МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка
до дипломної роботи
_____бакалавр_____
(освітньо-кваліфікаційний рівень)

на тему «Розробка інтерактивної гри на базі платформи Unity»

Виконав: студент 4 курсу, групи ІПЗ-17д
напряму підготовки 121 „Інженерія програмного забезпечення ”
_____ Жалдаков Д.А.
(підпис)

Керівник,
доцент, к.т.н._____ Іванов В.Г.
(підпис)

Рецензент,
ст. викл. ПМ _____ Батурін О.І.
(підпис)

СЄВЄРОДОНЕЦЬК – 2021

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ
бакалаврської роботи студента гр. ІПЗ-17д Жалдакова Д.А.

Науковий керівник
доцент, к.т.н._____ Іванова В.Г.
ПІБ, посада

Оцінка наукового керівника: _____

Рецензент Батурін О.І ст. викл . каф. ПМ СНУ ім. В. Даля
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту: _____

Голова ЕК,
проф. каф. _____ Захожай О.І.
підпис

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки

Кафедра програмування та математики

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 121 „Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ

Завідувач кафедри ПМ,

д.т.н., доцент

_____ Лифар В.О.

«___» _____ 2021 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Жалдакову Дмитру Андрійовичу

1. Тема роботи Розробка інтерактивної гри на базі платформи Unity керівник роботи к.т.н., доцент Іванов Віталій Григорович

затверджені наказом вищого навчального закладу від «23» квітня 2021 року № 68/14.04

2. Строк подання студентом роботи 15 червня 2021 р.

3. Вихідні дані до роботи

Об'єктом досліджень є клієнт-серверні технології на основі веб-додатку навчальної платформи.

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

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:

Поняття ігрової платформи.

Аналіз існуючих ігрових платформ.

Вимоги до ігрової платформ.

4.3 Основна частина, в якої висвітлити:

Визначення ігрового двигуна.

Архітектура написання скрипта.

Вимоги до написання коду.

Аналіз технологій реалізації програмної частини.

Розробка програмнох частини гри.

4.4 Висновки

4.4 Перелік використаних джерел

5. Перелік графічного матеріалу є

6. Дата видачі завдання 23 квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	23.04.21	
2	Укладання і погодження з керівником плану і етапів виконання роботи	23.04.21	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	26.04.21	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	02.05.21	
5	Проектування інфологічної моделі задачі що реалізується.	09.05.21	
6	Укладання програмного продукту	11.05.21	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	25.05.21	
8	Здача готової пояснювальної записки на кафедру	15.06.21	
9	Укладання доповіді і презентації	16.06.21	

Студент _____ Жалдаков Д.А.
(підпис)

Керівник роботи _____ Іванов В.Г.
(підпис)

РЕФЕРАТ

Робота містить: 51 сторінок основного тексту, 8 сторінок додатків, 23 рисунка, 11 використаних джерел. Метою випускної кваліфікаційної роботи є вивчення особливостей роботи з двигуном Unity, методів їх реалізації, оптимальних рішень, в плані програмування і підбору продуктивних інструментів. Було проаналізовано багато існуючих ігор, їх принцип роботи, перелік наданих можливостей і кращі рішення. Багато часу було приділено технічній частини. В результаті виконаної роботи, було реалізовано інтерактивну гру, що дозволяє побачити сучасну розробку ігор. Інтерактивна гра може бути легко масштабованою для подальшої розробки даного проекту. Система реалізована відповідно всім вимогам технічного завдання. Зроблено детальний опис процесу розробки системи, а також, продемонстрована робота готової гри.

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних.

СКБД – система керування базою даних.

C# – мова програмування від компанії Microsoft.

Unity – сучасний ігровий двигун.

XML – Extensible Markup Language, розширювана мова розмітки.

API – Application Programming Interface, інтерфейс прикладного програмування.

HTTP – HyperText Transfer Protocol, протокол передачі HyperText.

IP – Internet Protocol, міжмережевий протокол

Зміст

Вступ

Актуальність - У наш час майже у кожного другої людини є смартфон, вони стали невід'ємною частиною нашого життя. Раніше не кожна людина, у якого середньостатистична зарплата, міг дозволити собі придбати смартфон, але тепер ситуація кардинально змінилася. Діти, починаючи з малих років, починають знайомитися з портативними пристроями, в тому числі смартфонами, у багатьох з самого раннього дитинства з'являються телефони особистого користування. Люди більш старшого покоління почали звикати до життя зі смартфонами.

З плином часу, все більше людей купують собі новітні смартфони для особистого користування, тому що компанії, що виготовляють телефони, з кожним роком намагаються зробити товар більш доступним в ціновому сегменті і більш потужним, порівняно з попередніми моделями.

Раніше телефон використовувався, виключно, як засіб зв'язку, а не розваги і навчання, але тепер все навпаки. З кожним роком стали з'являтися все більше і більше додатків, та ігор вивчення історії. Саме тому розробка ігор для мобільного телефону в наш час є добре оплачуваною роботою.

Актуальність розробки ігор для смартфонів ОС Android підтверджується статистикою. В кінці 2017 року розподіл на ринку мобільних ОС частка Android пристроїв становила 87,7% ринку, а IOS - 12,1%.

Завдяки широкій лінійці продукції від світових брендів Samsung, Acer, HTC і багатьох інших в різних цінових категоріях, що сприяє поширенню смартфонів на базі Android серед усіх верств населення - від студентів до керівників великих компаній. Так як Android смартфони домінують на світовому ринку, кількість користувачів цієї системи буде тільки збільшуватися з кожним новим роком.

2д платформер - це жанр ігор, які вимагають від гравця задіяти його інтуїцію, стратегію і логіку. Також платформери дуже часто зустрічаются у смартфонах на операційних системах андроїд та ios. Тому актуальність платформерів та цих системах дуже велика.

Мета - метою даної роботи є розробка гри в жанрі «2д платформер» на ОС Android.

Об’єкт досліджень: Пізнання психологічного стану людини через гру.

Предмет досліджень: Цифрове рішення яке поєднує в собі інтерактивну гру та платформер.

Розділ 1 АНАЛІТИЧНА ЧАСТИНА

1.1 Класифікація жанру гри платформер

Ігри на платформі (часто спрощені як ігри на платформі або стрибки) - це жанр відеоігор та піджанр екшн-ігор, де основною метою є переміщення персонажа гравця між очками в подачі. Платформи характеризуються ступенем їх використання у стрибках та підйомі для керування оточенням гравця та досягнення мети. Рівні та середовища мають нерівну місцевість та підвісні платформи різної висоти, що вимагає використання навичок гравця для їх подолання. Інші акробатичні рухи також включені в ігровий процес, такі як підйом, ловлячись від таких речей, як вусики або гачки, стрибки зі стін, подряпини повітря, настройка в повітрі, стрільба зі зброї або сходження з водолазних дощок або батутів. Ігри, що мають повністю автоматичні стрибки, такі як 3D-ігри в серії The Legend of Zelda, не виходять за межі жанру[1].

Незважаючи на те, що це часто асоціюється з іграми-маркерами, для відеоаркадних ігор було випущено багато відомих платформних ігор, а також портативних ігрових консолей та домашніх комп’ютерів.

У той час, коли ігри на платформі були дуже популярними, ігри на платформах, за оцінками, становили від чверті до третини ігор символів, але з того часу мисливцями стали перші люди. У 2006 році вид став все більш популярним, що становило 2% частки ринку порівняно з 15% у 1998 році. Однак жанр все ще існує в комерційній обстановці, кілька ігор продаються мільйонами.

1.2 Тривимерні платформери

Термін «тривимірний платформер» зазвичай відноситься до ігор, які пропонують тривимірний ігровий процес і багато користувачів тривимірну графіку. Ігри з 3D-відтворенням тільки 2D-графіки зазвичай включаються в ізометричний платформер, в той час як ігри з 3D-графікою - це 2.5D, але грають в 2D-площині, оскільки вони являють собою комбінацію 2D і 3D.

Конго Бонго від Sega був одним з перших платформер з 3D-графікою в 1983 році. Перший платформер з 3D-видом і мобільного камерою з'явився на початку 80-х. Раннім прикладом цього може бути платформер Konami Antarctic Adventure, в якому гравець бере на себе управління пінгвіном в поточній сцені від третьої особи, перестрибуючи через ями і перешкоди. Спочатку випущений для ПК MSX в 1983 році, він був перенесений на різні платформи в наступному році, включаючи версію Arcade, NES і ColecoVision.

У 1986 році було випущено сиквел платформної гри Antarctic Adventure під назвою Penguin Adventure, розробленої Хідео Кодзіма. Він включав в себе більше ігрових елементів, більшу різноманітність рівнів, елементи рольової гри, такі як апгрейди устаткування, і розрахований на багато користувачів режим.

На початку 1987 року Square випустила WorldRunner 3-D, розроблений Хіронобу Сакагуті і Насир Гебеллі. Використовує ефект прокручування вперед, аналогічний ефекту рейкового шутера від третьої особи Space Harrier від Sega 1985 року. 3-D WorldRunner - це рання гра-платформер з псевдо-тривимірної прокруткою від третьої особи, яка дозволяла гравцям вільно переміщатися в напрямку руху вперед, стрибати і переслідувати перешкоди. Вона була відома як одна з перших стереоскопічних 3D-ігор. Пізніше в тому ж році Square випустила свою публікацію YY.

Alpha Waves (1990) була ранньої тривимірної ігровою платформою.

Самим раннім прикладом справжнього тривимірного платформера є французька комп'ютерна гра Alpha Waves, розроблена Крістофом де Дінешеном і опублікована Infogrames в 1990 році для Atari ST, Amiga і ПК.

Помилка! (1995) розширив традиційний платформер в усіх напрямках.

Bug!, Гра для Sega Saturn 1995 року, пропонувала більш консервативний підхід до реальних 3D-платформам. Це дозволяло гравцям рухатися у всіх напрямках, але не дозволяло переміщатися більш ніж по одній осі за раз; гравець міг рухатися ортогонально, але не по діагоналі. Її персонажі були пре-спрайтами, як раніше Зведений Лицар. Гра була дуже схожа на 2D-платформер, але вважалася справжньою 3D-грою і дозволяла гравцям підніматися по стінах і стелях. Є продовження під назвою Bug Too!.

У 1995 році компанія Delphine Software випустила 3D-продовження свого популярного 2D-платформер Flashback. Це була перша спроба перетворити популярну серію 2D-платформер в 3D з назвою «Fade to Black». При збереженні стилю дизайну на рівні головоломок і покрокового управління, вона не відповідала критеріям платформер і була включена як пригодницька гра. Він використовував справжніх тривимірних персонажів і декорації, але його оточення було визуализировано за допомогою жорсткого движка, схожого на Wolfenstein 3D, в якому він міг надати тільки квадратні горизонтальні коридори замість високих поверхів, між якими можна було стрибати.

Sony прийняла існуючий проект з розробниками Naughty Dog, який недавно випустив Hero Quiz. Гра Crash Bandicoot стала хітом нової консолі Nintendo для маркетингу в Північній Америці і була випущена якраз до японських свят. Crash залишався неофіційним талісманом Sony протягом наступних кількох років, перш ніж перейти до кросплатформним повідомленнями про токенах наступного покоління.

Sega попросила свою американську студію, Sega Technical Institute, принести Sonic the Hedgehog в 3D. Їх проект під назвою Sonic Xtreme повинен був використовувати зовсім інший підхід до серії, з вишуканою

камерою «риб'яче око» і розрахованої на багато користувачів грою, заснованої на Bug! нагадувати. Почасти через конфлікт з Sega Enterprises в Японії і поспішного графіка, гра так і не запустила її.

1.3 Аналіз існуючих ігор в жанрі платформер

Розглянемо найкращі ігри в жанрі платформер та проведемо аналіз існуючих рішеннь, використання цього жанру в ігровій індустріїх[2].

1.3.1 Лімбо

Limbo — це відеогра-головоломка, розроблена незалежною студією Playdead. Гра була випущена в Xbox Live Arcade в липні 2010 року і з тих пір була перенесена на кілька інших систем, включаючи PlayStation 3, Linux і Microsoft Windows. Limbo - це двомірний бічний скроллер, який містить фізичну систему, яка управляє оточенням гравця і персонажем. Гравець веде анонімного хлопчика через небезпечні середовища і пастки, шукаючи свою сестру. Розробник розробив головоломку гри, чекаючи, що гравець зазнає невдачі, перш ніж знайде правильне рішення. Playdead назвав стиль «Випробування і смерть» і використовував похмурі образи смерті хлопчика, щоб забрати гравця від непрацюючих рішень.

Гра представлена в чорно-білих тонах з використанням освітлення, ефектів зернистості плівки і мінімального зовнішнього шуму, щоб створити більш похмуре відчуття, часто пов'язане з жанром жахів. Журналісти високо оцінили похмуру презентацію і охарактеризували роботу як порівняння з фільмом нуар і німецьким виконанням. Через його краси рецензенти визначили лимбо як приклад відеоігор як виду мистецтва. "Limbo" отримав схвальні відгуки, але критики розійшлися з приводу його самого маленького сюжету; Деякі критики вважали, що відкрита робота має більш глибокий зміст, який добре пов'язаний з ігровою механікою, в той час як інші вважали, що їй не вистачало великого сюжету і раптового фіналу. Звичайна критика з боку оглядачів полягала в тому, що висока вартість гри в порівнянні з її невеликою довжиною може утримувати гравців від покупки гри, але деякі

огляди припускали, що Limbo чудова по довжині. Гра входить в число кращих ігор всіх часів.

Limbo була третьою за розміром продажем грою в аркадном сервісі Xbox Live в 2010 році, отримавши близько 7,5 мільйонів доларів доходу. Після публікації вона отримала кілька нагород від галузевих груп і була названа однією з кращих ігор 2010 року декількома публікаціями. Наступна гра від Playdead, Inside, була випущена в 2016 році і торкнулася багатьох з тих же тем, представлених в Limbo.

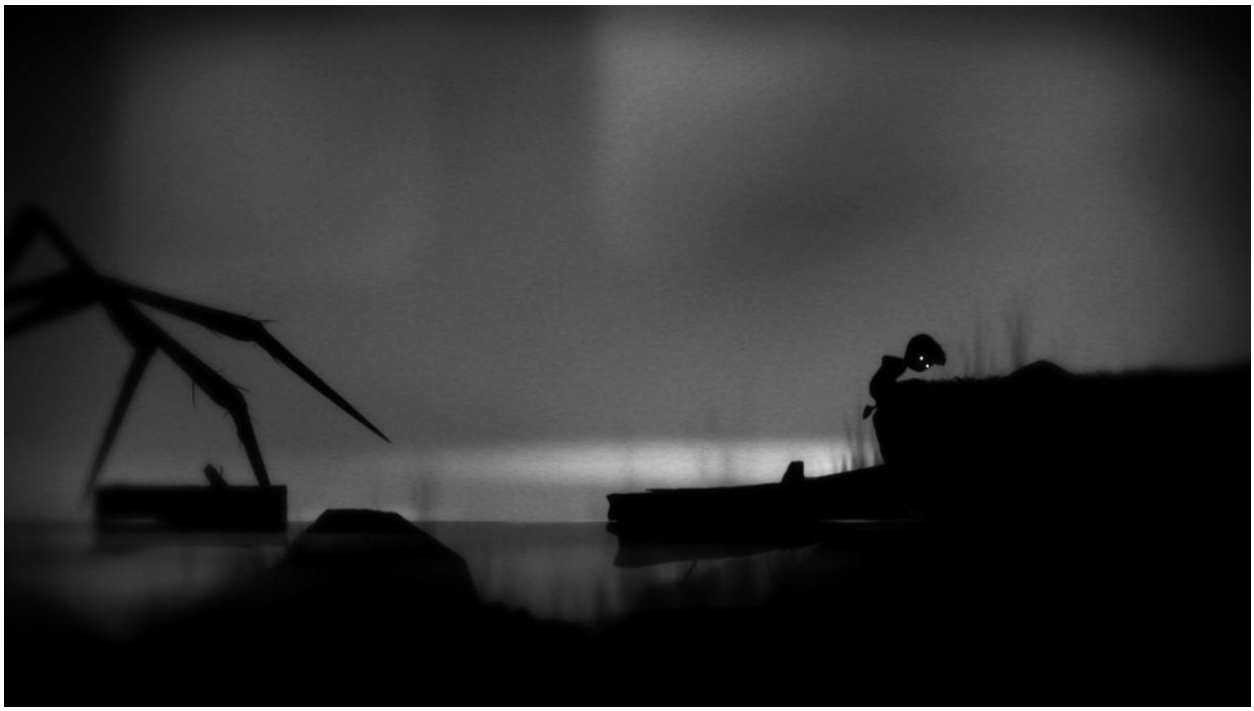

Рис. 1 – Гемплей гри Лібмо

Гравець керує хлопчиком під час гри. Як це часто буває в більшості платформерів з двостороннім рухом, хлопчик може бігати вліво або вправо, стрибати, дертися вгору і вниз по маленьких уступах, сходах і мотузках, а також штовхати або тягнути предмети. Лімбо представлений з використанням темної сіро-сірої графіки і мінімального навколишнього шуму, що створює дивну привабливе середовище. Зображення виглядають темними також приховують ряд смертельних чудес, включаючи екологічні та фізичні небезпеки, такі як смертоносні ведмежі пастки на лісовій підстилці або смертоносних монстрів. ховається в тіні. Небезпеки включають чисті

черви, які прикріплюються до голови хлопчика, змушуючи його блукати в одному напрямку, поки він не буде убитий.

У другій половині гри представлені механічні головоломки і головоломки-пастки з інструментами, електромагнітами і гравітацією. Багато з цих уловів не помітні до тих пір, поки вони не будуть стимульовані і часто вбивають дитинчат. Гравець може перезапустити гру на останній контрольній точці, з якою ви зіткнулися, без обмеження кількості випадків. Деяких пасток можна уникнути і використовувати пізніше в грі; Ведмежа пастка використовується для затиску деревного вугілля тварин, підвішується на кінці мотузки, відштовхує тушу від мотузки і підтягує гілку і мотузку вгору, дозволяючи хлопчикові підійматися по уступах поза досяжністю. Оскільки гравець може померти кілька разів, перш ніж вирішити кожну головоломку і завершити гру, розробники називають Limbo грою «Випробування і смерть». Деякі смерті анімовані зображеннями розпростертої голови або голови хлопчика, хоча на деяких платформах необов'язкові фільтри крові виходять за межі екрану замість того, щоб відображати ці смерті. Досягнення в грі (необов'язкові внутрішньоігрові мети) включають пошук захованих яєць комах і завершення гри п'ятьма або меншим числом смертей.

1.3.2 Super Meat Boy

Super Meat Boy - це платформер, в якому гравці беруть на себе управління маленького темно-червоного кубічного персонажа на ім'я Meat Boy, який хоче врятувати, його коханку в формі стрічки, Bandage Girl, яку забрав злий вченний Доктор. Плід. Гра розділена на глави, які разом складаються з більш ніж 300 рівнів. Гравці намагаються дійти до кінця кожного рівня, представленого дівчиною-пов'язкою, ухиляючись від кривих блоків, бачачи леза і різні інші смертельні перешкоди. Гравець може стрибати і бігати по платформах, а також стрибати або зісковзувати зі стін. Базова гра вимагає належного управління і часу в лічені секунди, і по

ігровому процесу і складності вона порівнянна з традиційними платформеними іграми, такими як Super Mario Bros. і Ghosts 'n Goblins.

Рівні в кожному розділі можна грати в будь-якому порядку, але вам потрібно пройти певну кількість рівнів, щоб дістатися до рівня боса, який відкриє наступну главу, коли вона буде завершена. У гравця є необмежена кількість спроб пройти кожен рівень. Якщо Meat Boy буде убитий, він негайно перезапустить рівень, навіть якщо декоративний сік червоного м'яса залишиться на поверхні, яку натер гравець. Повторювана дія, до якого можна отримати доступ після завершення рівня, показує одночасні спроби гравця завершити рівень. Завершення рівня протягом певного часу приносить рівень «A +», який вирішує складнішу версію рівня в «темному світі», додатковий набір рівнів особливо важкий. На приховані рівні, так звані ближні зони, можна потрапити, знайшовши гавані на певних рівнях. Ці закриті зони пропонують бонусні рівні, які або відповідають стилю старої відеоігри і закінчують три життя, або створені за зразком іншого інді-ігри, такий як Castle Crashers або Braid. Гравець може управляти як персонажами, так і Meat Boy, багато з яких вперше з'явилися в інших незалежних відеоіграх. У кожного персонажа є різні атрибути, такі як здатність Video Commander миттєво переміщатися по носію. Цих персонажів можна розблокувати, збираючи пов'язки на різних рівнях гри або роблячи кілька знімків крупним планом. Деякі смуги можна збирати зі спеціальними символами. На деяких рівнях, таких як ближні зони і рівні клану, можна грати тільки зі спеціальними персонажами. Доступні персонажі розрізняються залежно від версії гри.

Версія Xbox Live Arcade (XBLA) включає постійний мод під назвою «Teh internets», який оновлюється новими, безкоштовними, офіційно підтримуваними рівнями. У версії для ПК є розділ Super Meat World, який дозволяє користувачам грати в додаткові рівні, створені гравцями за допомогою редактора рівнів. Цей редактор був випущений в травні 2011 року. Гравці також можуть отримати доступ до Неподдерживается

внутріігрового режиму розробника, щоб налаштувати свої власні рівні за допомогою «грубих» інструментів, використовуваних Team Meat для створення гри.

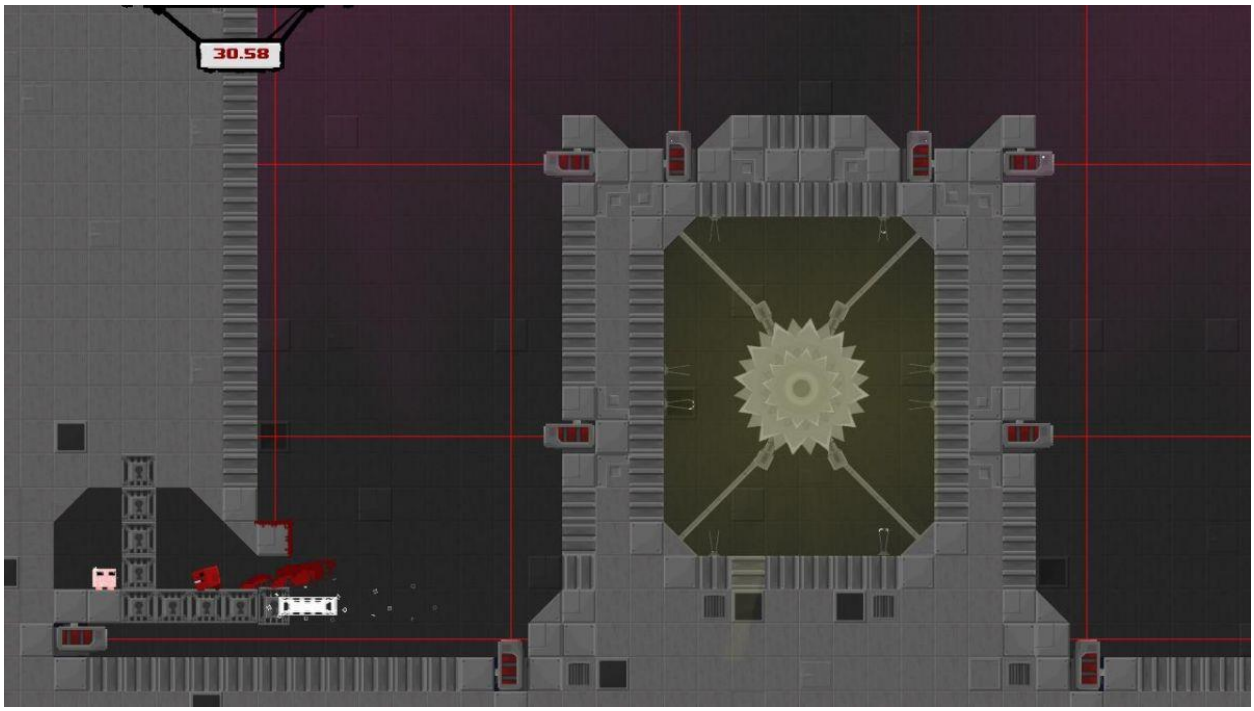

Рис. 2 – Гемплей гри Super Meat Boy

Meat Boy живе зі своєю дівчиною Bandage Girl. Одного разу д-р. Викрадений плід, розумний плід в костюмі життєзабезпечення. Потім М'ясник переслідує фетусов, поки фетусов не стріляє в ліс і не намагається вбити М'ясного хлопчика в своєму «Маленькому Слаггер», який знищується після того, як М'ясник заганяє його на лісопилку. Fetus збігає в занедбану лікарню, де Fetus C.H.A.D. випускає величезного монстра крові. М'ясник показує монстру сонце і викуповує його назад, але потім йде за плодом, який йде на його соляну фабрику. Там з бруду створюється клон Meat Boy під назвою Brownie, який призводить до гонки між ними, в якій Брауні, мабуть, убитий. Плід відправляється в пекло, де М'ясник виявляє, що його тіло потрапляє в пекло кожен раз, коли вмирає. Літтл Хорн, монстр, схожий на М'ясного Хлопчика, що складається з цих тіл, нападає на М'ясного Хлопчика. Фетусов збігає і випускає атомну бомбу, що відкриває шлях до вищого світ.

М'ясного хлопчика обманом змушують битися з Плачем Ларрі, трьома хробаками, провідними Ларрі-молодшого з «зв'язування Ісаака».

На заключному етапі Dr. Fetus Meat Boy, який руйнує міст, на якому сидить, і руйнує систему життєзабезпечення плода. Фетусов намагається вбити М'ясного Хлопчика і Бінтовщіка, підірвавши свій пристрій, але з'являється Брауні і рятує двох, жертвуючи собою. Коли Bandage Girl жує Meat Boy після його втечі, фетусов раптово виявляється нагорі Bandage Girl, намагаючись надути його, в той час як гра скорочується до титрів. Коли Dr. Фетусов переможений в Темному світі, з'ясовується, що атаки плода були неефективними, і що Дівчина-пов'язка зневажала його.

1.3.3 Cuphead

Cuphead - це відеогра для бігу та зброї 2017 року, створена і видана Studio MDHR. Натхнення до розробки стилю гри, було стилем анімації гумової змії, використовувався який в мультфільмах золотого століття американської анімації, таких як витвори Walt Disney Animation Studios і Fleischer Studios, і хотіла реалізувати їх підривні і сюрреалістичні якості.

У Cuphead є один або два гравці, які отримують під свій контроль головного героя і його брата Мугмана, щоб побитися на різних рівнях, дійти до вищої точки в битвах з головними злодіями(босами), для того щоб повернути свою позичку демону. Ігра взяла високу оцінку за художній стиль, ігровий процес, звуковий супровід і складність. Він мав як критичний, так і комерційний успіх: за три роки він отримав кілька нагород і був проданий більш шести мільйонів копій. Мультсеріал по грі знаходиться у виробництві Netflix.

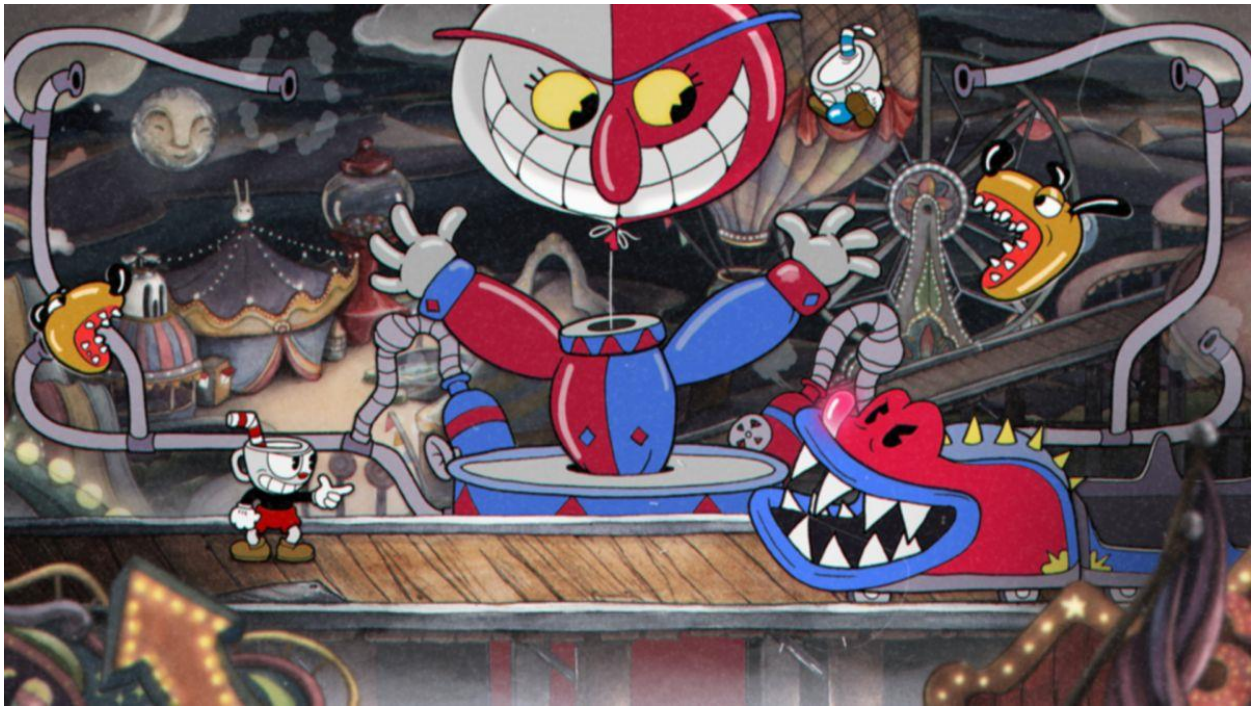

Рис. 3 – Гемплей гри Cuphead

Геймплей Cuphead зосереджений на безперервних битвах з босами, що перемежовуються рівнями «біжи і стріляй». У всіх світах, крім останнього, є мавзолей, де гравець повинен парирувати привидів, перш ніж ударити по урні, що винагороджує їх "Супер" ходом, щоб завершити випробування. Кожен з цих боїв з босами і рівнів «біжи і стріляй» знаходиться в одному з чотирьох різних світів, причому в четвертому світі йде фінальна битва з самим дияволом. Кожна битва з босом включає в себе простий, нормальний і експертний режими складності (за винятком двох останніх босів, у яких немає простого режиму). Для завершення гри необхідно перемогти будь-якого боса в звичайному режимі, а режим експерта розблокується після завершення. У той час як більшість боїв з босами відбуваються на суші, натомість кілька босів на літаках б'ються на рівнях, які грають роль стрілянина з бічної прокруткою. Гра також включає в себе рольові елементи і послідовність розгалужених рівнів. Фігурки гравців мають нескінченне життя і підтримують всі спорядження до смерті. Споряджати зброю і спеціальні здібності, відомі як Charms, можна придбати у внутрішньоігровому магазині Porkrind's Emporium, де можна знайти монети

на рівнях і в зовнішньому світі. У ігрових персонажів є парирує атака, яку можна використовувати на певних об'єктах, зазначених рожевим кольором для отримання різних ефектів, найважливіший з яких збільшує «суперметр», що дозволяє більш потужні атаки. Суперметр представлений гральними картами. Гравець може використовувати менш потужну суперздатностями, звану «EX» рухом з однієї грального картою, при цьому використовується рух залежить від екіпірованого зброї.

1.3.4 Hollow Knight

Hollow Knight - це гра з пригодами 2017 року, розроблена і видана Team Cherry, яка була розроблена для Microsoft Windows, macOS і Linux в 2017 році, а також для Nintendo Switch, PlayStation 4 і Xbox One в 2018 році. Розробка була продовжена. Частково фінансується за рахунок краудфандінговой кампанії Kickstarter, зібравши більш ніж 57 000 австралійських доларів в кінці 2014 года.

У грі розповідається про безіменного лицаря, який вторгається в древнє, заболівше чумою королівство, населене різними комахами, відоме як Hallownest. Лицар повинен подорожувати по Hallownest, битися з ворогами і відкривати нові можливості для просування, відкриваючи таємниці королівства. Hollow Knight був добре прийнятий критиками, і за станом на грудень 2020 року було продано більше 3 мільйонів копій. У розробці знаходиться продовження під назвою Hollow Knight: Silksong.

Рис. 4 – Гемплей гри Hollow Knight

Hollow Knight - це пригодницька гра в жанрі 2D Metroidvania, дія якої відбувається в Hallownest, вигаданому стародавньому королівстві. Гравець управляє схожим на комаха, безмовним і безіменним лицарем, досліджуючи підземний світ. Лицар використовує цвях - різновид меча, який використовується як в бою, так і при взаємодії з навколишнім середовищем.

У більшості областей гри гравці стикаються з ворожими помилками і іншими істотами. У рукопашному бою використовується шип для утримання ворогів з близької відстані. Гравець може вивчати заклинання, що дозволяють атакувати на великій відстані. У переможених ворогів випадає валюта під назвою Гео. Лицар починає з обмеженої кількості масок, які представляють місця зустрічі персонажа. «Осколки маски» можна збирати протягом всієї гри, щоб збільшити максимальне здоров'я гравця. Якщо лицар отримує шкоди від ворога або від навколишнього середовища, маска зменшується. Перемігши ворогів, лицар перемагає Душу, яка зберігається в Посудино Душі. Якщо все маски втрачені, лицар вмирає, і на цьому місці з'являється тінь. Гравець втрачає все гео і може зберегти зменшена кількість душі. Гравці повинні перемогти Тінь, щоб повернути втрачену монету і

забрати нормальна кількість Душі. Гра триває з останнього відвіданого банку, на якому вони сиділи, які розкидані по всьому ігровому світу і служать точками зберігання. Спочатку гравець може використовувати Душу тільки для відновлення «Фокуса» і масок, але по ходу гри гравці відкривають різні наступальні заклинання, які поглинають Душу.

У багатьох областях є більш складні вороги і боси, яких гравцеві, можливо, доведеться перемогти, щоб просунутися вперед. Перемога над деякими босами дає гравцеві нові навички. Пізніше в грі гравці отримують «цвях сновидінь», легендарний клинок, який «може прорізати завісу між сном і неспанням». Це дозволяє гравцеві створювати більш складні версії кількох босів і ламати все, що закриває шлях до фінального боса. Коли гравець перемагає останнього боса гри, званого «Порожній лицар», він отримує доступ до режиму «Сталева душа». В цьому режимі смерть необоротна, і якщо лицар втрачає все маски, блокування сховища скидається.

Під час гри гравець зустрічає персонажів (NPC), з якими він може взаємодіяти. Ці персонажі надають інформацію про сюжеті або історіях гри, пропонують допомогу і продають предмети або послуги. Гравець може поліпшити лицарський цвях, щоб нанести більше шкоди, або знайти судини душі, щоб нести більше душі. Протягом всієї гри гравцям видаються предмети, які забезпечують нові можливості пересування, в тому числі додатковий стрибок в повітрі (Крила монарха), чіпляння за стіни і стрибки з них (Кіготь богомола) і швидку серію (Плащ метелика). Гравець може вивчити інші бойові навички, відомі як нейл-арт, і вищезгадані заклинання. Щоб додатково налаштувати лицаря, гравці можуть екіпірувати різні талісмани, які можна знайти або купити у неігрових персонажів. Деякі з його ефектів включають поліпшені здатності або навички, більше масок або їх регенерацію, кращі навички пересування, більш легкий збір валюти, ніж у Душі, і трансформацію. Установка Чарма включає в себе певну кількість обмежених слотів, званих кришками. Можливо носіння оберегу, для якого потрібно більше, ніж наявна кількість клатч, але воно призводить до

«затемнення», в результаті чого лицар отримує подвійний шкоди з усіх джерел.

Hallownest складається з декількох великих взаємопов'язаних зон з унікальними темами. Завдяки нелінійного ігрового дизайну, Hollow Knight не прив'язує гравця до одного шляху в грі і не вимагає від нього дослідження всього світу, хоча існують перешкоди, що обмежують доступ гравця до певної області. Гравцеві може знадобитися просунутися по сюжету гри або придбати певний навик руху, навик або предмет, щоб рухатися далі. Щоб швидко подорожувати по ігровому світу, гравець може використовувати Stag Stations, термінали мережі тунелів; гравці можуть подорожувати тільки до раніше відвіданих і розблокованим станціям. Інші динамічні методи, такі як трамваї, ліфти і «Врата мрії», зустрічаються пізніше в грі.

Коли гравець входить в нову область, у нього немає доступу до карти свого оточення. Їм потрібно знайти картографа Корніфера, щоб купити приблизну карту. У міру того, як гравець досліджує область, карта стає більш точної і повної, хоча вона оновлюється тільки тоді, коли він сидить на лавці. Гравцеві потрібно буде купувати певні предмети, щоб заповнювати карти, переглядати пам'ятки і розміщувати маркери. Положення лицаря на карті можна побачити тільки в тому випадку, якщо у гравця є певний оберіг.

1.4 Засоби розробки

1.4.1 Unity

Що таке ігровий движок? Ви не знайдете визначення цього словника в усіх словниках. Однак, якщо ви хочете знайти відповідь на це питання, ви, ймовірно, задається питанням: «Ігровий движок - це програма, яка дозволяє створювати відеоігри і управляти ними». Ігровий рушій впливає на всі аспекти гри, включаючи презентацію, фізику, звуковий дизайн, освітлення та мережеве взаємодія. А якщо ви нічого не можете створити з його допомогою, ви можете створити це за допомогою спеціальної програми, а потім додати в свою гру.

Редактор юнітів має простий інтерфейс перетягування, його легко налаштувати і складається з безлічі вікон, так що ви можете обговорювати свою гру прямо в редакторі. Цей модуль підтримує дві мови написання: C # і JavaScript (змінений). Модульні проекти розбиті на сцени (рівні). Кожен файл відтворює ігровий світ зі своїми об'єктами, скриптами і настройками. Сцени можуть бути порожніми об'єктами (моделями) і ігровими об'єктами (немодельний об'єктами)[3].

У мене є набір елементів, які взаємодіють зі сценарієм. У них також є імена (Unity дозволяє використовувати кілька елементів з одним і тим же ім'ям) і може бути обкладинкою, яка відображається як ярлик (ярлик). Тому будь-який мотив у зовнішньому вигляді має в значній мірі мінливу частина. Він підтримує координати позицій, ніг і розмірів трьох осей. Елементи з візуальної геометрією також мають компонент відображення сітки, який відображає модель.

Пристрій також підтримує фізику стану і напруги, фізику тканинної ляльки. У редакторі старі системні настройки. Дочірній об'єкт кілька разів

змінює положення, поворот і масштаб основного об'єкта. Скрипти в редакторі пов'язані з елементами у вигляді окремого розділу.

Ви можете додавати текстури в движок для створення альфа-каналів, рівнів MIP, призначених для користувача карт, карт освітлення і карт відображень, але ви не можете пов'язувати текстури з вашою моделлю. Матеріал для модуля. Редактор модулів підтримує написання і редагування шейдеров. Також є розділ для створення анімації. Його також можна попередньо підготувати в 3D-редакторі, імпортувати разом з моделлю і розбити на файли.

Рис. 5 – Інтерфейс Unity

(A) Панель інструментів забезпечує доступ до найбільш важливих функцій вашого додатка. Зліва - основні інструменти для редагування сцени. Перегляд і ігрові об'єкти

в цьому. У центрі розташовані елементи управління відтворенням, зупинкою і кроком. Кнопки праворуч надають вам доступ до Unity Collaborate, Unity Cloud Services і вашого профілю Unity, за яким слід меню для видимості обкладинки і, нарешті, меню «Формат редактора» (яке має деякі інші

формати для вікон редактора і дозволяє вам для збереження ваших власних дизайнів).

(B) Вікно ієрархії - це ієрархічне текстове представлення кожного GameObject в поданні. Все в поданні має запис в ієрархії, тому два вікна тісно пов'язані. Ієрархія показує структуру того, як ігрові об'єкти пов'язані один з одним.

(C) Сцена гри схожа на те, що ваша остання гра демонструє через ваші камери огляду. Коли ви натискаєте кнопку Play, починається симуляція.

(D) Попередній перегляд сцени дозволяє переглядати і редагувати вид. види Ви можете відображати тривимірний або двомірний вигляд, в залежності від типу проекту, над яким ви працюєте.

(E) Ви можете переглядати і редагувати всі вибрані в даний момент властивості GameObject у вікні провідника. Оскільки різні типи ігрових об'єктів мають різні групи будівель, формат і вміст провідника розрізняються. перемикати вікна кожного разу, коли ви вибираєте інший GameObject.

(F) Вікно проекту показує вашу бібліотеку засобів, доступних у вашому проекті. Якщо ви збираєте кошти для свого проекту, вони будуть показані тут.

(G) Рядок стану надає повідомлення для різних процесів Unity і швидкий доступ до пов'язаних інструментів і налаштувань.

1.4.2 Microsoft Visual Studio

Microsoft Visual Studio - це середовище розробки для Microsoft (IDE). Він використовується для розробки веб-сайтів, веб-додатків, веб-сервісів і мобільних додатків, а також комп'ютерних програм. Visual Studio використовує платформи розробки програмного забезпечення Microsoft. Може управляти як кореневим, так і кореневих кодом[4].

Visual Studio має IntelliSense (фрагмент коду), а також редактор коду, який підтримує поновлення коду. Встановлене додаток працює як вихідний

код і виправлення пристрою. Інші інструменти програми включають код профілю, графічний дизайнер, веб-дизайнер, дизайнер класів і дизайнер даних. Вітаються розширення, що розширюють функціональність на всіх рівнях, включаючи підтримку систем управління контентом (таких як Subversion і Guitar), таких як редактори та програми перегляду на національних мовах або інші програмні інструменти. Встановіть новий пристрій. Прокляття. Життєвий цикл (Azure DevOps: для клієнтів Team Explorer).

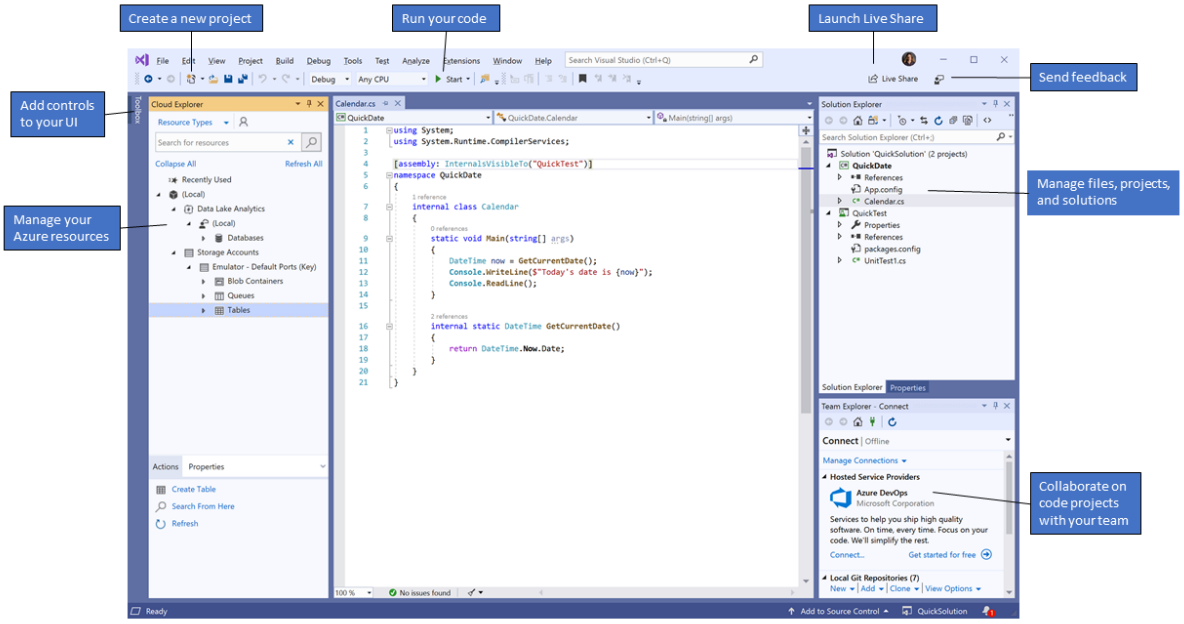

Рис. 6 – Інтерфейс Microsoft Visual Studio

Visual Studio (як і інші IDE) включає редактор коду, який підтримує інтерпретацію і витяг коду і використовує інтелектуальні параметри для варіативності, функціональності, режиму, кривої і запитів LINQ. Подумки рекомендує використовувати XML, стиль і JavaScript при вході в систему. Автоматичний дизайн не відображається в списку модулів у вікні редактора коду поруч з редактором.

Редактор Visual Studio підтримує параметри коду закладок для швидкої навігації. Інші інструменти навігації включають блоки тексту і розірваний звичайний текст, а також додатковий аналіз тексту з додатковими фразами.

Редактор скриптів також включає в себе панель управління багатозадачністю і список справ. Редактор коду підтримує фрагменти, які зберігаються як шаблони повтору коду і можуть бути прикріплені до коду і призначені поточного проекту. Також був створений віджет фрагмента коду, який виглядає як плаваюче вікно, яке може бути автоматично приховано, коли воно не використовується, або розміщено збоку від екрана. Редактор Visual Studio також підтримує завантаження коду, реструктуризацію параметрів, перейменування змінних і режимів, скорочення інтерфейсу і відображення членів класу в будівлі.

Є фон Visual Studio (також відомий як додатковий набір). Коли ви вводите код, Visual Studio компілює код ззаду і пропонує червоні лінії і помилки. Попередження позначається зеленою лінією. Безпека не створює виконуваний код, тому що для цього потрібно компілятор, відмінний від того, який використовується для створення виконуваного коду. Фонова колекція була вперше представлена в Microsoft Visual Basic, але тепер додана на всіх внутрішніх мовами.

Розділ 2 ПРОЕКТНА ЧАСТИНА

2.1 Характеристика потенційної аудиторії проекту

Цільовою аудиторією проекту є люди, які потенційно мають наступні характеристики:

- Вік: діти та підлітки, 10-21 років.
- Стать: здебільшого хлопці, тому що, переважно їм подобаються ігри з елементами стрільби, ніж дівчатам, яким в цьому віці більше подобаються казкові та мультиплікаційні персонажі.
- Інтереси: казуальні ігри для смартфонів, ігри на швидкість і спритність рук і ін. в жанрі платформер.
- Психологічний тип: дослідник (цей клас фокусується на знаннях, придбання ігрового контенту, вивченні ігрової механіки і ігрових навичок), пристроях пам'яті (люди, які стимулюють розвиток гри і прогрес).
- Чого очікувати від проекту: використовуйте вільний час, щоб зануритися в цікавий і дивно барвистий світ і поділитися своїми іграми.

2.2 Концепція

Далі мова піде про головні аспекти розробки ігри.Также про концепцію проекту в цілому. Основи проектування гри[5].

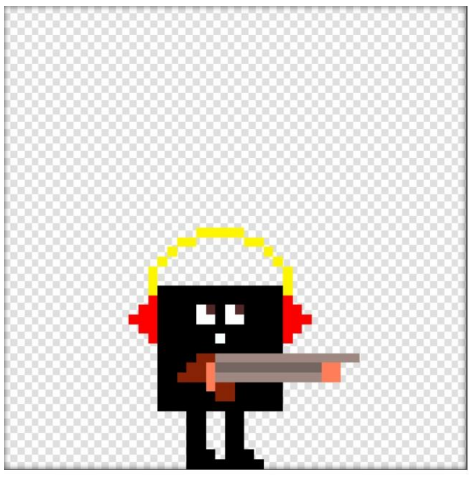

Рис. 7 – Головний персонаж гри

2.2.1 Вибір концепції платформеру

Вибір концепції платформеру було обрано із наступних видів під-жанру:

1. Чорно-білий стелс (2D) - Вид згори, з динамічним освітленням і двомірним звуком. Весь рівень і Неінтерактивні об'єкти - чорно-білі, інтерактивні - сині, пастки і вороги - помаранчеві і червоні, значення - жовті, союзники - зелені. Головне - складність і AI. Налаштування досі не зрозуміле.
2. Стелс-екшен (2D) - Вид збоку, аркадний платформер зі стрибками, гра на реакцію типу SMB, тільки замість пасток - охоронці. Ми граємо за ніндзя, команда повинна проникнути на базу і щось зібрати за обмежений час або дійти до фінішу. Травма - це смерть, і далі почнемо спочатку.
3. Темний дух (3D) - Дія в Лондоні 1930 року, ми граємо темношкірого персонажа, який може маніпулювати розумом ворога, відволікаючи його і створюючи ілюзії.
4. Гра без правил (2D) - Платформена сторона, незрозуміла гра з мінливою механікою і правилами. Мета гри - змусити гравця зупинитися якомога швидше.
5. Киберпанк-квест (2D) - Сторінка квесту з 4-ма синіми квіточками і ухилом під GameBoy. Гравець може розміщувати точки телепортації в місцях, щоб повернутися в раніше недоступне місце, необхідне для проходження. Час продовжує текти (день і ніч, одне сховище). Спочатку доступні всі локації.
6. Киберпанк-пригоди (2D) - Сюжет гри в 1-му виді змінюється в залежності від ситуації. Це вид збоку, як вгорі, так і внизу. У грі поєднуються квест, платформер, файтинг, стелс, гонки.
7. Cybercity (3D) - Кіберпанк, пригода в випадковому місті, гравець наданий самому собі, немає можливості пройти гру - гра цього не говорить. Проста графіка, як в SubRosa, розрахована на багатьох користувачів і в браузері.

8. Конвертор чар (2D) – Похмура гра про чарівника (лиходія) з трьома речами. Інша механіка у противників, зрушення в бік головоломок і пригодницьких квестів, ніж в бою. Додаткові кімнати, де втрачається здоров'я, гравець поступово перетворюється на скелет.
9. Класичний платформер (2D) - Це платформер по типу маріо. Де можно стрибати та вбивати стрибком ворогів, бігати та накопичувати очки, без інших доповнень.

Після аналізу було обрано класичний 2д платформер.

2.2.2 Назва та сюжет

Дуже важливо правильно придумати назву, тому що подальше розуміння залежності гри залежить від цього. Були розглянуті багато матеріалів по темі, складений список варіантів:

1. Справжній порядок.
2. Убивча подорож.
3. Платформер з перешкодами.
4. Чорний хлопець.
5. Чорний стрілець.
6. Вбивця негідників.
7. Врятування.
8. Хаос.
9. Дивний хлопчина (Strange boy).

Провівши невелике дослідження серед тих, хто знайомий з розробкою, було вирішино назвати гру «Strange boy», що перекладається як «Дивний хлопчина». Цей варіант добре поєднується з сюжетом гри, він короткий, легко читається і легко запам'ятовується.

Сюжет представляється гравцеві на початку гри в наступному форматі:
Головний герой заблукав, у просторі платформерного світу і йому треба вижити в цьому жахливому світі, повному мострів. Далі керуючи головним персонаж ігроку треба зібрати очки, вбити монстрів та дійти фінішу.

2.2.3 Блоки

Звичайні блоки слугують для того щоб персонаж по ним ходив, бігав і пригав. На них не має жодної строчки коду, а тільки один коллайдер. Вони можуть стояти на місці а можуть рухатись в гору-вниз або ліво-право.

Також є блоки які наносять шкоду персонажу їх ще називають пастками, таких блоків багата різноманітність є шипи, блоки червоного кольору, круглі пили, гармати. Коли персонаж доторкнеться до одного з таких блоків то він втратить одне життя(рис. 8).

Чим більше різноманітність блоків тим більш захоплююча гра.

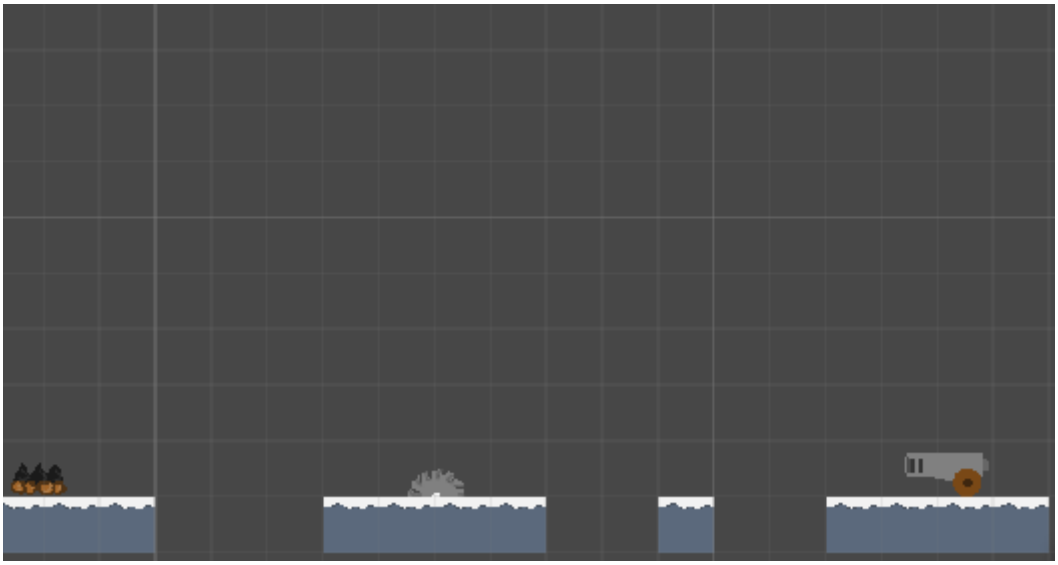

Рисунок 8 – блоки

2.2.4 Декорації

Декорація — об'єкт або дія, спрямована на збільшення краси обличчя, тіла і т. д. Декорації в грі - це об'єкти з якими неможна взаємодіяти персонажу, вони слугують лише для краси оточуючої середи та красивого дизайну. Їхня ціль додати грі атмосфери, передати емоції та допомогти вникнути в гру с головою. Без декорацій гра виглядала би страшною. Тому так важливо підібрати їхній правильний стиль.

2.2.5 Опис розробки проекту

Для розробки цифрового рішення потрібен дизайн, написання сюжету, намалювати графіку. Потім це все об’єднати в прототип гри, далі треба протестувати гру та виправити всі баги. Якщо ми виправили всі помилки у коді то випускаємо її у бета-тестування. Цей процес показаний на схемі idef0:

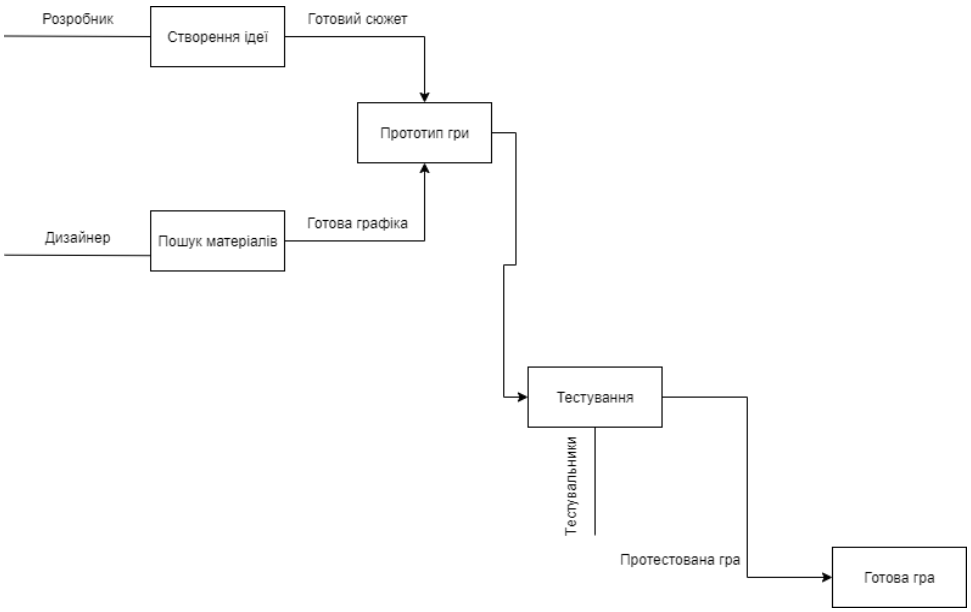

Рис. 9. Схема idef0

Розробник – це фахівець, що займається програмуванням, виконує розробку програмного забезпечення.

Дизайнер – це людина яка виконує пошук матеріалів для графікі та малює її.

Створення ідеї – це створення цілі гри, сюжету та механікі гри.

Готова графіка – це повне створення візуалу гри.

Прототи гри – це макет або начерк майбутньої гри, він покликаний знизити ризик створення неіграбельного проекту.

Тестування – це процес перевірки гри на баги та виправлення помилок у грі.

Готова гра – це гра яка повністю протестованна та готова до виходу у простір інтернету.

2.3 Функціонал проекту

Проект являє собою двомірну комп'ютерну гру жанру двухмірний платформер, призначений в першу чергу для розваги, релаксації і веселощів.

Перш за все, проект повинен відповідати наступним вимогам:

1. Гра повинна бути простою і зрозумілою.
2. Основна механіка гри - це здатність гравця стрибати по платформах та можливість винищувати ворогів.
3. Головного героя не можна занадто контролювати.
4. Вороги стоять в певних місцях, вони повинні ходити та атакувати героя якщо він поруч.
5. На сцені повинні бути присутнім предмети колекціонування. Ці предмети повинні збільшити рахунок гравця.
6. Якщо гравець виграє або програє, повинен з'явитися невеликий стіл із зазначенням рівня повторення або переходом до основного столу.
7. Таблицю не слід переоцінювати, але в той же час вона повинна бути бути інформативним.

Ці вимоги поширюються на базову функціональну частину,
це потрібно зробити в першу чергу.

Розділ 3 РАЛІЗАЦІЯ ПРОЕКТУ

3.1 Візуальна частина

Слід мати на увазі, що основні етапи реалізації проекту, це був розвиток графічного дизайну та розвиток програмного коду паралельно один одному. Цьому сприяло те, що для написання програмного коду, не обов'язково мати готові анімації та інші речі, але для перевірки функціональності самого коду достатньо стандартних примітивів Unity.

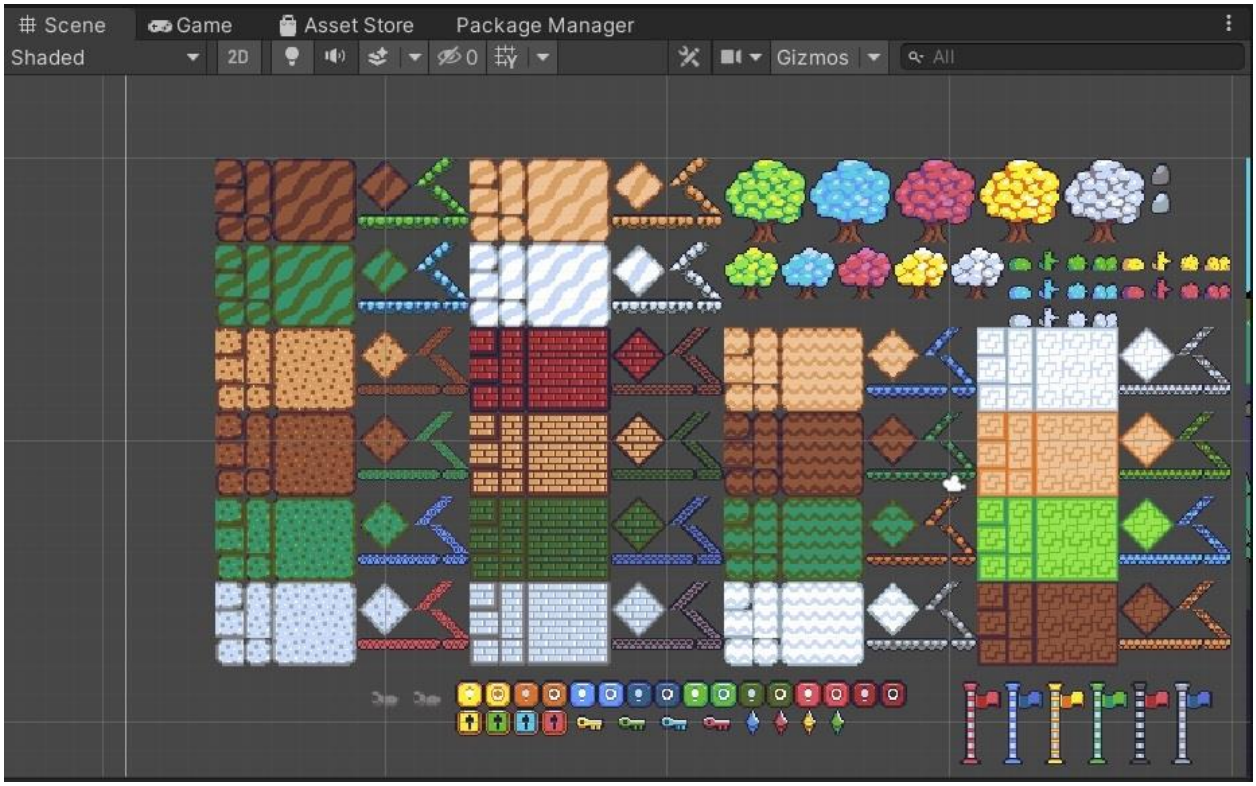

Рис. 10. Графіка оточення

Також такий спосіб реалізації проекту суттєво збільшує швидкість роботи. Перш за все, варто виділити такі основні поняття, як черепиця і спрайт (Sprite).

Tile - невеликий повторюваний фрагмент, який використовується для побудови великих зображень (плиточна графіка). Часто використовується для створення рівнів для 2D-ігор.

Спрайт (Sprite) - графічний об'єкт, який є растровим зображенням. Він використовується в комп’ютерній графіці як основний блок для анімації двовимірних об’єктів.

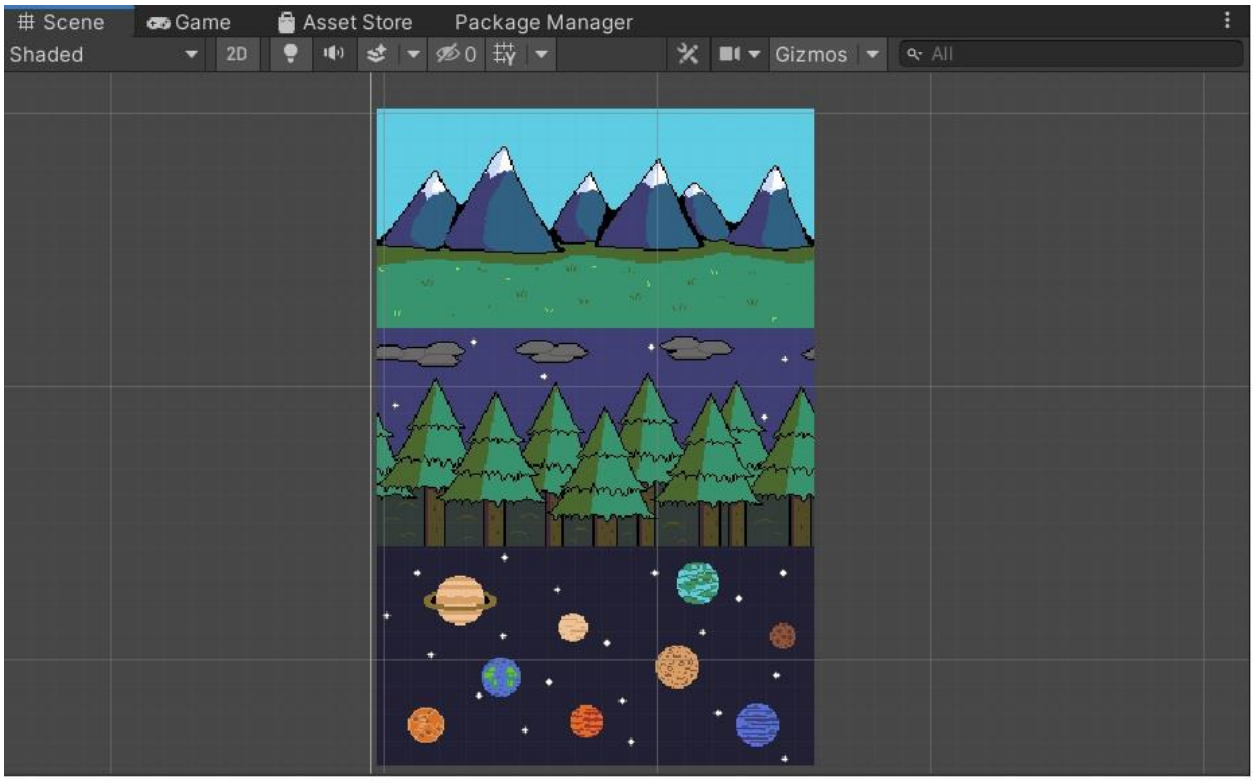

Рис. 11. Графіка заднього фону

Головний персонаж

Головний герой - це найважливіше в грі, тому до його розробки було задіяно багато зусиль і таким чинов вироблено анімація бігу (Рисунок 12).

Рис. 12. Спрайти для анімації головного персонажу

Монстри

Спочатку були зроблені малюнки персонажів, потім повне малювання істот і розбиття їх на складові частини, для подальшої анімації.

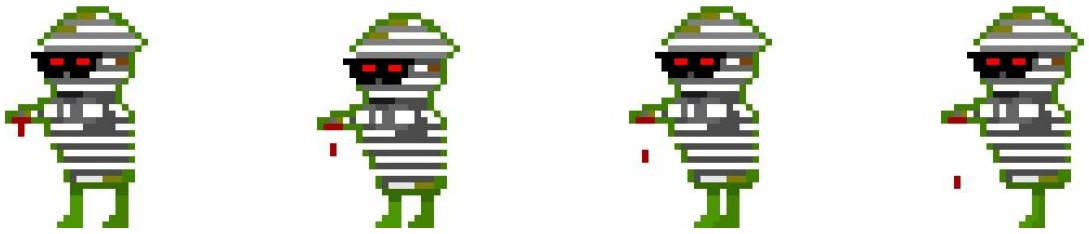

Рис. 13. Спрайти для анімації персонажу мумія

Наступний монстр це скелет, він буду найслабкішим серед інших за нього будуть давати менше всіх очків. Далі буде продемонстровано цого анімація (рисунок 14).

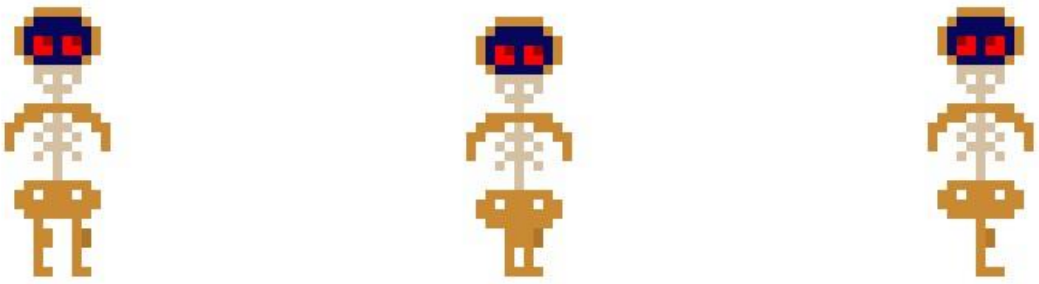

Рис. 14. Спрайти для анімації персонажу скелет

3.1.1 Розробка анімації головного персонажу

Після того я було намальовано спрайти для анімації головного персонажу теперь ми додамо анімацію руху у Unity (Рисунок 15).

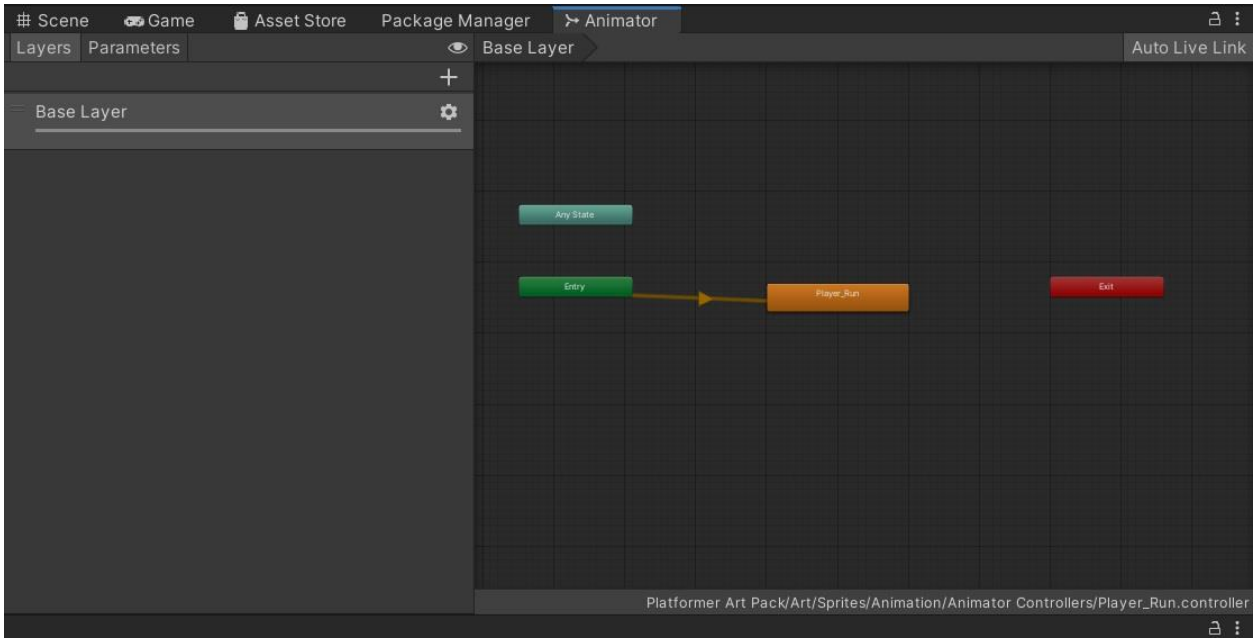

Рис. 15. Анімація руху головного персонажу

Теперь розробимо анімацію стрибка и додамо її у Unity. Ця анімація дуже важлива бо на неї будується половина механіки гри.

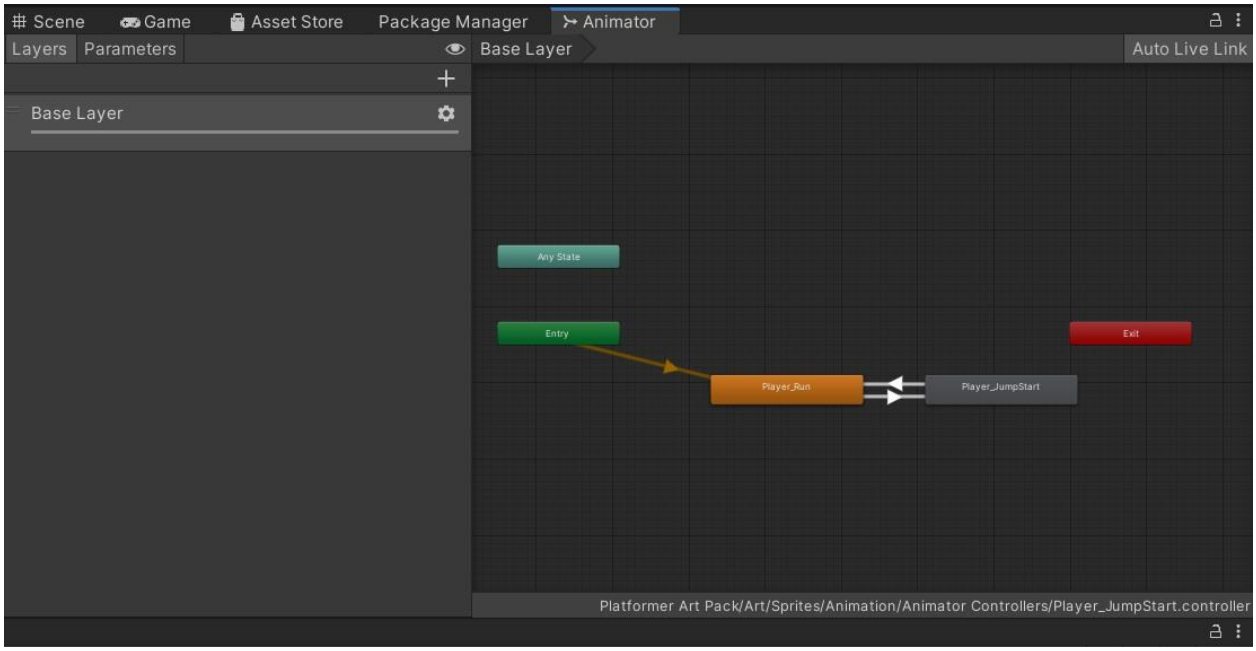

Рис. 16. Анімація стрибка головного персонажу

3.2 Програмна частина

Перш за все, нам необхідно додати в сцену платформу, за якою буде переміщатися наш герой і його вороги. Без неї гравець просто буде нескінченно падати в просторі. Спочатку нам необхідно додати сам спрайт платформи, який був створений раніше, для цього досить просто перетягнути його з папки в вікно проекту, після чого він повинен з'явитися серед всіх об'єктів вже доданих в гру. Після цього нам необхідно додати нашу платформу в поточну сцену. Це можна зробити 2-ма способами: перетягнути з вікна Проекту в вікно Сцени, або спочатку в вікні ієрархії створити окремий порожній об'єкт, до якого згодом і застосувати спрайт платформи (Рисунок 17).

Після цього нам необхідно додати до об'єкта коллайдер. Він необхідний для того, що б движок розумів межі об'єкта і гравець міг з ним взаємодіяти. У Unity існує два види коллайдеров: для тривимірних і двовимірних об'єктів. Так як створювана гра двовимірна, то і коллайдер потрібно додавати двовимірний. Що б додати коллайдер до об'єкта, необхідно виділити його у вікні ієрархії, а потім додати компонент коллайдера у вікні Інспектора. Для настройки розмірів коллайдера потрібно скористатися кнопкою Edit Collider.

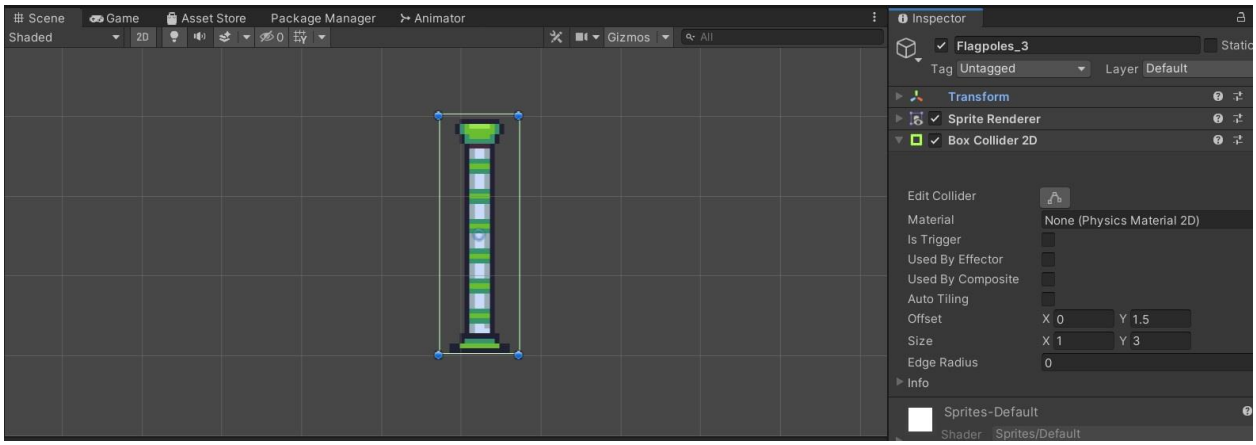

Рис. 17. Додавання колайдеру

Unity дозволяє зберігати об'єкти, з усіма доданими властивостями і скриптами для того, що б потім їх можна було повторно використовувати, просто перемістивши на сцену, не створюючи заново новий об'єкт. Такий об'єкт називається Префаб і для його створення досить просто перемістити створений об'єкт з вікна ієрархії в вікно Проекту. Для префабов має сенс створити окрему папку, щоб вони не губилися серед інших елементів гри.
Тепер створюємо героя гри. Для цього так само додаємо двовимірний об'єкт, і додаємо до нього спрайт героя. Для реалізації героя нам понадо- бится коллайдер, а так же компонент Rigidbody 2D, через нього налаштовуватися фізична модель об'єктів.

3.2.1 Реалізація прогамного коду персонажу

Так як головний герой повинен рухатися і здійснювати різні дії нам необхідно створити скрипт на мові C #. Це можна зробити декількома способами: створити у вікні Проекту, або створити відразу на необхідному об'єкті у вікні Інспектора. Коли скрипт буде створений, відкриваємо його і після цього повинен запуститися Microsoft Visual Studio. У коді вже будуть підключені основні бібліотеки, а так же створений стандартний метод Update. Записуємо код для руху персонажем (рисунок 18). Після чого додаємо його до нашого героя.

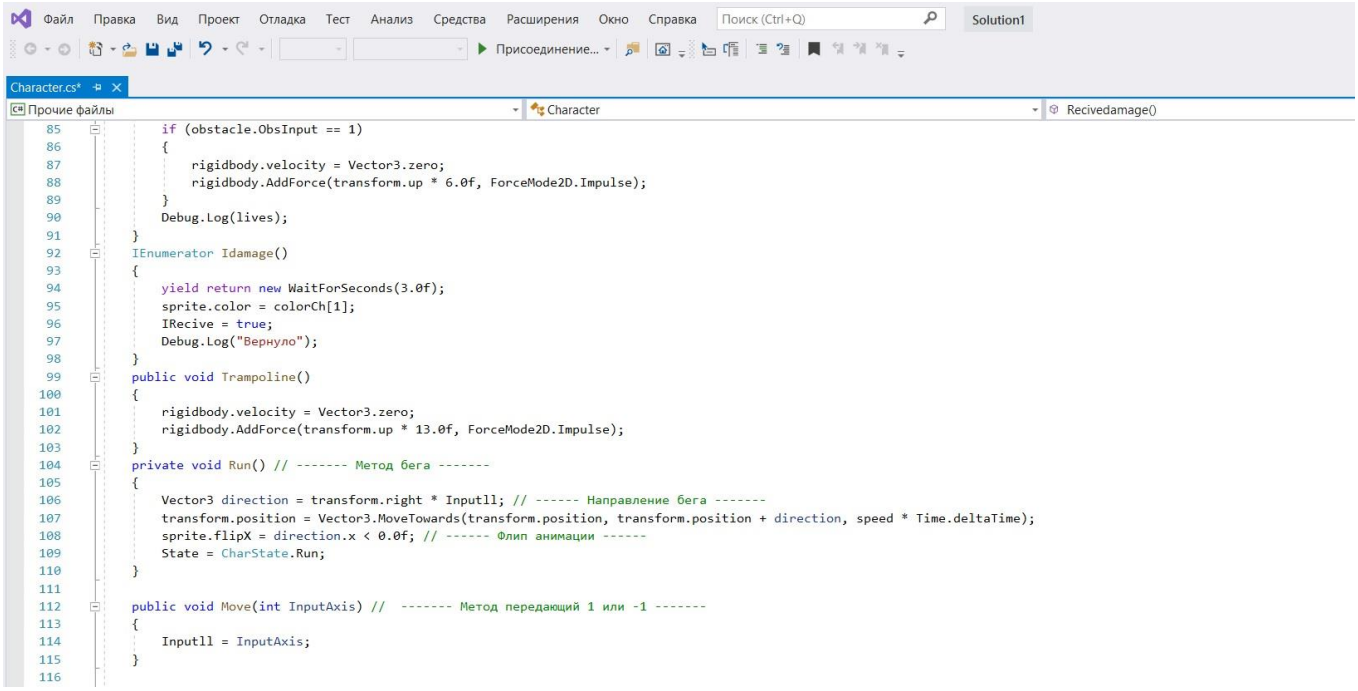

Рис. 18. Код руху

Для реалізації стрибків на необхідно створити у героя порожній дочірній об'єкт. Цей порожній об'єкт назвемо groundCheck, за допомогою нього ми визначатимемо, знаходиться персонаж на землі чи ні. Це необхідно, що б гравець не міг нескінченно підніматися вертикально вгору при постійному натисканні клавіші прижка.Теперь, якщо запустити проект, ми зможемо побігати по платформі з боку в бік.

Точно так же створюємо другий скрипт, який ми додаємо до камери і в ньому прописуємо умови, за якими камера буде слідувати за героєм по рівню. Якщо ми цього не зробимо, то не зможемо стежити за діями персонажа.

Тепер нам необхідно створити ворогів. Вороги створюються схожим способом, за тим винятком, що гравець ними управляти не може. Так само характерно особливістю ворогів є те, що в залежності від ситуації, вони самі вибирають необхідну дію доступне їм за правилами гри. Так само пересування ворогів обмежені кордонами їх місць проживання.

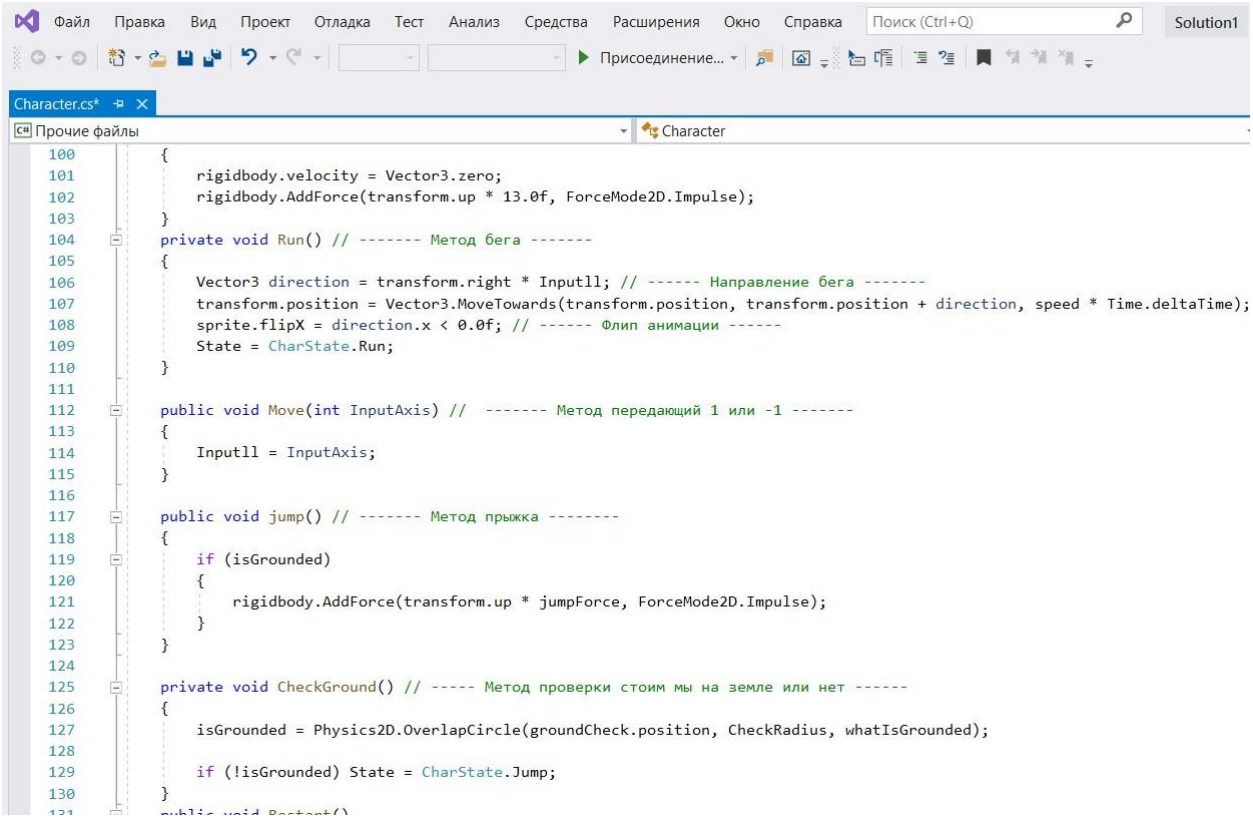

Рис. 19. Код стрибка

3.2.2 Реалізація прогамного коду монстрів

Багато з скриптів монстрів універсальні і підходять безлічі ворогів, різні в них лише початкові змінні, які є характери-стиками монстрів.

Спочатку розробимо рух монстрів, для цього ми дадому до скрипту метод Move(Рисунок 20).

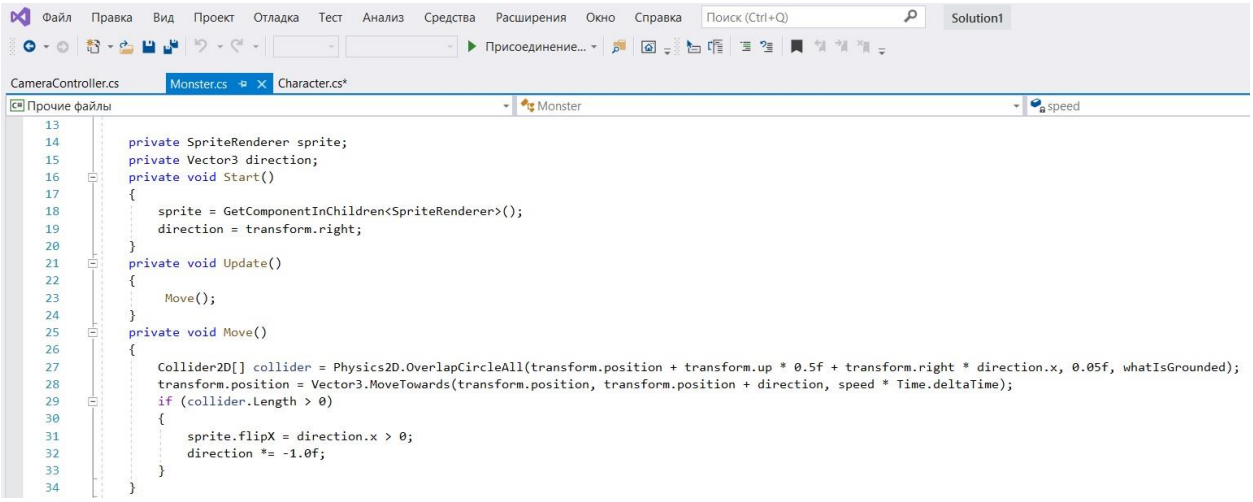

Рис. 20. Код руху монстрів

Далі було розроблено метод отримання шкоди, він дуже віжливий бо саме через цей метод ми маємо змогу вбити ворого.

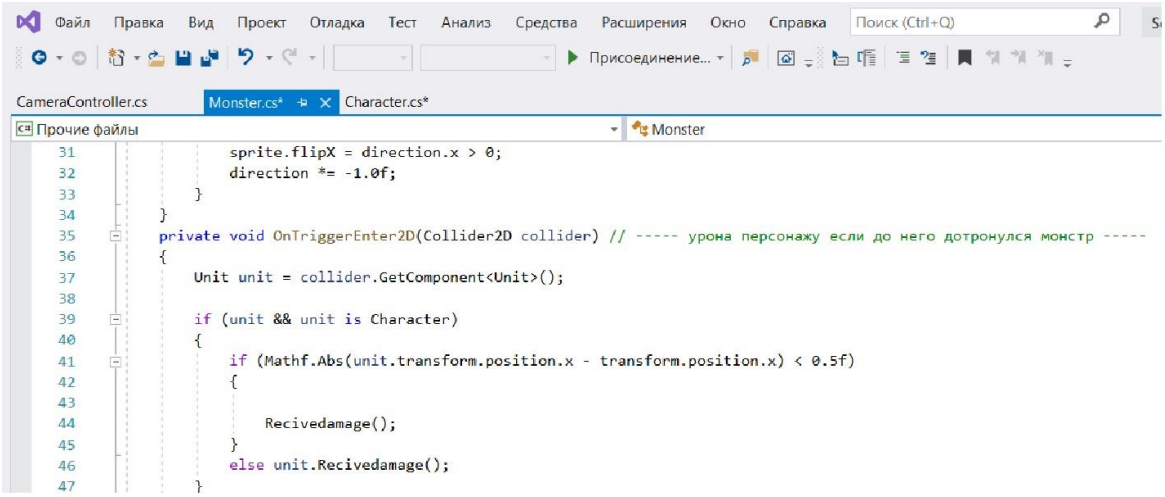

Рис. 21. Код отримання шкоди монстрів

3.2.3 Інші скрипти

Скрипт CameraController відповідає за переходи на наступні рівні і пересування камери. Весь алгоритм розташований в методі Update. Він відповідає за програвання ведення меню паузи, меню програшу, пересування камери слідом за ігроком і переходу гравця на наступний рівень. Про останній дії пого-ворім детальніше.

Скрипт має посилання на гравця і постійно перевіряє числове зна-ня кількості огнесвета в колбі, якщо воно перевищило поріг для прохожде-ня поточного рівня, відбувається наступне:

1. Виключається менеджер поточного рівня, який містить в собі підготовленні платформи, монстрів і бонусів на поточному рівні.
2. Змінюється глобальна змінна рівня.
3. На поточної позиції гравця активується заготівля початку наступного рівня з відповідними платформами і екранами навчання.

4. Коли ефект досягне гравця, його префаб замінюється на префаб наступного рівня, в цей же час зміну зовнішнього вигляду гравця закривають візуальні ефекти.

5. Камера включає вертикальний режим проходження за гравцем, поки гравець не виявиться на висоті наступного рівня.

6. Після цього видаляється пул об'єктів попереднього рівня.

7. Потім включається менеджер генерацій наступного рівня, котрий створює пул відповідних об'єктів.

8. Камера включає горизонтальний режим проходження за гравцем, і гравець продовжує проходження по горизонталі далі, однак якщо це був закінченний рівень, запускаються об'єкти, викликає при перемозі.

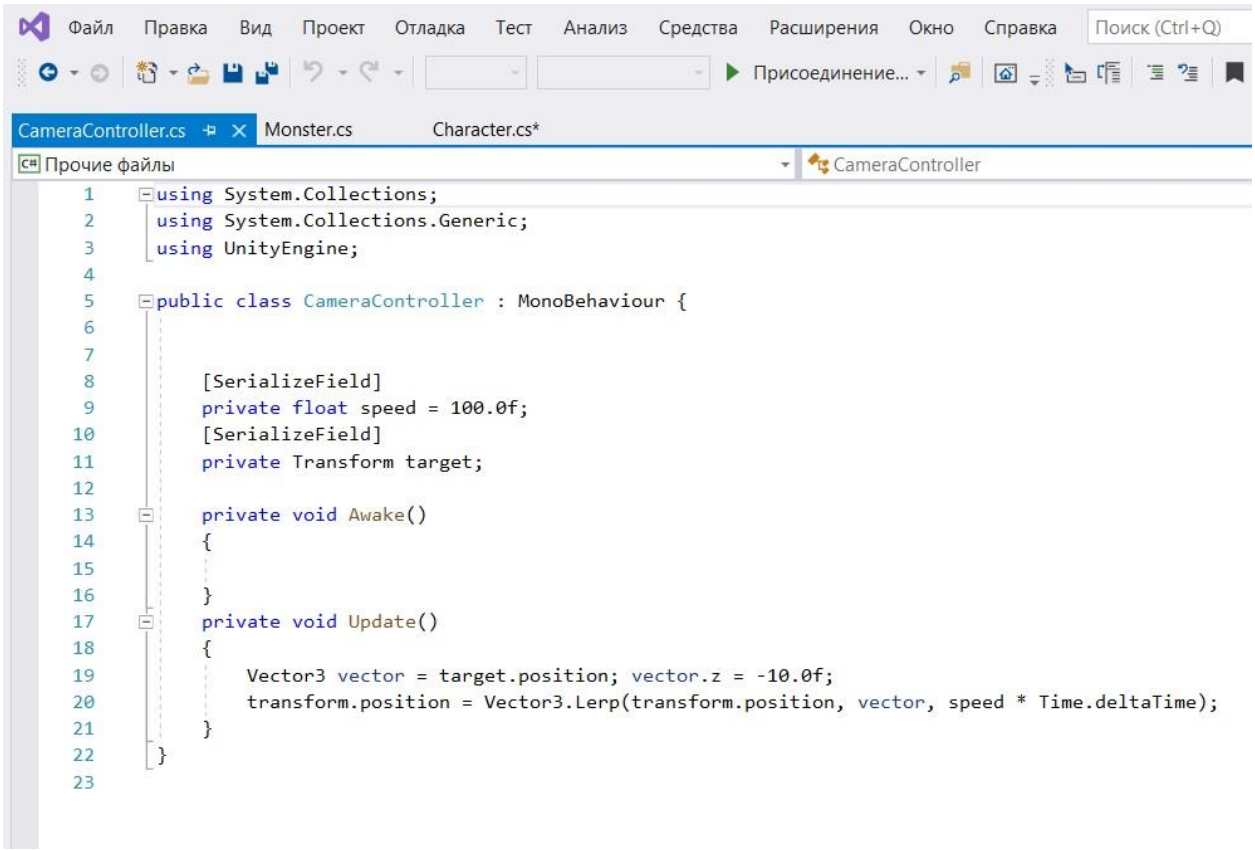

Рис. 22. Код камери

Далі розглянемо написання коду паузи. Після натискання на кнопку паузи гра зупиняє свій рух і викликається заздалегідь підготовлене меню.

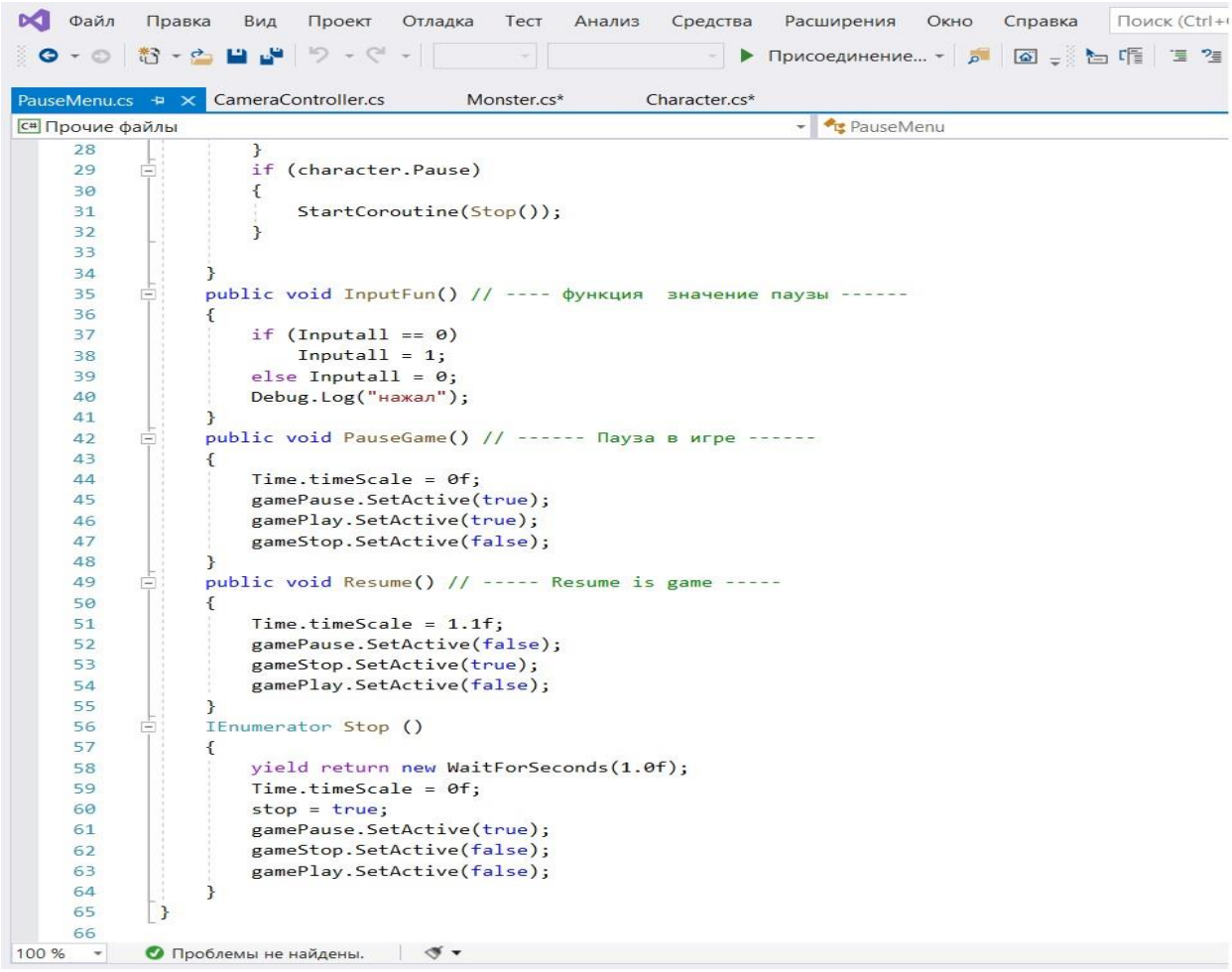

Рис. 23. Код паузи

3.2.4 Реалізація прогамного коду персонажу

Перш за все, варто розуміти, що даний проект не є повністю готовою до релізу комп'ютерною грою, і за своєю суттю є прототипом приблизно на рівні ранньої альфа-версії. Викликано це було тим, що створення якісного продукту займає колосально кількість часу. Так, наприклад, розробка невеликої гри для мобільних телефонів може займати більше року, при умови, що над проектом працює не більше трьох осіб. Збільшення кількості осіб може прискорити розробку, але не сильно.

На даний момент дуже складно визначити якими системними вимогами буде володіти проект в майбутньому, так як велика кількість елементів просто не реалізовані. На даний момент ми маємо лише елементи, які

відповідають за основну ігрову механіку, а значить, в майбутньому технічні характеристики для розроблюваної комп'ютерної гри можуть дуже сильно змінитися.

Зараз можна з упевненістю стверджувати лише те, що для запуску даного прототипу буде потрібно телефон на базі андроїд, що володіє системними характеристиками не нижче мінімально допустимих для коректної роботи ігрового движка Unity, для якого найбільш важливим аспектом є тільки відеокарта. Вона повинна підтримувати DirectX 9 з шейдерами не нижче версії 3.0. Це означає, що в даний момент створений прототип з великою ймовірністю піде на більшості телефонах.

ВИСНОВКИ

Під час аналізу доступних джерел було проведено дослідження поняття мобільна гра, під час якого була проведена класифікація ігор за 2 критеріями, але через порівняльної молодості ігрової індустрії, а так же того, що класифікація комп'ютерних ігор не була систематизована, скласти детальну класифікацію не вдалося. Додатково був складений алгоритм розробки відеоігор. Були проаналізовані популярні засоби розробки. В ході аналізу, було проведено їх порівняння і обрані найбільш актуальні види розробки для початківців розробників. Вибір пріоритетних засобів розробки проходив за двома критеріями: доступність і функціональність. При аналізі існуючих розробок, був проведено їх порівняння і виділені їхні переваги і недоліки. В ході аналізу стало ясно, що при розробці мобільної гри з простою ігровою механікою, варто звернути увагу на додаткові елементи гри, такі як сюжет і графічне оформлення. Це потрібно для того, що б утримати потенційного гравця і продовжити життєвий цикл розробки. Грунтуючись на все отриманої в ході дослідження інформації, було вирішено розробити прототип двовимірного платформера для одного гравця на ігровому движку Unity. Таке рішення було прийнято з кількох причин:

1) двовимірна графіка, на відміну від тривимірної легше в створенні;

2) по ігровій механіці, гра жанру платформер простіше реалізується;

3) ігровий движок Unity поширюється безкоштовно і дозволяє розробляти програми на мові програмування C #. Після вибору засобів розробки було розпочато вивчення Unity, і так само розробка самого проекту. В ході розробки був вивчений ігровий движок Unity і були придбані необхідні знання та вміння, а саме:

• створення сцен;

• створення анімацій;

• створення і написання Скриптів;

• настройка об'єктів;

• створення UI;

• компіляція проекту.

Освоєння середовища розробки Unity несе важливий характер, так як в сучасному світі індустрія розробки ігор все більше поширюється в нашому суспільстві. Ігри перестали бути лише предметом для розваг, і тепер використовуються і в інших областях, наприклад, в науці або в навчанні користувачів. Тому розвиток в даному напрямку можна вважати одним з найважливіших в сучасному суспільстві.

В ході реалізації проекту були виконані наступні завдання:

1) вивчені особливості і стан комп'ютерної індустрії;

2) обрані жанр, вид і платформа для комп'ютерної гри;

3) розроблено сценарій і концепція основних елементів;

4) вибрано і вивчено засіб реалізації;

5) підготовлені необхідні для гри анімації;

6) реалізований прототип гри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Платформер це [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Platform_game
2. Список платформеров [Електронний ресурс]. – Режим доступу: https://kanobu.ru/games/collections/platformery-na-android/
3. Unity [Електронний ресурс]. – Режим доступу: https://unity.com/ru
4. Visual Studio - docs.microsoft.com [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019
5. Створення графіки - http://www.piskelapp.com [Електронний ресурс]. – Режим доступу: http://www.piskelapp.com/
6. Microsoft docs [Електронний ресурс] – Режим доступу до ресурсу : https://docs.microsoft.com/uk-ua/dotnet/csharp/.
7. Unity docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/ru/current/Manual/index.html.
8. Шилдт Герберт - C# 4.0: полное руководство.:Пер. С англ – М.: ООО “И.Д Вильямс”, 2011 – 1056 с.: ил – Парал. Тит. Англ.
9. Створення платформеру [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/264611/
10. Ресурі до Unity [Електронний ресурс] – Режим доступу до ресурсу: https://assetstore.unity.com/
11. Оформлення диплоної роботи [Електронний ресурс] – Режим доступу до ресурсу: http://tef.kpi.ua/190

ДОДАТКИ

ДОДАТОК А1 Скрипт персонажу

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

public class Character : Unit
{
        [SerializeField]
        protected int lives = 5;
        public bool Pause = false;
        public Obstacle obstacle; // --- ссылка на obstacle -----
        public Color[] colorCh;
        bool IRecive = true;

        [SerializeField]
        Text text;

        [SerializeField]
        string scene;

        private float speed = 4.8f; // ------- Скорость бега главного персонажа -----
--
        private int Inputll; // -------- Значение которое проверяет бежать ему или
нет -------
        public float jumpForce = 8.0f; // ------- Сила прыжка главного персонажа -
-------

        private bool isGrounded = false; // -------- Проверка стоит ли главный
персонаж на змеле -------
        public Transform groundCheck; // ------- Определение позиции --------
        public float CheckRadius; // --------- радиус проверки есть ли под нами
блоки -------
        public LayerMask whatIsGrounded; // ------- номер маски --------

        private Rigidbody2D rigidbody;
        private Animator animator;
        private SpriteRenderer sprite;
```

```
        private CharState State // ------- Свойство выбора анимации и связки перечесление CharState и аниматора -------
        {
                get { return (CharState)animator.GetInteger("State"); }
                set { animator.SetInteger("State", (int)value); }
        }

        private void Awake()
        {
                // ------ Ссылки на компоненты --------
                rigidbody = GetComponent<Rigidbody2D>();
                animator = GetComponent<Animator>();
                sprite = GetComponentInChildren<SpriteRenderer>();
        }
        void Start() {

        }




        void Update()
        {
                text.text = lives.ToString();
                if (lives <= 0)
                {
                        State = CharState.Die;
                        Pause = true;
                }
        }
        void FixedUpdate()
        {

                if ((Inputll == 1 || Inputll == -1) && lives > 0) Run(); // -------- Проверка на бег --------
                else State = CharState.Idle; // -------- Если нет то проигрывать анимацию стойки --------
                CheckGround(); // ----- Вызов метода -----
        }

        public override void Recivedamage()
```

```
        {

            if (IRecive) {
                lives--;
                sprite.color = colorCh[0];
                StartCoroutine(Idamage());
            }
            IRecive = false;
            if (obstacle.ObsInput == 1)
            {
                rigidbody.velocity = Vector3.zero;
                rigidbody.AddForce(transform.up * 6.0f,
ForceMode2D.Impulse);
            }
            Debug.Log(lives);
        }
        IEnumerator Idamage()
        {
            yield return new WaitForSeconds(3.0f);
            sprite.color = colorCh[1];
            IRecive = true;
            Debug.Log("Вернуло");
        }
        public void Trampoline()
        {
            rigidbody.velocity = Vector3.zero;
            rigidbody.AddForce(transform.up * 13.0f, ForceMode2D.Impulse);
        }
        private void Run() // ------- Метод бега -------
        {
            Vector3 direction = transform.right * Inputll; // ------ Направление
бега -------
            transform.position = Vector3.MoveTowards(transform.position,
transform.position + direction, speed * Time.deltaTime);
            sprite.flipX = direction.x < 0.0f; // ------ Флип анимации ------
            State = CharState.Run;
        }

        public void Move(int InputAxis) //  ------- Метод передающий 1 или -1 ----
---
        {
            Inputll = InputAxis;
        }

        public void jump() // ------- Метод прыжка --------
```

```
        {
            if (isGrounded)
            {
                rigidbody.AddForce(transform.up * jumpForce,
ForceMode2D.Impulse);
            }
        }

        private void CheckGround() // ----- Метод проверки стоим мы на земле
или нет ------
        {
            isGrounded = Physics2D.OverlapCircle(groundCheck.position,
CheckRadius, whatIsGrounded);

            if (!isGrounded) State = CharState.Jump;
        }
        public void Restart()
        {
            SceneManager.LoadScene(scene);
        }


}
public enum CharState // ------- Перечесление анимаций -------
{
        Idle, Run, Jump, Die
}
```

ДОДАТОК А2 Скрипт паузи

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

public class PauseMenu : MonoBehaviour {

        public GameObject gamePause; // ------- Картинки паузы ------
        public GameObject gameStop; // ------- Картинки стопа ------
        public GameObject gamePlay; // ------- Картинки стопа ------
        int Inputall = 0; // ---- значение паузы ------
        public Character character; // ------ ссылка на обьект Character ------
        bool stop = false;
        private void Start()
        {
```

```
        gamePause.SetActive(false);
        gameStop.SetActive(true);
        gamePlay.SetActive(false);
}
void Update () {
        if (Inputall == 0 && !stop)
        {
                Resume();
        }
        else if (Inputall == 1 && !stop)
        {
                PauseGame();

        }
        if (character.Pause)
        {
                StartCoroutine(Stop());
        }

}
public void InputFun() // ---- функция  значение паузы ------
{
        if (Inputall == 0)
                Inputall = 1;
        else Inputall = 0;
        Debug.Log("нажал");
}
public void PauseGame() // ------ Пауза в игре ------
{
        Time.timeScale = 0f;
        gamePause.SetActive(true);
        gamePlay.SetActive(true);
        gameStop.SetActive(false);
}
public void Resume() // ----- Resume is game -----
{
        Time.timeScale = 1.1f;
        gamePause.SetActive(false);
        gameStop.SetActive(true);
        gamePlay.SetActive(false);
}
IEnumerator Stop ()
{
        yield return new WaitForSeconds(1.0f);
        Time.timeScale = 0f;
```

```
            stop = true;
            gamePause.SetActive(true);
            gameStop.SetActive(false);
            gamePlay.SetActive(false);
        }
}
```

ДОДАТОК А3 Скрипт монстрів

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

public class Monster : Unit {

        [SerializeField]
        private float speed = 3.5f;

        [SerializeField]
        private LayerMask whatIsGrounded;


        private SpriteRenderer sprite;
        private Vector3 direction;
        private void Start()
        {
            sprite = GetComponentInChildren<SpriteRenderer>();
            direction = transform.right;
        }
        private void Update()
        {
             Move();
        }
        private void Move()
        {
            Collider2D[] collider =
Physics2D.OverlapCircleAll(transform.position + transform.up * 0.5f +
transform.right * direction.x, 0.05f, whatIsGrounded);
            transform.position = Vector3.MoveTowards(transform.position,
transform.position + direction, speed * Time.deltaTime);
            if (collider.Length > 0)
            {
                sprite.flipX = direction.x > 0;
                direction *= -1.0f;
```

```
            }
        }
        private void OnTriggerEnter2D(Collider2D collider) // ----- урона
персонажу если до него дотронулся монстр -----
        {
            Unit unit = collider.GetComponent<Unit>();

            if (unit && unit is Character)
            {
                if (Mathf.Abs(unit.transform.position.x - transform.position.x)
< 0.5f)
                {

                    Recivedamage();
                }
                else unit.Recivedamage();
            }
        }
        /*
          private void OnTriggerEnter2D(Collider2D collider) // ----- урона
персонажу если до него дотронулся монстр -----
        {
            Character character = collider.GetComponent<Character>();
            Monster monster = collider.GetComponent<Monster>();
            if (character)
            {
                monster.Recivedamage();
                character.Recivedamage();

            }
        }
         */
}
```

ДОДАТОК А4 Скрипт гармати

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

public class CircleGun : MonoBehaviour {

    [SerializeField]
    private float speed = 5f;
```

```
        // Use this for initialization
        void Start()
        {

        }

        // Update is called once per frame
        void Update()
        {
                transform.position += (transform.right * -1)  * Time.deltaTime *
speed;
        }
        private void OnTriggerEnter2D(Collider2D collider) // ----- урона
персонажу если до него дотронулся шар -----
        {
                Character character = collider.GetComponent<Character>();

                if (character)
                {
                        character.Recivedamage();
                        Destroy(gameObject);
                }
        }
}
```