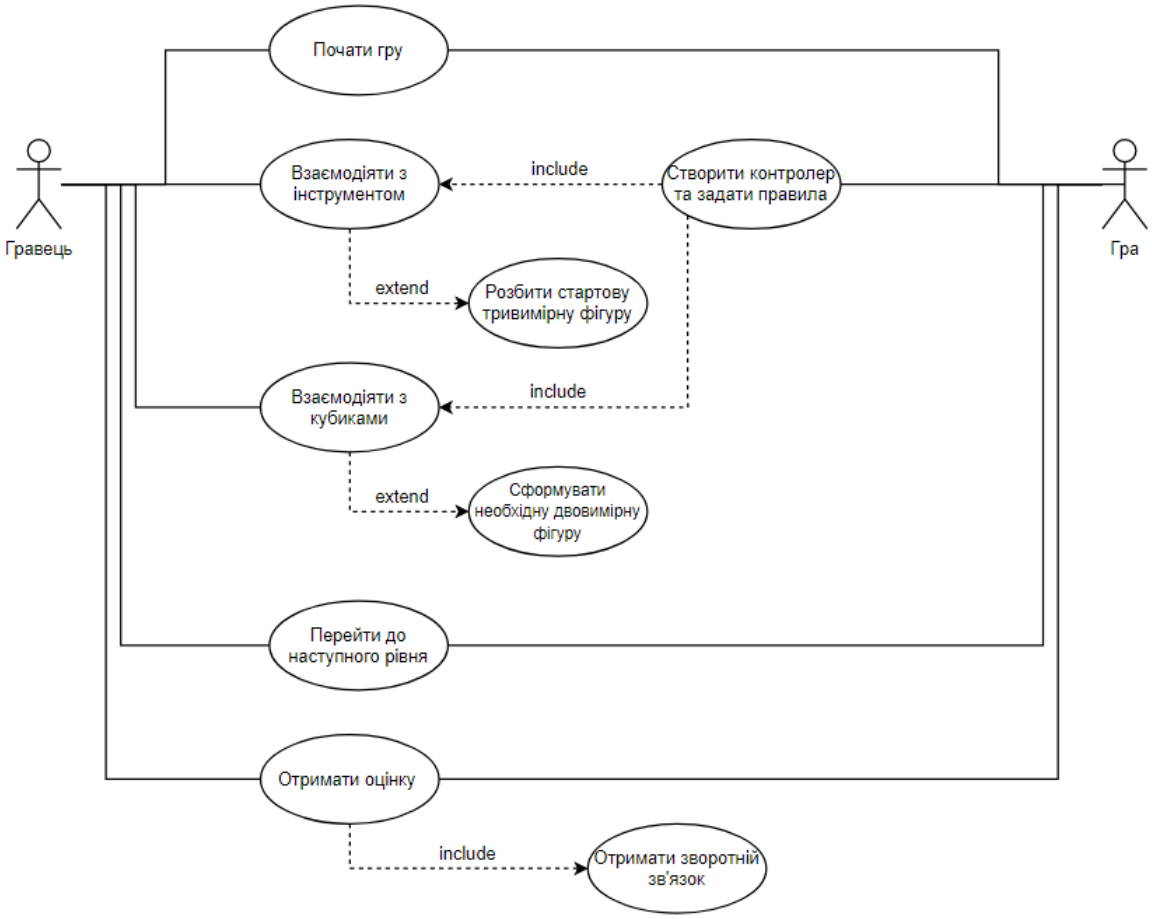

Рис.2.13. Діаграма варіантів використання

Так як логіка роботи гри доволі проста, вона не потребує додаткових схем чи діаграм. Надалі для проектування буде достатньо інформації в текстовій формі для опису особливостей гри, та візуальної інформації, щоб наглядно показати конкретні ідеї, які вже були реалізовані в аналогічних продуктах.

РОЗДІЛ 3. РОЗРОБКА ВІДЕОГРИ

3.1. Розробка графічного контенту

Для створення графічного контенту використовується графічний редактор Adobe Photoshop, який найкраще підходить для реалізації даної задачі. Photoshop – це зручний, комплексний та інтуїтивно зрозумілий редактор растрової графіки, що дозволяє розділяти векторні малюнки на досить великі квадрати різного кольору для подальших маніпуляцій всередині гри. Такий стиль графічного контенту називається піксель-арт (англ. pixel art). Тобто весь малюнок складається виключно з квадратів однакового розміру, але різного кольору, що імітує графічний стиль комп'ютерних ігор 80-х та 90-х років минулого століття (рис.3.1.).

Рис. 3.1. Малюнок пончика в стилі піксель-арт

Для даної відеогри розроблено 15 малюнків такого формату (рис.3.2.). Кожен малюнок – це репродукція одного конкретного образу сучасної популярної культури в стилі піксель-арту. Кожен малюнок збережено у форматі .psd (формат проектів Adobe Photoshop) та інтегровано в проектів Unity 3D. Слід відмітити, що платформа Unity 3D чудово працює в симбіозі з Photoshop, що економить багато часу розробникам.

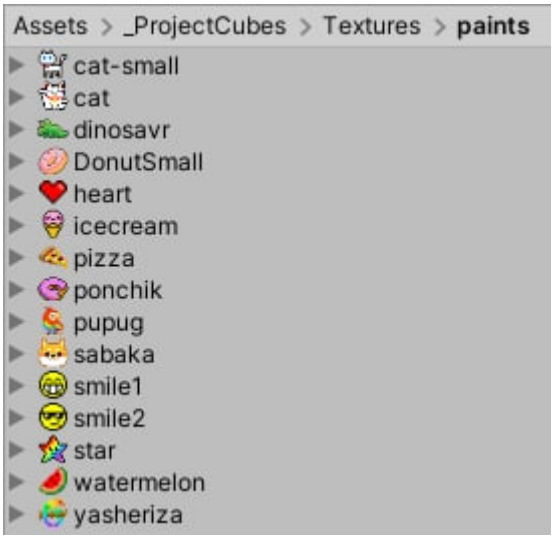

Рис.3.2. Список розробленого графічного контенту

З точки зору візуального мистецтва, найбільш цікавими картинками, які створені в процесі переформатування оригінальних образів, стали малюнки ящірки, кавуна та сердечка (рис.3.3.).

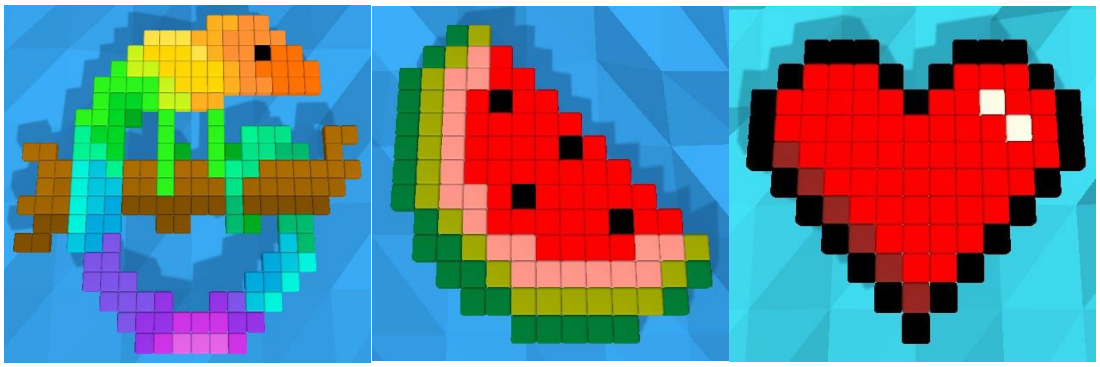

Рис.3.3. Малюнки ящірки, кавуна та сердечка

3.2. Розробка основного ігрового циклу

Основний ігровий цикл або кор-геймплей (англ. core gameplay) – це набір правил гри, послідовність дій гравця та реакцій ігрових систем на ці дії, які необхідні для створення мінімального ігрового процесу (рис.3.4.).

Основний ігровий цикл гри «Paint Cube» складається з декількох етапів:

1) гра формує тривимірну фігуру, яка складається з різнокольорових кубиків;

2) гра формує відповідну двовимірну фігуру;

3) гравець, за допомогою інструменту, розбиває стартову тривимірну фігуру;

4) гравець збирає кубики в групи і перетягує в зону двовимірної фігури;

5) коли кубик знаходиться над коміркою підходящого кольору – кубик падає вниз і займає цю комірку;

6) основний цикл закінчується, коли всі кубики розміщенні в комірках відповідно до їхнього кольору.

Рис.3.4. Основний ігровий цикл

Після загального опису основного ігрового циклу слід детально розібрати всі його складові, додаткові механіки, функції та побічні можливості.

3.3. Розробка основних ігрових підсистем

Першим ділом, гра формує стартову тривимірну фігуру (рис.3.5.). Алгоритм формування наступний: спочатку створюється куб з довжиною грані 1 кубик, потім 2 і так далі. Побудова куба починається з лівого нижнього боку, тому майже всі фігури мають незавершену праву грань (за

виключенням випадків, коли кількість кубиків відповідає кубу цілого числа, яке більше нуля).

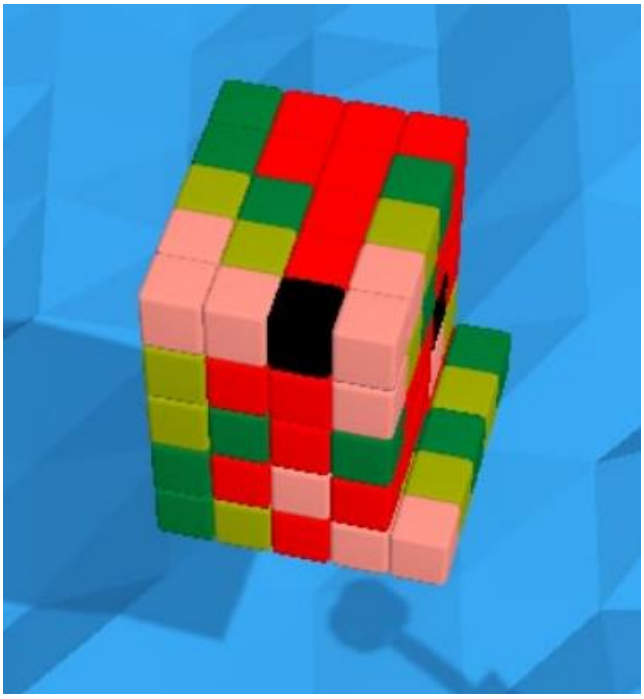

Рис.3.5. Стартова фігура

Так як необхідно мати конкретну кількість кубиків певних кольорів, для цього розроблена підсистема задання кольору стартовим кубикам, основною функцією якої є SetColor() (рис.3.6.). Функція визначає які кольори присутні в двовимірній фігурі і потім рендерить кубики відповідного кольору. В даному випадку використовуються не просто кольори, а матеріали, які мають додаткові характеристики. Наприклад, крім кольору кожен матеріал має коефіцієнт поглинання світла, може мати додаткові декоративні властивості типу ефектів матового чи глянцевого покриття.

```
public void SetColor(Color color)
{
    _targetColor = color;
    var renderer = GetComponent<MeshRenderer>();
    var materialPropertyBlock = new MaterialPropertyBlock();
    renderer.GetPropertyBlock(materialPropertyBlock);
    materialPropertyBlock.SetColor("_Color", _targetColor);
    renderer.SetPropertyBlock(materialPropertyBlock);
}
```

Рис.3.6. Функція SetColor()

Гра пропонує два інструмента для розбивання тривимірної фігури на початку рівня: дриль (візуально більше схожий на шуруповерт) та молоток (рис.3.7.). Інструмент залежить від рівня та має фізичні властивості, які схожі

на реальний інструмент. Тобто дриль має маленьку зону ураження і постійний дію, в той час як молоток має велику зону ураження і може використовуватися не частіше певного періоду, який відповідає часу замаху.

Рис.3.7. Інструменти для розбивання стартової фігури

Кожен інструмент управляється за допомогою модуля FingerForce, в основі якого лежить відповідний скрипт (рис.3.8.). Основні параметри скрипта – це радіус взаємодії, сила ураження та радіус ураження. Для дриля та молотка значення цих параметрів різні.

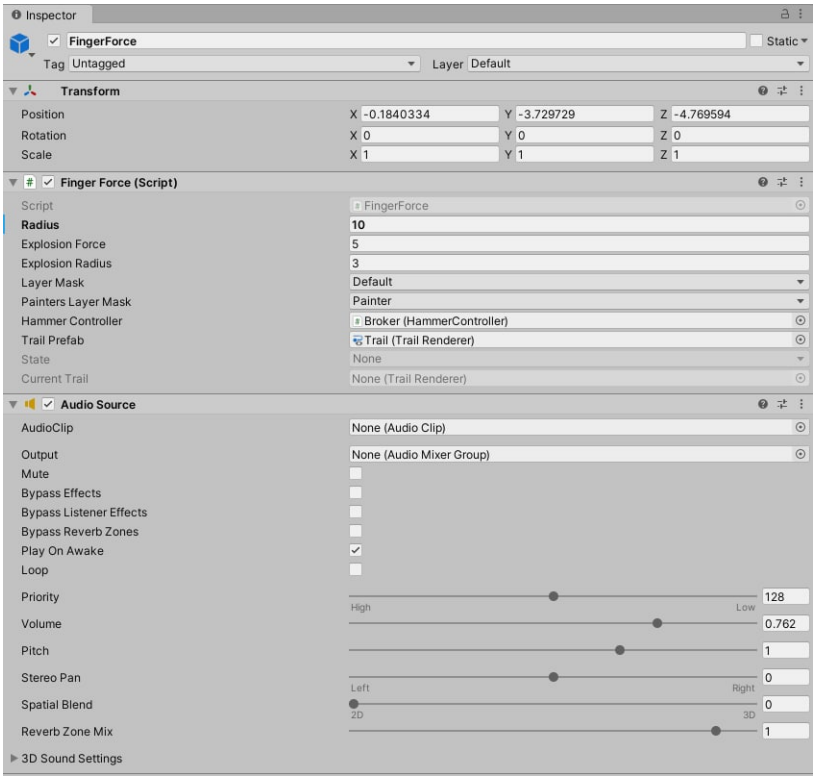

Рис.3.8. Модуль FingerForce

Для покращення ігрового досвіду гравця, зручного моніторингу прогресу рівня та заохочення гравця створено функцію графічного

відображення прогресу рівня, яка реалізована у вигляді прогрес-бару в верхній частині екрану (рис.3.9.). Даний графічний елемент відображає не тільки прогрес, а також показує номер рівня.

```
private void Start()
{
  _txtLevelNumber.text = $"Level {LevelManager.CurrentLevelNumber+1}";
  _imageLoader.Progress.Subscribe(value =>
  {
    float viewValue = value; //Mathf.Pow(value,2);
    viewValue = Mathf.Lerp(0.01f, 1, viewValue);
    _filler.fillAmount = viewValue;
    _filler.color = Color.white;
    _filler.DOKill();
    _filler.DOColor(_progressUpdatedColor, 0.1f).SetLoops(2, LoopType.Yoyo);
  });
```

Рис.3.9. Програмування відображення прогресу ігрового рівня

Ще однією додатковою функцією є можливість почати рівень спочатку. Для цього гравцю слід натиснути на відповідну кнопку з малюнком стрілочки у верхньому лівому куті екрану (рис.3.10.).

Рис.3.10. Інтерфейс ігрового рівня

Після того, як всі кубики будуть розміщені у відповідних комірках, рівень успішно закінчується і гравець отримує зворотній зв'язок щодо результатів проходження рівня. По-перше, на екран зворотного зв’язку виводиться створена фігура. По-друге, успішність проходження рівня виражається в кількості зірочок – чим їх більше, тим краще. Додатковим

ідентифікатором результативності є текстова інформація типу “Amazing!” (рис.3.11.).

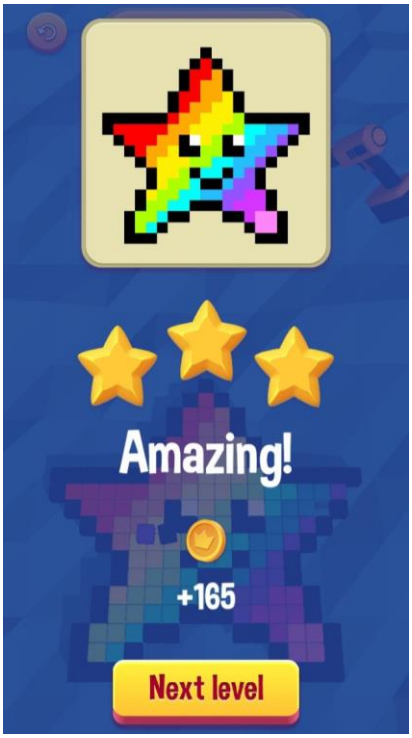

Рис.3.11. Меню зворотного зв’язку

Зірочки – це динамічні графічні об’єкти, які мають запрограмовану анімацію появи на екран (рис.3.12.). По-перше, вони в процесі появи вони змінюють свій розмір з маленького до потрібного, дещо рухаються вгору та обертаються навколо своєї осі. Всі ці дрібні деталі створюють приємну для гравця анімацію, чим ще більше радують його після завершення рівня.

```
private IEnumerator Start()
{
   var scale = _star.localScale;
   _star.localScale = Vector3.zero;
   yield return new WaitForSeconds(_delay);
   var image = _star.GetComponent<Image>();
   image.color = new Color(0,0,0,0);
   _star.localEulerAngles = new Vector3(0,0,25);
   image.DOColor(Color.white, 0.1f).OnComplete(() =>
   {
      _star.DORotate(Vector3.zero, 0.1f);
   });
   _star.DOScale(scale, 0.25f).SetEase(_curve);
}
```

Рис.3.12. Програмування анімації появи зірочок

Окремою і, на перший погляд, несуттєвою підсистемою є аудіо-менеджер, тобто підсистема контролю звуку (рис.3.13.). Аудіо-менеджер не є критично необхідною частиною гри, але він відіграє важливу роль у

формуванні повноцінного досвіду гравця. Хороший звук, так же само як і анімація, «оживляє» дії на екрані, дає можливість гравцю краще відчути ігровий процес.

Дана відеогра має два основних звука: тихий приглушений звук розбивання стартової фігури та звук зіткнення будь-якого кубика з площиною, яка в грі зображена у вигляді водної поверхні. Обидва звука попередньо штучно переведені в формат моно, тобто лівий і правий канал звуку в цих файлах звучить однаково. Це потрібно для правильного позиціонування звуку в просторі, адже саме інформацію про різницю між звуковими сигналами в правому і лівому вусі дає мозку зрозуміти, з якого боку та на якій відстані знаходиться джерело звуку. Також додатково додана невелика реверберація (ефект відбивання звуку від стін приміщення) для більш м’якого та приємного звучання.

```
public class AudioManager : MonoBehaviour
{
    public static AudioManager Instance { get; private set; }

    private AudioSource _audioSource;

    private void Awake()
    {
        if (Instance != null)
        {
            Destroy(gameObject);
            return;
        }

        Instance = this;
        DontDestroyOnLoad(gameObject);
        _audioSource = GetComponent<AudioSource>();
    }

    public void PlayOneShot(AudioClip clip)
    {
        _audioSource.PlayOneShot(clip);
    }

    public void PlayOneShot(AudioClip clip, float volume)
    {
        _audioSource.PlayOneShot(clip, volume);
    }
```

Рис.3.13. Програмування аудіо-менеджера

Також для створення приємного, розслабленого гри настрою підібрано підходящу музичну тему в стилі Ambient, яка гармонічно звучить разом з основними звуковими ефектами і чудово вписується в атмосферу.

Ambient – це жанр музики, який характеризується використанням фактурних, атмосферних, «широких» звуків. Дуже часто, крім традиційних для електронної музики синтезаторів, використовуються також натуральні записи, типу звуків дощу, вітру, індустріального звукового фону тощо. В більшості випадків Ambient не має чіткого ритму, інколи навіть темпу, що додає композиціям цього жанру додаткової ефемерності, атмосферності та легкості.

Існує ще один прийом, який частково пов’язаний зі звуковим дизайном та слугує всі тій же цілі – покращити ігровий досвід гравця. Цей прийом називається вібрація. На даний час вібрація – це майже єдиний спосіб доставляти до гравця тактильний зворотній зв’язок. Наприклад, сучасні консольні контролери DualSense від компанії Sony (рис.3.14.) мають дуже складну систему вібрацій, що дає змогу передати гравцю ігровий процес буквально «на кінчиках пальців».

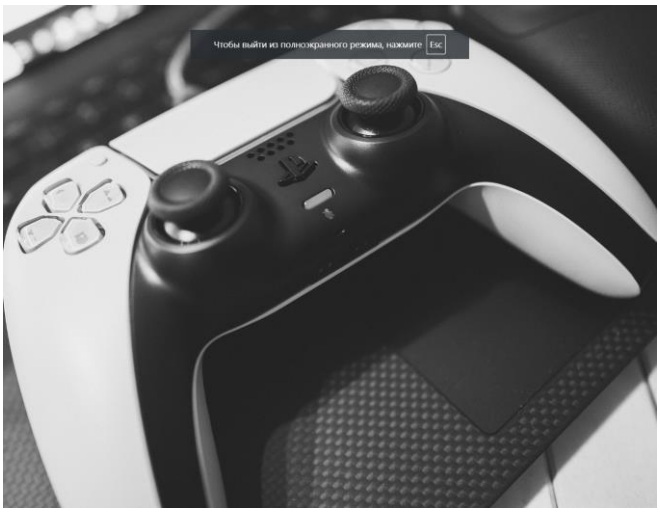

Рис.3.14. Консольний контролер DualSense від компанії Sony

Віддача після вистрілу вогнепальної зброї, стрибки, приземлення, падіння – ці всі і багато інших ігрових подій доповнюються вібрацією.

Вібрація також реалізовані в даній грі (рис.3.15.), спрацьовує вона в момент взаємодії з кубиками. Слід відмітити, що сила вібрації різних моделей телефонів відрізняється, тому обрано середній рівень сили, щоб не створювати дискомфорту гравцю.

```
.OnComplete(() =>
{
   cube.Paint();
   AudioManager.Instance.PlayOneShot(_audioClip);
   Vibration.VibratePop();
   Destroy(gameObject);
});
```

Рис.3.15. Програмування вібрації

За допомогою знімків екрану важко показати рух, але реалізована анімація синьої поверхні ігрового рівня, яка симулює рух водної поверхні, відповідає всім сучасним тенденціям анімації мобільних ігор. Для створення такого ефекту площина була поділена за допомогою спеціального алгоритму на трикутники, які формуються в тривимірні фігуру. Також кожен трикутник має свій відтінок, що створює ефект об’ємної фігури.

Кожен ігровий рівень має спеціальне освітлення (рис.3.16.). Освітлення складається з двох компонентів: неправленого та направленого джерела світла. Перший компонент дає можливість налаштувати інтенсивність та колір світла, в той час як другий – задає точку розміщення джерела світла, таким чином створюючи тіні потрібної довжини.

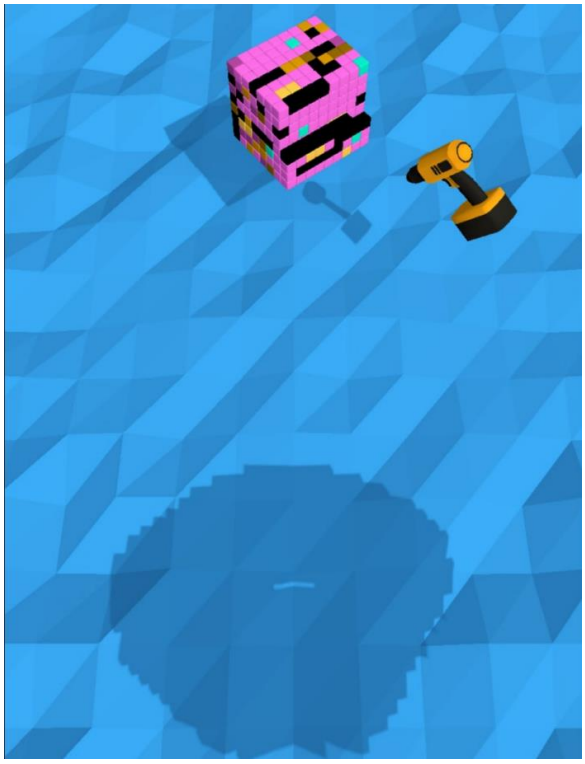

Рис. 3.16. Демонстрація освітлення ігрового рівня

3.6. Тестування

Процес тестування включає в себе, в першу чергу, багатократне проходження ігрового рівнів, в даному випадку це 10 контрольних ітерацій для кожного рівня. Обов'язковою умовою якісного тестування є генерація нестандартних ситуацій під час ігрового процесу.

В процесі тестування виявлена одна невідповідність проектній документації: занадто сильних рух пальцем по екрану може призвести до вильоту кубиків за межі ігрового поля, що призводить до неможливості завершити рівень (рис.3.17.). Дана невідповідність не є критичною, так як така ситуація з'являється дуже рідко та є можливість перезапустити ігровий рівень і почати все спочатку.

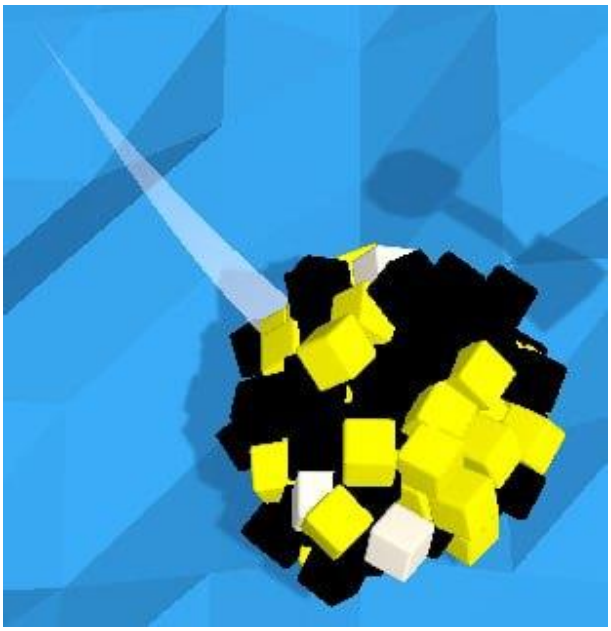

Рис.3.17. Демонстрація знайденої невідповідності

Дослідним шляхом визначено середню тривалість успішного проходження ігрового рівня – 20 секунд. Зважаючи на високу реіграбельність рівнів та специфіку геймплею, можна сказати, що така тривалість більш ніж достатня, особливо враховуючи вже існуючу кількість рівнів – 14, та можливість відносно просто створювати нові рівні.

В результаті тестування можна зробити висновок, що всі ігрові механіки працюють правильно, керування дружелюбне й зручне, критичних багів не виявлено.

ВИСНОВКИ

Результатом виконання дипломного проекту є тривимірна мобільна відеогра “Paint Cube” на платформі Unity 3D.

У першому розділі досліджено предметну область та процес створення відеоігор, а саме етапи розробки ігрового дизайну, наративного дизайну, , графічного дизайну, програмного коду та етап тестування.

У другому розділі описано безпосередній процес проектування відеогри “Paint Cube”, процес формування вимог до розробки, вибір програмного забезпечення та технологій.

У третьому розділі відображені етапи розробки даної відеогри, починаючи зі створення графічного контенту й програмування підсистем та закінчуючи тестуванням фінальної версії продукту.

Завдяки правильному підходу до проектування та розподілу ресурсів, спроектовано та розроблено відеогру, що повністю відповідає сучасним тенденціям індустрії, описаним характеристикам та має відповідний ігровий функціонал.

Відеогра “Paint Cube” – конкуренто спроможний продукт індустрії мобільних відеоігор, що поєднує в собі цікаву ігрову механіку, зручний інтерфейс, приємний графічний стиль та звукове оформлення.

Проект має перспективу розширюватися завдяки створенню нових рівнів, цікавого графічного контенту та ігрових механік.