МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерної алгебри та дискретної математики

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему

«Кроссплатформенна гра під назвою "The last chance"»

«Cross-platform game called "The last chance"»

Виконав: студент денної форми навчання
спеціальності 123 – Комп'ютерна інженерія .

(шифр і назва напряму підготовки, спеціальності)

Руденко Юрій Андрійович

(прізвище, ім'я, по-батькові)

Керівник канд. техн. наук, доц. Левченко А.О

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент доктор техн. наук, проф. Гунченко Ю.О

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2021 р.

Завідувач кафедри

_____ _Ю.О. Гунченко

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «_ _» ____ 2021 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Н.Ф. Казакова_

(підпис) (прізвище, ініціали)

Одеса – 2021

АНОТАЦІЯ

Цей проект був створений для того щоб поліпшити у користувачів навички такі як стресостійкість, а так само розвинути дрібну моторику рук. Для розвитку дрібної моторики рук відповідає реакція і швидкодія гравця який взаємодіє з інтерфейса проект. За рахунок аркадного жанру проекту користувачеві весь час прийдеться взаємодіяти з інтерфейсом. Також завдяки складності ігрового процесу користувач буде в постійній напрузі що і буде сприяти на його психічний стан. У супроводі нагнітаючої музики користувачеві доведеться вчитися контролювати себе для того щоб досягти підсумкової мети.

Для створення даного проекту була підібрана платформа на якій буде збиратися весь проект. Гра є кроссплатформенною завдяки інструменту для розробки відеоігор і застосунків. Так само був продуманий зручний інтерфейс управління. Безліч елементів проекту були анімовані і вручну намальовані. Ігрова зона була продумана таким чином щоб гравцеві доводилося запам'ятати розташування видимих і невидимих елементів.

В результаті була створена гра яка запускається на двох платформах, а саме на персональному комп'ютері і на андроїд пристрої. Елементи управління і геймплей вимагатимуть реакції і концентрації. Підсумковий продукт має свій стиль, жанр і сюжет, який дозволить скласти конкуренцію на ринку.

Матеріалів по даній роботі має публікацію в "SWorldJournal".

SUMMARY

This project was created in order to improve user skills such as stress resistance, as well as develop fine motor skills of the hands. For the development of fine motor skills of the hands, the reaction and speed of the player who interacts with the project interface correspond. Due to the arcade genre of the project, the user will have to interact with the interface all the time. Also, due to the complexity of the gameplay, the user will be in constant stress, which will contribute to his mental state. Accompanied by pumping music, the user will have to learn to control himself in order to achieve the final goal.

To create this project, a platform was selected on which the entire project will be assembled. The game is cross-platform thanks to a tool for developing video games and applications. A user-friendly control interface was also designed. Many elements of the project were animated and hand-drawn. The game area was designed so that the player had to remember the location of visible and invisible elements.

As a result, a game was created that runs on two platforms, namely on a personal computer and on an Android device. Controls and gameplay require reaction and concentration. The final product has its own style, genre and plot, which will compete in the market.

Materials on this work have been published in "SWorldJournal".

АНОТАЦИЯ

Этот проект был создан для того чтобы улучшить у пользователей навыки такие как стрессоустойчивость, а так же развить мелкую моторику рук. Для развития мелкой моторики рук соответствует реакция и быстродействие игрока который взаимодействует с интерфейса проект. За счет аркадного жанра проекта пользователю все время придется взаимодействовать с интерфейсом. Также благодаря сложности игрового процесса пользователь будет в постоянном напряжении что и будет способствовать его психическое состояние. В сопровождении нагнетающей музыки пользователю придется учиться контролировать себя для того чтобы достичь итоговой цели.

Для создания данного проекта была подобрана платформа на которой будет собираться весь проект. Игра является кроссплатформенной благодаря инструмента для разработки видеоигр и приложений. Так же был продуман удобный интерфейс управления. Множество элементов проекта были анимированные и вручную нарисованы. Игровая зона была продумана таким образом, чтобы игроку приходилось запомнить расположение видимых и невидимых элементов.

В результате была создана игра которая запускается на двух платформах, а именно на персональном компьютере и на андроид устройства. Элементы управления и геймплей требовать реакции и концентрации. Итоговый продукт имеет свой стиль, жанр и сюжет, который позволит составить конкуренцию на рынке.

Материялы по даной работе имеет публикацию в “SWorldJournal”.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Коллайдер — определяют форму объекта для целей физических столкновений.

Інді-розробник — це фахівець, який займається самостійною розробкою ігор або входить до складу невеликої команди розробників, що випускає комп'ютерні та мобільні ігри без фінансової підтримки великих компаній.

Платформер – жанр відеоігор, ігровий процес в якому складається зі стрибків персонажа по різноманітних платформах (звідси і назва) та через перешкоди, збирання предметів, звичайно необхідних для завершення рівня.

ПЗ (Програмне Забезпечення) — програма або безліч програм

Скіннінг мешів — програмування анімації об'єкта

ВСТУП

Сучасні комп’ютерні ігри – яскравий приклад бурхливого розвитку інформаційних аудіовізуальних технологій XXI століття. Неймовірно популярні зараз комп’ютерні ігри в своєму розвитку пройшли довгий шлях від примітивних аркад до повноцінних віртуальних світів типу The Elder Scrolls V: Skyrim, Grand Theft Auto, Far Cry, Mass Effect, та інші, для повноцінного освоєння яких потрібно не один місяць «реального» часу. З кожним роком популярність комп’ютерних ігор зростає, все більше і більше людей грають в комп’ютерні ігри, які вже давно перестали бути тільки розвагами для дітей і підлітків. Сучасні комп’ютерні ігри вриваються в суміжні громадські та культурні сфери – мистецтво, освіту, етику, психологію, соціальні комунікації і навіть спорт (в світі давно проводяться повноцінні кіберспортивні чемпіонати з солідними бюджетами, як наприклад, турнір по Dota 2, The International 2016 – призовий фонд склав $20 770 460).

У прикладному контексті комп’ютерні ігри можуть знайти застосування в інноваційно-освітньому процесі, наприклад, в якості спеціальних ігрових навчальних програм, що використовуються як в ході лекцій, так і в ході заліків, екзаменаційних тестів. Інший продуктивний варіант – використання ігрових навчальних програм студентами під час профорієнтаційної практики.

Сучасні комп’ютерні ігри виявляють широкий спектр інноваційних можливостей, які можуть бути актуальні не тільки в світлі технологічної модернізації країни, але і в контексті перспективних соціально-гуманітарних досліджень.

Розробкою відеоігор може займатися як одна людина, так і фірма (колектив розробників). Часто комерційні ігри створюються командами розробників, найнятими однією фірмою. Фірми можуть спеціалізуватися на виробництві ігор для персональних комп’ютерів, ігрових приставок або планшетних комп’ютерів. Часто розробка фінансується інший, більш великою фірмою – видавцем. Фірма-видавець після закінчення розробки займається

розповсюдженням копій гри, бере на себе пов’язані з цим витрати. Буває, що фірми-видавці наймають команди розробників, або фірма-розробник самостійно (без участі видавців) поширює копії ігор, наприклад, засобами цифрової дистрибуції (інді-ігри).

До складу типової сучасної команди розробників зазвичай входять представники різних спеціалізацій:

- один або кілька продюсерів для спостереження за виробництвом;
- дизайнери;
- звукооператори:
- композитори;
- творці звукових ефектів;
- тестувальники.

Деякі члени команди можуть виконувати кілька функцій. Наприклад, продюсер може бути дизайнером або провідним програмістом. Багатофункціональність була звичайним явищем на початку епохи відеоігор, а зараз, при розробці професійних ігор, зустрічається все рідше і рідше.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Історія розвитку комп’ютерних ігор

1968 – була розроблена Magnavox Odyssey – перша комерційно успішна домашня ігрова консоль, була запатентована в 1968 в 1972 і вийшла на ринок. Odyssey була розроблена Ральфом Г. Баєром за допомогою інженерів Вільяма Харрісона і Вільяма розра. Було продано більше 350 000 Odyssey.

1972 – було розроблено ПОНГ – одну з перших аркадних відеоігор. Оригінальна гра була розроблена компанією Atari Incorporated (Atari) році. Аллан Алкорн створив Pong в якості тренування навичок, яку йому запропонував засновник Atari Нолан Бушнелл. На відміну від випущених раніше аркадних відеоігор Pong чекав неймовірний успіх і визнання гравців.

«Золотий вік» аркадних відеоігор – 1973-1986 – прийнято вважати піком популяризації та технічного розвитку ігор цього жанру. У той час величезні зусилля інженерів і розробників кидалися на те, щоб представити публіці все більш просунуті на ті часи гри. І немає чітких меж у цієї періоду, більшість джерел називають дати з 1970-х до середини 1980-х.

У 1985 в Америці ігровий ринок консолей отримав новий поштовх розвитку завдяки тому, що Nintendo випустила свою 8-ми бітову приставку Famicom, в Азії відому як Nintendo Entertainment System (NES).

16-ти бітна приставка була випущена Nintendo в 1990 році в Японії. SNES чекав світовий успіх, перетворивши її в одну з найбільш продаваних консолей в світі, не дивлячись на її пізній старт і жорстку конкуренцію з боку Північно-американської та Європейської приставки Sega.

Ігрова приставка PlayStation (офіційне скорочення PS) – ігрова приставка п’ятого покоління, розроблена компанією Sony Computer Entertainment під керівництвом Кена Кутараги. Реліз консолі відбувся 3 грудня 1994 року в Японії. Приставка виявилася дуже популярною, було продано більше 100 млн. приставок.

Портативні консолі мали легку вагу, вбудований екран, елементи управління і динаміки. Вони були меншого розміру ніж домашні консолі, що дозволяло людям брати їх з собою в дорогу в будь-який час.

В даний час ринок ігрових консолей ділять Sony, Microsoft і Nintendo. PS4 від Sony лідирує, на другому місці Xbox One від Microsoft і замикає гонку Wii U від Nintendo.

Три десятиріччя ринок відеоігор активно розвивається і сучасні технології відкривають нам фантастичні можливості, про які раніше можна було тільки мріяти. Зараз ринок відеоігор захоплює потихеньку віртуальна реальність, але всі пам'ятають з чого починалося індустрія. І платформери залишаться як просто Класика до якої завжди приємно повернутися.

1.2 Студії-розробники ігор

Naughty Dog – студія, яка була заснована в 1984 році. У той час вона ще називалася Джеймс Software, і засновники компанії працювали в своєму гаражі. Компанія була перейменована в 1989 році в Naughty Dog. Популярність компанії принесла перша частина гри Crash Bandicoot розробка якої почалася в 1994 році. У 2001 році студія була придбана Sony для розробки ексклюзивних ігор для консолей компанії. На даний момент Naughty Dog славиться серією популярних ігор Uncharted і постапокаліптичній драмою The Last of Us.

Capcom – компанія заснована в 1983 році. В той час зароджувалася ігрова індустрія і з’явилося багато справді легендарних компаній, таких, як Nintendo. Capcom зробила собі ім’я, коли вийшла ігрова приставка Ness. На рахунку компанії безліч пам’ятних олдскульних ігор, включаючи Megaman. Але найбільший успіх компанії принесла серія постапокаліптичних ігор Resident Evil, по якій було знято безліч фільмів.

Bethesda Softworks – студія, яка займає одне з провідних місць в світі по розробці рольових ігор і гонок. Ця компанія відома в першу чергу як

розробник всесвітньо відомої серії рольових ігор The Elder Scrolls, а так само постапокаліптичних шутерів Fallout.

Infinity World – американська компанія, заснована в 2002 році, яка створює гри для різних ігрових приставок і ПК. Найважливішим брендом компанії є серія ігор Call of Duty. У 2003 році компанія була придбана видавцем ігор Activision Кожна гра з серії знаменитої «чаклують» незмінно продається мільйонними тиражами.

Nintendo – компанія, заснована в 1983 році. Слово «легендарна» мало для опису цієї студії, яка провела відразу кілька революцій в історії ігрової індустрії. Нінтендо створили Супер Маріо, Легенди Зельди, Метроід і безліч інших брендів. Компанія по суті створила індустрію відеоігор в ранніх вісімдесятих. До того ж Нінтендо змінила спосіб взаємодії з іграми.

Blizzard – компанія, яка подарувала світу стратегії, в які грає весь світ, творці StarCraft і WarCraft. Компанія, що з'явилася в результаті злиття Vivendi Games з Activison в 1994 році. У тому ж році вийшла їх сама легендарна гра Варкрафт, яка принесла компанії світову популярність і вивела в лідери. Кожна гра цієї студії є бестселером, починаючи з самого першого Варкрафта. У 1996 році компанія вдало викупила студію Contra Games, яка розробляла не менше легендарну Diablo. А в 1998 Blizzard випустила СтарКрафт – гру, яка стала найбільш продаваною в той рік, отримавши дику популярність в Південній Кореї і по всьому світу. Сплески популярності Блізард були і в 2002 році з виходом третього WarCraft і, зрозуміло, в 2004, коли з'явилася одна з найпопулярніших MMORPG World of Warcraft.

Valve Corporation – компанія-розробник, заснована Гейб Ньюелом. Початок компанії Валве поклали екс-співробітники Майкрософт Гейб Ньюел і Майк Харінгтон в 1996 році. Купивши ліцензію движка Квейк, вони стали розробляти Half-Life, а до працівників сценарію був запрошений знаменитий письменник фантаст Марк Лейдлоу. Гру показали на виставці E3 в 1997 році, де вона викликала справжній фурор, такий же, як і через рік, коли остаточно вийшла. Після успіху Valve випустила кілька ігор і модифікацій, в тому числі,

знамениту Counter-Strike. У 2003 році була анонсована друга частина Халф Лайф, яка була поділена на кілька епізодів. Ну а в наш час компанія годується за рахунок CS:GO і Доти 2.

Electronic Arts – компанія, що розробляє ігри в різних жанрах, починаючи спортивними симуляторами і закінчуючи стратегіями, такими, як знаменита EA Games. Одна з найстаріших ігрових компаній, заснована Тріп Копкінсом в 1982 році. Майже цілком стартовий капітал був зібраний з його особистих накопичень. Спочатку EA була тільки компанією видавцем ігор, але вже в кінці 80-х вона почала підтримувати консольні проекти. В даний час під маркою EA вийшли багато спортивних симулятори, такі, як серія FIFA, NHL, а так же серії ігор Гаррі Поттер, Need for Speed і The Sims. Під логотипом цієї компанії виходило безліч ігор і франшиз.

RockStar Games – ті самі розробники, які радують нас одним з найпопулярніших екшнів. Компанія була створена в 1998 році як об’єднання відразу безлічі студій. Рокстар прославилася своєю основною франшизою, яка пов’язана з компанією з самого виходу – GTA – гра в жанрі екшн з цікавим сюжетом, де ви, граючи за бандита, проходите всю кар’єрну злочинну сходинку від автовикрадача до легендарного мафіозі. Дана серія ігор виходить з 1997 року і до сих пір з кожним разом збирає величезні доходи і все нових фанатів

Ubisoft – творці Асасинів, Фаркраев і шостих Героїв – європейська компанія Юбісофт, чиї офіси знаходяться в більш ніж 20 країнах, а головний штаб у Франції. Історія компанії бере початок від п’яти братів, які заснували Ubisoft у Франції в 1986 році. У 1994 вони відкрили офіси в Канаді і розробили Реймонд – гру, чия франшіза залишається у Ubisoft досі. У 2000 році, купуючи компанію з правом видання ігор за мотивами книг Тома Кленсі, Убісофт починає розробляти нові ігри. А в 2011 році компанія створила дочірню студію, яка збирається знімати фільми по мотивам ігор.

1.3 Опис предметної області

Гра має назву “The last chance”, що в перекладі означає “ останній шанс”. Це 2D-гра в жанрі Hardcore Platformer, тобто платформер з підвищеним рівнем складності.

1.3.1 Сюжет гри

Перед вами стоїть персонаж уособлює наше життя і шлях в якій належить пройти мало не кожному з нас. Персонажа звуть "Furry", це миле і беззахисне створіння, яке всього лише хоче пройти до кінця і залишитися в живих. Весь його шлях це найскладніші і оманливі пастки і мета всього – вижити. Ми так само намагаємося вижити в цьому неосяжному і важкому світі, в якому нас чекають обман, зради, труднощі і байдужості всіх хто міг би допомогти. Кожен день і кожен раз чекаємо якогось дива або щастя, але все так само засмучуємося, що все що сталося хорошого відразу перекриває суцільний негатив нашому житті. Тільки найсильніший і терплячий зможе дійти до кінця ні дивлячись ні на що.

1.3.2 Жанр гри

Гру буде розроблено у жанрі платформер. Платформер – жанр відеоігор, ігровий процес в якому складається зі стрибків персонажа по різноманітних платформах (звідси і назва) та через перешкоди, збирання предметів, звичайно необхідних для завершення рівня.

Платформери розрізняються за кількома ознаками:

– За глибиною переміщення – в тривимірних платформерах персонаж може переміщатися в трьох вимірах, в двомірних – тільки в двох, а в деяких

іграх герой може поглиблюватися і наближатися, перемикаючись між двома-трьома лініями глибини – так зване 2.5D.

– По виду графіки – двомірні або тривимірні. Сучасні двомірні платформери (де персонаж переміщається тільки в двох вимірах) часто використовують тривимірну графіку. Бувають також тривимірні платформери (де персонаж переміщається в трьох вимірах) з двомірної графікою, зазвичай в ізометричної перспективі, як, наприклад, «Sonic 3D Blast».

– За свободою переміщення – лінійні і вільні. В лінійних персонаж повинен пройти певний шлях від початку до кінця рівня. Якщо це шлях справа наліво і зліва направо, то гра називається сайд-скроллер (side-scroller, «з горизонтальним рухом екрану»). У вільних платформер персонаж може переміщатися за рівнем без обмежень, і для перемоги часто потрібно відвідати різні місця в будь-якій послідовності.

Першими платформерами були: Frogs (1978) де були стрибки, але не було платформ і Space Panic (1980) де були платформи, але не було стрибків. В Donkey Kong (1981) вперше з'явився скроллінг. Першим справжнім платформером стала гра Pitfall, але справжня слава прийшла до жанру після виходу Super Mario Bros. 1985 року. З того часу двовимірні платформери продовжують випускати до цих пір на різних ігрових платформах.[1]

2 РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ

2.1 Технічне завдання

2.1.1 Загальні відомості

Повне найменування системи: «Розробка комп’ютерної гри на движку Unity 5».

Коротке найменування системи: Програма, Програмне забезпечення (ПЗ), Програмний комплекс, Система.

Порядок оформлення та передачі замовнику результатів робіт по розробці програмного забезпечення: програмне забезпечення передається у вигляді функціонуючого комплексу на базі засобів обчислювальної техніки Замовника у визначені строки. Приймання програмного забезпечення здійснюється комісією у складі уповноважених осіб Замовника. Робочий проект, вихідні коди та супутня документація повинна бути збережена на носії, наприклад лазерному компакт диску.

2.1.2 Призначення і мета створення системи

Призначення системи: дана розробка є дипломним проектом. Програмний комплекс призначений для створення комп’ютерної гри на движку Unity.

Мета створення системи:

Розробка виконується з метою:

– Створити гру для перевірки своєї реакції, та витримки.

2.1.3 Характеристика об’єкту автоматизації

Розробка ведеться на підставі завдання на дипломний проект та може бути застосована в ігровій індустрії, тільки у безкоштовному вигляді.

2.1.4 Вимоги до системи

Вимоги до функціональних характеристик

Програмний продукт повинен володіти наступними функціональними характеристиками:

– сюжет;

– переміщення головного героя;

– головне меню;

Організація вхідних і вихідних даних

У процесі роботи програми вхідною інформацією для програми повинні бути:

– панель переміщення головного персонажу;

– дві панелі на екрані для стрибка та усилювач стрибка;

– основні меню для старту та виходу з гри;

Вимоги до надійності

Надійне (стійке) функціонування програмного комплексу має бути забезпечене шляхом:

– контролю коректності та повноти вхідних даних – всі дані, що вводяться користувачем, перевіряються на формальну коректність;

– ведення протоколів (архівів) дій користувачів;

– відновлення після відмови – в разі виникнення програмного збою система повинна відновлювати роботу з останнього зафіксованого стабільного стану;

– надання можливості періодичного створення резервних копій інформаційної бази (періодичність встановлюється адміністратором системи).

Контроль вхідної і вихідної інформації

Програма повинна контролювати вибір користувачем пункту меню «Вихід».

Вимоги до технічного забезпечення

Програмний комплекс розрахований на функціонування при такому наборі технічних засобів – мінімальна апаратна конфігурація смартфона:

– система на базі андроід не нижче версії 4.2;

– вільного місця не меньше ніж 40 мб.

Вимоги до середовищ програмування

Розробка програми повинна вестися мовою програмування C# у середовищі розробки Microsoft Visual Studio – Unity, приєднаний компілятор до движку Unity. Вибір інших середовищ розробки недоцільний.

Вимоги до програмних засобів, використовуваних програмою

Для роботи програми необхідна операційна система WINDOWS 7 і більш пізня.

2.1.5 Склад і зміст робіт по створенню системи

Розробка системи виконується в наступній послідовності, представленій у табл. 1.1.

Таблиця 1.1 – Етапи робіт по створенню системи

Етап	Зміст робіт	Результат
1	Розробка основних текстур та стилю гри	Намальована більша кількість текстур та зроблена анімація
2	Розробка технічного проекту	Проектні рішення по системі і її частинам. Узгодження проекту.
3	Реалізація технічного проекту (робочий проект)	Реалізація проектних рішень. Заповнення Пояснювальної Записки.
4	Тестування	Результати проведення тестів. Виправлення недоліків на стадії робочого проекту.
5	Здача-приймання дипломного проекту	Оформлення документації. Передача проекту Замовнику.

2.1.6 Порядок контролю і приймання системи

Прийом проекту виконується шляхом проведення приймальних випробувань. Приймальні випробування виконуються приймальною комісією у складі уповноважених осіб Замовника, здатних засвідчити факт відповідності створеного програмного продукту всім пунктам технічного завдання.

Мета приймальних випробувань полягає у підтвердженні відповідності проведених робіт затвердженому Технічному завданню.

Види, склад та об’єм випробувань визначаються програмою та методикою випробувань.

2.1.7 Вимоги до документування

Виконавець повинен надати Замовнику звіти по створенню програмного комплексу. Обов’язковим елементом звіту є Пояснювальна Записка, яка включає: загальний розділ, технічне завдання, технічний проект і реалізацію технічного проекту (робочий проект), охорону праці, інформацію про технічні характеристики продукту (архітектура, застосовані технології), опис функцій, тестування, контрольний приклад роботи та інші дані, інформація про які необхідна для обслуговування і використання продукту Замовником.

Документація надається українською мовою на електронному та паперовому носіях.

2.2 Обґрунтування вибору середовища розробки

Гра The last chance розроблена на ігровому движку версіі Unity 2017.3.1f1. Було обрано саме цей движок, бо він безкоштовний, багатофункціональний і простий у навчанні.

Unity – багатоплатформовий інструмент для розробки та тривімирних додатків та ігор, що працює на операційних системах Windows та OS X. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360. Є можливість створювати інтернет-додатки за допомогою спеціального модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL.

Технічні характеристики:

– сценарії на C#, JavaScript та Boo;

– ігровий двигун, повністю пов’язаний із середовищем розробки, що дозволяє випробовувати гру прямо в редакторі;

– робота з ресурсами можлива через звичайний Drag&Drop.

– система успадкування об’єктів;

– підтримка імпортування великої кількості форматів файлів;

– вбудований генератор ландшафтів;

– вбудована підтримка мережі;

існує рішення для спільної розробки – Asset Server. Також можна використовувати зручний для користувача спосіб контролю версій. Наприклад, SVN або Source Gear.

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

Проект в Unity поділяється на сцени (рівні) – окремі файли, що містять свої ігрові світи зі своїм набором об’єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об’єкти (моделі), так і порожні ігрові об’єкти – тобто ті, які не мають моделі. Об’єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у них є назва (в Unity допускається наявність двох і більше об’єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов’язково присутній компонент Transform – він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об’єктів з видимою геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель видимою.

Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об’єктів, дочірні об’єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об’єкта. Скрипти в редакторі прикріплюються до об’єктів у вигляді окремих компонентів.

При імпорті текстури в двигуні можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на

модель текстуру прикріпити не можна – буде створено матеріал, з яким буде призначений шейдер і тільки потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Графічний рушій використовує DirectX (Windows), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), та спеціальне власне API для Wii. Також підтримуються bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), динамічні тіні з використанням shadow maps, render-to-texture та повноекранні ефекти post-processing.

Unity підтримує файли 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks та Allegorithmic Substance. В ігровий проект Unity можна імпортувати об’єкти цих програм та робити налаштовування за допомогою графічного інтерфейсу.

Для написання шейдерів використовується ShaderLab, що підтримує шейдерні програми написані на GLSL або Cg. Шейдер може включати декілька варіантів реалізації, що дозволяє Unity визначати найкращий варіант для конкретної відеокарти. Unity також має вбудовану підтримку фізичного рушія Nvidia PhysX (колишнього Ageia), підтримку симуляції одягу в системі реального часу на довільній та прив’язаній полігональній сітці (починаючи з Unity 3.0), підтримку системи ray casts та шарів зіткнення.

Скриптова система ігрового рушія зроблена на Mono – вільний відкритий проект з реалізації .NET Framework. Програмісти можуть використовувати UnityScript (власна скриптова мова, подібна до JavaScript та ECMAScript), C# або Boo (мова програмування, подібна до Python). Починаючи з версії 3.0, до Unity входить перероблена версія MonoDevelop для зневадження скриптів.

Із виходом версії 5.2 передбачається вбудована можливість редагувати скрипти у середовищі Visual Studio

В Unity включено систему контролю версій Unity Asset Server для ігрових об’єктів та скриптів. Система використовує PostgreSQL, роботу зі звуком, побудовану на основі бібліотеки FMOD (з можливістю програвати Ogg Vorbis аудіофайли), відеопрогравач із кодеком Theora, рушій для побудови ландшафтів рослинності, вбудовану систему карт освітлення (Beast), мережу для мультиплеєру (RakNet) та вбудовані навігаційні меші для пошуку шляху. [5]

2.3 Обґрунтування вибору мови програмування

В якості мови програмування в Unity 5 присутні такі мови: C# та JavaScript.

У грі The last chance скрипти будуть написані на мові C#, бо ця мова зараз дуже актуальна та позволяє раціонально створювати різноманітні інтернет-додатки. Також C# інтегрувала в собі переваги мови Java і C++, що й обумовлює популярність даної мови серед розробників. При цьому в об’єднаній мові виключені деякі суперечливі директиви, макроси, скасовані глобальні змінні.

C# – об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# є дуже близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив задачу роззосереджених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі). Для розосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності і була ліцензована також компанією Microsoft. Але з плином часу Sun почала винуватити Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і тому з'ясування стосунків набуло статусу судового процесу. Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від позаліцензійного використання Java.

У цій ситуації в Microsoft вирішили, користуючись своєю вагою на ринку, створити свій власний аналог Java, мови, в якій корпорація стане повновладним господарем. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов

програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.[4]

3 РЕАЛІЗАЦІЯ ТЕХНІЧНОГО ПРОЕКТУ

3.1 Розробка скриптів

Скрипт – це програма або програмний файл сценарій, які автоматизують деяку задачу, яку користувач робив би вручну, використовуючи інтерфейс програми. Скрипти пишуться на скриптових мовах, які відрізняються за своїм синтаксисом, сферами застосування та можливостями. До скриптових мов належать: AngelScript, Perl, Python, PHP, JavaScript, JScript та інші.

Сфера застосування скриптів величезна. Наприклад:

– можливість звертатися до баз даних;

– seo-скрипти, що допомагають просувати сайти, використовуючи спеціальні програми автоматизації браузера, наприклад XHE;

– лічильники відвідування, які допомагають спостерігати статистику відвідувань;

– можливість створювати записи в гостьових книгах;

– можливість залишати коментарі до вподобаним статей;

– на скриптах засновані всі cms і форуми;

– динамічне відображення веб-сайту;

– організація зміни частини сайту без перевантаження всієї сторінки тощо.

Поведінка ігрових об’єктів в Unity контролюється за допомогою компонентів (Components), які приєднуються до них. Незважаючи на те, що вбудовані компоненти Unity можуть бути дуже різнобічними, їх недостатньо, щоб реалізувати усі власні особливості геймплея. Unity дозволяє створювати свої компоненти, використовуючи скрипти. Вони дозволяють активувати ігрові події, змінювати параметри компонентів, і відповідати на введення користувача яким завгодно способом.

Unity підтримує три мови програмування:

– C#, стандартна в галузі мова, подібна до Java або C++;

– UnityScript, мова, розроблена спеціально для використання в Unity за зразком JavaScript;

– Boo, мова програмування, яка є діалектом Python.

Також у Unity можуть бути використані багато інших мов сімейства .NET.

Приведено скрипт, за допомогою якого, «персонаж» рухається та стрибає по ігровому полю, написаний на мові C#.

```
using UnityEngine;
using System.Collections;
using UnityStandardAssets.CrossPlatformInput;
public class PlayerController_m : MonoBehaviour {
    public float maxSpeed = 6f; public float jumpForce =
10f;
    public Transform groundCheck; public LayerMask
whatIsGround;
    public float verticalSpeed = 20; public bool
lookingRight = true;
    bool doubleJump = false; public GameObject Boost;
    public GameObject Cloud; private Rigidbody2D rb2d;
    private Animator anim; public static bool isGrounded =
false;

    void Start () {
        rb2d = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        Cloud = GameObject.Find("Cloud");
    }
    void OnCollisionEnter2D(Collision2D collision2D) {
        if (collision2D.relativeVelocity.magnitude > 20){
Boost = Instantiate(Resources.Load("Prefabs/Cloud"),
transform.position, transform.rotation) as GameObject;
        }}
void Update () {
if (CrossPlatformInputManager.GetButtonDown ("Jump") &&
(isGrounded || !doubleJump))
        {
            rb2d.AddForce(new Vector2(0,jumpForce));
            if (!doubleJump && !isGrounded)
            { doubleJump = true;
Boost = Instantiate(Resources.Load("Prefabs/Cloud"),
transform.position, transform.rotation) as GameObject;
```

```
                    }
                }
                if
(CrossPlatformInputManager.GetButtonDown("Boost") &&
!isGrounded)
                { rb2d.AddForce(new Vector2(0,-jumpForce));
                    Boost =
Instantiate(Resources.Load("Prefabs/Cloud"), transform.position,
transform.rotation) as GameObject;
                }
            }
            void FixedUpdate()
            {
                if (isGrounded)
                    doubleJump = false;
                float hor = CrossPlatformInputManager.GetAxis
("Horizontal");
                anim.SetFloat ("Speed", Mathf.Abs (hor));
                rb2d.velocity = new Vector2 (hor * maxSpeed,
rb2d.velocity.y);
                isGrounded = Physics2D.OverlapCircle
(groundCheck.position, 0.15F,
     whatIsGround); anim.SetBool ("IsGrounded", isGrounded);
                if ((hor > 0 && !lookingRight)||(hor < 0 &&
lookingRight))
                    Flip ();
                anim.SetFloat ("vSpeed",
GetComponent<Rigidbody2D>().velocity.y);
            }
          public void Flip()
            {    lookingRight = !lookingRight;
                Vector3 myScale = transform.localScale;
                myScale.x *= -1; transform.localScale = myScale; }
     }
```

В цьому коді ми задаємо панелями на екрані для направління руху персонажа або стрибку, тобто при натисканні на екран у певній зоні персонаж починає рухатися вліво, вправо по осям координат та стрибає при натисканні на інші дві панелі. Також описується швидкість пересування самого персонажа, силу його стрибка і поява анімації під назвою "Cloud".

На рисунку 3.1 представлено меню скрипта та параметри персонажа в самому редакторі Unity.

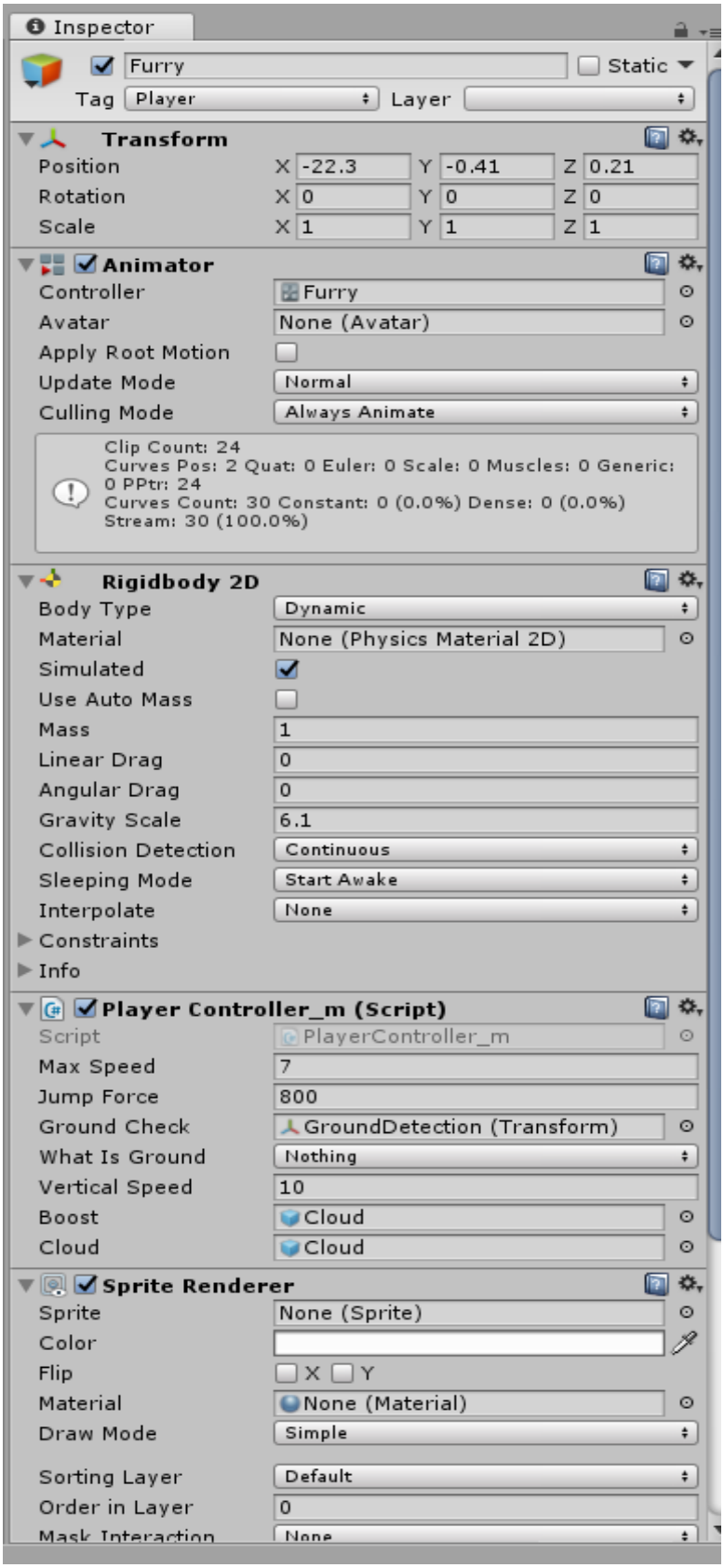

Рисунок 3.1 – Меню скрипта

Скрипт під назвою “Floor” відповідає тригером для “Furry”:

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

public class Floor : MonoBehaviour {
     // Use this for initialization
     void Start () {
     }
     // Update is called once per frame
     void Update () {
     }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.name == "Furry")
        {
            PlayerController_m.isGrounded = true;
        };
    }
}
```

Ця частина коду відповідає тригером для виявлення персонажем краю землі. Без цього тригера персонаж при виконання стрибка не бачить краю заданого колайдера.

Скрипт під назвою “LevelManeger” та “Spawn” виконують важливу функцію для взаємодії головного персонажу та перепон які йому зустрічаються:

```
“LevelManeger”
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class LevelManger : MonoBehaviour {
    public GameObject currentspawn;
    private PlayerController_m player;
     void Start () {
        //Обнаружения всех частей кода глав персонажа
        player = FindObjectOfType<PlayerController_m>();
     }
     void Update () {
```

```
        }
        public void RespawnPlayer() {
            Debug.Log("Player Respawn");
            player.transform.position =
currentspawn.transform.position;
        }
    }
    “Spawn”
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    public class spawn : MonoBehaviour {
        public LevelManger LevelManger;
        private void Start()
        {
            LevelManger = FindObjectOfType<LevelManger>();
        }
        void Update()
        {
        }
        private void OnTriggerEnter2D(Collider2D other)
        {
            if (other.name == "Player")
            {
                LevelManger.currentspawn = gameObject;
            };
        }
    }
```

Ці частини коду відповідають за виявлення персонажа і при виконання функкій коду "Trapes" переміщують об'єкт в зазначену точку званої "respawn".

Точка "Respawn" і "LevelManeger" персонажа зображується на рисунку 3.2.

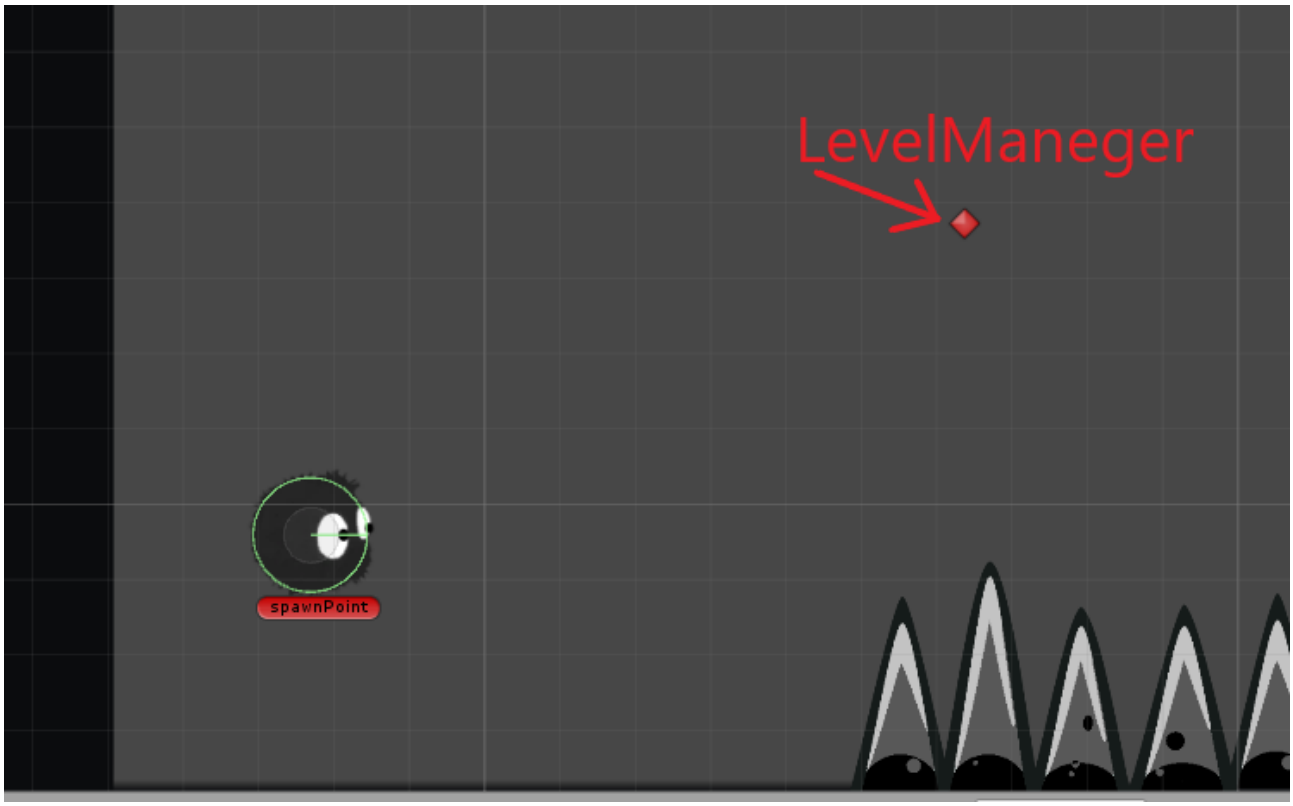

Рисунок 3.2 – Зображення "Respawn" і "LevelManeger"

Скрипт “Trapes” тісно взаємодіє з “LevelManeger” та “Spawn”, виконоючи функцію тригера яка викликає функції з вище сказаних скриптів:

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

public class Trapes : MonoBehaviour
{
    public LevelManger LevelManger;

    private void Start()
    {
        LevelManger = FindObjectOfType<LevelManger>();
    }
    private void Update()
    {

    }
    private void OnTriggerEnter2D(Collider2D other)
```

```
    {
        if (other.name == "Furry") {
            LevelManger.RespawnPlayer();
        };

    }
}
```

Основне завдання цього скрипта, це при досягненні коллайдера персонажа до коллайдер пасток, персонаж переміщується на "spawn".

Скрипт “ Dvigplat” виконує просту функцію пересування нерухомого об'єкта:

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

public class Dvigplat : MonoBehaviour {

    public GameObject platform;
    public float moveSpeed;
    public Transform currentPoint;
    public Transform[] points;
    public int pointSelection;
    // Use this for initialization
    void Start () {
        currentPoint = points[pointSelection];
     }

     // Update is called once per frame
     void Update () {
platform.transform.position =
Vector3.MoveTowards(platform.transform.position,
currentPoint.position, Time.deltaTime * moveSpeed);
        if (platform.transform.position==
currentPoint.position) {
            pointSelection++;
            if (pointSelection==points.Length) {
                pointSelection = 0;
            };
            currentPoint = points[pointSelection];
        };
     }
}
```

В скрипті описано переміщення об'єкта з однієї позиції на іншу завдяки стартовою і кінцевій точці. За цим точкам вказуються траєкторії руху, їх може бути як 2 так і більше, але при виставлені 0 або 1 рух може не відбутися. Відбудеться лише в разі якщо сам об'єкт перемістити в стартову позицію на відстані від 1 точки, тільки в такому випадку об'єкт перемістити до неї. Приклад наведено на рисунку 3.3:

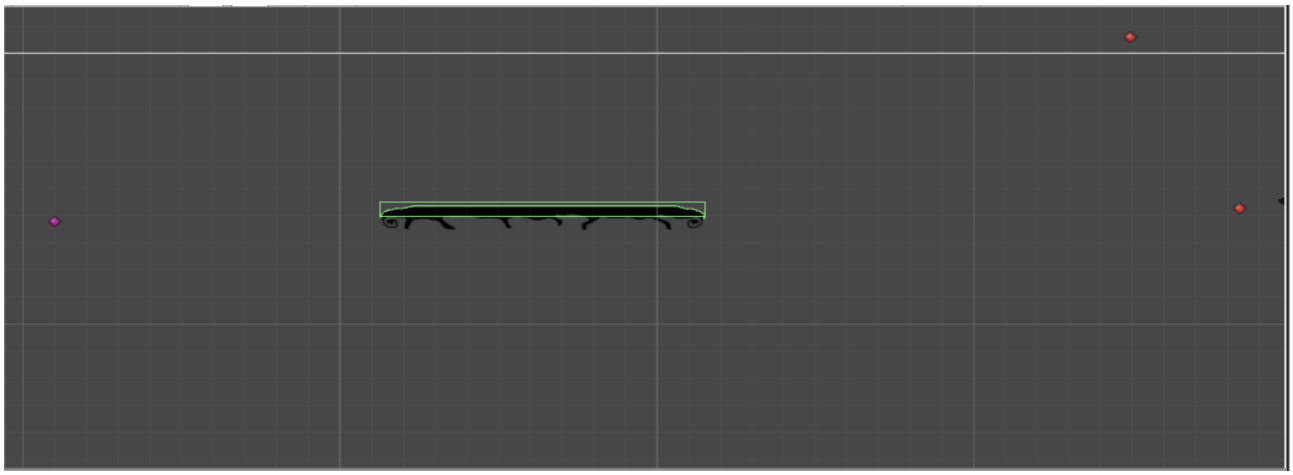

Рисунок 3.3 – Зображення точок по яким повинна рухатись платформа

3.2 Структура проекту

UML-діаграма проекту представлена на рис. 3.4.

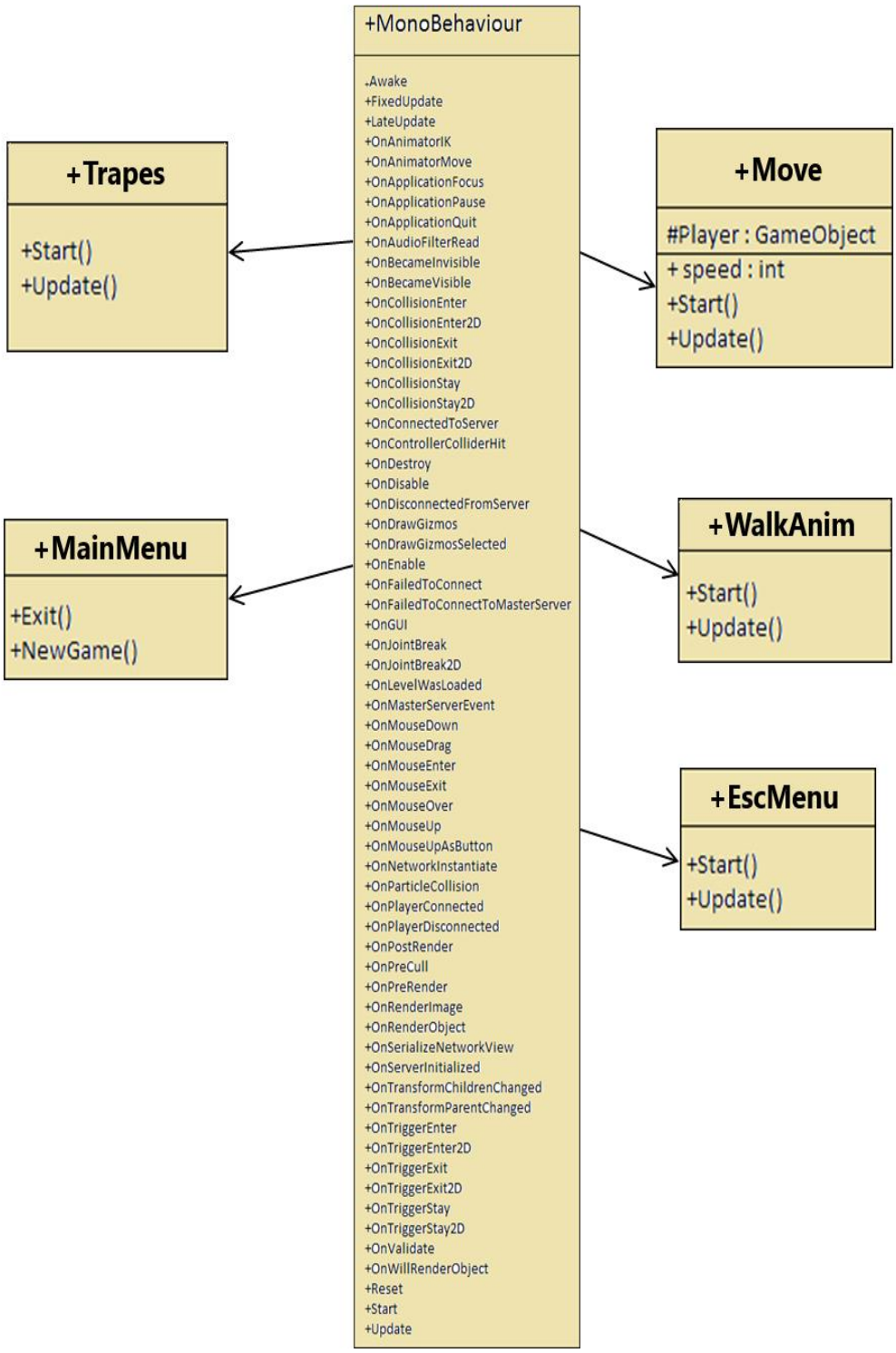

Рисунок 3.4 – UML-діаграма проекту

3.3 Створення інтерфейсу

Сучасні ігри та програми часто потребують підтримки широкого спектра різних дозволів екрану і, особливо в даній ситуації, потребують інтерфейси цих ігор. Система створення інтерфейсів в Unity забезпечена рядом різних інструментів для реалізації цих можливостей, які також можна комбінувати між собою масою різних способів.

Інтерфейс користувача (UI – англ. User interface) – інтерфейс, що забезпечує передачу інформації між користувачем-людиною і програмно-апаратними компонентами комп'ютерної системи.

Під сукупністю засобів і методів інтерфейсу користувача маються на увазі:

– засоби виведення інформації з пристрою до користувача – весь доступний діапазон впливів на організм людини (зорових, слухових, тактильних, нюхових і т. д.): екрани (дисплеї, проектори) і лампочки, динаміки, зумери і сирени, вібромотор тощо;

– засоби введення інформації / команд користувачем в пристрій – безліч всіляких пристроїв для контролю стану людини: кнопки, перемикачі, потенціометри, датчики положення і руху, сервоприводи, жести особою і руками, навіть знімання мозкової активності користувача. За наявністю тих чи інших засобів введення, інтерфейси поділяються на типи: жестовий, голосовий, брейн та інші. Можливі змішані варіанти. Ці засоби повинні бути необхідними і достатніми, бути зручними і практичними, розташованими і скомпонованими розумно і зрозуміло, відповідати фізіології людини, не повинні призводити до негативних наслідків для організму користувача (все це входить в поняття ергономіки).

– методи – набір правил, закладених розробником пристрою, згідно з якими сукупність дій користувача повинна привести до необхідної реакції пристрою і виконання необхідного завдання – т. зв. логічний інтерфейс.

Правила ці повинні бути досить ясні для розуміння, природні і легкі для запам’ятовування (все це входить в поняття юзабіліті).

Збільшення в пристрої (при рівній функціональності) засобів вводу-виводу дає спрощення побудови методів управління і спрощення правил користування, але зате призводить до складності сприйняття інформації користувачем – інтерфейс стає перевантаженим. І навпаки – зменшення засобів відображення і контролю призводить до ускладнення правил управління – кожен елемент несе на собі занадто багато функцій. Тому проектувальники інтерфейсів намагаються прийняти компромісне рішення між цими двома крайностями у кожному окремому випадку.

Одним із основних напрямків досліджень в області забезпечення безпеки користувальницьких інтерфейсів, і, зокрема, візуальних інтерфейсів користувача є розробка моделей інформаційної безпеки за умови комплексного обліку інформаційних, функціональних, психофізіологічних і екологічних аспектів безпеки. Це пов’язано, перш за все, з включенням інформаційного фактору до складу факторів середовища систем людина-комп’ютер і інформаційним характером майже всього, що відбуваються в області поширення ІТ-процесів. До областей проблематики захисту інформації в системі людина-комп’ютер (СЧК) відносять такі загрози, як:

– спотворення сприймається користувачем інформації за рахунок її зашумлення джерелами середовища на робочому місці користувача;

– втрата або спотворення сприймається користувачем інформації через фізичну, семантичної або синтаксичної неузгодженості її уявлення користувачеві;

– спотворення уявлень користувача про реальний стан об’єкта управління за рахунок прихованих інформаційних впливів і неадекватне прийняття ним рішень в процесі вирішення завдань в рамках СЧК.

3.4 Інтерфейс керування та меню

У грі The last chance в грі зроблена своя система управління яка має нестандартний спосіб керування. Для цього треба було створювати вручну інтерфейс який відрізняється повністю від стандартів на рис. 3.5 та 3.6 зобаржен інтерфейс керування персонажем:

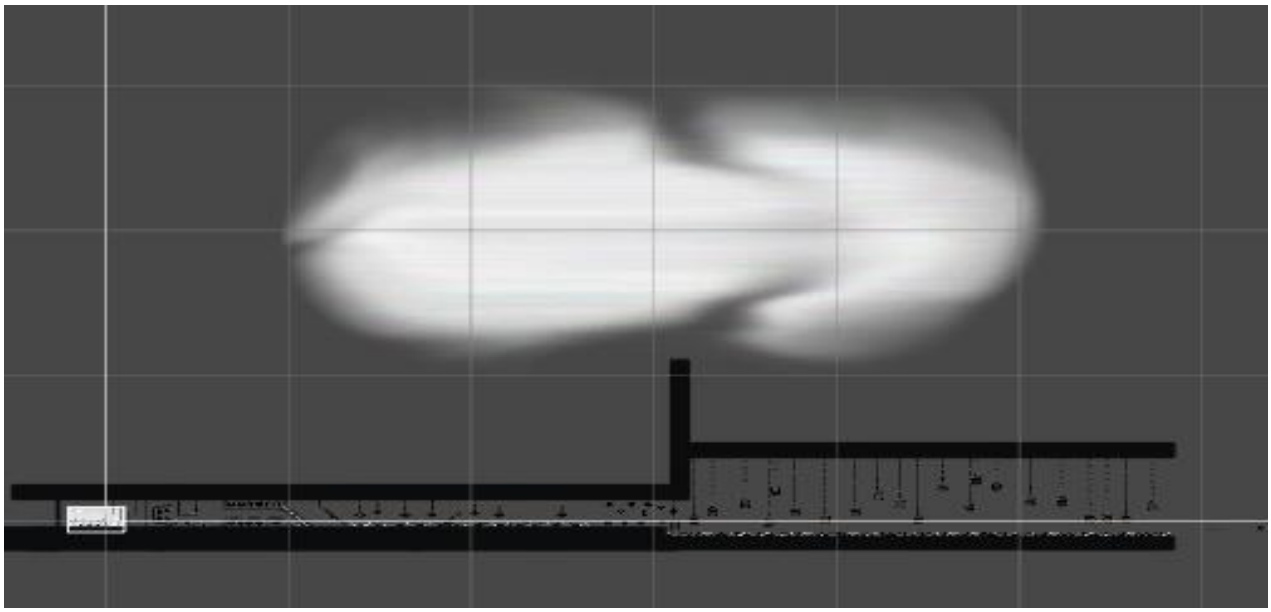

Рисунок 3.5 – Джойстик для переміщення

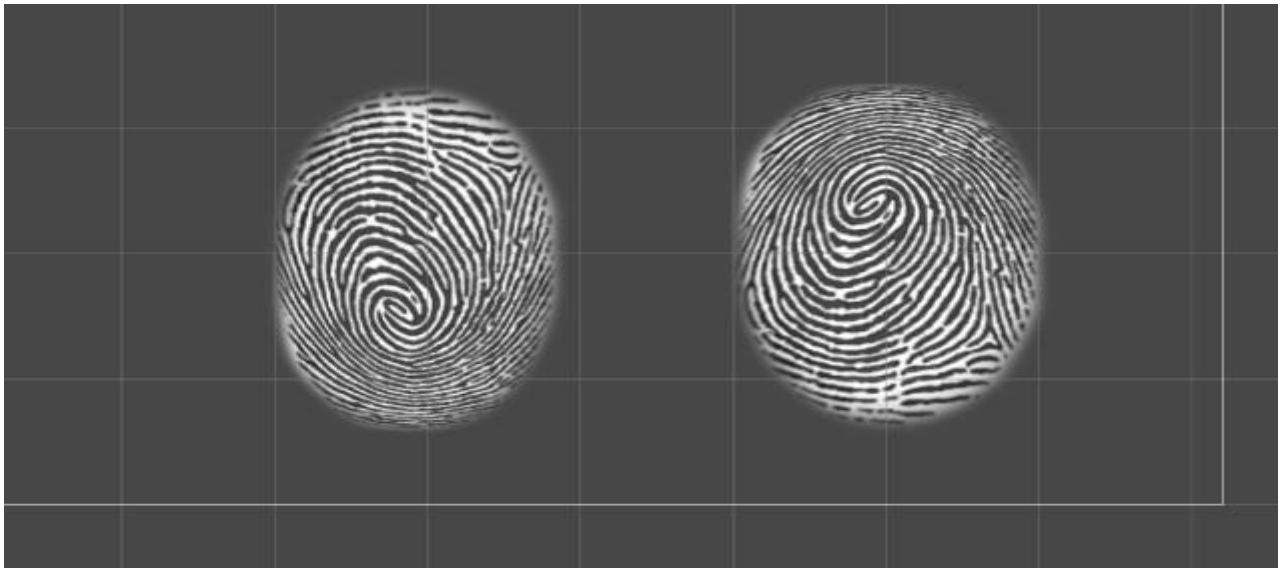

Рисунок 3.6 – Кнопки для стрибку та посилювачу стрибку

Так само було створено стартове меню використовують функції і оформленні відрізняючихся від стандарту на на рис. 3.7 та 3.8 зобаржено інтерфейс керування персонажем:

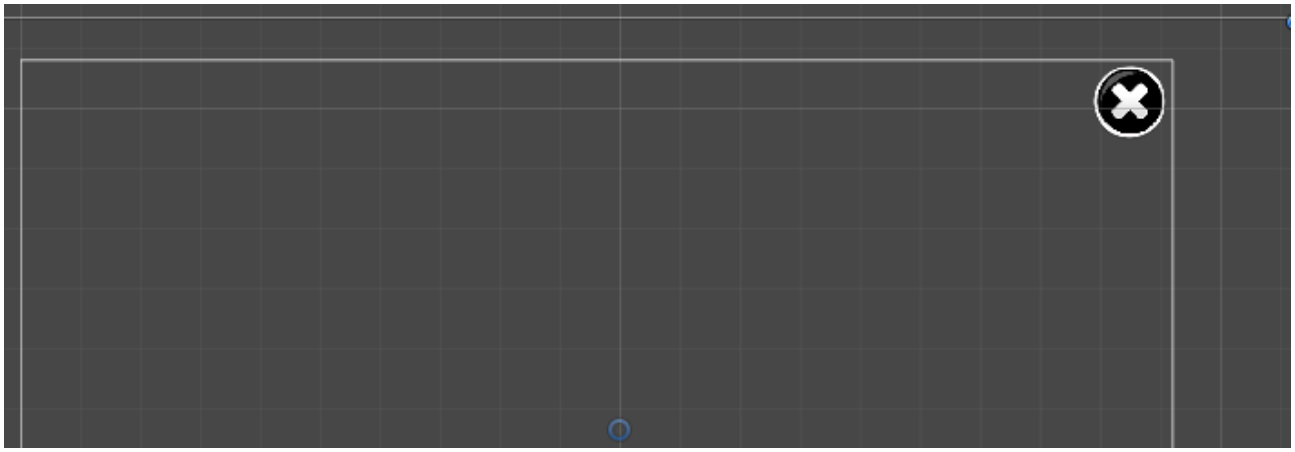

Рисунок 3.7 – Кнопка для вихода з самої гри

Рисунок 3.8 – Головне меню гри

Кнопки активуються при натисканні на екран в певній зоні і виконують функції описані в скрипті:

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

public class menu : MonoBehaviour {
    public void PlayPressed()
    {
        SceneManager.LoadScene("Furry");
    }
    public void ExitPressed()
    {
        Application.Quit();
        Debug.Log("Exit pressed!");
    }

}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Buttonexit : MonoBehaviour {

    void Start()
    {

        {
            switch (gameObject.name)
            {
                case "Exit":
                    Application.Quit();
                    break;
            }
        }
    }
}
```

Функція SceneManager.LoadScene означає, перехід на іншу сцену які описані в Build Settings. На рисунку 3.9 показано як виглядає Build Settings та сцени Unity.

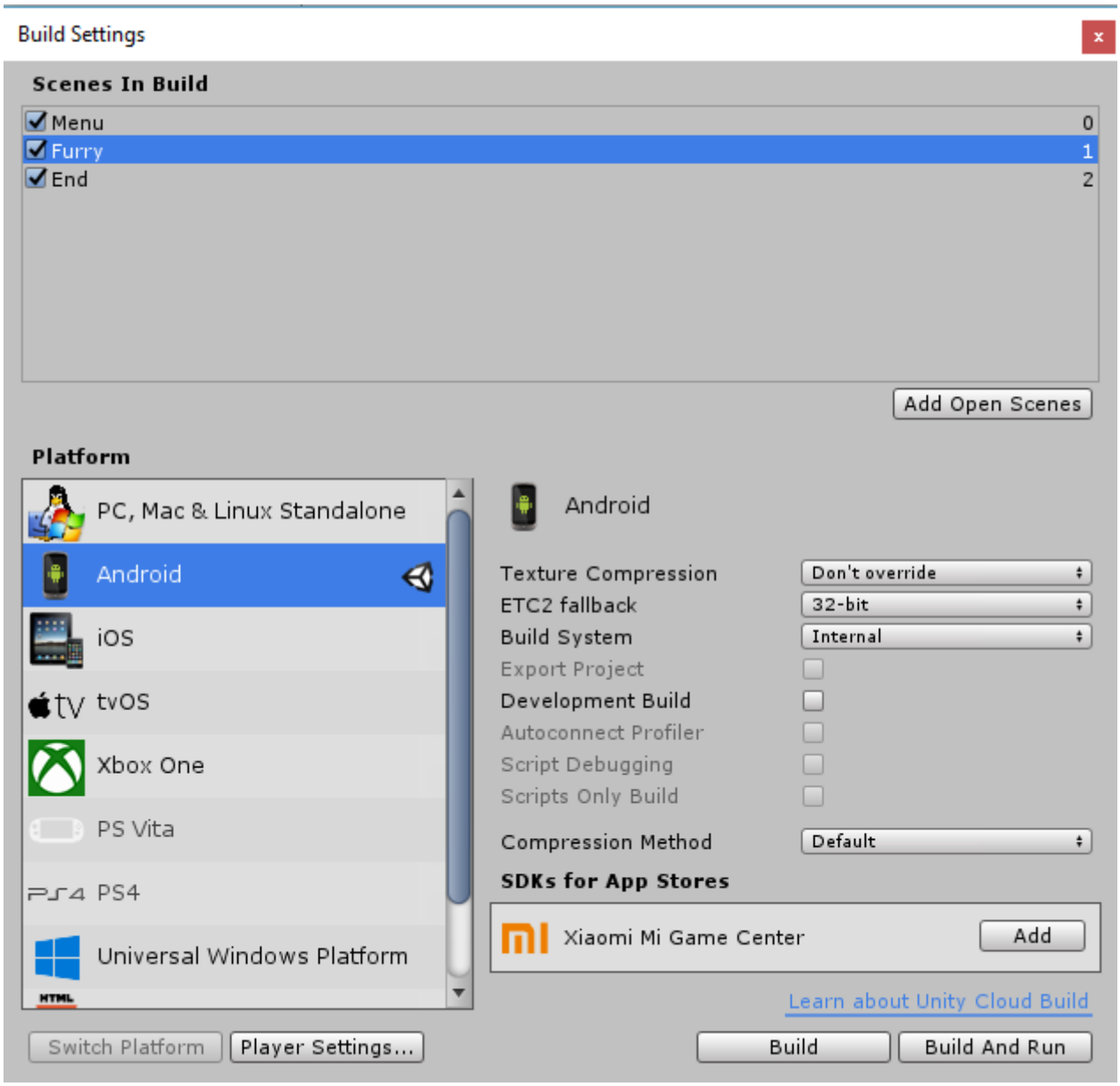

Рисунок 3.9– Меню Build Settings

В грі дуже багато об'єктів, наприклад на рис 3.10 показано кількість статичних і динамічних пасток.

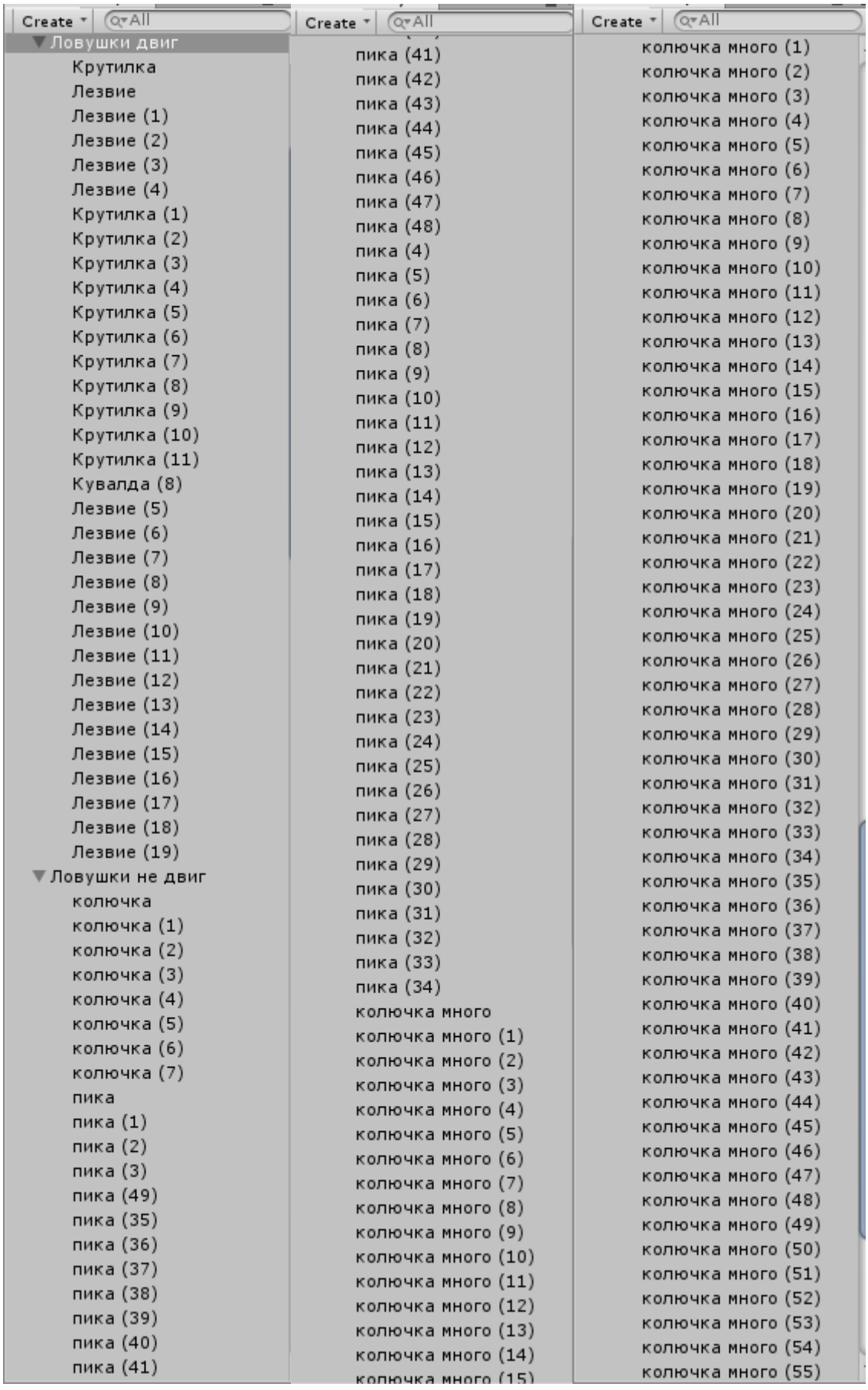

Рисунок 3.10 – Кількість статичних і динамічних пасток

3.5 Анімація

Комп'ютерна анімація – мистецтво створення рухомих зображень, за допомогою комп'ютерів. Є підрозділом комп'ютерної графіки та анімації. На відміну від більш загального поняття «графіка CGI», що відноситься як до нерухомих, так і до рухомих зображень, комп'ютерна анімація має на увазі тільки рухомі. На сьогодні отримала широке застосування як в області розваг, так і у виробничій, науковій та діловій сферах. Будучи похідною від комп'ютерної графіки, анімація успадковує ті ж способи створення зображень:

- векторна графіка;
- растрова графіка;
- фрактальна графіка;
- тривимірна графіка (3D).

Розстановка ключових кадрів проводиться аніматором. Проміжні ж кадри генерує спеціальна програма. Цей спосіб найбільш близький до традиційної мальованої анімації, тільки роль фазовщика бере на себе комп'ютер, а не людина.

Система анімації в Unity дозволяє створювати чудово анімованих персонажів. Вона підтримує Блендінг, мікшування, додавання анімацій, синхронізацію циклу ходьби, анімаційні шари, контроль всіх аспектів програвання (час, швидкість, ваги блендінга), скіннінг мешів з 1, 2 або 4 кістками на вершину, а також засновані на фізиці rag-dolls (ганчіркові ляльки) і процедурну анімацію.

У грі The last chance використовували спрайтову анімацію – анімацію 2D картинок. Два спрайти, які використовувалися для створенні анімації ходьби персонажу показані на рисунку 3.11.

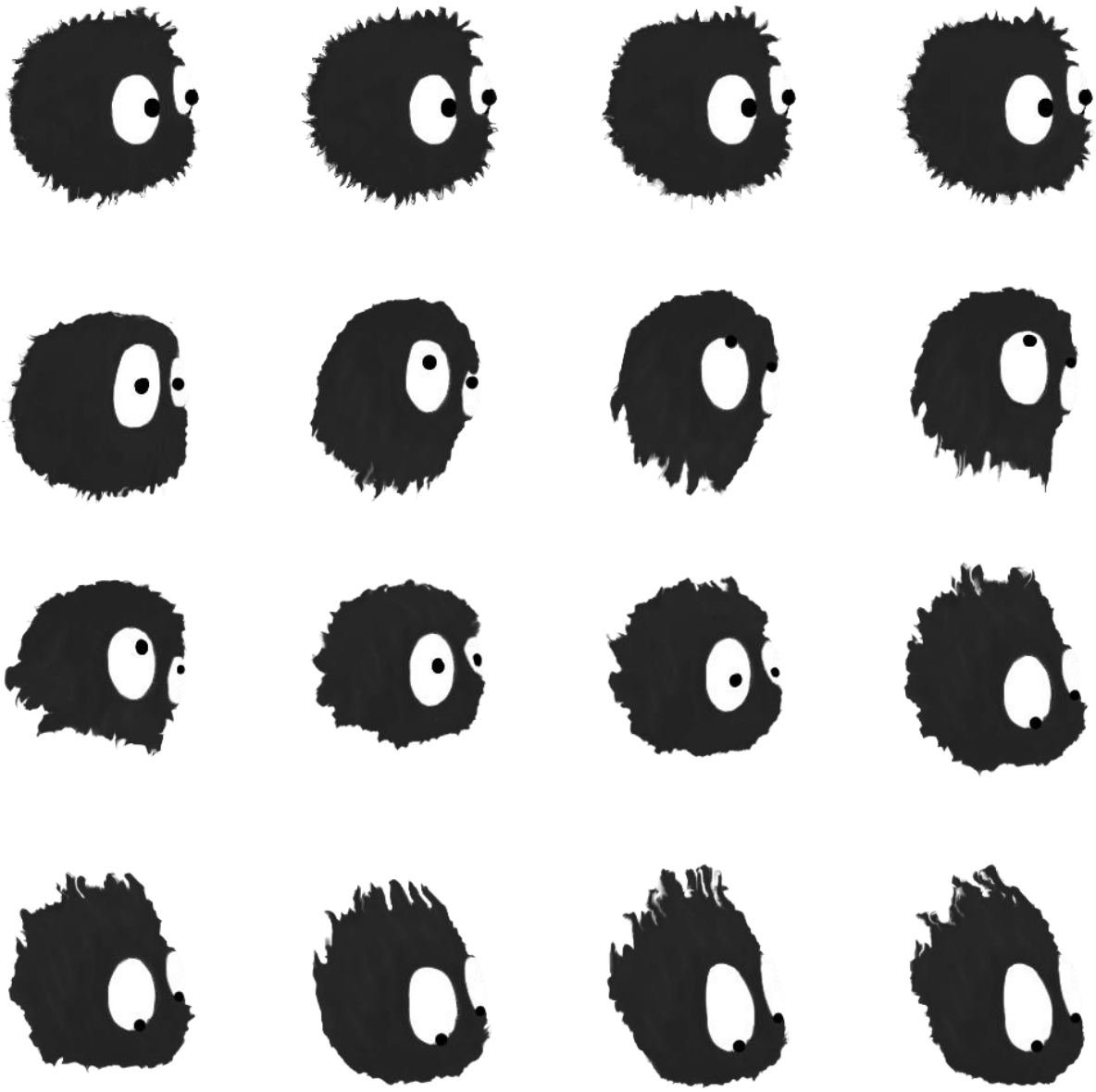

Рисунок 3.11 – Анімація стрибку персонажа

На рисунку 3.12 показано вікно створення спрайтової анімації.

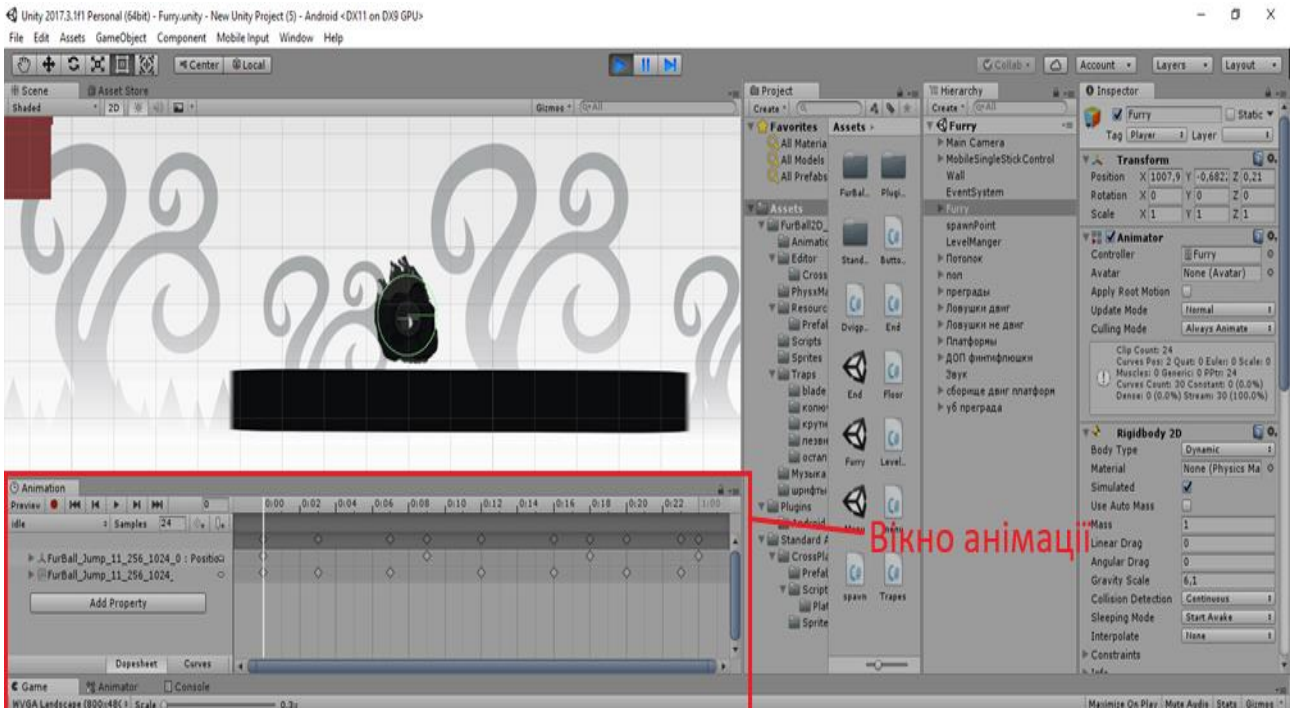

Рисунок 3.12– Вікно анімації

Як показано на рисунку 3.12, щоб створити анімацію руху, потрібно перемістити на часову доріжку спрайти в такій послідовності, в якій буде відтворюватися анімація.

3.6 Тестування програми

Мета тестування: виявлення функціональних помилок, невідповідностей технічному завданню і очікуванням Замовника шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Класифікація функцій, які будуть протестовані:

1 Переміщення персонажу по ігровому полю – буде проведення тестування переміщення головного героя по всім направленням:

1.1 Стрибки вгору.

1.2 “Буст” вниз.

1.3 Переміщення вправо.

1.4 Переміщення вліво.

Представлено у табл. 3.1.

Таблиця 3.1 – Календарний план робіт.

Номер тесту	Дата початку	Дата кінця	Часи				
			12:00	13:00	14:00	15:00	16:00
1	20.05.2021	28.05.2021					
2							
3							
4							

2 Тестування багів в підлозі і перешкод:

2.1 Рішення проблеми з стрибком від полу.

2.2 Зменшення до нуля способів проходження скрізь текстуру.

2.3 Оптимізація звичаних перешкод.

Представлено у табл. 3.2.

Таблиця 3.2 – Календарний план робіт.

Номер тесту	Дата початку	Дата кінця	Часи			
			16:10	16:11	16:12	16:20
1	28.05.2021	06.06.2021				
2						
3						

3 Тестування усіх активних кнопок та переходу між сценами:

3.1 Тестування старту та виходу з гри.

3.2 Рішення проблем з переходм між сценами.

3.3 Тестування активного тригера для кінцевої сцени.

Представлено у табл. 3.3.

Таблиця 3.3 – Календарний план робіт.

Номер тесту	Дата початку	Дата кінця	Часи			
			10:30	10:35	10:40	11:00
1	08.06.2021	10.06.2021				
2						
3						

4 Тестування усього ігрового процессу:

4.1 Знаходження непроходимих частин гри.

4.2 Рішення спростити чи створити checkpoints у грі.

Представлено у табл. 3.4.

Таблиця 3.4 – Календарний план робіт.

Номер тесту	Дата початку	Дата кінця	Часи		
			16:40	16:41	16:42
1	10.06.2021	14.06.2021			
2					

Результати тестів наведено таблиці 3.5.

Таблиця 3.5 – Успішність виконання тестів.

Відмітка на виконання	Види тестів	Примітка
✓	Переміщення персонажу по ігровому полю	При натисканні відповідних панелей на екрані, персонаж рухався по ігровому полю.
✓	Тестування багів в підлозі і перешкод	Присвоення до полу скрипту Floor та додатковий RigBody.

Продовження таблиці 3.5

✓	Тестування усіх активних кнопок та переходу між сценами	Тестування реакції виконання функцій при натискання на кнопки.
✓	Тестування усього ігрового процессу	Проходження гри та спрощення її проходження.

4 КРОСПЛАТФОРМЕНІСТЬ

4.1 Понятие кросплатформеності

Крос-платформенне програмне забезпечення - програмне забезпечення, яке працює більш ніж на одній апаратній платформі і / або операційній системі.

На сьогоднішній день багато користувачів переходять на мобільні платформи через їх популярності і зручності використання. З цієї причини розробникам програмного забезпечення доводиться створювати все нові програми. При цьому використовуються існуючі засоби розробки, які не завжди дозволяють написати код, стерпний на різні платформи без втрат продуктивності і швидкості роботи. Останнім часом широко використовуються так звані крос-компілятори, здатні виробляти на одній і тій же платформі код для відмінних від неї платформ. Попередня збірка такого крос-компілятора вимагає великих витрат часу і ресурсів, а робота з ним не гарантує абсолютно успішного результату, так як готове програмне забезпечення може потребувати змін

Якщо не акцентувати увагу саме на платформах для мобільних пристроїв, то і для настільних робочих станцій широке коло розроблюваних продуктів створюється строго під певні операційні системи, через що користувачеві доводиться іноді відмовлятися від звичного йому робочого оточення заради програмного продукту, який йому необхідний на даний момент для вирішення конкретного завдання.

В даний час існують такі засоби розробки програм, як мова Java, за допомогою якого можливо створити програмний продукт, що переноситься між платформами без необхідних додаткових змін вихідного коду. Однак програми, реалізовані на мові Java, вимагають досить багато ресурсів для своєї роботи. З появою кроссплатформенного інструментарію Qt стало можливим створювати стерпні програмні продукти на мовах сімейства C / C ++, Java або Python. Основним недоліком використання Qt є розмір готового програмного

продукту, що в ряді випадків неприйнятно. Крім того, програми, розроблені з використанням перерахованих мов програмування, не завжди можуть працювати під управлінням більш ранніх версій операційних систем. Провівши аналіз запропонованих рішень для створення кроссплатформенних додатків, можна запропонувати два основні варіанти:

1 створення програмних продуктів на мові Java зі зменшенням продуктивності роботи;

2 створення програм з використанням інструментарію Qt в зв'язці з мовами Java або C / C ++.

Альтернативою використання вже стали стандартними методів вирішення даної проблеми може стати створення спеціалізованого прикладного інтерфейсу програмування, використовуючи який в подальшому, можна буде створювати програмні продукти, що переносяться між різними мобільними платформами без втрат продуктивності, додаткових змін коду і збільшення розміру. Крім цього, такі програми могли б функціонувати і на платформах для настільних комп'ютерів.

Проект, що розробляється, в першу чергу, являє собою необхідний інструментарій, який дозволить створювати Кросплатформені програмні продукти різного призначення і може широко застосовуватися в різних областях науки, де використовуються і розробляються програм ні продукти для математичного моделювання реальних фізичних процесів, створюються наочні моделі і проводяться обробка і аналіз різних результатів вимірювання. Такі програмні продукти можуть бути зібрані для різних платформ, що знімає обмеження на використання різних операційних систем і прив'язку до техніки певної марки.

4.2 Переваги виборної платформи для реалізації проекту на різних пристроях.

Unity є дуже відомим двигуном серед інді-розробників. Це багатоплатформовий движок, який дозволяє розробляти 3D і 2D-ігри. Особливістю, яка виділяє Unity серед інших движків, є низький поріг входження для новачків при наявності багатого інструментарію для професіоналів. Кросплатформеність ж дозволяє розробляти додатки під будь-яку платформу, починаючи з десктопних ігор і закінчуючи мобільними. Unity надала можливість реалізовувати проекти які підтримуються на Windows, MacOS, Wii, iPhone, iPod, iPad, Android, PS3 і XBox 360 і веб-плагін. Але найголовнішим плюсом є те щоб перенести проект на іншу платформ не потрібно писати код, а достатньо всього лише вибрати в меню компіляції платформу під яку буде збиратися проект.

ВИСНОВКИ

Головна мета проекту була досягнена. Під час роботи над дипломним проектом було проаналізовано велику кількість джерел інформації щодо розробки програмних продуктів на платформі Unity3D, а в особливості – розробки інтерфейсу та написання комп’ютерних ігор. Можна зробити висновок, що Unity3D є лідером в індустрії серед існуючих програмних рішень. Ця платформа має низький поріг входження, тобто не передбачає наявності масштабної бази знань у розробника. Вона має зручний, інтуітивно зрозумілий інтерфейс, що сприяє швидкому вивченню різноманітності елементів програми. Також, постійна підтримка розробників та своєчасне оновлення функціоналу дає змогу створити у найкоротший термін стабільний, якісний продукт, який відповідає сучасним запитам.

На сьогоднішній день ринок комп'ютерних ігор задає досить високу планку для розроблюваних продуктів. Сучасна гра повинна мати сильні сторони в області графіки, сюжетної складової, особливостей геймплея.

Часто проблемою для розробників стає не складність розробки, а її масштабність. Не завжди у розробника виходить довести до ідеалу всі аспекти свого продукту, але певна частина з них, зазвичай, виділяється на тлі інших.

Як програмний додаток до дипломного проекту була розроблена комп’ютерна тривимірна гра у жанрі Platformer, сутність виконаної роботи полягала в створенні гри. Гра має звід правил, за якими проходить ігровий процес і може представляти інтерес будь-якому поціновувачу ігор, чи кінострічок такого, або наближеного жанру.

Підводячи підсумок, можна зауважити, що комп’ютерні ігри вже тривалий час є частиною нашого життя. Кількість їх жанрів і різновидів часом важко перелічити. Звичайно, ігри впливають на нас як позитивно, так і негативно. Але, якщо дивитися на них з боку розробки, то це вкрай захоплюючий і творчий процес, часом цікавіше самої гри. Він вимагає великої

кількості різноманітних витрат, результати яких не завжди виправдовують очікування. Однак, в разі успіху, розробники не тільки набувають нових знанн в даній сфері, а й сприяють розвитку апаратних характеристик пристроїв, які прямо впливають на можливість найбільш обширної демонстрації можливостей продукту.

ЛІТЕРАТУРА

1. [Електронний ресурс] https://jak.waykun.com/articles/unity-kerivnictvo-skripting-animacii-legacy.html

2. [Електронний ресурс] https://docs.unity3d.com/ru/530/Manual/

3. Рейнбоу В., "Комп'ютерні ігри". Енциклопедія. – С .: "Пітер" 2005. - 732с.

4. Миколаєва О. "Епідемія XXI століття: телебачення, інтернет і комп'ютерні ігри" .- Ростов н / Д: Фелікс, 2008.-254с.

5. Роллінгз, Ендрю. "Проектування та архітектура ігор": пров. з англ. / Е. Роллінгз, Д. Морріс. – М .: Вільямс, 2006.- 1040с.

6. Дашко Ю.В., Заїка А.А. "Основи розробки комп'ютерних ігор" – М .: "Форум" 2009.-350С.

7. “Unity 3D and PlayMaker Essentials: Game Development from Concept to Publishing (Focal Press Game Design Workshops)” – Jere Miles, 2016

8. “Unity 2D Game Development Cookbook” – Claudio Scolastici, 2015.