МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Савченко А.С.

“ ”_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ “МАГІСТРА”

ЗА СПЕЦІАЛІЗАЦІЄЮ “ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)”

Тема: “Технологія розробки ігрового додатку в середовищі Unity 3D з використанням піксельної графіки”

Виконавець: Приймак Роман Сергійович

Керівник: к.т.н., доцент Малежик Олександр Іванович

Нормоконтролер: _____ Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за галузями)".

ЗАТВЕРДЖУЮ
Завідувач кафедри
Савченко А.С.
" " _____ 2020 р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Приймака Романа Сергійовча .

1. Тема роботи: "Технологія розробки ігрового додатку в середовищі Unity3D з використанням піксельної графіки".

Затверджена наказом ректора від " 02 " жовтня 2020р за № 1891/ст .

2. Термін виконання роботи: з 05 жовтня 2020р. до 27 грудня 2020р.

3. Вихідні данні до роботи: кросплатформовий ігровий додаток, завдання створення працездатного програмного продукту по технології Unity3D з використанням мови програмування C#.

4. Зміст пояснювальної записки: 1)Проаналізовано ринок операційних систем. 2) Створено алгоритм розробки ігрового додатку. 3)Проаналізовано існуючі алгоритми та методи розробки додатків, також описано та аргументовано вибір програмного забезпечення. 4) Спроектовано покроковий метод розробки ігрового додатку, з реалізацією різноманітних ігрових механік.

5. Перелік обов'язкового ілюстративного матеріалу: рисунки, діаграма, а також слайди презентації доповіді у PowerPoint.

6. Календарний план-графік

№ з/п	***Завдання***	***Термін виконання***	***Підпис керівника***
1.	Формування теми дипломної роботи, постановка задачі та узгодження з дипломним керівником.	05.10.20	
2	Формування структури розділів дипломної роботи	06.10.20– 10.10.20	
3	Формування та оформлення першої частини дипломної роботи	11.10.20 – 15.11.20	
4	Збір науково-технічного матеріалу до другої частини дипломної роботи	16.10.20 – 20.10.20	
5	Формування та оформлення другої частини дипломної роботи	08.11.20 – 16.11.20	
6	Розробка ігрового додатку згідно з завдання дипломної роботи	16.11.20 – 31.11.20	
7	Формування та оформлення третьої частини дипломної роботи	19.11.20 – 31.11.20	
8	Формування звіту та графічних матеріалів	23.11.20 – 08.12.20	
9	Підписання необхідних документів	09.12.20 – 15.12.20	
10.	Підготовка до захисту дипломної роботи	16.12.20 – 21.12.20	

7. Консультація з окремого(мих) розділу(ів) роботи:

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

8. Дата видачі завдання 05 жовтня 2020р.

Керівник дипломної роботи . _____ Малежик О.І.

Завдання прийняв до виконання _____ Приймак Р.С.

(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи "Технологія розробки ігрового додатку в середовищі Unity3D з використанням піксельної графіки" містить 76 сторінок, 39 рисунків, 7 бібліографічних посилань.

Об'єкт дослідження: є технології та методи розробки кросплатформового ігрового додатку з використання Unity3D.

Мета роботи: Розробити та вдосконалити ігровий додаток, з можливістю його використання на різних операційних платформах

Методи дослідження: Концепція технології створення ігрового мобільного додатку, розбиття його на рівні, та опис розроблених ігрових механік для можливості більш активної взаємодії персонажа з навколишнім середовищем, які були описані в ході розробки.

Результати магістерської роботи: В ході написання дипломної роботи, було розроблено ігровий додаток з можливістю підтримки на різних операційних платформах. Також було розглянуто технології розробки та налаштування характеристик в середині ігрового движку Unity3D.

Ключові слова: UNITY3D, ANDROID, GODOT, UNREALENGINE, BLENDER, ІНТЕРАКТИВНИЙ ГРАФІЧНИЙ ОБ’ЄКТ, C#.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

- GPU (General Public License) – ліцензія програмного забезпечення з відкритим програмним кодом.
- API (Application programing interface) – сукупність засобів та правил, що взаємодіють між окремими складовими програмного забезпечення.
- Framework – інфраструктура програмних рішень.
- Script – частина програмного коду, яка відповідає за базовий набір команд.
- Compiler (Компілятор) – інструмент для складання та генерації програмного коду.
- Rendering (Рендерінг) – процес отримання зображення за створеною моделлю.
- SEO (Search engine optimization) – комплекс робіт, задача яких вивід ресурсі на перші позиції в пошукових системах.
- SMM (Social Media Marketing) – один із видів контекстної реклами, цілю якої є способи залучення аудиторії для використання вашого програмного продукту.
- Bugs (помилка) – помилка, або дефект в програмному коді.

ВСТУП

Мобільні технології стали частиною нашого життя, з кожним роком збільшується обсяг програмних продуктів, що використовуються. Неможливо уявити, що в найближчому майбутньому може змінитися ця ситуація, і можна сміливо стверджувати, що спеціалісти в цій сфері будуть необхідні, як мінімум наступне десятиліття.

Для того щоб програмний продукт міг приносити доходи, важливо, щоб він міг підтримувати велику кількість телефонів та інших гаджетів. При цьому важливо врахувати той нюанс, що технічні можливості різних апаратів значно відрізняються один від одного.

В більшості випадків ігровий додаток розроблюють для одного типу пристроїв а потім модифікують під інші. Для перенесення між платформами змінюється розширення екрану, положення та коди клавіш, підключаються нові бібліотеки, які підтримує той пристрій. Якщо програмний продукт переноситься на більш слабкий або сильний пристрій змінюється його функціонал в ту чи іншу сторону, додають або прибираються певні графічні ефекти, які можуть впливати на параметри продуктивності пристроїв.

У багатьох навчальних програмах в основі лежить ігрова складова. У таких програмах, ігри представлені в структурованій текстовій інформації, або у вигляді повноцінних додатків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Вдосконалення апаратного забезпечення, таких як звукові карти, графіка та швидші процесори, означають пов'язане зростання та розвиток ігрової індустрії. Як результат, сучасні ігри, особливо ті, що базуються на ПК, стали дуже вимогливими, оскільки додатки та серйозні геймери серед тих, хто купує потужні персональні комп’ютери, щоб не відставати від найновіших ігор.
Види ігор

Протягом багатьох років відеоігри розвивалися, включаючи низку різних засобів. Еволюція призвела до того, що консолі ставали все більш досконалими протягом десятиліть, ігри для ПК стають багатокористувацькими та складнішими, і ціла екосистема мобільних ігор. Давайте подивимось, що це все означає.

- Ігри для ПК

  Ігри для ПК або персональні комп’ютерні ігри відтворюються на комп’ютері за допомогою клавіатур, миші, джойстика або геймпада.

Ці ігри характеризуються відсутністю центрального контролюючого органу та більшими можливостями для введення, обробки та виведення. Комп'ютерні ігри ненадовго втратили ринок консолей в середині 90-х років, але знову з'явилися в середині 2000-х завдяки цифровому розповсюдженню. У ігри можна грати локально або в Інтернеті.

- Консольні ігри

Пристрій, який використовується для гри, називається відеоігровою консоллю. Консоль виводить відеозображення, яке відображає гру. До консолей належать ті, що використовуються для гри вдома, портативні консолі, мікроконсолі та спеціальні консолі. Консоль тепер стала одним пристроєм, який можна використовувати як приставку, веб-браузер та програвач компакт-дисків та DVD-дисків.

- Мобільні ігри

Відеоігра, в яку грають за допомогою мобільного пристрою, такого як смартфон, планшет або навіть калькулятор або розумний годинник, відома як мобільна гра. Найпершою мобільною грою стала "Тетріс" на пристрої Hagenuk MT-2000 в 1994 році, а за нею - "Snake" від Nokia у 1997 році. На даний момент мобільні ігри завантажуються на смартфон або планшет за допомогою магазину додатків або вбудовуються в пристрій при покупці.

1.1 Аналіз операційних систем

Операційна система - це програмне забезпечення, яке керує кожною частиною комп’ютерної системи - усім апаратним та іншим програмним забезпеченням. Якщо бути конкретним, він контролює кожен файл, кожен пристрій, кожен розділ основної пам'яті, кожну наносекунду часу обробки та кожне мережеве підключення. Він контролює, хто і як може використовувати систему.

Кожна операційна система, незалежно від її розміру та складності, може бути представлена пірамідою, що показує, як її п'ять основних функцій (так звані менеджери) працюють разом. Менеджер пам’яті, диспетчер процесорів, диспетчер пристроїв та менеджер файлів складають основу піраміди; мережеві операційні системи також додають менеджера мережі. Інтерфейс користувача - частина операційної системи, яка взаємодіє з користувачем - підтримується іншими чотирма або п’ятьма менеджерами.

Операційна система комп’ютера є однією з найважливіших “частин” комп’ютера. Майже кожному типу комп’ютерів, включаючи мобільні телефони, системи відеоігор, пристрої зчитування електронних книг та відеореєстратори, потрібна операційна система для належної роботи. Операційна система дозволяє користувачеві працювати на комп’ютері, не знаючи всіх подробиць про те, як працює апаратне забезпечення.

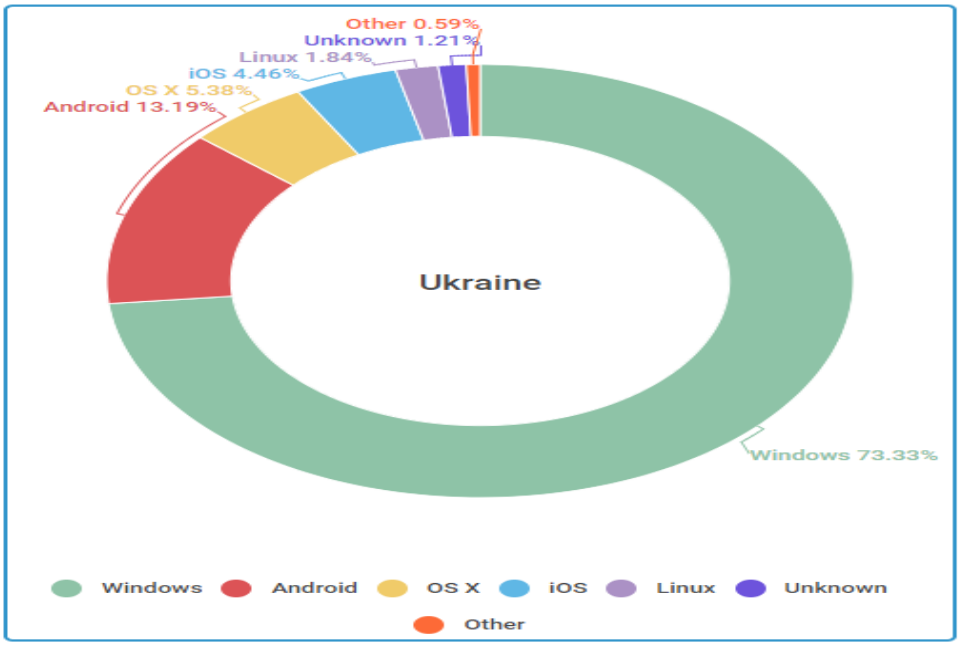


Рис.1.1. Статистика використання операційних систем в Україні

В Україні лідируючої по використанню операційною системою є звичайно ж Windows – 73.33% всіх пристроїв, від настільних до кишенькових. І так як мобільні технології беруть своє, Android – в Україні теж взяв 13,19%. Операційна система виробництва компанії Apple OS X займає третє місце – 5.38%, і навіть якщо до нього додати показник iOS – 4,46% – це не дасть йому обігнати за популярністю в Україні Андроїд.

1.1.1 Операційні системи для ПК

В наш час існує значна кількість різноманітних операційних систем для персональних комп’ютерів, кожна з яких має свої особливості, переваги та недоліки, детальніше про це ми розглянемо пізніше. Операційні системи (далі ОС) – були створенні для ефективного використання ресурсів і можуть включати в себе програмне забезпечення для обліку та для розподілу витрат часу процесору.

Домінуючою настільною операційною системою являється Microsoft Windows, яка охоплю 83,3% всього ринку операційних систем. MacOS від

компанії Apple займає друге місце з часткою ринку в 11,2%, на третьому місці знаходяться різновиди операційної системи Linux з часткою в 1,55%.

Розглянемо більш детально можливості кожної з найбільш вживаних операційних систем.

1.1.1.1 Microsoft Windows

Microsoft представила операційне середовище під назвою Windows 12 серпня 1981 року як графічну оболонку операційної системи для MS-DOS у відповідь на зростаючий інтерес до графічних інтерфейсів користувача (GUI). Microsoft Windows стала домінуючою на світовому ринку персональних комп'ютерів (ПК) з часткою ринку понад 90%, випередивши Mac OS, яка була представлена в 1984 році. На ПК Windows досі є найпопулярнішою операційною системою. Однак це порівняння може бути не повністю актуальним, оскільки дві операційні системи традиційно націлені на різні платформи. Тим не менше, цифри для серверного використання Windows (які можна порівняти з конкурентами) показують третину частки ринку, подібно до кінцевого споживача. На серпень 2020 року останньою версією Windows для ПК, планшетів, смартфонів та вбудованих пристроїв є Windows 10. Найновішою версією для серверних комп’ютерів є Windows Server 2019.

Розглянемо переваги та недоліки даної операційної системи.

До переваг можна віднести наступні пункти:

- Широкий вибір програмного забезпечення;
- Повна сумісність з усіма устаткуваннями;
- Технічна підтримка;
- Широка сфера використання;
- Легкість у налаштуванні;

До недоліків можна віднести наступні пункти:

- Поганий рівень захищеності операційної системи (вразлива для вірусів, та шкідливих програм);

- Завищені системні вимоги;
- Велика кількість обмежень від компанії розробника (наприклад контроль цифрового контенту);

Широке використання операційної системи Microsoft виграло від того, що вона не пов’язана з успіхом одного виробника обладнання та готовністю Microsoft ліцензувати операційну систему для виробників. Це на відміну від Apple Computer, яка не надає ліцензію Mac OS X іншим виробникам, та Sun Microsystems, які не ліцензували Solaris до того, як її зробили безкоштовною та відкритою.

Однак широкий спектр можливих апаратних перестановок з Microsoft Windows також є основним джерелом проблем із комп'ютером через несумісність апаратно-програмного забезпечення для споживачів.

Завдяки великим ліцензійним угодам Microsoft з багатьма постачальниками комп’ютерів, в даний час Windows постачається попередньо встановленою на більшості комп’ютерів у комплекті з OEM-версією, що робить її типовим або єдиним вибором для більшості ринків.

Підтримання сумісності в новому випуску Windows із цією великою колекцією програмного забезпечення, призначеного для роботи на старих версіях Windows, забирає значну частину ресурсів команди розробників Windows.

Найбільш популярними і вживаними версіями Windows є версії Windows7 та Windows10.

У жовтні 2009 року Windows 7 була випущена як наступник Vista, і вважається набагато стабільнішою та придатнішою для використання, ніж її попередник. Одночасно було запущено Windows Server 2008 R2 як оновлення для серверної лінії.

Windows 10 вийшов 29 липня 2015 року як наступник Windows 8.1. Windows 10 має підтримку універсальних програм. Користувацький інтерфейс був розроблений для обробки змін між інтерфейсом на основі миші та інтерфейсом із сенсорним екраном, а також були внесені інші нові зміни. За цей час було додано Windows Server 2016 та Windows Server 2019.

1.1.1.2 Mac OS

Mac OS – друга по популярності в світі операційна система (після Windows) від компанії Apple, яка представляє собою десяте покоління для комп'ютерів Macintosh. Операційна система Mas OS має досить не поганий зовнішній інтерфейс, який не підвищує її функціональність. В операційній системі Mac OS використовується ядро XNU (X is Not Unix) яке засноване на мікро ядрі Mach, і програмного коду розробленого компанією Apple.

До впровадження Mac OS X Apple експериментувала з кількома іншими концепціями, випускаючи різні продукти, призначені для перенесення інтерфейсу Macintosh або додатків до Unix-подібних систем або навпаки, A / UX, MAE та MkLinux. Зусилля Apple розширити та розробити заміну класичної Mac OS у 1990-х роках призвели до кількох скасованих проектів під кодовою назвою Star Trek, Taligent та Copland.

Хоча вони мають різну архітектуру, операційні системи Macintosh мають спільний набір принципів графічного інтерфейсу, включаючи рядок меню у верхній частині екрана; оболонка Finder, що містить метафору на робочому столі, яка представляє файли та програми за допомогою піктограм та пов'язує такі поняття, як каталоги та видалення файлів, із реальними об'єктами, такими як папки та кошик для сміття; та перекриття вікон для багатозадачності.

До переваг та недоліків даної операційної системи можна віднести наступне.

Переваги:

- Досить легке налаштування системи;
- Не вимагає від користувача фундаментальних технічних знань;
- Легка у використанні;
- Зручна організація інтерфейсу;
- В системі встановлено базовий набір програмного забезпечення;

Недоліки:

- Висока вартість комп’ютерів з Mac OS X;
- Закрита архітектура комп’ютерів;

1.1.1.3 Linux

Linux є Unix – подібною операційною систему, але був розроблений без використання коду Unix. Завдяки тому, що система Linux має відкритий код ядра, це дає можливість для детального вивчення і модифікації, що призвели до його використання в різних сферах обчислювальної техніки.

З моменту свого первісного розвитку Linux прийняв умови авторського права Free Software Foundation, який створив GNU GPL General Public License (GPL). Copyleft говорить, що все, що взято безкоштовно і модифіковано, має в свою чергу поширюватися безкоштовно. На практиці, якщо Linux або інші компоненти GNU розробляються або модифікуються для створення нової версії Linux, ця нова версія повинна поширюватися безкоштовно. Це основа розробки з відкритим вихідним кодом, яка заважає розробнику або іншим групам отримувати прибуток від вільно доступною роботи інших.

Операційна система Linux має модульну конструкцію, яка є ключем до її численних варіантів і дистрибутивам. Завантажувач відповідає за запуск ядра Linux. Ядро є ядром системи Linux, що забезпечує доступ до мережі, планування процесів або додатків, управління основними периферійними пристроями та спостереження за службами файлової системи.

Програмне забезпечення диспетчера пакетів зазвичай додає, оновлює або видаляє програмні компоненти в операційній системі Linux. Приклади менеджерів пакетів включають dpkg, OpenPKG, RPM Package Manager і Zero Install.

До переваг даної операційної системи можна віднести:

- Вільне поширення;
- Стабільність в роботі;
- Низькі системні вимоги;

- Відсутність вірусів\програм шкідників;
- Велика кількість дистрибутивів;
- Практично все програмне забезпечення є безкоштовним;
- Підтримує безліч апаратних платформ;

До недоліків можна віднести:

- Складність початкового налаштування системи;
- Можливі конфлікти з устаткуванням яке не підтримується;
- Відсутність аналогів деяких програм, які наявні в інших операційних системах;

Провівши аналіз операційних систем для персонального комп'ютеру, можна зробити наступний висновок, що більш зручною системою для звичайного користувача є Microsoft Windows через свою простоту та легкість у використанні, операційна система Linux є більш складною, і звичайний користувач може зіштовхнутися з проблемою правильного налаштування, та вибору правильного або необхідного дистрибутиву для своєї роботи. Операційна система Mac OS є також доволі легкою у використанні, але має один великий недолік, продукція компанії Apple має досить вискоку вартість і не всі користувачі можуть собі дозволити її використання.

1.1.2 Мобільні операційні системи

Останні декілька років смартфони набуваю все більшу і більшу популярність серед користувачів різноманітних гаджетів і все більше розпочали наздоганяти користувачів персональних комп'ютерів.

Мобільна операційна система - це ОС, побудована виключно для мобільних пристроїв, таких як смартфон, персональний цифровий помічник (КПК), планшет чи інша вбудована мобільна ОС.

Мобільна ОС відповідає за ідентифікацію та визначення функцій та функцій мобільних пристроїв, включаючи клавіатури, синхронізацію додатків, електронну пошту, колесо та текстові повідомлення. Мобільна ОС схожа на стандартну ОС

(наприклад, Windows, Linux та Mac), але порівняно проста та легка і в першу чергу управляє бездротовими варіаціями локальних та широкосмугових з'єднань, мобільним мультимедіа та різними методами введення.

Для адаптації до властивих середовищ мобільних пристроїв мобільна ОС працює на обмежених ресурсах, наголошуючи на комунікації, таких як оперативна пам'ять (RAM), швидкість зберігання та центральний процесор (CPU).

Нижче наведено приклад, що описує, як текстові повідомлення працюють у мобільній ОС:

- Мобільний додаток дозволяє користувачеві читати та писати повідомлення для доставки на мобільний пристрій через радіохвильові хвилі;
- Після того, як пристрій отримує сигнали повідомлень, пристрій повідомляє мобільну ОС, яка зберігає повідомлення та повідомляє програму обміну повідомленнями;
- Користувач читає повідомлення та відповідає відповідним повідомленням;
- ОС використовує апаратні антени для передачі повідомлення.

1.1.2.1 Android

Android - це операційна система на базі ядра Linux, з проміжним програмним забезпеченням, бібліотеками та API, написаними на мові C, і прикладним програмним забезпеченням, що працює на фреймворку програми, що включає сумісні з Java бібліотеки на базі Apache Harmony.

Android, як ОС, дозволяє запускати програми та програми на своєму пристрої. Це носій, за допомогою якого ви взаємодієте зі своїм пристроєм. Він повідомляє пристрою, що робити, коли ви виконуєте певну дію.

Наприклад, коли ви вибираєте пісню, яку ви хотіли б прослухати, і натискаєте на неї, Android повідомляє пристрою відтворити цю пісню та призупинити її, коли ви натискаєте кнопку «призупинити».

Android - це проект з відкритим кодом, який називається AOSP, тобто Android Open Source Project. В рамках цього проекту Google випускає вихідний код Android, який тоді може завантажити та скомпілювати кожен.

У четвертому кварталі 2010 року компанія Canalys зареєструвала Android як найбільш продавану платформу для смартфонів у світі з понад 200 мільйонами пристроїв Android, які використовувались до листопада 2011 року в декількох версіях операційної системи.

Розглянемо переваги та недоліки операційної системи.

До переваг можна віднести:

- Доступність системи для апаратних платформ MIPS, x86, ARM.
- Швидка робота.
- Велика кількість доповнень і додатків, що сприяють розширенню функціоналу.
- Здатність системи підлаштовуватися під користувачів.
- Оптимізація, відсутність жорстких вимог до самих пристроїв.
- Відкритий вихідний код, завдяки якому розробники можуть створювати будь-які рішення.

Але є і мінуси, які теж можуть виявитися важливими для користувача:

- Легкий пошук вразливостей, розробка вірусів.
- Сильний витрата енергії, якщо оптимізація постраждала.
- Виробники створюють оболонки, які не завжди можуть похвалитися високою продуктивністю.

1.1.2.2 IOS

iOS - це операційна система, розроблена компанією apple.inc. iOS працює на всіх мобільних пристроях Apple. IOS - друга за величиною операційна система у світі після Android.

IOS повністю відрізняється від інших операційних систем мобільних телефонів. Тому що він зберігає всі програми на своєму пристрої всередині

захисної оболонки. Щоб додатки трималися подалі один від одного і не втручалися в роботу один одного. IOS розроблений таким чином, що якщо на пристрій випадково потрапить вірус через програми, він може не нашкодити іншим програмам. Хоча в інших операційних системах такої функції не спостерігається. Оскільки одна програма не може спілкуватися безпосередньо з іншою програмою.

До переваг та недоліків даної операційної системи можна віднести.

Переваги:

- Простота оновлення системи;
- Інтегрована взаємодія між пристроями;
- Зручний та простий інтерфейс користувача;
- Надійність та безпека;

Недоліки:

- Закрита файлова система;
- Платні додатки та інші сервіси (наприклад музика\фільми);

1.2 Вибір програмного забезпечення

Перед початком роботи над певним програмним продуктом необхідно провести аналіз, який програмний інструментарій вам може знадобитись для виконання поставлених завдань. Так як практичною частиною моєї дипломної роботи являється гра. Розглянемо з яких основних частин і компонентів вона складається, для того щоб вибрати програмне забезпечення для подальшої розробки. Відеогра складається з 2 основних частин, це візуальна частина (графіка гри) та програмна (програмний код гри), більш детально розглянуто ці питання в розділі 3.

Перед вибором інструментів для розробки необхідно визначитись з яким типом графіки буде йти роботи. Розглянемо більш детально які види графіки існують.

1.2.1 Аналіз можливостей використання редактора Unity3D

Unity – це більше ніж ігровий движок, це середовище для розробки ігор на ПК та мобільних ігор, в якому об’єднанні різні програмні компоненти, які використовуються при створенні програмного забезпечення.

Перевагою Unity є те, що ти можеш почати розробляти власний програмний продукт не маючи за плечами великого досвіду розробки. Тобі не потрібні фундаментальні знання для того, щоб почати творити свій продукт. В Unity використовується компонентно-орієнтований підхід, в рамках якого ви створюєте якийсь об’єкт, а потім в порядку необхідності можете його кастомізувати. Другим великим плюсом Unity є великий вибір плагінів та асетів за допомогою яких можна сильно пришвидшити розробку.

Третім плюсом Unity є кросплатформеність – це підтримка великої кількості платформ, технологій та інтерфейсів користувача.

Великим недоліком при розробці ігор за допомогою Unity є те, що при розробці якось маштабного проекту, команді знадобиться досвідчений програміст на мові C#, який зможе написати скрипти до компонентів та налаштувати їх правильне функціонування.

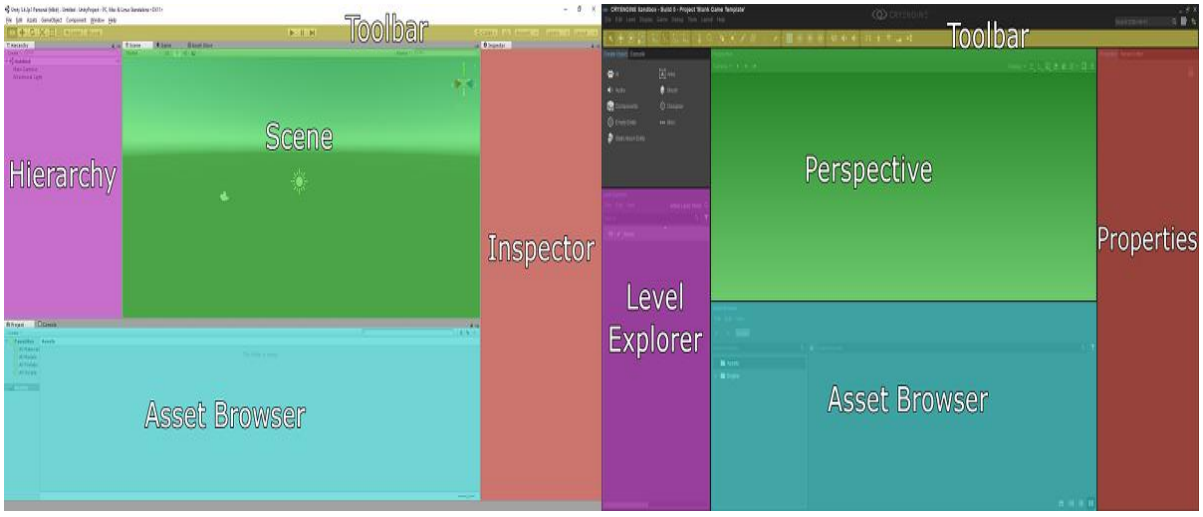


Рис.1.2. Інтерфейсне вікно Unity3D

Більш детально про функціональні можливості розглянуто в розділі 2.1.1.

1.2.2 Принципи формування графічного зображення

Розрізняють три види комп'ютерної графіки: растрову графіку, векторну графіку і фрактальну графіку.

Растрову графіку застосовують при розробці електронних (мультимедійних) і поліграфічних видань.

На відміну від растрової графіки, в якій зображення формується з сукупності точок, в векторній графіці зображення являє собою сукупність простих елементів: прямих ліній, дуг, кіл, еліпсів, прямокутників. Які називаються графічними примітивами. Положення і форма графічних примітивів задаються в системі координат, пов'язаних з екраном.

Фрактальна графіка, як і векторна, - обчислюється, але відрізняється тим, що ніякі об'єкти в пам'яті комп'ютера не зберігається. Зображення будується по управлінню, тому нічого, крім формули зберігати не треба. Зміна коефіцієнтів в рівнянні дозволяє отримувати абсолютно іншу картину.

Висновок до розділу 1

В ході написання даного розділу дипломної роботи було розглянуто, операційні системи персональних комп’ютерів та мобільних пристроїв. Було проведено аналіз використання операційних систем, які зображені на рисунку 1. В процесі аналізу операційних систем було прийняте рішення, що розроблений додаток буде інтегровано під операційну систему Android.

Також було проведено короткий аналіз можливостей та переваг редактору Unity3D, завдяки чому було вирішено остаточно працювати на даному движку. Було розглянуто і проаналізовано принципи формування графічних зображень.

РОЗДІЛ 2 АЛГОРИТМ РОЗРОБКИ КРОСПЛАТФОРМОВОЇ ГРИ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

2.1 Аналіз інструментів для розробки

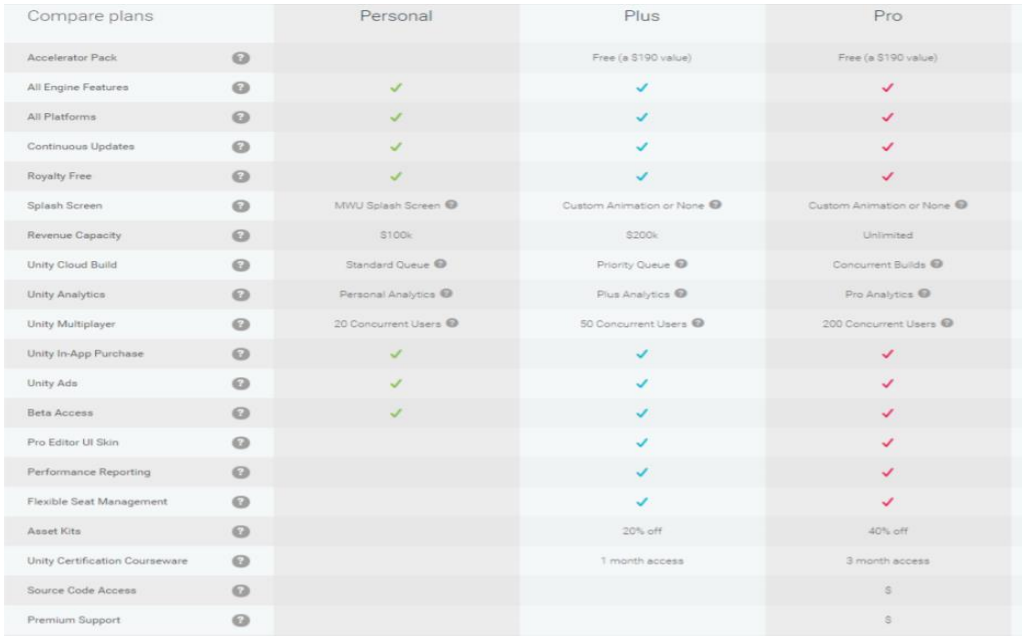
Перед початком розробки будь-якого програмного продукту необхідно проаналізувати поставлені задачі, далі вибрати необхідне програмне забезпечення яке відповідає всім вимогам і критеріям, а також зручне у використанні для самого розробника.

Аналізуючи поставленні задачі для свого дипломного проекту, я вибрав ряд програмного забезпечення, яке використовувалося на кожному з етапів розробки програмного продукту. Розглянемо їх:

- Blender – програма для створення анімацій;
- Visual Studio – інтегроване середовище розробки;
- Unity3D – графічний движок з можливістю використання 2D та 3D графіки.

2.1.1 Опис движку Unity3D

Перед початком роботи з Unity3D, я обрав одну з трьох доступних версій Unity, в розробці буду використовувати версію Personal, так як вона дає можливість працювати з даною програмою не вкладуючи кошти на отримання ліцензії. На відміну від версій Plus і Pro в неї обмежений функціонал, але це не заважало розробці. На малюнку зображено відмінність функціональності версій.



Compare plans	Personal	Plus	Pro
Accelerator Pack		Free (a $190 value)	Free (a $190 value)
All Engine Features	✓	✓	✓
All Platforms	✓	✓	✓
Continuous Updates	✓	✓	✓
Royalty Free	✓	✓	✓
Splash Screen	MWU Splash Screen	Custom Animation or None	Custom Animation or None
Revenue Capacity	$100k	$200k	Unlimited
Unity Cloud Build	Standard Queue	Priority Queue	Concurrent Builds
Unity Analytics	Personal Analytics	Plus Analytics	Pro Analytics
Unity Multiplayer	20 Concurrent Users	50 Concurrent Users	200 Concurrent Users
Unity In-App Purchase	✓	✓	✓
Unity Ads	✓	✓	✓
Beta Access	✓	✓	✓
Pro Editor UI Skin		✓	✓
Performance Reporting		✓	✓
Flexible Seat Management		✓	✓
Asset Kits		20% off	40% off
Unity Certification Courseware		1 month access	3 month access
Source Code Access			$
Premium Support			$

Рис.2.1. Функціональні можливості версій Unity

2.1.1.1 Інтерфейсне вікно Unity

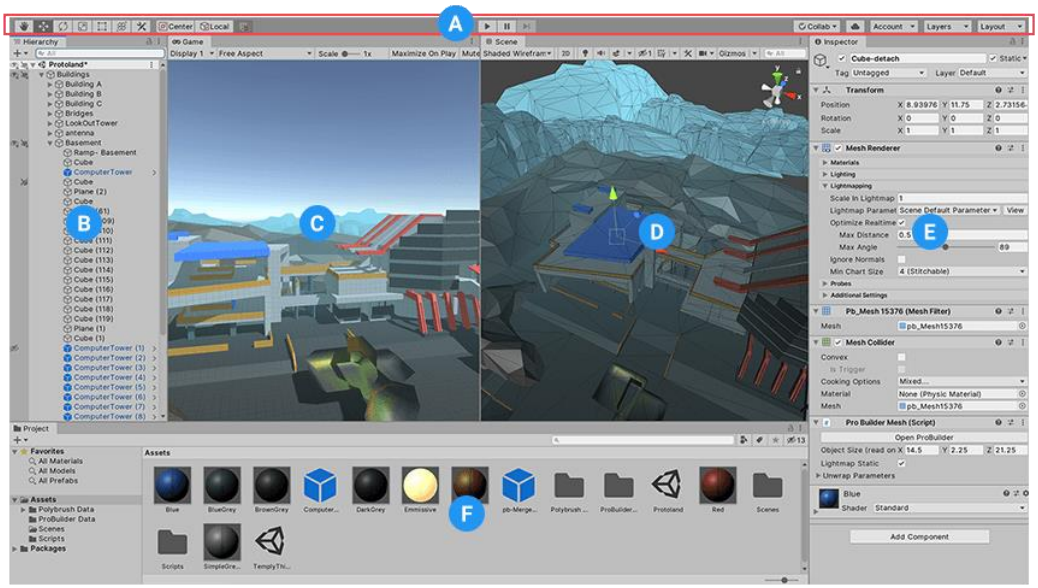


Рис.2.2. Інтерфейсне вікно

(A) Панель інструментів забезпечує доступ до найважливіших робочих функцій. Зліва він містить основні інструменти для керування видом сцени та об'єктами GameObjects у ньому. У центрі розташовані елементи управління відтворенням, паузою та кроком. Кнопки праворуч дають вам доступ до Unity Collaborate, Unity Cloud Services та вашого облікового запису Unity, за яким слідує меню видимості шарів і, нарешті, меню макета редактора (яке надає деякі альтернативні макети для вікон редактора та дозволяє зберігати ваші власні власні макети).

(B) Вікно ієрархії - це ієрархічне текстове представлення кожного GameObject у Сцені. Кожен елемент у Сцені має запис в ієрархії, тому два вікна невід’ємно пов’язані між собою. Ієрархія розкриває структуру того, як GameObjects приєднуються один до одного.

(C) Перегляд гри імітує, як буде виглядати ваша остаточна відтворена гра через ваші камери сцен. Після натискання кнопки «Відтворити» починається моделювання.

(D) Перегляд сцени дозволяє вам візуально орієнтуватися та редагувати вашу сцену. Вигляд сцени може показувати тривимірну або 2D-перспективу, залежно від типу проекту, над яким ви працюєте.

(E) Вікно інспектора дозволяє переглядати та редагувати всі властивості вибраного на даний момент GameObject. Оскільки різні типи GameObjects мають різні набори властивостей, макет та вміст вікна інспектора змінюються кожного разу, коли ви вибираєте інший GameObject.

(F) У вікні проекту відображається ваша бібліотека активів, доступних для використання у вашому проекті. Коли ви імпортуєте активи у свій проект, вони з’являються тут. [1]

2.1.1.2 Графіка

Для налаштування графіки в Unity необхідно відкрити Edit/Project Settings і далі вибрати вкладку Graphics. Далі розглянуте покрокове налаштування.

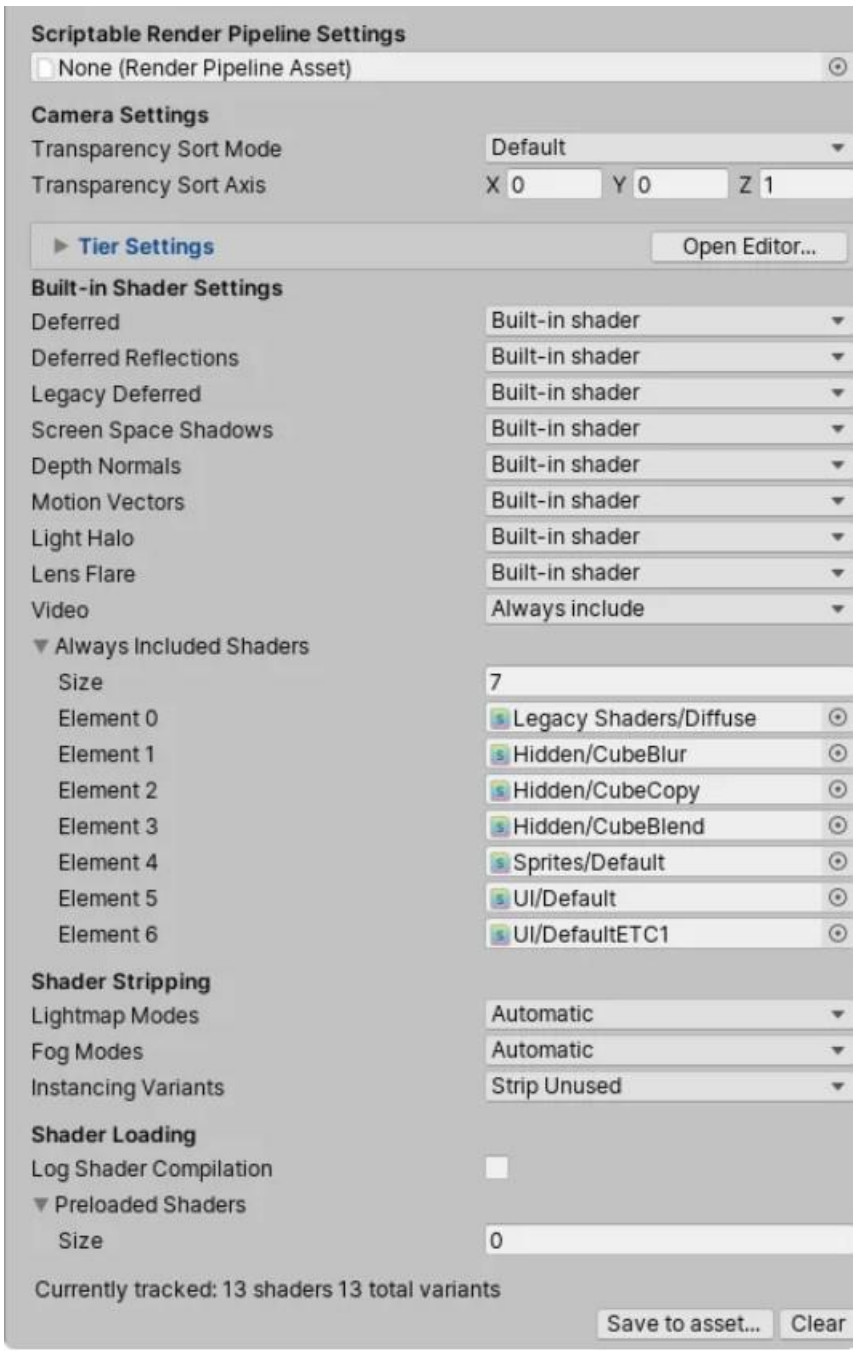


Рис.2.3. Налаштування графіки

Розглянемо більш детально доступні для використання параметри:

1) Scriptable Render Pipeline Settings (Налаштування параметрів візуалізації та скриптів) – даний параметр дозволяє визначити ряд команд, для того щоб точніше керувати чергою відображення об’єктів в сцені.

2) Camera Settings (налаштування камери) – група налаштувань, яка відповідає різним видам рендерінга.

2.1) Transparency Sort Mode – визначає порядок відображення об’єктів по

їх відстані відносно певної осі. Рендерінг в Unity сортується по певним кретеріям, таким як номер слоя або відстань до камери.

- Default – сортування об’єктів в залежності від положення камери.
- Perspective – сортування об’єктів в залежності від види перспективи.
- Orthographic – сортування об’єктів на основі ортографічного виду.
- Transparency Sort Axis – сортування прозорості за бажанням користувача.

3) Tier Settings (Налаштування рівня)

Рис.2.4. Налаштування Tier Settings відображаються всередині налаштувань Player Settings

Ці налаштуваня дозовляють вносити, спецефічні для платформи коригування в рендерінг і компіляції шейдерів, шляхом налаштування

вбудованих визначень. Рівні (Tiers) визначені в бібліотеці Rendering.GhaphicsTier.

Standart Shader Quality – встановлення якості стандартного шейдера на високий, середній або низький рівенью

Reflection Probes Box Projection – включає проекцію для відображення UV карт на Reflection Probes.

Reflection Probes Blending – включає змішування Reflection Probes.

Detail Normal Map – включає дану функцію якщо вона призначена і необхідна.

Enable Semitransparent Shadows – включає напівпрозорі тіні. Ця функція додає або видаляє визначення для компілятора шейдерів.

Enable Light Probe Proxy Volume – включає рендерінг 3D сітки з інтерпольованих Light Probes.

Cascaded Shadows – включає використання каскадних тіней.

Prefer 32 bit shadow maps – дає можливість роботи з 32 бітною картою тіней, коли ви націлені розробляти програмний продукт для консолі (PS4), або будете використовувати платформи, які використовують DirectX 11 або DirectX 12. Більшість платформ мають фіксований формат тіней, які ви не можете налаштувати. Вони відрізняються за форматами і можуть бути 12, 24 або 32 бітні.

Use HDR – включає рендерінг з високим динамічним діапазоном.

HDR Mode – дає можливість вибору формату, який буде використовуватися для буфера HDR. За базовим налаштуванням використовується формат FP16.

FP16 – формат який використовує 16 біт з плаваючою точкою на канал.

Rendering Path - виберіть як Unity повинен відображати графіку. Різні шляхи рендеринга впливають на продуктивність вашої гри, а також на те, як розраховуються освітлення і затінення. Деякі шляхи більше підходять для різних платформ і устаткування, ніж інші.

Відкладений (Deferred) рендеринг не підтримує при використанні ортогональної проекції. Якщо режим проекції камери встановлено на

ортогональний, то ці значення перевизначаються, і камера завжди використовує прямий (Forward) рендеринг.

- Forward - Традиційний шлях рендеринга. Він підтримує всі типові графічні функції Unity (карти нормалей, попіксельне освітлення, тіні і так далі). Однак відповідно до налаштувань за замовчуванням тільки невелика кількість найяскравіших джерел світла. Інші джерела світла обчислюються на вершинах об'єкта або на кожному об'єкті.
- Deferred - відкладене затінення володіє найбільшою точністю освітлення і тіней, і краще за все підходить, якщо у вас є багато джерел світла в реальному часі. Це вимагає певного рівня апаратної підтримки.
- Legacy Vertex Lit - це шлях рендеринга з найнижчою точністю освітлення і відсутністю підтримки тіней в реальному часі. Це підмножина шляху прямого (Forward) рендеринга.
- Legacy Deferred (light prepass) - Відкладений рендеринг (застарілий) схоже на його побратима відкладене затінення, просто використовує іншу техніку з різними компромісами. Він не підтримує стандартний шейдер Unity 5 на основі фізики (PBR шейдер з'явився з 5 версії Unity).

Realtime Global Illumination CPU Usage - виберіть який обсяг використання ЦП слід призначити для розрахунків освітлення під час виконання. Збільшення цього показника змушує систему швидше реагувати на зміни освітлення за рахунок використання більшої кількості процесорного часу. Деякі платформи дозволяють всім процесорам бути зайнятими робітниками потоками, тоді як на інших застосовують максимуми.

- Low - 25% потоків ЦП використовуються в якості робочих потоків.
- Medium - 50% потоків ЦП використовуються в якості робочих потоків.
- High - 75% потоків ЦП використовуються в якості робочих потоків.
- Unlimited - 100% потоків ЦП використовуються в якості робочих потоків.

Built-in shader settings –параметр, необхідний щоб вказати, який шейдер слід використовувати для кожної з перерахованих функцій.

Deferred – функція з відкладеним затінюванням.

Deferred Reflection - функція з Reflection Probes уздовж відкладеного освітлення.

Screen Space shadows - функція з каскадними тіньовими картами для спрямованого освітлення на платформах ПК / консолі.

Legacy deferred - функція із застарілим відкладеним освітленням.

Motion vectors - використовується для розрахунку вектора руху на основі об'єкта.

Для кожної з цих функцій можна вибрати, який тип шейдера буде використовуватися:

No Support - відключає ці розрахунки. Використовуйте цей параметр, якщо ви не використовуєте відкладене затінення або освітлення. Це дозволить заощадити деякий простір у вбудованих файлах ігрових даних.

Built-in Shader - для виконання обчислень використовуються вбудовані шейдери Unity. Це значення за замовчуванням.

Custom Shader - для виконання обчислень використовується ваш власний сумісний шейдер. Це дозволяє виконувати глибоку настройку відкладеного рендеринга.

Коли ви вибираєте пункт Custom Shader, починає відображатися властивість reference під властивістю feature, в якому можна задати посилання на шейдер, який ви хочете використовувати.

Always-included Shaders – список шейдеров, які завжди зберігаються разом з проектом, навіть якщо вони фактично не використовуються у вашій сцені. Важливо додати шейдери, використовувані в AssetBundles в цей список, щоб забезпечити до них доступ.

Щоб додати шейдер в список, збільште його значення Size. Для видалення навпаки зменшите властивість Size. Щоб видалити шейдер, який не є останнім у списку, можна встановити йому значення None.

Shader stripping – Зменшіть розмір даних збірки і збільште час завантаження, видаливши деякі шейдери. За замовчуванням Unity дивиться на ваші сцени і настройки lightmapping, щоб з'ясувати, які варіанти туману і

lightmapping не використовуються, і пропускає відповідні варіанти шейдеров. Однак ви можете вибрати певні режими, якщо ви створюєте бандли Ассет (asset bundles), щоб гарантувати, що режими, які ви хочете використовувати, включені.

За замовчуванням властивість Lightmap modes має значення Automatic, що означає, що Unity вирішує, які варіанти шейдера пропустити.

Щоб вказати, які режими використовувати самостійно, змініть цей параметр на призначений для користувача і увімкніть чи вимкніть такі режими lightmapping:

- Baked Non-Directional
- Baked Directional
- Realtime Non-Directional
- Realtime Directional
- Baked Shadowmask
- Baked Subtractive

Та ж історія стосується і туману, тому для вказівки режиму використання самостійно, змініть параметр на призначений для користувача і увімкніть чи вимкніть такі режими fog:

- Linear
- Exponential
- Exponential Squared

Strip Unused (Default value) - коли Unity створює білд, він включає в себе тільки ті варіанти шейдеров, які щонайменше мають хоча б один матеріал з посиланням на шейдер і включеним Enable instancing. Unity видаляє всі шейдери, на які не посилаються матеріали з відключеними Enable instancing.

Strip All - видалення всіх примірників варіантів шейдера, навіть якщо вони використовуються.

Keep All - зберігає всі варіанти примірників шейдеров, навіть якщо вони не використовуються.

Shader preloading.

Вкажіть список варіантів шейдеров з колекції Ассет для попереднього завантаження під час завантаження гри. Зазначені в цьому списку варіанти шейдеров завантажуються на протязі всього терміну життя додатки. Використовуйте його для попереднього завантаження дуже часто використовуваних шейдеров.

2.1.1.3 Налаштування звуку

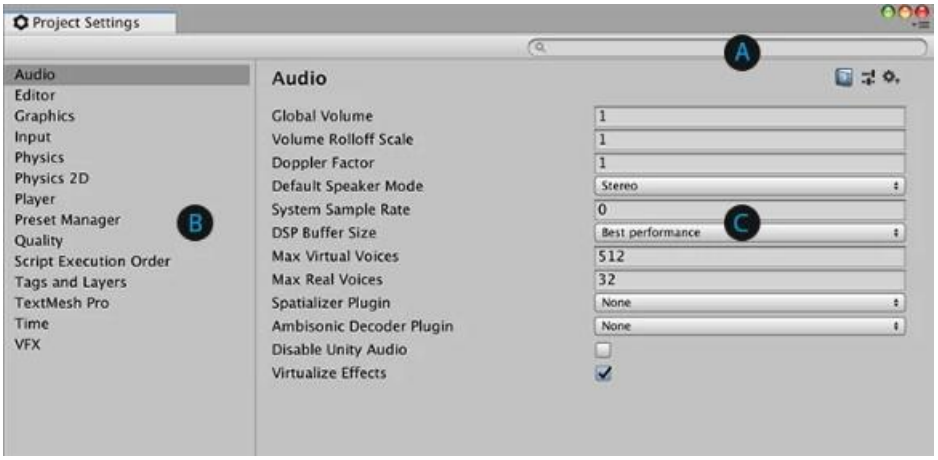


Рис.2.5. Вікно налаштування

Вікно налаштування поділено на три області, розглянемо кожну із них:

(A) За допомогою пункту A, можна фільтрувати список налаштувань, по категоріям зліва і виводи пошукові слова з права.

(B) Всі налаштування розбиті по катерогіям, які відображаються в списку категорій. При виборі певної категорії налаштування будуть відображатися з права.

(C) В даній області показані доступні налаштування, для вибраної категорії.

Налаштування звуку дозволяють налаштувати максимальну гучність всіх звуків котрі буду використовуватися на сцені. Для початку в налаштуваннях Project Settings виберіть категорію Audio.

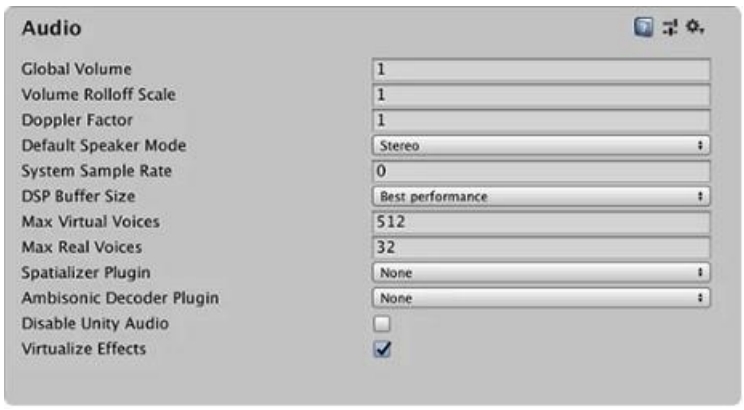


Рис.2.6. Інтерфейсне вікно налаштування звуку.

Розгянемо, для чого необхідне кожне з налаштувань більш детально:

Global Volume – установка гучності звуку, для всіх звуків які будуть програватися.

Volume Rolloff Scale – цей параметр являє собою спад частотної характеристики коефіцієнта затухання. Чим більше значення даного коефіцієнта тим швидше зменшується гучність.

Doppler Factor – цей параметр використовується для ефекта Допплера.

Default Speaker Mode – цей параметр дозволяє змінити режим відтворення звуку.

- Mono – використовує динаміки в одноканальному режимі.
- Stereo – використовує динаміки в двоканальному режим.
- Quad – використовує чотири канали. Включає в себе передні і задні, лівий і правий динаміки.
- Surround – використовує п’ять каналів. Включає в себе ті же динаміки що і Quad але має на відміну від них додатковий центральний.

System Sample Rate – установка частоти дискретизації* (*частота дискретизації - взяття відліків неперервного за часом сигналу при його дискретизації)

DSP Buffer Size – вибір розміру буфера DSP (Digital signal processing – процесор цифрових сигналів) для оптимізації між затримкою і продуктивністю.

- Default – стандартний розмір буферу;
- Best Latency – компроміс між продуктивністю і затримкою. Перевага при виборі цього режиму надається затримці;

- Good Latency – баланс між затримкою і продуктивністю;
- Best Perfomance – працює аналогічно як і Best Latency але в іншу сторону. Перевага надається продуктивності.

Max Virtual Voices – дана функція дає можливість встановлювати кількість віртуальних голосів, якими може керувати аудіосистема. Значення цього параметру завжди має бути більше ніж кількість використовуваних голосів.

Max Real Voices – функція дає можливість встановлювати кількість реальних голосів, котрі зможуть програватись одночасно. В кожному кадрі буде вибраний найгучніший голос.

Virtualize Effect – включення та виключення динамічного відключення ефектів в просторі AudioSources які відкидаються для того щоб зберегти продуктивність роботи процесора.

Disable Unity Audio – включає або відключає аудіосистему на персональних комп'ютерах.

2.1.1.4 Фізика 2D

При створенні проекту є можливість вибору між двома режимами роботи редактора це 2D і 3D. Але в будь-який момент часу ви можете переключатись між цими режимами.

Ці два режими визначають налаштування редактора з якими вони будуть працювати. Розглянемо ці два режими роботи більш детально.

В режимі 2D редактор має наступні властивості:

- Будь-яке імпортоване зображення, рахується як 2D об'єкт (спрайт) і виставляється в режимі Sprite;
- Активований режим Sprite Packer;
- Вікно відображення сцени працює в режимі 2D;
- При створенні нової сцени створюється камера яка знаходиться в положенні по координатам 0;0;-10 і має ортографічну проекцію;

В режимі 3D редактор має наступні властивості:

- Будь-які імпортовані зображення в проект не рахуються спрайтами;
- Режим Sprite Packer деактивується;
- Вікно сцени працює в режимі 3D;
- При створенні нової сцени створюється камера яка знаходиться в положенні по координатам 0;1;-10 і має перспективну проекцію;

Зазвичай 3D ігри використовують трьохвимірну геометрію, а матеріали і текстури візуалізуються на поверхності цих ігових об’єктів, для того щоб вони виглядали як оточення цих ігрових об’єктів. Камери може вільно переміщуватися разом з світлом і тінями, які є досить реалістичними. 3D ігри зазвичай візуалізують за допомогою камер в режимі перспективи, через це об’єкти які знаходяться на екрані візуально мають більший розмір чим мають в реальності. Іноді в іграх використовується трьохвимірна геометрія, але замість перспективної використовують ортографічну камеру. Це розповсюджена техніка де камера має вид зверху, ці ігри називають 2.5D іграми.

Більшість 2D ігор використовують плоску графіку (спрайти), яка взагалі немають трьохвимірної геометрії. Сцена будується у виді плоских зображень які накладуються один на одне. Деякі 2D ігри використовують трьохвимірну геометрію для створення персонажів, но при цьому обмежують ігровий процес двома вимірами.

Ще один із доступних можливостей при розробці 2D ігор, використовується 2D графіка разом з перспетивною камерою, для того щоб отримати ефет параллакса (об'єкти які знаходяться ближче до камери рухаються швидше ніж об'єкти які знаходяться далі)

В моєму дипломному проекті є створив 2D гру, розгянемо налаштування 2D фізики в середовищі Unity:

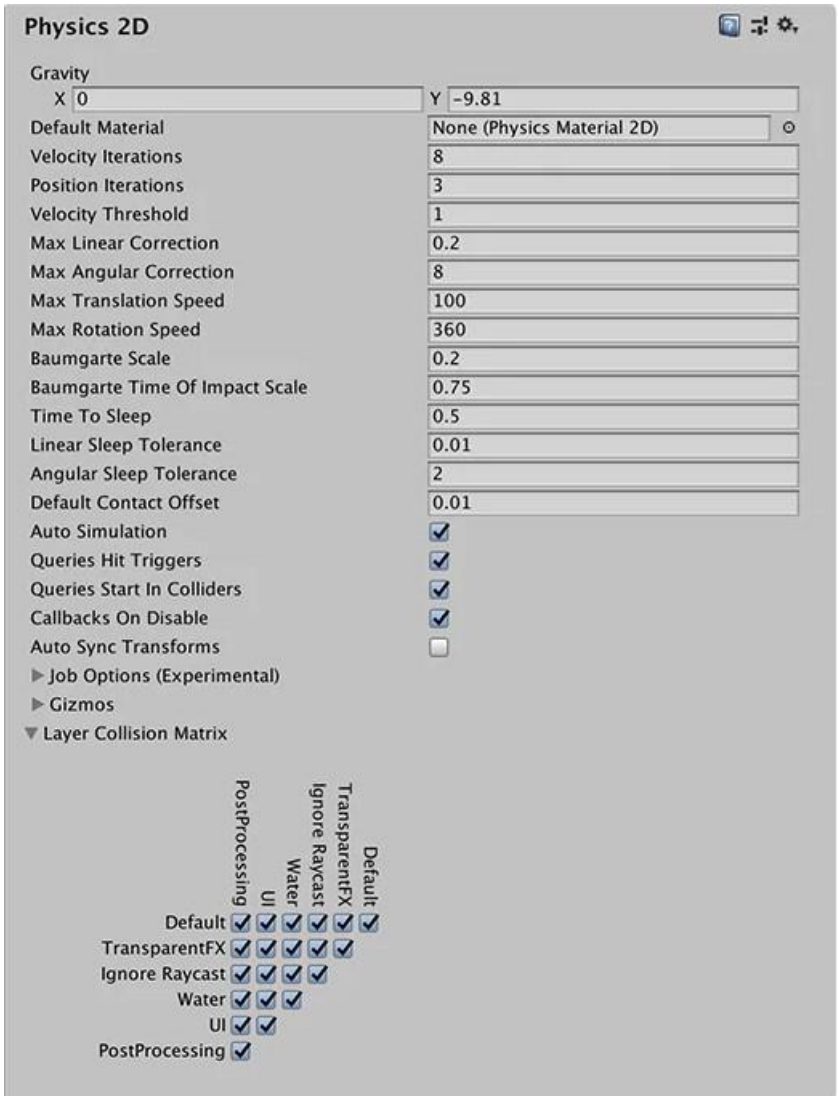


Рис.2.7. Секція налаштування Physics2D

Налаштування Physics2D визначають межі точності фізичного моделювання. Більш точне моделювання потребує більших затрат на обробку, через це налаштування пропонують спосіб обміняти точність на швидкодію.

Розглянемо параметри налаштування більш детально:

Gravity – параметр дозволяє встановити величину тяжіння для всіх доступних ігрових об’єктів з компонентом RigidBody2D. Як правило, вставновлюється гравітація для напрямку осі У;

Default Material – дозволяє встановити посилання на фізичний матеріал PhysicsMaterial2D для використання за замовчуванням, якщо жоден з них не був призначений окремому Collider2D;

Velocity Iterations – дозволяє встановити число ітерацій, виконуваних фізичним движком для вирішення ефектів швидкості. Більш високі числа призводять до більш точної фізики, але за рахунок процесорного часу;

Position Iterations - дозволяє встановити кількість ітерацій, виконуваних фізичним движком для вирішення змін положення. Більш високі числа призводять до більш точної фізики, але за рахунок процесорного часу;

Velocity Threshold – дозволяє встановити порогове значення для пружних зіткнень. Unity розглядає зіткнення з відносною швидкістю нижче цього значення як неупругие зіткнення;

Max Linear Correction – дозволяє встановити максимальну лінійну корекцію положення, використовувану при вирішенні обмежень (від 0,0001 до 1000000);

Max Angular Correction – дозволяє встановити максимальну кутову поправку, яка використовується при вирішенні обмежень (від 0,0001 до 1000000);

Max Translation – дозволяє встановити максимальну лінійну швидкість ігрового об'єкта з компонентом RigidBody2D при будь-якому оновленні фізики;

Max Rotation Speed – дозволяє встановити максимальну швидкість обертання ігрового об'єкта з компонентом RigidBody2D при будь-якому оновленні фізики;

Baumgarte Scale – дозволяє встановити коефіцієнт масштабу, що визначає, як швидко Unity дозволяє накладення колізій;

Baumgarte Time of Impact Scale – дозволяє встановити коефіцієнт масштабу, що визначає, як швидко Unity дозволяє перекриття часу впливу;

Time to Sleep - час (в секундах), яке повинно пройти після того, як Rigidbody2D перестане рухатися, перш ніж він засне;

Linear Sleep Tolerance – дозволяє встановити лінійну швидкість, нижче якої Rigidbody2D переходить в сплячий режим після закінчення часу сну;

Angular Sleep Tolerance – дозволяє встановити швидкість обертання, нижче якої Rigidbody2D переходить в сплячий режим після закінчення часу сну;

Default Contact Offset – дозволяє встановити значення відстані близькості для коллайдеров, які будуть вважатися контактирующими, навіть якщо вони

фактично не перебувають у контакті. Колайдери, відстань між якими менше суми їх значень зміщення контактів, генерують контакти. Це дозволяє системі виявлення зіткнень прогнозувати і застосовувати обмеження контакту навіть при незначному поділі об'єктів;

Auto Simulation - – дозволяє автоматично запускати фізичну симуляцію або дозволити явне управління нею;

Queries Hit Triggers – дозволяє встановити Collider2D, зазначений як тригер, повертав потрапляння, коли будь-який фізичний запит (наприклад, LineCast або RayCast) перетинається з ними;

Queries Start In Colliders – дозволяє встановити фізичні запити, які починаються всередині Collider2D, виявляли коллайдер, в якому вони починаються;

Callbacks On Disable – дозволяє виробляти зворотні виклики зіткнень, коли коллайдер з контактами відключений;

Auto Sync Transforms – дозволяє автоматично синхронізувати зміни перетворення (Transform) з фізичної системою;

Layer Collision Matrix – дозволяє визначити, як поводиться система виявлення зіткнень на основі шарів;

Налаштування Job options.

Параметри даних налаштувань дозволяють налаштувати використання системи Job System для настройки багатопотокової фізики.

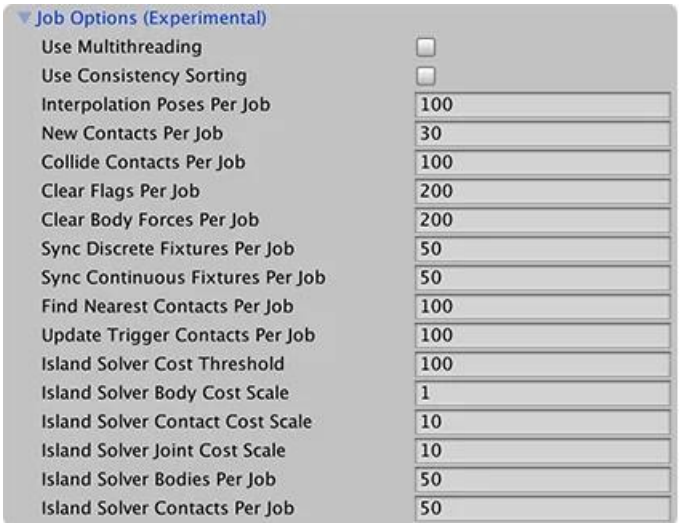


Рис.2.7. Секція Job options

Use Multithreading – параметр для виконання кроків моделювання (simulation) фізики за допомогою системи завдань (Job System) і використовуйте інші параметри для управління тим, як це зробити.

Use Consistency Sorting – параметр для підтримки узгодженого порядку обробки стає важливим для моделювання. Виконання кроків моделювання на декількох потоках ЦП призводить до отримання окремих пакетів даних.

Обробка цих окремих пакетів зменшує детермінізм в порядку обробки, хоча і дає більш швидкі результати.

Interpolation Poses Per Job - встановіть мінімальну кількість ігрових об'єктів з компонентів RigidBody2D, інтерпольованої в кожному завданні моделювання;

New Contacts Per Job – дозволяє встановити мінімальну кількість нових контактів для пошуку в кожному завданні моделювання;

Collide Contacts Per Job – дозволяє встановити мінімальну кількість контактів для зіткнення в кожному завданні моделювання;

Clear Flags Per Job – дозволяє встановити мінімальну кількість прапорів, які будуть очищені в кожному завданні моделювання;

Clear Body Forces Per Job – дозволяє встановити мінімальну кількість тіл, що очищаються в кожному завданні моделювання;

Sync Discrete Fixtures Per Job – дозволяє встановити мінімальну кількість пристосувань для синхронізації в широкій фазі при вирішенні дискретних острівців в кожному завданні моделювання;

Sync Continuous Fixtures Per Job – дозволяє встановити мінімальну кількість пристосувань для синхронізації в широкій фазі при безперервному вирішенні острівців в кожному завданні моделювання;

Find Nearest Contacts Per Job – дозволяє встановити мінімальну кількість найближчих контактів для пошуку в кожному завданні моделювання;

Update Trigger Contacts Per Job – дозволяє встановити мінімальну кількість контактів тригера для поновлення в кожному завданні моделювання;

Island Solver Cost Threshold – дозволяє встановити мінімальну порогову вартість всіх тіл, контактів і з'єднань;

Island Solver Body Cost Scale – дозволяє встановити шкалу витрат для кожного тіла при вирішенні дискретного острова;

Island Solver Contact Cost Scale – дозволяє встановити шкалу вартості кожного контакту при вирішенні дискретних острівців;

Island Solver Joint Cost Scale – дозволяє встановити шкалу витрат для кожного з'єднання під час дискретного острівного рішення;

Island Solver Bodies Per Job – дозволяє встановити мінімальну кількість тіл для вирішення в кожному завданні моделювання при виконанні острівного рішення;

Island Solver Contacts Per Job – дозволяє встановити мінімальну кількість контактів для вирішення в кожному завданні моделювання при виконанні острівного рішення.

2.1.1.5 Інтеграція між Visual Studio і Unity

Visual Studio – це інструмент інтегрованого середовища розробки. Дане середовище розробки випускається в трьох версіях:

- Community (Безкоштовне);

- Professional (Платне);
- Enterprise (Платне);

Поддерживаемые возможности	Visual Studio Community Скачать бесплатно	Visual Studio Professional Купить	Visual Studio Enterprise Купить
⊕ Поддерживаемые сценарии использования	●●●○	●●●●	●●●●
Поддержка платформ разработки [2]	●●●●	●●●●	●●●●
⊕ Интегрированная среда разработки	●●●○	●●●○	●●●●
⊕ Расширенная отладка и диагностика	●●○○	●●○○	●●●●
⊕ Средства тестирования	●○○○	●○○○	●●●●
⊕ Кроссплатформенная разработка	●●○○	●●○○	●●●●
⊕ Инструменты и возможности для совместной работы	●●●●	●●●●	●●●●

Рис.2.8. Функціональні можливості безкоштовних і платних версій Microsoft Visual Studio

Інтеграція Unity з Visual Studio дозволяє автоматично створювати і підтримувати файли проектів Visual Studio.

Під час написання скриптів ви можете відреагувати файл, зберегти його і переключитися в Unity, для того щоб перевірити і переглянути внесені вами зміни.

Не дивлячись на те, що Visual Studio має свій власний компілятор C#, Unity використовує використовує свій власний компілятор для перевірки вашого коду. Використання Visual Studio досить корисне, у вас немає необхідності постійно перемикатися на Unity, для того щоб зрозуміти є у вас помилки чи ні.

Компілятор Visual Studio має трохи більше можливостей ніж компілятор Unity. Це означає, що певні елементи коду не будуть видавати помилки в Visual Studio, але можуть видавати помилки в Unity.

Unity автоматично створює і підтримує в Visual Studio файли форматів .sln і .csproj. Кожен раз коли користувач виконує дії (додає\вилучає\переміщає\перейменовує файл), Unity автоматично відновлює його з файлів .sln і .csproj. Потім імпортує ці нові файли, і в наступний раз коли Unity створить файли проекту, вони будуть відтворені разом з ними.

2.1.1.5.1 Опис вибраної мови програмування

Движок Unity підтримує 7 мов програмування. Для написання коду вам не буде необхідності вивчати нову мову програмування з нуля. Для розробки свого проекту я вибрав мову програмування C# по причині того, що вона має достатньо великий фунціонал для розробки і через те що з цих доступних семи мов програмування зручніше всього мені буде працювати с C# так як я знаю цю мову програмування.

Розглянемо і проаналізуємо доступні мови програмування:

- C# – являється об’єктно-орієнтованою мовою програмування, з можливістю підтримки компонентно-орієнтованого програмування. C# являється мовою програмування яка має спільні корні з C подібними мовами програмування такими як C, C++, Java.

Розробка сучасних додатків все більше направлена до створення програмних компонентів у формі автономних і самопишущих пакетів, що реалізують окремі функціональні можливості. Важлива особливість таких компонентів - це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. C # надає мовні конструкції для безпосередньої підтримки цих концепцій, роблячи C # дуже природна мова для створення і використання програмних компонентів. [2]

- Boo – об’єктно-орієнтована мова програмування з сильною статичною типізацією для платформи .NET. Синтаксис побудований на основі синтаксисів CPython і IronPyton, за рахунок цього його синтаксис не являється динамічно-

типізованим за умовчанням. Використовувався в Unity до версії 5 для створення трьохвимірних додатки для різних операційних платформ.

- JavaScript – це невибаглива до ресурсів мову програмування з функціями першого класу, код якої інтерпретується і компілюється під час виконання. Хоча JavaScript перш за все відома як скриптова мова для веб-сторінок, вона також використовується в багатьох небраузерніх середовищах на кшталт Node.js, Apache CouchDB і Adobe Acrobat. JavaScript - прототип-орієнтована динамічна мова, має кілька парадигм і підтримує об'єктно-орієнтована, імперативний і декларативний (тобто функціональне програмування) стилі. [3]

- Lua – швидка и компактна скриптова мова програмування. Інтерпретатор мови є вільно пошірюванім, з відкрітім сирцевої кодом на мові C. За можливости, ідеологією и реалізацією, мова найближче до JavaScript, проти Lua відрізняється могутнішімі и набагато гнучкішімі конструкціямі. Хоча Lua НЕ містіть Поняття класу и об'єкта в явному виде, Механізми об'єктно-орієнтованого програмування (ООП) з підтрімкою прототіпів (включаючі множини Успадкування) легко реалізуються з Використання метатабліць, Які також дозволяють перевантаження операцій, тощо. [4]

- IronPython – це реалізація Python з відкритим кодом для .NET CLR та Mono. IronPython використовує Dynamic Language Runtime, структуру для написання динамічних мов для .NET, що виникла в IronPython 1. Він також працює на Silverlight, плагіні для браузера .NET, який працює на Windows і Mac (а монопорт, що називається Moonlight, працює на Linux). Це означає, що IronPython можна використовувати для скриптів на стороні клієнта у браузері. Продуктивність порівнянна з CPython - набагато швидша для деяких речей (де він може скористатися перевагами компілятора JIT в базовій платформі), але повільніша для інших речей (особливо вбудованих типів контейнерів, де було зроблено багато роботи з оптимізації Типи CPython). [5]

- Rust – експериментальна мова програмування, що розробляється

Mozilla. Мова позиціонується як альтернатива C/C ++. У Rust підтримуються функіцональние, паралельне, процедурне і об'єктно-орієнтоване програмування, тобто майже весь спектр реально використовуваних в прикладному програмуванні парадигм. [6]

- C/C++ – компільована статично типізована мова програмування

загального призначення. Спочатку був розроблений для реалізації операційної системи UNIX, але згодом був перенесений на безліч інших платформ. Згідно дизайну мови, його конструкції близько зіставляються типовим машинним інструкціям, завдяки чому він знайшов застосування в проектах, для яких був властивий мову асемблера, в тому числі як в операційних системах, так і в різному прикладному програмному забезпеченні для безлічі пристроїв - від суперкомп'ютерів до вбудованих систем. Мова програмування C справив значний вплив на розвиток індустрії програмного забезпечення, а його синтаксис став основою для таких мов програмування, як C ++, C #, Java і Objective-C. [7]

2.1.1.6 Порівняння Unity з аналогами

При проведені порівняльної характеристики розглянемо два найбільш популярних аналогів Unity це Unreal engine і Godot. Розглянемо більш детально їх можливості в підрозділах нижче.

2.1.1.6.1 Порівняння Unity з Unreal Engine

Перед початком роботи багато програмістів-початківців зіштовхуються з проблемою який ігровий движок краще вивчати для подальшої роботи та створення власних проектів. По власному досвіду зауважу, що на вирішення цього питання витрачається досить велика кількість часу.

Як і у випадку з Unity, Unreal engine дозволяє створювати ігри для більшості операційних систем, консолі і мобільних платформ. Завдяки підтримці різних

систем рендерінга графіки Direct3D, OpenGL, підтримки різних систем відтворення звуку і можливості для створення мережевих ігор.

Наприклад на ігровому движку Unreal Engine була створена в свій час одна із найпопулярніших MMOPRG Lineage2.

До Складу Unreal Engine входить середовище розробки(SDK) і редактор.

На відміну від Unity, Unreal Engine має відкритий код, який написаний на C++. Якщо говорити про переваги Unreal Engine, потрібно розуміти, що Unity більше підходить для мобільних 2D ігор а Unreal Engine дозволяє працювати з складною просунотою графікою.

В Unreal Engine є примітна система скриптів, які створюються без використання мов програмування, з допомогою якої можна описати все, що завгодно від дій персонажа до процедурної генерації рівнів гри. За рахунок використання движку Unreal Engine можна створювати фотореалістичну графіку при архітектурній візуалізації.

2.1.1.6.2 Порівняння Unity з Godot

Для сценаріїв у Godot є власний вбудований редактор сценаріїв та візуальний сценарій. Unity потребує встановлення окремих програм (Monodeveloper, Visual Studio, notepad ++) сценарії і єдність скриптів скасовується на корисить C#.

Великою перевагою Unity є можливість призначати та змінювати змінні за допомогою інспектора (перетягування об’єктів на сцену), тоді як у Godot змінні необхідно встановлювати в сценарії. Godot часто критикують за те, що він може використовувати лише один сценарій на об’єкт, вузли працюють більше ніж компоненти, ніж GameObjects, тому вам необхідно створити тут кількість вузлів, скільки буде необхідно вам для роботи.

Для управління проектом такими як матеріали, обидва механізми створюють папку в Godot (res://) а в Unity (activities). Коли ви створюєте щось на зразок матеріалі, Unity збегірає його в папці (materials) а Godot за замовчанням

зберігає матеріали в самій сцені. І нарешті, можливості 3D у більш нових версіяї Godot зараз майже на рівні з Unity.

2.1.2 Blender програма для створення анімації

Blender – один із найбільш функціональних безкоштовних 3D редакторів. Редактор вміє майже все необхідне:

- Створювати моделі та анімацію;
- Створювати текстури;
- Працювати з освітленням об’єкту;
- Можливість створювати власні матеріали для текстур.

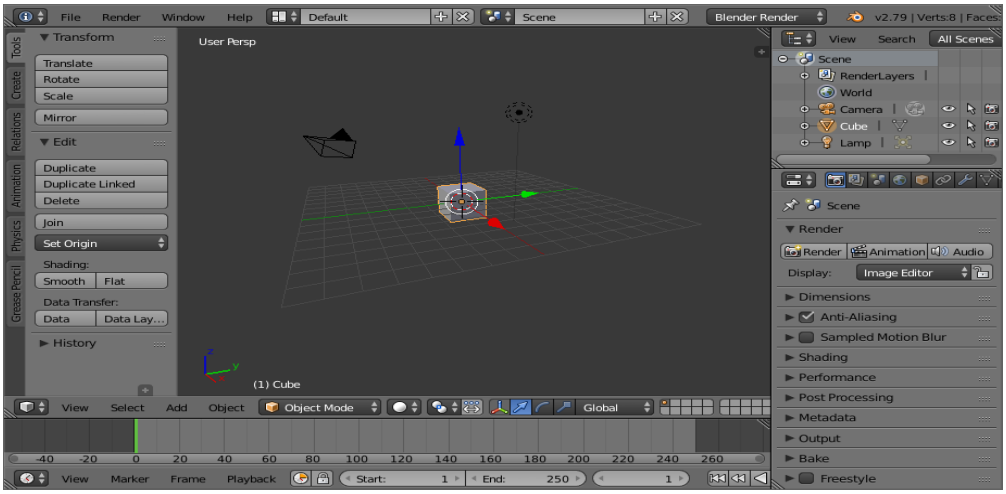


Рис.2.9. Інтерейсне вікно Blender

Інтерфейс програми майже нічим не відрязняється від інших 3D редакторів, має велику кількість панелей і кнопок. Необхідно зауважити те, що для різних режимів роботи можна обрати різні інтерфейси.

На відміну від більшої кількості безкоштовних 3D редакторів, Blender також уміє створювати анімацію.

Blender позиціонується як додаток для створення і редагування тривимірної графіки, візуалізації, анімації, створення комп'ютерних ігор і навіть скульптінга. Цілком серйозна програма, для якої потрібні серйозні ресурси апаратного забезпечення, скажете ви. Але перший сюрприз пакет підносить на етапі скачування настановних файлів - їх вага становить близько 70 Мбайт. Немислимо для програмного забезпечення такого рівня! Проте, розробники примудрилися забезпечити програму усіма необхідними функціями, які повноцінно функціонують і демонструють відмінну продуктивність.

Функції програми:

- 3D моделювання – доступне проектування об’єктів на основі примітивів, полігонів, NURBS-кривих, кривих Базьє, метасфер, булевих операцій;
- Анімація – доступні інструменти для створення скелетної анімації, інверсної кінематики, сіткової деформації, анімації по ключовим кадра. Реалізована динаміка тверди та м’яких тіл.
- Створення текстур – програма дозволяє накладати декілька текстур на один об’єкт і оснащена рядом інструментів для створення текстур.
- Можливість малювання – програма дає можливість створювати ескізи різними типами кісточок у вікні програми.
- Візуалізація – пакет оснащений певною кількістю вбудованих інструментів візуалізації, а також доступна можливість інтеграції різних зовнішніх рендерів.
- Базовий відеоредактор – в програму вбудований відеоредактор з деякими можливостями для створення простих відео.
- Ігровий движок – вбудований ігровий движок для створення інтерактивних 3D додатків.

2.2 Етапи та методи розробки мобільного додатку

Перед початком розбору алгоритму розробки мобільного додатку необхідно розглянути які обов’язки є у самого розробника:

- Створення технічного завдання;
- Побудова архітектури додатку;
- Програмування;
- Створення дизайну;
- Підтримка власне випущеного програмного продукту;
- Оформлення документації.

Можна виділити наступні стадії розробки додатку:

- Пошук основної ідеї, тобто концепції додатку.

На цьому етапі проводиться аналіз ринку а також уже існуючих продуктів. Перш ніж перейти до наступного етапу, обов'язково потрібно детально описати бізнес-вимоги продукту.

- Визначення базового функціоналу.

Мобільні додатки створюються поступово, спочатку виходить тестова Demo-версія зі спрощеним функціоналом, а потім поступово в наступних оновленнях додаються функції які були допрацьовані і відтестовані та готові для використання користувачами. Розробник збирає інформацію від користувачів, аналізує і потім продумує, що краще удосконалювати на наступних етапах розробки та підтримки.

На цьому етапі визначається набір базових функцій які доступні користувачеві, та можливість користувача взаємодії з ними.

- Створення дизайну.

На даному етапі починається робота з конкретних фунціоналом і з чітким розумінням того які кроки у використанні буде робити користувач. Створюється інформаційна архітектура, тобто основна структура системи:

1. Прописується сценарії використання;
2. Підключаються системні аналітики;
3. Створюється протип і дизайн взаємодії.

Потім проводиться тестування продукту. Аналізується чи бачуть користувачі цінність додатку, та чи здатні вони розібратися як ним користуватися. За результатами тестів можливі внесення змін в дизайн.

- Розробка мобільного додатку.

В процесі створення мобільного додатку продовжується тестування і покращення, але важливо дотримуватися плану поетапного випуску, причиною цього є можливість внесення змін в функціонал та дизайн продукту, які дуже часто впливають на час розробки.

- Маркетинг.

На даному етапі підключається відділ маркетингу, і вони займаються просуванням програмного продукту на різних платформах це SMM(контекста реклама), SEO(розсилка листів можливим користувачам), реклама на різних ринкових майданчиках (AppStore, PlayMarket).

Далі проходить фінальне тестування програмного продукту, оцінка проекту, створення документації, виправлення багів (помилок у роботі) і додаток переходить у передрелізний стан.

- Реліз (випуск).

Коли серія тестів і виправлення помилок завершено і кожен з відділів розробки задоволений отриманим результатом проект випускається для широкої маси користувачів, але на цьому створення додатку не завершується, далі здійснюється підтримка працездатності, оновлення, та розширення функціоналу.

Розглянемо методи розробки мобільних додатків, всього існує два методи:

1. Нативна розробка.

Нативна розробка включає в себе створення програми для мобільного пристрою на конкретній мові програмування під конкретну платформу. Нативні проекти досить продуктивні і немають ніякиї обмежень в розробці. До плюсів можна віднести швидку реакцію на дії користувача. До недоліків можна віднести високу вартість розробки і підримки програмного продкуту, і досить тривалий час який не обіхдний для розробки.

2. Гібридна розробка.

Розробляти один і той же додаток під різні платформи довго і дуже дорого, ще одними недоліком є те, що тестування займе тривалий час, виходячи з цього, якщо необіхдно створити простий додатоку відразу для декількох платформ

використовують метод гібридної розробки. Гібридна розробка проводиться за допомогою web-технологій – HTML, CSS, JavaScript, які дозволяють створювати додаток відразу на кілька платформ. До переваг можне віднести низьку вартість розробки продукту. А до недоліків можна віднести труднощі в роботі всіх функцій і затримки реакції на дії користувача.

Висновок до розділу 2

В ході написання другого розділу дипломної роботи були проаналізовані можливості і причини вибору програмного забезпечення для розробки дипломного проекту, були розглянуті функціональні можливості вибраного програмного забезпечення та проведено порівняння зі схожими аналогами, також було розглянуто покрокове налаштування кожного з етапів розробки та налаштування проекту.

Також були розглянуті мови програмування які могли використовуватись при написанні програмного коду проекту, відмінність роботи з та графікою та розглянуто порядок етапів і алгоритмів для розробки фінальної цілі дипломного проекту – створення власної 2D гри.

РОЗДІЛ 3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ

В рамках даної дипломної роботи необхідно розробити систему взаємодії між логічною та візуальною частинами. Для реалізації візуальної частини використовується ігровий движок Unity3D, виділимо основні функції, за які буде відповідати ігровий движок. В першу чергу Unity буде відображати всі елементи і сцени в середині гри.

Також за допомогою високорівневої мови програмування C# будуть задаватися основні правила взаємодії ігрових елементів в середині сцени, їх властивості і характеристики, ці властивості не будуть змінюватися в рамках гри. Ігровий сценарій має зберігати в собі ігрову логіку, правила взаємодії гравця з ігровими предметами, властивості ігрових сцен та предметів в середині них.

3.1 Підготовка і розміщення графічних елементів на ігровій сцені та створення ігрових перешкод

Перед початком розробки ігрового додатку не обхідно чітко визначити цілі і можливості які будуть реалізовуватися протягом всього процесу розробки. Також заздалегідь потрібно підготувати базові графічні елементи для подальшої можливості використання та взаємодії їх на ігровій сцені.

Для реалізації ігрового додатку не обхідно спочатку створити ігрову сцену, на якій буде проходити основний сюжет гри. Ключовими елементами гри буде можливість вибору одного із п’яти доступних рівнів, ігровий персонаж та інтерактивні об’єкти з якими гравець може вести взаємодію. Прототип вибору рівнів представлений на рис.13.

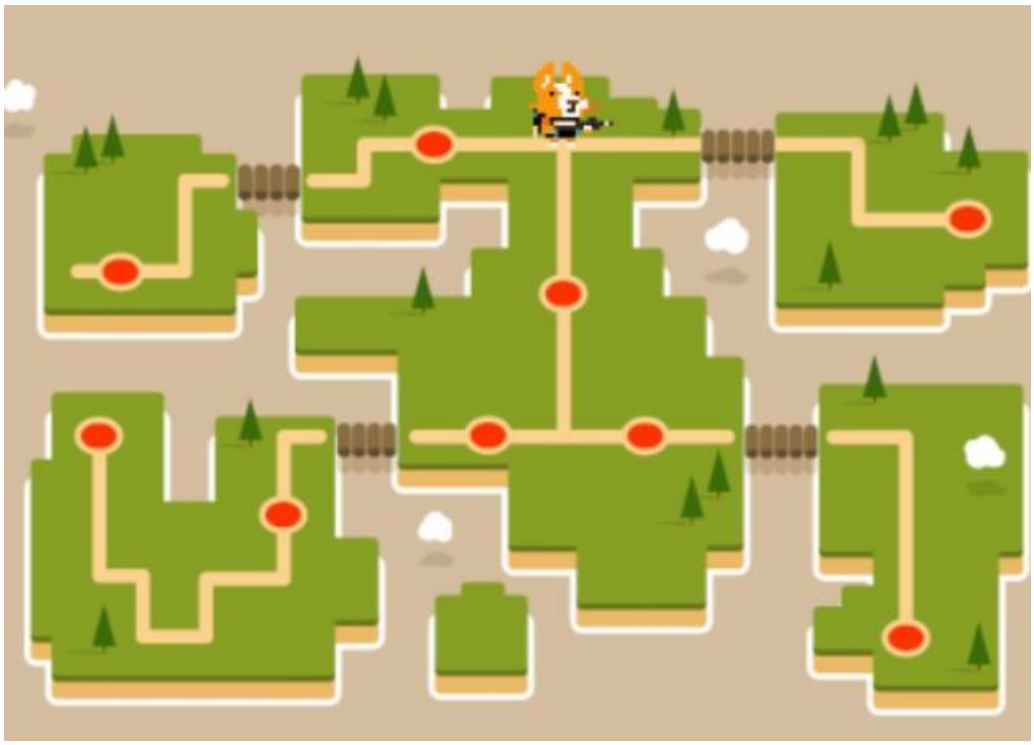


Рис. 3.1 Меню вибору доступних рівнів

Взаємодія з меню вибору рівня має дві властивості:

1) Меню знаходиться в режимі очікування, тобто користувач в даний момент не взаємодії з ігровою сценою, або знаходиться на стадії вибору одного з можливих рівнів гри.

2) Меню використовується, тобто гравець вибрав рівень який він буде використовувати і анімований персонаж створений за допомогою 2.5D графіки переходить на вибраний рівень, далі обхідно натиснути клавішу підтвердження вибору рівня (для ПК версії базова кнопка є – Enter). Далі вже безпосередньо проходить перехід на вибраний рівень гри.

Як видно на малюнку 3.2 представлений прототип ігрового рівня. На цьому рівні гравець може взаємодіяти з ігровим персонажем, навколишнім середовищем та при можливості з інтерактивними ігровими об’єктами.

Рис.3.2 Прототип ігрового рівня

Розглянемо з яких графічних елементів складається ігровий рівень.

1) Після того як пройшов запуск гри перший об’єкт який ви зустрічає на ігровій сцені знаходиться NPC (No Player Control) який дає вам базові настанови і цілі для проходження рівня.

Рис 15. Перший об’єкт взаємодії NPC.

Далі на ігровій сцені знаходяться платформи по яким може пересуватися ігровий персонаж, на деяких платформах розташовані інтерактивні об’єкти які впливають на ігровий процес, перший інтерактивний об’єкт – трамплін, за допомогою якого персонаж може змінювати положення висоти на якій він власне знаходиться.

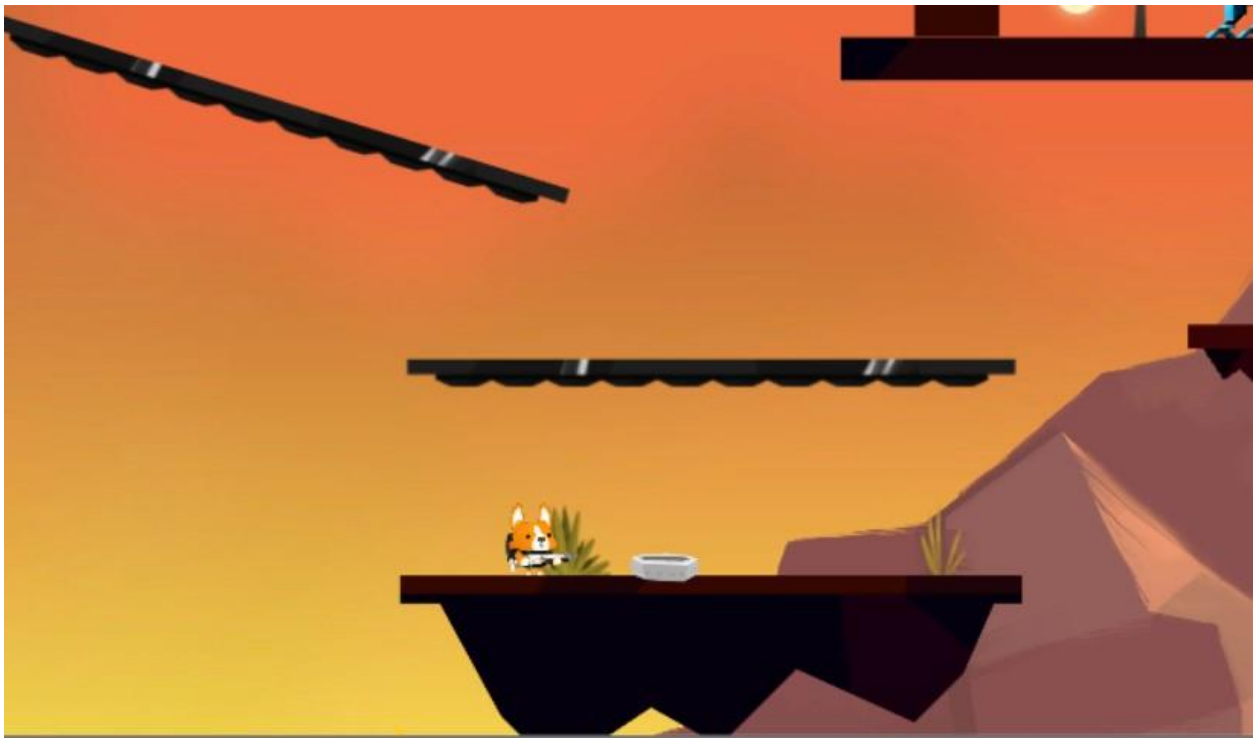


Рис. 3.3. Платформи та інтерактивний об’єкт трамплін.

На сцені розташовані статичні та динамічні елементи, які можуть завдати шкоду ігрового персонажу, при чому вони мають певну властивість, а саме їм не можливо завдати фізичну шкоду, тобто їх не обхідно оминути доступними способами, щоб уникнути шкоди або смерті ігрового персонажу.

Рис. 3.4 Динамічна перешкода гармата (з ліва) та статична перешкода йорш (з права)

Механіка перешкоди гармата реалізована таким чином, що персонаж отримує пошкодження поступово, і в нього пропорційно зменшується кількість одиниць здоров’я, тобто від цієї перешкоди не можливо отримати смертельні пошкодження одразу. На відміну від перешкоди гармата, перешкода йорш наносить значні пошкодження, які миттєво знімають всі одиниці здоров’я і це означає те, що ви програли і рівень буде перезапущено. За цей ігровий момент відповідає скрипт перезапуску рівня.

```
[AddComponentMenu("Corgi Engine/Spawn/Auto Respawn")]
public class AutoRespawn : MonoBehaviour, Respawnable
{
    [Header("Respawn when the player respawns")]

    public bool RespawnOnPlayerRespawn = true;

    [Header("Auto respawn after X seconds")]

    public float AutoRespawnDuration = 0f;

    public GameObject RespawnEffect;

    public AudioClip RespawnSfx;

    protected MonoBehaviour[] _otherComponents;
    protected Collider2D _collider2D;
    protected Renderer _renderer;

    protected bool _reviving = false;
    protected float _timeOfDeath = 0f;
    protected bool _firstRespawn = true;


    protected virtual void Start()
    {
        _otherComponents = GetComponents<MonoBehaviour>() ;
        _collider2D = GetComponent<Collider2D> ();
        _renderer = GetComponent<Renderer> ();
    }
```

Властивістю цього скрипта є автоматичний перезапуск рівня через певний інтервал часу (декілька секунд) і він дозволяє почати проходження рівня в з початку або з останнього збереженого місця, в залежності від положення персонажу на ігровій сцені.

На ігровій сцені є декілька типів перешкод, які впливають на ігровий процес в цілому, деякі з них можна знищити(рис. 18), деякі можна оминути (рис.19), а для проходження деяких перешкод краще знайти можливість обійти їх (рис.20).

Рис.3.5. Знищуванні перешкоди

Рис. 3.6 Перешкода яку можна уникнути

Рис.3.7 Перешкода, яку варто оминути.

Також в грі реалізовані допоміжні елементи (аптечка) та елементи які можуть накласти на персонажа негативні властивості (поле уповільнення часу).

Рис. 3.8 Аптечка

Рис.3.9. Поле уповільнення часу

Після успішного проходження рівня (рис.3.10), гравець автоматично буде направлений в меню вибору проходження рівня(зображено на рис. 3.1), після чого у нього є можливість вибрати інший рівень для проходження.

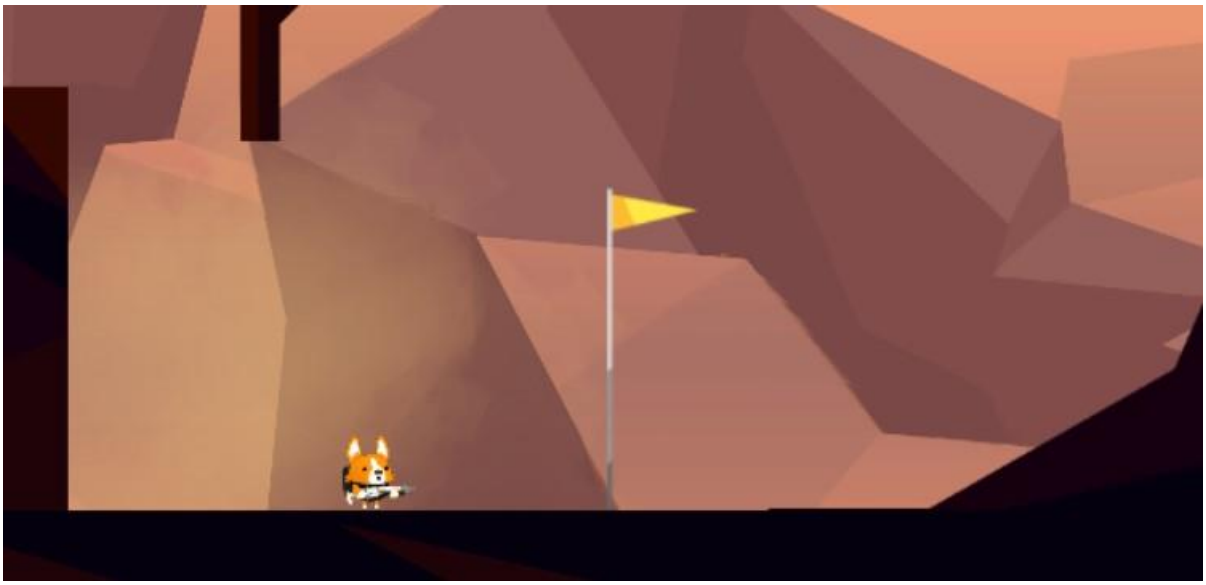


Рис. 3.10. Завершення рівня.

3.2 Інтерактивні ігрові об'єкти

Перед початком опису даного розділу, розглянемо поняття інтерактивного ігрового об’єкту.

Інтерактивний об’єкт – це об’єкт, в результаті взаємодії з яким, персонаж має можливість виконувати певний алгоритм дій, які прописані механікою програмного продукту, та дозволяють взаємодіяти з ігровими об’єктами для отримання результату.

В розробленому програмному продукту(відеогрі) реалізовано декілька інтерактивних ігрових об’єктів. Перший з них зображений на рис.16, властивістю даного інтерактивного об’єкту є переміщення персонажу(або його союзників) на певний проміжок вгору миттєво (по осі координат Y), на базову висоту яка заздалегідь прописана в програмному коді.

Наступний інтерактивний ігровий об’єкт – драбина, вона має такі ж значення як попередній об’єкт, але на відміну від нього, на взаємодію з цим об’єктом відходить певний проміжок часу (залежить від самого гравця, скільки він вважає потрібним знаходиться на даному об’єкті), зображено на рисунку 3.11..

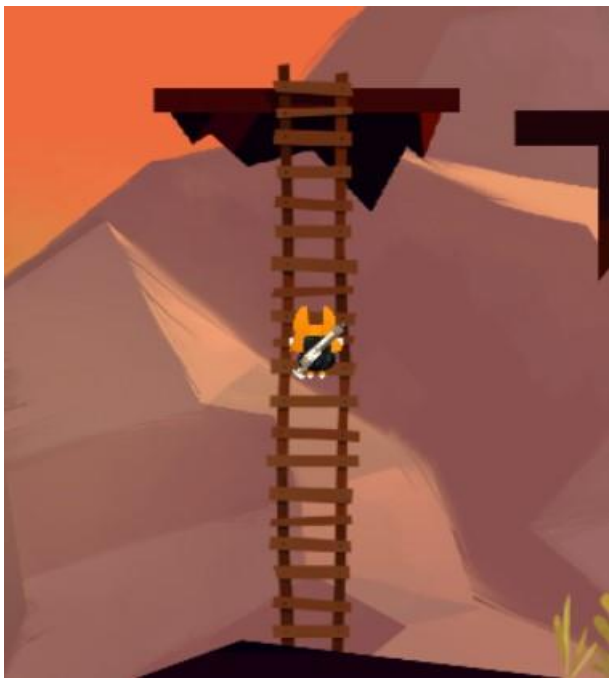


Рис. 3.11. Інтерактивний об’єкт драбина

Останній з інтерактивних об’єктів які задіяні на даному рівні є платформа підйому, її властивістю є доставка персонажу за певний проміжок часу в конкретну точку відносно осей координат X та Y, тобто платформа може переміщуватися як вертикально так і горизонтально, переміщувати платформу по діагоналі немає ніякої необхідності на даному рівні так, як цей рівень 2D платформер, і в ньому задіяні лише 2 осі координат. На рисунку 3.12 зображено платформу підйому.

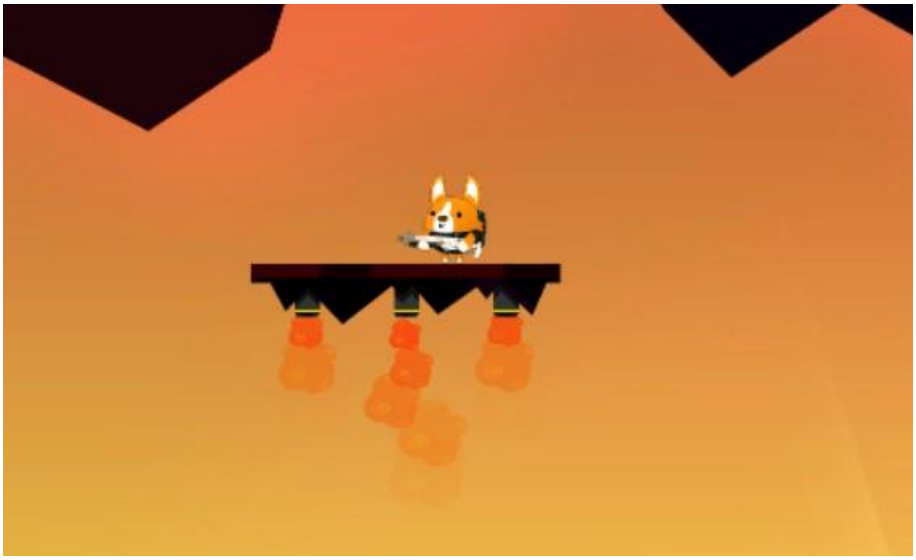


Рис. 3.12. Платформа підйому

3.3 Створення ігрового персонажу та його моделі поведінки

Ігровий персонаж – це об’єкт з яким в першу чергу буде взаємодіяти гравець та використовувати його властивості. І для більш зручного використання користувачем потрібно продумати можливості персонажа, які спростять проходження гри користувачеві, тобто продумати можливості руху персонажу, додаткові механіки та об’єкти які дають певні властивості персонажу. В розробленій відеогрі реалізовано декілька цікавих механік для персонажу, але розглянемо на що здатен персонаж поступово:

1) Переміщення ігрового персонажу – персонаж може змінювати свою

траєкторію відносно координат X та Y, тобто персонаж може пересуватися вертикально та горизонтально. Можливості ходьби персонажу представлені в скрипті нижче.

```
public float MovementSpeed { get; set; }
[Header("Speed")]

public float WalkSpeed = 6f;

[Header("Effects")]

public ParticleSystem TouchTheGroundEffect;

public AudioClip TouchTheGroundSfx;

protected float _horizontalMovement;
protected float _normalizedHorizontalSpeed;

protected override void Initialization()
{
    base.Initialization ();
    MovementSpeed = WalkSpeed;
}

public override void ProcessAbility()
{
    base.ProcessAbility();
    HandleHorizontalMovement();
}

protected override void HandleInput()
{
    _horizontalMovement = _inputManager.PrimaryMovement.x;
}
tive : will move to the right, negative : will move left </param>
public virtual void SetHorizontalMove(float value)
{
    _horizontalMovement=value;
}
```

Цей скрипт відповідає за горизонтальне переміщення персонажу, та взаємодії його з навколишнім середовищем, тобто переміщення по різним об’єктам і приземлення або підйом після виконання певного алгоритму дій.

2) Біг – у персонажу реалізована можливість швидкого переміщення по ігровій зоні, це дозволяє швидко уникати небезпечних моментів, без ризику втрати одиниці здоров’я. Функціональний скрипт представлений нижче.

```
[Header("Dash")]
public float DashDistance = 3f;
public float DashForce = 40f;
public float DashCooldown = 1f;
protected float _cooldownTimeStamp = 0;
protected CharacterHorizontalMovement _characterHorizontalMovement;
protected float _startTime ;
protected Vector3 _initialPosition ;
protected float _dashDirection;
protected float _distanceTraveled = 0;
protected bool _shouldKeepDashing = true;
protected float _computedDashForce;
protected override void Initialization()
{
    base.Initialization();
    _characterHorizontalMovement = GetComponent<CharacterHorizontalMovement>();
}
protected override void HandleInput()
{
    if (_inputManager.DashButton.State.CurrentState == MMInput.ButtonStates.ButtonDown)
    {
        StartDash();
    }
}
public override void ProcessAbility()
{
    base.ProcessAbility();
    // If the character is dashing, we cancel the gravity
    if (_movement.CurrentState == CharacterStates.MovementStates.Dashing)
    {
        _controller.GravityActive(false);
        _controller.SetVerticalForce(0);
    }
}
```

3) Стрибок – у персонажа є три варіативності використання стрибків:

➢ Звичайний стрибок (рис. 3.13) – дозволяє персонажу застрибувати на платформи які знаходяться на малій висоті.

Рис. 3.13 Звичайний стрибок

➢ Подвійний стрибок – дозволяє персонажу покорювати більш високі об’єкти для подальшого проходження рівню. (не можливо відобразити и вигляді зображення)

➢ Стрибок від платформи – дає можливість персонажу відштовхуватися від платформи, з цілю пересування угору по спеціальним блокам. (рис. 3.14)

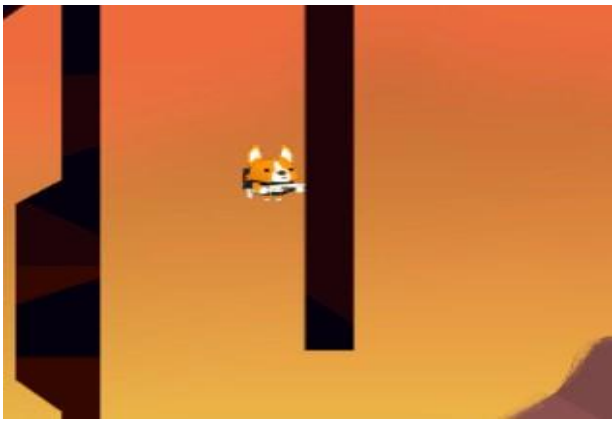


Рис. 3.14. Стрибок від платформи.

Реалізація в програмному коді зображена нижче:

```
public class CharacterJump : CharacterAbility
{
    public override string HelpBoxText() { return "This component handles jumps.
    public enum JumpBehavior
    {
        CanJumpOnGround,
        CanJumpAnywhere,
        CantJump,
        CanJumpAnywhereAnyNumberOfTimes
    }

    [Header("Jump Behaviour")]
    public float JumpHeight = 3.025f;
    public int NumberOfJumps=3;
    public JumpBehavior JumpRestrictions;
    public float JumpTimeWindow = 0f;
    [Header("Proportional jumps")]
    public bool JumpIsProportionalToThePressTime=true;
    public float JumpMinimumAirTime = 0.1f;
```

4) Стрільба – персонаж оснащений зброєю, не обхідним елементом для ведення вогню по супротивникам з метою їх знищення та успішного завершення рівня (рис. 3.15).

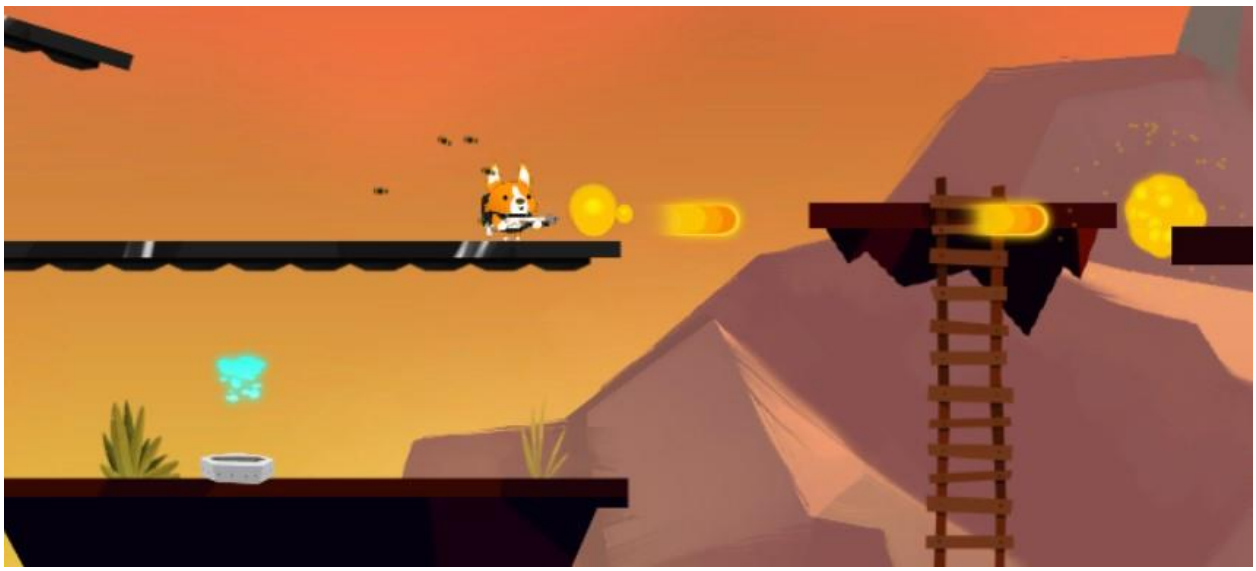


Рис.3.15. Ведення вогню.

Базовий скрипт використання зброї:

```
public virtual void Setup()
{
    if (WeaponAttachment==null)
    {
        WeaponAttachment=transform;
    }
    if (_animator != null)
    {
        _weaponIK = _animator.GetComponent<WeaponIK> ();
    }
    if (InitialWeapon != null)
    {
        ChangeWeapon(InitialWeapon);
    }
    _character = gameObject.GetComponentNoAlloc<Character> ();
}
public override void ProcessAbility()
{
    base.ProcessAbility ();
    UpdateAmmoDisplay ();
}
protected override void HandleInput ()
{

    if (_inputManager.ShootButton.State.CurrentState == MMInput.ButtonStates.ButtonDown)
    {
        ShootStart();
    }

    if (_inputManager.ShootButton.State.CurrentState == MMInput.ButtonStates.ButtonUp)
    {
        ShootStop();
    }
```

5) Зависання на поверхності – для уникнення перешкод, або ухилення від від динамічних ворогів (рис. 17), реалізована механіка зависання на короткий інтервал часу на доступних поверхностях (рис.3.16).

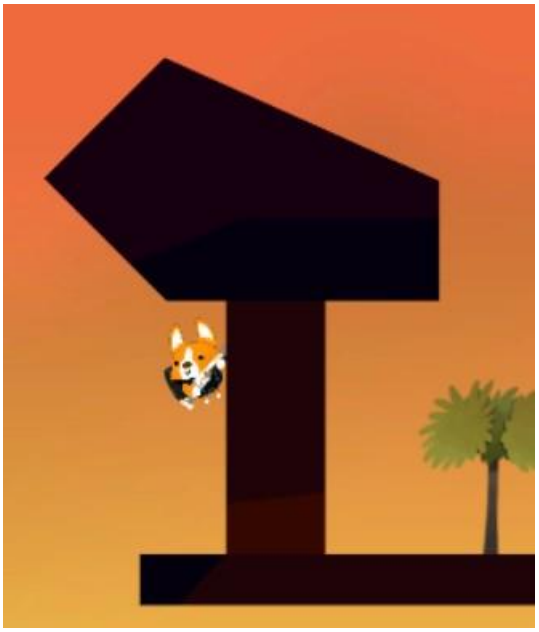


Рис.3.16. Зависання на об’єктах

В програмному коді воно реалізовано наступним чином:

```
public float WallClingingSlowFactor=0.6f;
public float WallClingingTolerance = 0.3f;
protected override void HandleInput()
{
    if ( (_inputManager.PrimaryMovement.x <= -_inputManager.Threshold.x) || (_inputManager.PrimaryMovement.x >= _inputManager.Threshold.x) )
    {
        WallClinging();
    }
}
public override void ProcessAbility()
{
    base.ProcessAbility();
    ExitWallClinging();
    WallClingingLastFrame ();
}
protected virtual void WallClinging()
{
    if (!AbilityPermitted
        || (_condition.CurrentState != CharacterStates.CharacterConditions.Normal)
        || (_controller.State.IsGrounded)
        || (_controller.Speed.y >= 0) )
    {
        return;
    }
```

6) Використання реактивного ранцю – ця додаткова функція дозволяє використовувати на короткий реактивний ранець, який по базовому функціоналу дозволяє оминути перешкоди, або піднятись на потрібну висоту якщо не можливо потрапити туди використовуючи інтерактивні об’єкті, або властивості стрибків

персонажу. Час використання обмежено і відновлення шкали заряду використання займає певний проміжок часу, який заданий константою (рис.3.17).

Рис.3.17. Використання реактивного ранцю.

Програмно реалізація даної можливості виглядає наступним чином:

```
protected bool _stillFuelLeft = true;
protected bool _refueling = false;
protected bool _jetpacking = true;
protected Vector3 _initialPosition;
protected AudioSource _jetpackUsedSound;
protected WaitForSeconds _jetpackRefuelCooldownWFS;
protected override void Initialization ()
{
    base.Initialization();
    if (ParticleEmitter!=null)
    {
        _initialPosition = ParticleEmitter.transform.localPosition;
        ParticleSystem.EmissionModule emissionModule = ParticleEmitter.emission;
        emissionModule.enabled=false;
    }
    JetpackFuelDurationLeft = JetpackFuelDuration;
    _jetpackRefuelCooldownWFS = new WaitForSeconds (JetpackRefuelCooldown);

    if (GUIManager.Instance!= null && _character.CharacterType == Character.CharacterTypes.Player)
    {
        GUIManager.Instance.SetJetpackBar(!JetpackUnlimited, _character.PlayerID);
        UpdateJetpackBar();
    }
}
```

В процесі розробки знаходяться ще декілька можливих для використання функцій, але вони знаходяться на стадії тестування та налаштування і поки не можуть використовуватися як основні елементи:

- Переміщення по канату – дозволяє гравцю переміщувати персонажа по спеціальним виділеним місцям в локації, щоб уникнути перешкод які поступово знімають у персонажа одиниці здоров’я, або і зовсім можуть його вбити.
- Плавання – дає можливість персонажу пересуватися по водному

простору на певних рівнях гри без втрати одиниць здоров’я, але швидкість руху персонажу значно зменшується, на даному етапі гри відсутня можливість отримати шкоду від навколишнього середовища, але є ймовірність отримати шкоду від об’єктів класу ворог.

- Телепортація – миттєве переміщення в просторі з однієї точки в іншу, в випадку необхідності цього моменту в сюжеті гри.

3.3.1 Створення класу ворог та моделі його поведінки

Клас ворог реалізований схожим чином, як клас персонаж (підконтрольний реальному гравцеві), але з деяким відмінностями. Ворог має дещо зменшений функціонал дій, він не може використовувати аптечки, немає вбудованого реактивного ранцю, має обмежену область пострілів, не може використовувати всі види стрибків на відміну від ігрового персонажу, і головною властивістю цього класу є прив’язка до певної зони на локації, яку даний персонаж залишити не в змозі, на даний момент в грі реалізовано два види даного класу, але з розширенням і оновлення гри буде додаватись, функціонал даного класу, і різноманітні види для більш цікавого і довго проходження гри. Нижче представлені зображення двох класів ворогів.

Рис. 3.18. Клас ворога Стрілок (Archer)

Даний клас характеризується найбільшою (на даному етапі готовності продукту) дальністю пострілу, але має низький рівень пошкодження по персонажу гравця.

Рис. 3.19. Клас ворога бульбашка (Bubble)

Цей клас ворога має харектерну відмінність від попереднього класу, у нього малий радіус атаки і він дуже вразливий до атак гравця, але наносить середній рівень пошкоджень, якщо гравець зіштовхнеться з ним.

Два класа ворогів мають базовий набір алгоритмів який керує їх поведінкою, в залежності від радіуса положення персонажу яким керує гравець персонаж виконує свої дії, але якщо підконтрольний персонаж потрапляє в радіус провокації, ворог починає його атакувати.

3.4 Аналіз отриманих результатів

Вході розробки ігрового додатку, було розроблено чотири рівні гри порядок проходження яких гравець може вибирати сам, і один рівень який розблоковується після того як гравець набере не обхідну кількість очок, що дозволяє перейти на бонусний рівень. На зображеннях нижче показані зовнішні вигляди розроблених рівнів.

Рис. 3.20. Рівень 1

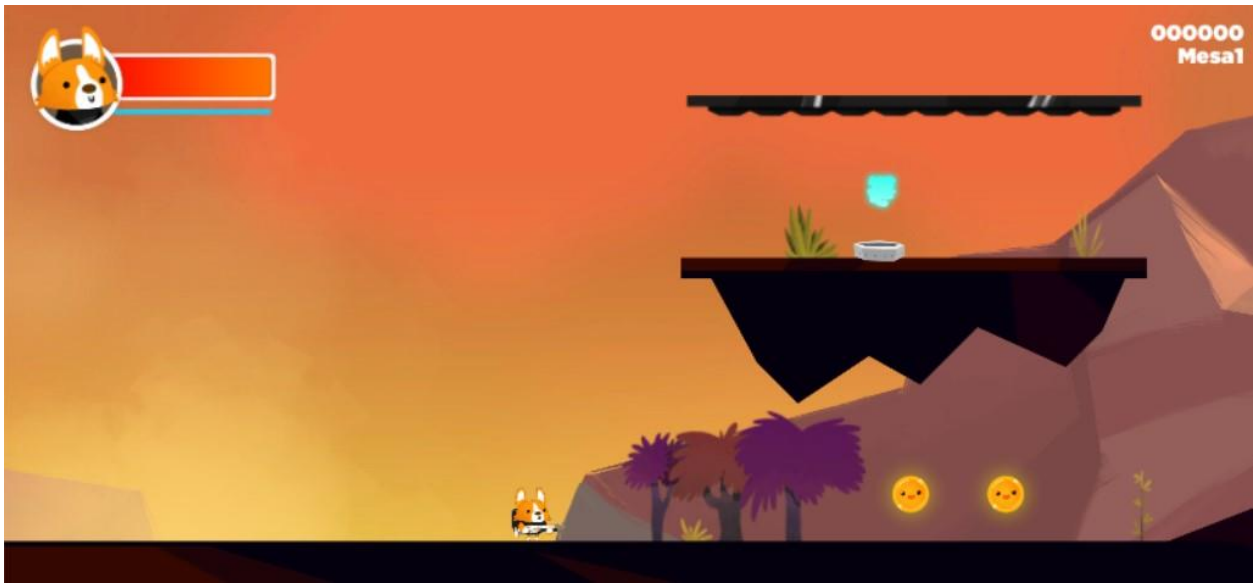


Рис.3.21. Рівень 2

Рис. 3.22. Рівень 3

Рис. 3.23. Рівень 4

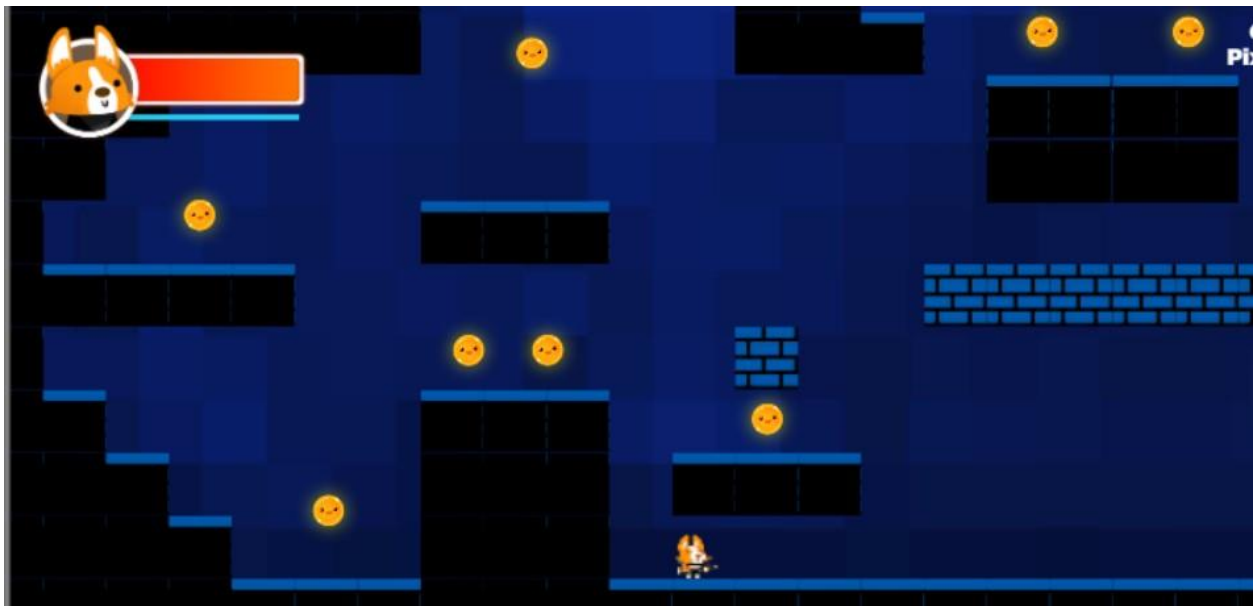


Рис. 3.24. Бонусний рівень

Кожен з чотирьох рівнів, окрім бонусного, мають однаковий майже однаковий сюжет але з різними рівнями складності, що поступового підготовлює гравця до більш складного проходження наступних рівнів.

Бонусний рівень розблоковується по досягненню певної кількості ігрових очок, на цьому рівні доступний збір коінів (монет), але на це відведений певний проміжок часу, і при завершенні відрахунку часу, гравець має достигнути фінішу, в іншому випадку результат бонусного рівня буде анальовано.

В грі реалізована велика кількість механік, які додають різноманітні можливості взаємодії з навколишнім середовищем, всі розроблені механіки гри детально розписані в розділі 3.3.

В подальших планах на вдосконалення програмного продукту є декілька варіантів:

1) Доробити два рівні, які знаходяться на даному етапі проекту, в в розробці та відлажуванні.

2) Реалізувати розроблені механіки, з можливістю інтеграції їх, уже в готові і функціональні рівні.

3) Реалізувати магазин для можливості витрачати зібрані монети, на кастамізацію та покращення характеристик персонажу.

Висновок до розділу 3

В ході написання даного розділу дипломного проекту, було розроблений програмний продукт (відеогру), а саме графічну і функціональну частину (програмну), описані функціональні можливості керування персонажем.

Детально розглянуто можливості одного з доступних рівнів гри, та опис працездатності і функціоналу взаємодії ігрової сцени з реальною людиною (гравцем). Розписані плани на подальші розробки і покращення ігрового додатку, з рошришенням його можливостей та функціоналу.

ВИСНОВКИ

В ході написання першого розділу було розглянуто функціональні можливості різноманітних операційних систем для персональних комп’ютерів та мобільних пристроїв. Було проведено аналіз, на основні якого було прийняте рішення розробляти програмний продукт на операційну систему Android.

В процесі написання другого розділу дипломної роботи були проаналізовані можливості різноманітного програмного забезпечення для розробки розділених завдань: створення графічних елементів, створення анімації, створення звуку, і написання програмного коду. Також були наведені порівняльні характеристики з існуючими аналогами програмного забезпечення. І було розглянуте по етапне налаштування проекту.

В ході розробки третього розділу дипломного проекту, була розроблена відеогра (графічна та програмна частина), і також було проведено опис функціональних можливостей розробленого програмного продукту. Також були наведені плани на подальші плани розробки проекту.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Unity Manual [Електронний ресурс] / режим доступу: https://docs.unity3d.com/Manual/UsingTheEditor.html

2. Microsoft documentation Справочник по языку C# [Електронний ресурс] / режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/introduction

3. Веб технології для розробників JavaScript [Електронний ресурс] / режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript

4. Wikipedia мова програмування Lua [Електронний ресурс] / режим доступу: https://uk.wikipedia.org/wiki/Lua

5. IronPython [Електронний ресурс] / режим доступу: https://wiki.python.org/moin/IronPython

6. Обзор языка программирования Rust [Електронний ресурс] / режим доступу: https://habr.com/ru/post/135712/

7. Си язык программирования [Електронний ресурс] / режим доступу: https://wiki2.org/ru/Си_(язык_программирования)