ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

Бакалаврська роботи
на ступінь вищої освіти бакалавр

на тему:

«Розробка гри „Great Walk" в жанрі 2D платформер з використанням ігрового двигуна UNITY та мови програмування c#»

Виконав: студент 4 курсу, групи ПД-42
спеціальності:
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

________________Черниш В. В.________
(прізвище та ініціали)

Керівник ________Негоденко О. В.______
(прізвище та ініціали)

Рецензент________________________________
(прізвище та ініціали)

Нормоконтроль___________________________
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____________О.В.Негоденко

"____"________20____року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Чернишу Володимиру Володимировичу

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка гри „great walk" в жанрі 2D платформер з використанням ігрового двигуна UNITY та мови програмування C#»

Керівник роботи к.т.н., доцент Негоденко О.В. ,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "12" березня року №65.

2. Строк подання студентом роботи 01.06.2021.

3. Вихідні дані до роботи:

3.1. Положення побудови гри;

3.2. Методи побудови гри;

3.3. Розробка моделі гри;

3.4. Науково-технічна література.

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

4.1. Загальні положення побудови гри;

4.2. Аналіз технології і методів побудови гри;

4.3. Висновки

5. Перелік графічного матеріалу.
   5.1. Основні характеристики роботи;
   5.2. Актуальність задачі;
   5.3. Реалізація алгоритмів гри;
   5.4. Висновки.

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір джерел інформації	19.04.2021	Виконано
2	Вимоги до встановленого додатку	23.04.2021	Виконано
3	Оцінка якості тестування до систем	29.04.2021	Виконано
4	Концепція та архітектура додатку	02.05.2021	Виконано
5	Вступ, висновки, реферат	05.05.2021	Виконано
6	Розробка презентації	06.05.2021	Виконано
7	Перед захист диплому	11.05.2021	
8	Задача роботи	01.06.2021	

Студент ________________ Черниш В.В.

(підпис) (прізвище та ініціали)

Керівник роботи ______________ Негоденко О.В.

підпис) (прізвище та ініціали

РЕФЕРАТ

Текстова частина бакалаврської роботи 64 стр., 7 рис., 49 джерел

Ключові слова: ГРА, ВІДЕОІГРИ, ПЛАТФОРМЕР, UNITY, C#

Об'єкт дослідження – процес розробки двомірного ігрового застосунку у жанрі «платформер» для Android.

Предмет дослідження – методи, прийоми та інформаційні технології розробки ігрового програмного забезпечення для пристроїв на базі Android платформи.

Мета роботи – розробити та вдосконалити ігровий додаток, з можливістю його використання на платформі Android.

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

Для реалізації мети було сформульовано та вирішено наступні завдання:

1. Провести аналіз схожих робіт.
2. Дослідити середовище розробки Unity3d та мову C#.
3. Розробити гру для проходження тестування.
4. Провести тестування.

Предметом дослідження є гра платформер.

Практичне значення отриманих результатів полягає у написанні функціоналу для роботи з інтерполяцією візуального контенту гри Great Walk з використання ігрового двигуна UNITY та мови програмування C# на мобільну платформу. Також реалізовану гру можна поширювати серед гравців, виклавши її до магазинів мобільних ігор.

В роботі розглянуто основні етапи створення ігор і досліджено можливості технічних засобів для розробки гри.

Розроблено модель застосунку та функціональні вимоги до гри та підібрані методи інтерполювання для їх реалізації.

Галузь використання – завдяки розвитку IT індустрії, будь-який користувач може завантажити та використовувати гру Great Walk в жанрі 2D платформер.

Зміст

ВСТУП

Актуальність теми. Перші відеоігри з'являлися на комп'ютери та приставки. Та й зараз при слові «відеогра» на думку спадає саме комп'ютерна гра. Але останнім часом мобільні ігри набирають шалену популярність. Це пов'язано з тим, що в останні роки спостерігається зростання попиту на мобільні пристрої, через доступність всіх можливостей, включаючи вихід в Інтернет, не тільки через стаціонарні ПК, а й через планшети, смартфони і звичайні телефони. Потужний комп'ютер для роботи потрібен далеко не всім, а ось мобільний пристрій, який займає мало місця, завжди з вами і все може – зовсім інша справа. Цілком природно, що постійно з'являються нові застосунки для Android, IOS і Windows, що роблять процес експлуатації пристроїв ще більш комфортним і простим.

У зв'язку з цим розвивається й ринок розваг для мобільних пристроїв. Кожного дня в магазини мобільних ігор виходить 1-2 гри. Тому щоб створювати конкурентоспроможні та якісні ігри – потрібно генерувати нові ідеї та мати багато досвіду для втілення їх у життя. Потрібно розроблювати універсальні модулі, що можуть використовуватись у майбутніх розробках, покращувати візуальну складову ігор задля привернення більшої аудиторії до ігрового застосунку. Саме тому темою дипломного проекту я обрав розробку мобільної гри. Адже ринок ігор сьогодні один із самих прибуткових. Гаджети 2 стрімко розвиваються щоб полегшити життя людини та зайняти дозвілля. Популярність мобільних телефонів зростає.

Мета та завдання. Метою є створення ігрової платформи на Unity на основі неї написання невеликої демонстраційної казуальної гри, яка може залучити потенційних покупців в стрімко розширюється ринку комп'ютерних розваг. Так само створена платформа є основою для створення наступних програмних продуктів значно скоротивши час їх створення на 70%.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- вибрати жанр, вид і платформу для гри;

- розробити сценарій, концепцію основних елементів;
- підготувати необхідні для гри анімації;
- реалізація прототипу гри.

Об'єктот роботи виступає процес розробки двомірного ігрового застосунку у жанрі «платформер» для Android.

Предмет – методи, прийоми та інформаційні технології розробки ігрового програмного забезпечення для пристроїв на базі Android платформи.

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

Практичне значення отриманих результатів. Написаний функціонал для роботи з інтерполяцією візуального контенту гри Great Walk з використання ігрового двигуна UNITY та мови програмування C# на мобільну платформу. Також реалізовану гру можна поширювати серед гравців, виклавши її до магазинів мобільних ігор. Теоретичний матеріал може бути використаний під час викладання дисципліни за спеціальністю «Інженерія програмного забезпечення».

Структура роботи. Структуру роботи складають: вступ, три розділи, висновки, перелік використаної літератури та додатки. Загальний обсяг роботи 62 сторінок.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Відеоігри

Відеогра - це електронна гра, яка передбачає взаємодію з користувальницьким інтерфейсом або пристроєм введення - наприклад, джойстиком, контролером, клавіатурою або пристроєм, що зчитує рух - для створення візуального зворотного зв'язку для гравця. Цей відгук відображається на пристрої для відображення відео, наприклад на телевізорі, моніторі, сенсорному екрані або гарнітурі віртуальної реальності. Відеоігри часто доповнюються звуковим зворотним зв'язком, що надходить через динаміки або навушники, а іноді й іншими типами зворотного зв'язку, включаючи тактильні технології.

Відеоігри визначаються на основі їх платформи, яка включає аркадні ігри, консольні ігри для ПК. Зовсім недавно галузь розширилася на мобільні ігри за допомогою смартфонів та планшетних комп’ютерів, систем віртуальної та доповненої реальності та віддалених хмарних ігор. Відеоігри класифікуються на широкий діапазон жанрів залежно від їх типу гри та призначення.

Перші відеоігри були простими розширеннями електронних ігор з використанням відеоподібного виводу з великих комп'ютерів розміру в приміщенні в 1950-х і 1960-х роках, тоді як перші відеоігри, доступні для споживачів, з'явилися в 1971 році завдяки випуску аркадної гри Computer Space, наступного року від Pong, і з першою домашньою консоллю Magnavox Odyssey у 1972 р. Сьогодні для розробки відеоігор потрібні численні навички виведення гри на ринок, включаючи розробників, видавців, дистриб’юторів, роздрібних торговців, консольних та інших сторонніх виробників, і інші ролі.

З 2010-х років комерційне значення індустрії відеоігор зростає. Ринки Азії, що розвиваються, і мобільні ігри на смартфонах, зокрема, є рушієм зростання індустрії. Станом на 2018 рік відеоігри щорічно продавали 134,9 млрд. Доларів США у всьому світі [1] і були третім за величиною сегментом на американському ринку розваг, після трансляції та кабельного телебачення.

Ранні ігри використовували інтерактивні електронні пристрої з різними форматами дисплея. Найбільш ранній приклад - з 1947 р. - "Пристрій для розваг з електронно-променевою трубкою" було подано на патент 25 січня 1947 р. Томасом Т. Голдсмітом-молодшим та Естл Рей Манн і видано 14 грудня 1948 р. Як патент США 2455992. [2] Натхненний технологією радіолокаційного відображення, він складався з аналогового пристрою, який дозволяв користувачеві управляти векторною крапкою на екрані для імітації ракети, що стріляла по цілях, які були закріплені на екрані. [3] Серед інших ранніх прикладів можна назвати: гру «Крістофер Стейчі» «Чернетки», комп’ютер «Німрод» на Британському фестивалі 1951 року; OXO - комп’ютерна гра “тик-так”, виконана Олександром Дугласом для EDSAC у 1952 році; «Теніс для двох» - електронна інтерактивна гра, розроблена Вільямом Хігінботем у 1958 році; "Космічна війна!", Написана студентами Массачусетського технологічного інституту Мартіном Грацем, Стівом Расселом та Уейном Війтаненом на комп'ютері DEC PDP-1 у 1961 році; і хіт-понг у стилі пінг-понг, гра Atari 1972 року. У кожній грі використовувались різні засоби відображення: NIMROD використовував панель вогнів, щоб грати в гру Нім, [4] OXO використовував графічний дисплей для гри в хрестики-нулики [5] Теніс для двох використовував осцилограф для відображення виду збоку тенісного корту [3] та Spacewar! використовував векторний дисплей DEC PDP-1, щоб два космічні кораблі билися між собою. [6]

Ці попередні винаходи відкрили шлях для походження відеоігор сьогодні. Ральф Х.Бер, працюючи в компанії Sanders Associates в 1966 році, виступив з ідеєю використовувати систему управління, щоб грати елементарну гру в настільний теніс на телевізійному екрані. З благословення Сандерса Баер розробляє прототип "Коричневої коробки". Сандерс запатентував винаходи Баера і дозволив їх Magnavox, який комерціалізував його як першу домашню відеоігрову консоль, Magnavox Odyssey, випущену в 1972 році [3] [7]. Окремо, Нолан Бушнелл і Тед Дабні, натхненні побаченим Spacewar! Працюючи в Стенфордському університеті, він придумав створити подібну версію, що працює в меншій шафі, використовуючи менш дорогий комп'ютер з функцією монети. Це було випущено під назвою Computer Space, першою аркадною грою, в 1971 році [8]. Бушнелл і Дабні створили Atari, Inc., а разом з Алланом Алькорном створили свою другу аркадну гру Pong в 1972 році, яка була безпосередньо натхненна грою в настільний теніс на "Одісеї". Сандерс та Магнавокс подали до суду на Atari за порушення патентів на патенти Baer, але Atari вирішив це поза судом, заплативши за безстрокові права на патенти. Після їх домовленості Atari продовжив планувати зробити домашню версію Pong, однак був випущений до Різдва 1975 року. [3] Успіх "Одісеї" та "Понга" як аркадної гри, так і домашньої машини започаткував індустрію відеоігор. [9] [10] І Баер, і Бушнелл за свій внесок отримали титул "Батько відеоігор" [11] [12].

Термін "відеоігра" був розроблений для розрізнення цього класу електронних ігор, які відтворювались на певному типі відеодисплея, а не тих, що використовували вихідні дані телетайпного принтера або подібного пристрою.

Перша поява терміна з’явилася приблизно в 1973 р. Оксфордський словник англійської мови цитував статтю BusinessWeek від 10 листопада 1973 р. Як перше друковане використання цього терміна [14]. Хоча Бушнелл вважав, що цей термін вийшов з огляду вендингових журналів Computer Space в 1971 р. [15], огляд найбільших вендингових журналів Vending Times і Cashbox показав, що цей термін з’явився набагато раніше, з’явившись у цих журналах масово в березні 1973 р. використання в тому числі виробниками аркадних ігор. Як проаналізував історик відеоігор Кіт Сміт, раптова поява припустила, що цей термін був запропонований та охоче прийнятий причетними. Здавалося, це прослідковується Едом Адлумом, який керував монетизованою секцією Cashbox до 1972 року, а потім заснував журнал RePlay, що охоплює поле розваг про монети, в 1975 році. У вересневому випуску RePlay, Adlum приписується першому називанню цих ігри як "відеоігри": "Едді

Адлум від RePlay працював у" Cash Box ", коли" ТВ-ігри "вперше вийшли. Персонажі в ті часи були Бушнелл, його менеджер з продажів Пат Карнс і ще кілька виробників" ТВ-ігор " Генрі Лейзер і брати Макьюени. Здавалося, незручно називати їх продукцію "телевізійними іграми", тому, запозичивши слово з "Білбордівського опису музичних автоматів", Adlum почав називати цю нову породу розважальних машин "відеоіграми". Фраза застрягла ". [16] У Японії, де консолі на зразок" Одісеї "спочатку імпортувались, а потім виготовлялись у межах країни великими виробниками телевізорів, такими як Toshiba та Sharp Corporation, вони також були відомі як" телевізійні ігри "або телевізійний геему або terebi geemu. [17]

Оскільки кожна відеогра відрізняється, досвід гри у кожну відеоігру неможливо узагальнити в окремому твердженні, але існує багато загальних елементів. Більшість ігор запускаються на титульному екрані і дають гравцеві можливість переглянути варіанти, такі як кількість гравців, перед початком гри. Більшість ігор поділяються на рівні, на яких гравець повинен пробити свій аватар, набираючи очки, збираючи бонуси для підвищення вроджених атрибутів аватара, при цьому використовуючи спеціальні атаки для перемоги над ворогами або рухаючись, щоб уникнути їх. Нанесення шкоди призведе до виснаження здоров’я їхнього аватара, і якщо це впаде до нуля або якщо аватар в іншому випадку потрапить у неможливе для втечі місце, гравець втратить одне з своїх життів. Якщо вони втратять все своє життя, не отримавши зайвого життя або "1-UP", тоді гравець дістанеться до екрану "game over". Багато рівнів, а також фінал гри закінчуються типом боса, якого гравець повинен перемогти, щоб продовжити. У деяких іграх проміжні очки між рівнями пропонують очки збереження, де гравець може створити збережену гру на носії даних, щоб перезапустити гру, якщо вони втратять все життя або їм потрібно зупинити гру та перезапуститись пізніше. Вони також можуть бути у формі уривку, який можна записати та повернути на головний екран.

Оскільки ігри є програмними продуктами, вони все ще можуть поставлятися із програмними помилками. Вони можуть проявлятися як збої в грі, які може бути використаний гравцем; це часто є основою швидкісного запуску відеоігри. В інших випадках ці помилки, поряд із чіт-кодами, писанками та іншими прихованими секретами, які були навмисно додані до гри, також можуть бути використані [18] [19] [20] [21]. На деяких консолях чіт-картриджі дозволяють гравцям виконувати ці чіт-коди, тоді як розроблені користувачем трейнери дозволяють аналогічне обхід для комп'ютерних програмних ігор, і те, і інше, що може полегшити гру, надати гравцеві додаткові підсилення або змінити вигляд гри . [19]

Для роботи відеоігор потрібна платформа, певна комбінація електронних компонентів або комп'ютерного обладнання та відповідного програмного забезпечення. [22] Також широко використовується термін система. Ігри, як правило, розроблені для гри на одній або обмеженій кількості платформ, а ексклюзивність платформи використовується як конкурентне перевагу на ринку відеоігор. [23] Наведений нижче список не є вичерпним і не включає інші електронні пристрої, здатні грати у відеоігри, такі як КПК та графічні калькулятори.

Комп'ютерна гра

Більшість комп’ютерних ігор є іграми для ПК, маючи на увазі ті, які передбачають взаємодію гравця з персональним комп’ютером (ПК), підключеним до відеомонітора. [24] Персональні комп’ютери не є спеціальними ігровими платформами, тому можуть існувати відмінності, коли одна і та ж гра працює на різному обладнанні. Крім того, відкритість дозволяє розробникам деякі функції, такі як зниження вартості програмного забезпечення, [25] підвищена гнучкість, збільшення інновацій, емуляція, створення модифікацій або модифікацій, відкритий хостинг для онлайн-ігор (в якому людина грає у відеогру з людьми, які перебувають у інше домогосподарство) та інші. Ігровий комп'ютер - це ПК або ноутбук, призначений спеціально для ігор, як правило, використовуючи високопродуктивні та дорогі компоненти. Окрім ігор на персональних комп’ютерах, існують також ігри, які працюють на мейнфреймових комп’ютерах та інших подібних спільних системах, коли користувачі віддалено входять у систему, щоб користуватися комп’ютером.

Домашня консоль

Консольна гра проводиться на домашній консолі, спеціалізованому електронному пристрої, який підключається до звичайного телевізора або композитного відеомонітора. Домашні консолі спеціально розроблені для гри в ігри з використанням спеціального апаратного середовища, що надає розробникам конкретну апаратну ціль для розробки та гарантує, які функції будуть доступні, спрощуючи розробку порівняно з розробкою ігор на ПК. Зазвичай консолі запускають лише ігри, розроблені для неї, або ігри з іншої платформи, виготовлені тією ж компанією, але ніколи не ігри, розроблені її прямим конкурентом, навіть якщо одна і та ж гра доступна на різних платформах. Він часто постачається з певним ігровим контролером. Основні консольні платформи включають Xbox, PlayStation та Nintendo.

Портативна консоль

Портативний ігровий пристрій - це невеликий автономний електронний пристрій, який є портативним і може триматися в руках користувача. У ньому є консоль, невеликий екран, динаміки та кнопки, джойстик або інші ігрові контролери в одному блоці. Як і консолі, КПК - це виділені платформи і мають майже однакові характеристики. Портативне обладнання зазвичай є менш потужним, ніж апаратне забезпечення ПК або консолі. Деякі портативні ігри кінця 1970-х та початку 1980-х могли грати лише в одну гру. У 1990-х і 2000-х роках у ряді портативних ігор використовувались картриджі, що дозволяло використовувати їх для різноманітних ігор.

Аркадна гра

Аркадна гра, як правило, відноситься до гри, яка ведеться на ще більш спеціалізованому типі електронного пристрою, який, як правило, призначений для гри лише в одну гру і укладений у спеціальну велику шафу на монети, яка має одну вбудовану консоль, контролери (джойстик , кнопки тощо), ЕЛТ-екран, а також підсилювач звуку та динаміки. Аркадні ігри часто мають яскраво розписані логотипи та зображення, що стосуються теми гри. Хоча більшість аркадних ігор розміщуються у вертикальному шафі, перед яким зазвичай грає користувач, деякі аркадні ігри використовують настільний підхід, в якому екран дисплея розміщений у настільному шафі з прозорою стільницею. . У настільних іграх користувачі зазвичай сидять грати. У 1990-х і 2000-х роках деякі аркадні ігри пропонували гравцям на вибір кілька ігор. У 1980-х роках відеоаркади були бізнесами, в яких гравці ігор могли використовувати ряд аркадних відеоігор. У 2010-х відеоарк стає набагато менше, але в деяких кінотеатрах та центрах сімейних розваг вони все ще є.

Браузерна гра

Браузерна гра використовує переваги стандартизації технологій для функціонування веб-браузерів на декількох пристроях, що забезпечують міжплатформене середовище. Ці ігри можуть бути ідентифіковані на основі веб-сайту, на якому вони з’являються, наприклад, для ігор Miniclip. Інші називаються на основі платформи програмування, яка використовується для їх розробки, наприклад, Java та Flash ігри.

Мобільна гра

З появою смартфонів та планшетних комп’ютерів, стандартизованих для операційних систем iOS та Android, мобільні ігри стали важливою платформою. Ці ігри можуть використовувати унікальні функції мобільних пристроїв, які не є необхідними на інших платформах, такі як акселерометри, інформація про глобальне розміщення та камери для підтримки геймплею доповненої реальності.

Хмарні ігри

Для хмарних ігор потрібен мінімальний апаратний пристрій, такий як базовий комп’ютер, консоль, ноутбук, мобільний телефон або навіть спеціальний апаратний пристрій, підключений до дисплея з хорошим підключенням до Інтернету, який підключається до апаратних систем постачальником хмарних ігор. Гра обчислюється і відображається на віддаленому обладнанні, використовуючи ряд методів прогнозування, щоб зменшити мережеву затримку між входом і виходом програвача на пристрої відображення.

Віртуальна реальність

Ігри з віртуальною реальністю (VR) зазвичай вимагають від гравців використання спеціального блоку, встановленого на голові, який забезпечує стереоскопічні екрани та відстеження руху, щоб занурити гравця у віртуальне середовище, яке реагує на їхні рухи головою. Деякі системи VR включають в себе блоки управління для рук гравця, що забезпечують прямий спосіб взаємодії з віртуальним світом. Для систем VR зазвичай потрібен окремий комп’ютер, консоль або інший обробний пристрій, який поєднується з головним блоком.

Емуляція

Емулятор дозволяє запускати ігри з консолі або іншої системи у віртуальній машині на сучасній системі, імітуючи апаратне забезпечення оригіналу, і дозволяє відтворювати старі ігри. Хоча самі емулятори були визнані законними у прецедентній практиці США, акт отримання ігрового програмного забезпечення, яким ще не володіє, може порушити авторські права. Однак є деякі офіційні випуски емульованого програмного забезпечення від виробників ігор, такі як Nintendo з його віртуальною консоллю або пропозиції Nintendo Switch Online.

Ранні аркадні ігри, домашні консолі та портативні ігри були виділеними апаратними блоками з логікою гри, вбудованою в електронні компоненти апаратного забезпечення. З тих пір більшість платформ відеоігор мають засоби для використання декількох ігор, розподілених на різних типах носіїв або форматів. Фізичні формати включають картриджі ПЗУ, магнітне сховище, включаючи зберігання даних магнітної стрічки та дискети, формати оптичних носіїв, включаючи CD-ROM та DVD, та карти флеш-пам'яті. Крім того, цифровий розподіл через Інтернет або інші способи зв'язку, а також хмарні ігри полегшують потребу в будь-яких фізичних носіях. У деяких випадках носій служить безпосередньо пам’яттю для гри лише для читання, або це може бути форма інсталяційного носія, який використовується для запису основних ресурсів у локальну пам’ять платформи гравця для швидшого періоду завантаження та подальших оновлень.

Ігри можна розширити за допомогою нового контенту та програмних виправлень за допомогою будь-яких пакетів розширення, які зазвичай доступні як фізичний носій, або як завантажуваний вміст, номінально доступний за допомогою цифрового розповсюдження. Їх можна пропонувати безкоштовно або використовувати для монетизації гри після її первинного випуску. Кілька ігор пропонують гравцям можливість створювати створений користувачем вміст, щоб ділитися з іншими гравцями. Інші ігри, в основному на персональних комп’ютерах, можуть бути розширені за допомогою створених користувачем модифікацій або модифікацій, які змінюють або додають у гру; вони часто є неофіційними і були розроблені гравцями із зворотного інжинірингу гри, але інші ігри надають офіційну підтримку модифікації гри. [26]

Відеогра може використовувати кілька типів пристроїв введення для перекладу людських дій у гру. Найбільш поширеними є використання ігрових контролерів, таких як геймпади та джойстики для більшості консолей. Портативні консолі матимуть вбудовані кнопки та напрямні накладки, аналогічно аркадні ігри матимуть елементи управління, вбудовані в сам блок консолі. Багато ігор на персональних комп’ютерах можуть скористатися перевагами клавіатури та миші. Інші ігрові контролери зазвичай використовуються для таких ігор, як гоночні колеса, легкі гармати чи танцювальні майданчики. Цифрові камери також можуть використовуватися як ігрові контролери, що фіксують рухи тіла гравця.

По мірі того, як технологія продовжує прогресувати, на контролер можна додавати все більше, щоб надати гравцеві більш захоплюючий досвід під час гри у різні ігри. Є деякі контролери, які мають попередньо встановлені налаштування, щоб кнопки накладалися на певний спосіб, щоб полегшити гру в певні ігри. Поряд із пресетами, гравець іноді може власноруч зіставити кнопки, щоб краще відповідати їх стилю гри. На клавіатурі та миші різні дії в грі вже задані для клавіш на клавіатурі. Більшість ігор дозволяють гравцеві змінити це, так що дії відображаються на різні клавіші, які їм більше до вподоби. Компанії, що розробляють контролери, намагаються зробити контролер візуально привабливим, а також почуватись комфортно в руках споживача.

Прикладом технології, яка була включена в контролер, був сенсорний екран. Це дозволяє гравцеві мати можливість взаємодіяти з грою інакше, ніж раніше. Людина могла б легше пересуватися в меню, і вона також могла взаємодіяти з різними предметами в грі. Вони можуть підбирати одні предмети, екіпірувати інші або навіть просто пересувати предмети з шляху гравця. Інший приклад - датчик руху, де рух людини можна зафіксувати і ввести в гру. Деякі ігри з датчиками руху засновані на тому, де знаходиться контролер. Причиною цього є те, що існує сигнал, який надсилається від контролера на консоль або комп’ютер, щоб виконані дії могли створювати певні рухи в грі. Інший тип ігор з датчиком руху - це стиль веб-камери, коли гравець рухається перед собою, і дії повторюються ігровим персонажем.

За визначенням, всі відеоігри призначені для виведення графіки на зовнішній відеодисплей, наприклад, телевізори з електронно-променевою трубкою, нові телевізори з рідкокристалічним дисплеєм (LCD) та вбудовані екрани, проектори чи монітори комп’ютерів, залежно від типу платформа, на якій грається. Такі функції, як глибина кольору, частота оновлення, частота кадрів та роздільна здатність екрану - це поєднання обмежень ігрової платформи та пристрою відображення та ефективності програми самої гри. Результат гри може варіюватися від фіксованих дисплеїв із використанням світлодіодних або рідкокристалічних елементів, текстових ігор, двовимірної та тривимірної графіки та дисплеїв доповненої реальності.

Графіка гри часто супроводжується звуком, виробленим внутрішніми динаміками на ігровій платформі або зовнішніми динаміками, прикріпленими до платформи, відповідно до програмування гри. Сюди часто входять звукові ефекти, пов’язані з діями гравця для забезпечення звукового зворотного зв’язку, а також фонова музика для гри.

Деякі платформи підтримують додаткові механізми зворотного зв'язку з гравцем, якими може скористатися гра. Це найчастіше тактильна технологія, вбудована в ігровий контролер, наприклад, що викликає тремтіння контролера в руках гравця для імітації тремтячого землетрусу, що відбувається в грі.

Жанри

Відеоігри, як і більшість інших засобів масової інформації, можна класифікувати за жанрами. Однак, на відміну від фільму чи телебачення, які використовують візуальні або розповідні елементи, відеоігри, як правило, класифікуються за жанрами на основі їхньої ігрової взаємодії, оскільки це основний засіб взаємодії з відеоігрою. [27] [28] [29] Розповідь не впливає на ігровий процес; гра-шутер - це все-таки гра-шутер, незалежно від того, відбувається вона у світі фантазії чи в космічному просторі. [30] [31]

Зазвичай жанрові назви описують себе з точки зору типу ігрового процесу, наприклад, бойовик, рольова гра або стрілянина, хоча деякі жанри мають похідні від впливових творів, які визначили цей жанр, наприклад, roguelikes від Rogue, [32] Клони Grand Theft Auto з Grand Theft Auto III, [33] та гра Battle Royale з фільму Battle Royale. [34] Імена можуть змінюватися з часом, коли гравці, розробники та засоби масової інформації придумують нові умови; наприклад, шутерів від першої особи спочатку називали "клонами Doom" за мотивами гри 1993 р. [35] Існує ієрархія ігрових жанрів, серед яких є жанри найвищого рівня, такі як «гра-шутер» та «екшн-гра», що широко охоплюють основний стиль гри, а також декілька піджанрів конкретної реалізації, наприклад, в шутері від першої особи і третього -особистий стрілець. Існують також деякі міжжанрові типи, які потрапляють до багатьох жанрів вищого рівня, таких як екшн-пригодницька гра.

Режим

Режим відеоігри описує, скільки гравців можуть використовувати гру одночасно. Це в першу чергу відрізняється однокористувацькими відеоіграми та багатокористувацькими відеоіграми. В рамках останньої категорії в багатокористувацькі ігри можна грати різними способами, в тому числі локально на одному пристрої, на окремих пристроях, підключених через локальну мережу, наприклад, учасників локальної мережі, або в Інтернеті через окремі підключення до Інтернету. Більшість багатокористувацьких ігор засновані на конкурентному геймплеї, але багато хто пропонує спільні та командні варіанти, а також асиметричний геймплей. Інтернет-ігри використовують серверні структури, які також можуть дозволити масовим багатокористувацьким онлайн-іграм (MMO) підтримувати сотні гравців одночасно.

Намір

Більшість відеоігор створені для розважальних цілей, але є невелика частина ігор, розроблених для додаткових цілей, крім розваги. До них належать:

Казуальні ігри

Казуальні ігри розроблені для легкої доступності, простих для розуміння ігрових процесів та швидких для сприйняття наборів правил і спрямовані на аудиторію масового ринку, на відміну від хардкорної гри. Вони часто підтримують можливість стрибати і виходити з гри на вимогу, наприклад, під час поїздок на роботу або в обідні перерви. Численні браузерні та мобільні ігри потрапляють у зону казуальних ігор, і казуальні ігри часто походять із жанрів з низькою інтенсивністю ігрових елементів, таких як матч три, прихований об'єкт, управління часом та ігри-головоломки. [36] У причинно-наслідкових іграх часто використовується механіка ігор соціальних мереж, де гравці можуть залучати допомогу друзів у своїх соціальних мережах для додаткових поворотів або ходів щодня. [37] Більш нещодавні - гіпер-казуальні ігри, які використовують ще більш спрощені правила для берегових, але нескінченно відтворюваних ігор.

Навчальні ігри

Освітнє програмне забезпечення використовувалось у будинках та класах, щоб допомогти навчати дітей та студентів, а відеоігри також були адаптовані з цих причин, і всі вони розроблені для забезпечення форми інтерактивності та розваги, пов’язаної з елементами ігрового дизайну. Існують різноманітні відмінності в їх конструкціях та тому, як вони навчають користувача. Вони широко розподілені між навчально-розважальними іграми, які, як правило, зосереджуються на цінність розваги та вивченні на повну, але навряд чи займаються критичним мисленням, та навчальними відеоіграми, які спрямовані на вирішення проблем за допомогою мотивації та позитивного підкріплення, применшуючи цінність розваги. [38] Крім того, ігри, спочатку не розроблені для навчальних цілей, потрапили в клас після випуску, часто ті, що мають відкриті світи або віртуальні пісочниці, такі як Minecraft. [39]

Серйозні ігри

Подальшими від навчальних ігор є серйозні ігри, де фактор розваги може бути доповнений, затьмарений або навіть усунутий іншими цілями гри. Ігровий дизайн використовується для підсилення нерозважальної мети гри, наприклад, використання технології відеоігор для інтерактивного світу гри або гейміфікації для навчання посилення. Навчальні ігри - це форма серйозних ігор, але інші типи серйозних ігор включають фітнес-ігри, що включають значні фізичні вправи, що допомагають підтримувати гравця у формі, тренажери польоту, що імітують пілотування комерційних та військових літаків, рекламні ігри, які побудовані на рекламі товару та новинні ігри, спрямовані на передачу конкретного пропагандистського повідомлення. [40]

Рейтинг вмісту

До відеоігор можуть застосовуватися національні та міжнародні вимоги щодо рейтингу вмісту. Як і у випадку з рейтингами кінофільмів, введення рейтингів відеоігор визначає цільову вікову групу, яку національна або регіональна комісія з рейтингу вважає доцільним для гравця, починаючи від різного віку, закінчуючи підлітками чи дорослими, дозріваючими та рідко баченими назви лише для дорослих. Більшість оглядів вмісту базується на рівні насильства як за типом насильства, так і наскільки наочно воно може бути представлене, так і за сексуальним вмістом, але також можуть бути визначені інші теми, такі як вживання наркотиків та алкоголю та азартні ігри, які можуть вплинути на дітей. Первинний ідентифікатор, заснований на мінімальному віці, використовується майже у всіх системах, а також додаткові дескриптори для ідентифікації конкретного вмісту, про який повинні знати гравці та батьки.

Нормативні акти різняться залежно від країни, але, як правило, це добровільні системи, що підтримуються практикою постачальників, із штрафом та штрафами, що виписуються органом з рейтингу видавця відеоігор за неправильне використання рейтингів. Серед основних систем оцінки вмісту можна назвати:

- Дошка оцінок програмного забезпечення для розваг (ESRB), яка контролює ігри, випущені в США. Рейтинги ESRB є добровільними та оцінюються за E (для всіх), E10 + (для всіх від 10 років), T (для підлітків), M (для дорослих) та AO (лише для дорослих). Спроби встановити рейтинг відеоігор у США згодом призвели до знакової справи Верховного суду Браун проти Асоціації торговців розвагами у 2011 році, яка постановила, що відеоігри є захищеним видом мистецтва, що є ключовою перемогою для індустрії відеоігор [41].
- Загальноєвропейська ігрова інформація (PEGI), що охоплює Великобританію, більшу частину Європейського Союзу та інші європейські країни, замінюючи попередні національні системи. Система PEGI використовує вміст, оцінений на основі рекомендованих мінімальних вікових груп, які включають 3+, 8+, 12+, 16+ та 18+.
- Австралійська класифікаційна комісія (ACB) контролює рейтинги ігор та інших робіт в Австралії, використовуючи рейтинги G (Загальний), PG (Батьківські настанови), M (Зрілий), MA15 + (Дорослі в супроводі), R18 +

(Обмежений) та X (Обмежено для порнографічних матеріалів). ACB також може відмовити в присвоєнні оцінки грі (RC - Відмовлена класифікація). Рейтинги ACB застосовуються згідно із законом, і що важливо, ігри не можна імпортувати або купувати цифровим способом в Австралії, якщо вони не отримали рейтинг або отримали рейтинг RC, що призвело до ряду помітних заборонених ігор.

- Організація рейтингу комп'ютерних розваг (CERO) оцінює ігри для Японії. Їх рейтинги включають A (для всіх вікових груп), B (від 12 років і старше), C (від 15 років), D (від 17 років) та Z (від 18 років і старше).

Крім того, основна система контенту працює над створенням Міжнародної коаліції вікових рейтингів (IARC), засобу впорядкування та вирівнювання системи оцінки вмісту між різними регіонами, так що видавцеві потрібно буде лише пройти огляд рейтингу вмісту для одного постачальника та використовуйте перехід IARC для підтвердження рейтингу вмісту для всіх інших регіонів.

Деякі країни мають ще більш обмежувальні правила, пов'язані з політичним чи ідеологічним змістом. Примітно, що сегмент відеоігор Китаю в основному ізольований від решти світу через цензуру уряду, і всі опубліковані там ігри повинні дотримуватися суворого урядового контролю, забороняючи вміст, такий як змазування іміджу комуністичної партії Китаю. Іноземні ігри, опубліковані в Китаї, часто вимагають модифікації розробниками та видавцями, щоб відповідати цим вимогам. [42]

1.2. Розробка відеоігор

Розробка відеоігор - це процес розробки відеоігор. Зусилля докладає розробник, починаючи від однієї людини і закінчуючи міжнародною командою, розподіленою по всьому світу. Розробка традиційних комерційних ігор для ПК та консолей зазвичай фінансується видавцем, і до завершення може піти кілька років. Інді-ігри зазвичай займають менше часу та грошей, і їх можуть виробляти приватні особи та менші розробники. Незалежна ігрова індустрія зростає, чому сприяє зростання доступного програмного забезпечення для розробки ігор, таких як платформа Unity та Unreal Engine [1], та нових систем розповсюдження в Інтернеті, таких як Steam та Uplay, а також ринку мобільних ігор для Android та пристрої iOS.

Перші відеоігри, розроблені в 1960-х роках, зазвичай не комерціалізувались. Вони вимагали, щоб працювали комп'ютери головного комп'ютера, і вони не були доступні для широкої громадськості. Розробка комерційних ігор розпочалася в 70-х роках з появою відеоігрових консолей першого покоління та ранніх домашніх комп’ютерів, таких як Apple I. У той час, завдяки низьким витратам і низьким можливостям комп’ютерів, самотній програміст міг розробити повний і повний гра. Однак наприкінці 80-х та 90-х років постійно зростаюча обчислювальна потужність комп’ютера та підвищені очікування від геймерів ускладнювали для однієї людини створення звичайної консолі чи гри на ПК. Середня вартість виробництва відеоігри Triple-A повільно зростала з 1–4 млн. Доларів США у 2000 р. До понад 5 млн. Дол. США у 2006 р., а потім до понад 20 млн. Дол. США до 2010 р.

Поширені комерційні ігри для ПК та консолі, як правило, розробляються поетапно: по-перше, у попередньому виробництві, питчі, прототипи та документи на ігровий дизайн; якщо ідея схвалена, а розробник отримує фінансування, то починається повномасштабна розробка. У розробці повної гри зазвичай бере участь команда з 20–100 осіб з різними обов’язками, включаючи дизайнерів, художників, програмістів та тестувальників.

Ігри створюються в процесі розробки програмного забезпечення. [2] Ігри розробляються як креативна торгова точка [3] та для отримання прибутку. [4] Створення ігор вважається як мистецтвом, так і наукою. [5] [6] Розробка зазвичай фінансується видавцем [7]. Добре зроблені ігри легше приносять прибуток. [5] Однак важливо оцінити фінансові потреби гри, [8] такі як витрати на розробку окремих особливостей. [9] Відсутність чітких наслідків очікувань гри може призвести до перевищення виділеного бюджету. [8] Насправді більшість комерційних ігор не приносять прибутку. [10] [11] [12] Більшість розробників не можуть дозволити собі змінити свій графік розробки посередині і вимагають оцінки своїх можливостей за допомогою наявних ресурсів до виробництва. [13]

Ігрова індустрія вимагає інновацій, оскільки видавці не можуть отримати прибуток від постійного випуску повторюваних продовжень та імітацій. [14] [нейтралітет оскаржується] Щороку відкриваються нові незалежні девелоперські компанії, а деяким вдається розробляти хітові заголовки. Подібним чином багато розробників закриваються, оскільки вони не можуть знайти видавничий контракт або їх виробництво не приносить прибутку. Важко заснувати нову компанію через високі початкові інвестиції. [16] Проте зростання ринку казуальних і мобільних ігор дозволив розробникам із меншими командами вийти на ринок. Як тільки компанії стануть фінансово стабільними, вони можуть розширюватись і розробляти великі ігри. [15] Більшість розробників починають з малого і поступово розширюють свій бізнес. [16] Розробник, який отримує прибуток від успішного титулу, може накопичувати капітал для розширення та реструктуризації своєї компанії, а також терпіти більше невдалих термінів. [17]

Середній бюджет розвитку мультиплатформенної гри становить 18-28 мільйонів доларів США, а гучні ігри часто перевищують 40 мільйонів доларів. [18]

На початку ери домашніх комп’ютерів та відеоігор на початку 1980-х років один програміст міг вирішувати майже всі завдання розробки гри - програмування, графічний дизайн, звукові ефекти тощо [19] [20] [21] Розробка гри може зайняти лише шість тижнів. [20] Однак високі очікування та вимоги користувачів [20] сучасних комерційних ігор значно перевищують можливості одного розробника і вимагають розподілу відповідальності. [22] Команда з понад ста людей може працевлаштуватися на повний робочий день для одного проекту. [21]

Розробка, виробництво або дизайн ігор - це процес, який починається з ідеї чи концепції. [23] [24] [25] [26] Часто ідея заснована на модифікації існуючої ігрової концепції. [23] [27] Ідея гри може входити в один або кілька жанрів. [28] Дизайнери часто експериментують з різними жанровими комбінаціями. [28] [29] Дизайнер ігор зазвичай пише початковий документ із пропозицією гри, який описує основну концепцію, ігровий процес, перелік функцій, налаштування та історію, цільову аудиторію, вимоги та графік і, нарешті, оцінку персоналу та бюджету. [30] Різні компанії мають різні офіційні процедури та філософію щодо дизайну та розробки ігор. [31] [31] [32] Не існує стандартизованого методу розробки; однак спільні риси існують. [32] [33]

Розробник ігор може варіюватися від окремо взятої особи до великої багатонаціональної компанії. Існують як незалежні, так і видавничі студії. [34] Незалежні розробники покладаються на фінансову підтримку видавця ігор. [35] Зазвичай їм доводиться розробляти гру від концепції до прототипу без зовнішнього фінансування. Потім офіційна пропозиція гри надходить видавцям, які можуть фінансувати розробку гри від декількох місяців до років. Видавець зберігав би ексклюзивні права на розповсюдження та реалізацію гри і часто володів правами інтелектуальної власності на ігрову франшизу. [34] Компанія видавця може також володіти компанією розробника [34] [36], або вона може мати внутрішню студію (и) розвитку. Як правило, видавцем є той, хто володіє правами інтелектуальної власності гри. [11]

Усі, крім найменших компаній-розробників, працюють одразу над кількома назвами. Це необхідно через час, що пройшов між доставкою гри та отриманням виплат роялті, який може становити від 6 до 18 місяців. Невеликі компанії можуть структурувати контракти, вимагати авансових платежів за роялті, використовувати розподілене програмне забезпечення, наймати працівників, що працюють за сумісництвом, та використовувати інші методи для задоволення вимог до заробітної плати.

Виробники консолей, такі як Microsoft, Nintendo або Sony, мають стандартний набір технічних вимог, яким гра повинна відповідати, щоб бути схваленою. Крім того, концепція гри повинна бути схвалена виробником, який може відмовити у затвердженні певних назв. [38]

Для завершення більшості сучасних ПК чи консольних ігор потрібно від трьох до п'яти років. [Потрібне цитування], де як мобільна гра може бути розроблена за кілька місяців [39]. На тривалість розробки впливає ряд факторів, таких як жанр, масштаб, платформа розробки та кількість активів.

Деякі ігри можуть зайняти набагато більше часу, ніж середній час. Ганебним прикладом є "Duke Nukem Forever" 3D Realms, який, як було оголошено, буде випущений у виробництві в квітні 1997 року, а випущений через чотирнадцять років у червні 2011 року [40]. Планування гри Maxis Spore розпочалося наприкінці 1999 року; гра була випущена через дев'ять років у вересні 2008 р. Гра Prey була коротко представлена у випуску PC Gamer 1997 р., але випущена лише в 2006 р., і лише тоді у сильно зміненому вигляді. Нарешті, Team Fortress 2 розроблявся з 1998 р. До випуску 2007 р. І вийшов із заплутаного процесу розробки, який включав "ймовірно три-чотири різні ігри", за словами Гейба Ньюелла [41].

Дохід від роздрібних продажів ігор розподіляється між сторонами по ланцюжку розподілу, такими як - розробник, видавець, роздрібна торгівля, виробник та консольний роялті. Багато розробників не отримують від цього прибутку і збанкрутують. [37] Багато розробників шукають альтернативні економічні моделі за допомогою Інтернет-маркетингу та каналів розповсюдження для поліпшення прибутковості. [42], оскільки за допомогою мобільного каналу розподілу частка розробника може становити до 70% від загального доходу [39] та через Інтернет-канал розподілу майже на 100%.

1.3. Ігри-платформери

Ігри на платформі (часто спрощені як платформер, або стрибати та запускати) - жанр відеоігор та піджанр екшн-ігор. Платформери характеризуються великим використанням стрибків та підйому для навігації в оточенні гравця та досягнення своєї мети. Рівні та середовища, як правило, мають нерівну місцевість та підвісні платформи різної висоти, що вимагає використання здібностей персонажа гравця для пересування. Інші акробатичні маневри часто також враховують ігровий процес, наприклад, підйом, відхилення від таких предметів, як виноградні лози або гачки, стрибки зі стін, порив повітря, ковзання по повітрю, стрілянина з гармат або стрибки з трамплінів або батутів. Ці механіки, навіть в контексті інших жанрів, зазвичай називають платформерними. Найпоширенішим об’єднуючим елементом ігор цього жанру є вміння стрибати. Ігри, де стрибки повністю автоматизовані, такі як 3D-ігри в серії The Legend of Zelda, виходять за межі жанру.

Хоча зазвичай це асоціюється з консольними іграми, існує багато важливих ігор на платформах, випущених на відеоаркади, а також для портативних ігрових консолей та домашніх комп’ютерів.

Одного разу ігри на платформі були найпопулярнішим жанром відеоігор. На піку їх популярності, за підрахунками, від однієї чверті до однієї третини консольних ігор були платформери, [1] але з тих пір вони були витіснені шутерами від першої особи. [2] Починаючи з 2006 року, жанр став набагато менш домінуючим, представляючи двовідсоткову частку ринку порівняно з п'ятнадцятьма відсотками в 1998 році [3], однак, цей жанр все ще є комерційно життєздатним, а кількість ігор продається мільйонами одиниць. Починаючи з 2010 року, безліч нескінченних платформних платформ для мобільних пристроїв принесли новій популярності жанру.

Як випливає з назви, гра на платформі вимагає від гравця маневрувати своїм персонажем на різних платформах, щоб досягти мети. У ці ігри зазвичай грають збоку, використовуючи 2D-рух, або в 3D-камері з камерою в перспективі від першої особи або третьої особи.

Найпоширенішими варіантами руху в жанрі є ходьба, біг, стрибки, атака та сходження. Стрибки можна класифікувати як "скоєні" або "змінні"; де траєкторія скоченого стрибка не може бути змінена в повітрі, а змінна стрибка може. Залежно від гри, падіння зі значної висоти може завдати шкоди падінню, можливо, призвести до смерті. У багатьох іграх на платформі також є, здавалося б, бездонні ями або інші особливості в ігровому світі, які негайно вбивають персонажа, якщо він потрапляє в них. [4]

Окрім цих елементів, компоненти гри на платформі можуть сильно відрізнятися, включаючи різні додаткові варіанти пересування, бойові дії та інші особливості.

1.4. Платформери з одноекранним рухом

Ігри на платформі виникли наприкінці 1970-х - на початку 1980-х. Більшість, але не всі ранні приклади ігор на платформі були приурочені до статичного ігрового поля, як правило, розглядалися в профілі, і базувалися на механіці сходження між платформами, а не на стрибках.

Space Panic, аркадний реліз 1980 року від Universal, іноді зараховують як першу гру на платформі [10], хоча ця відмінність суперечлива. Хоча гравець має можливість падати, немає можливості стрибати, тому гра не задовольняє більшості сучасних визначень жанру. Однак це явно вплинуло на жанр, ігровий процес був зосереджений на підйомі по сходах між різними поверхами, що є загальним елементом у багатьох ранніх іграх на платформі. Важка гра для вивчення, Space Panic залишалася неясною як аркадна гра, але несанкціонований клон Apple Panic 1981 року став хітом для домашніх комп'ютерів.

Ще одним попередником жанру з 1980 року став божевільний альпініст Нічібуцу, в якому гравець масштабує вертикально прокручувані хмарочоси [11].

Donkey Kong, аркадна гра, створена Nintendo і випущена в липні 1981 року, була першою грою, яка дозволила гравцям перестрибувати через перешкоди та через прогалини, що робить її першим справжнім платформером. [12] [13] Він представив Маріо, сучасну ікону жанру, під назвою Jumpman. На той час Donkey Kong переносили на багато консолей та комп’ютерів, зокрема як систему, що продає пакетну гру для ColecoVision [14], а також портативну версію від Coleco в 1982 році [15]. Гра допомогла закріпити позицію Nintendo як важливої назви у галузі відеоігор на міжнародному рівні.

Наступного року "Donkey Kong" отримав продовження "Donkey Kong Jr.", а пізніше "Mario Bros" - гру на платформі, яка пропонувала спільну гру для двох гравців. Цей титул заклав основу для інших двох гравців кооперативних платформерів, таких як Fairyland Story та Bubble Bobble.

Починаючи з 1982 року, з'явилися перехідні ігри, в яких не використовувалася графіка прокрутки, але були рівні, що охоплюють кілька підключених екранів. Підводний камінь Девіда Крейна для Atari 2600 з 256 горизонтально підключеними екранами стала однією з найбільш продаваних ігор в системі і стала проривом для жанру. Smurf: Rescue in Castle of Gargamel's Castle був випущений на ColecoVision того ж року, додавши нерівну місцевість і прокручуючи каструлі між статичними екранами. Manic Miner (1983) та його продовження Jet Set Willy (1984) продовжили цей стиль багатоекранних рівнів на домашніх комп’ютерах. Розшукується: Монті Мол виграв першу в історії нагороду за найкращу гру на платформі у 1984 році від журналу Crash.

РОЗДІЛ 2. ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Мова програмування C#

C # (вимовляється як музична нота C♯, але написаний зі знаком числа) - це універсальна багатопарадігмальна мова програмування, що включає строгу типізацію, лексичну область видимості, імператив, декларативний, функціональний, універсальний об'єкт -орієнтовані (на основі класів) та компонентно-орієнтовані дисципліни програмування. Він був розроблений Microsoft приблизно в 2000 році в рамках ініціативи .NET, а потім затверджений в якості міжнародного стандарту Ecma (ECMA-334) і ISO (ISO / IEC 23270: 2018). Mono - це назва безкоштовного проекту з відкритим вихідним кодом для розробки компілятора і середовища виконання для мови. C # є одним з мов програмування, розроблених для Common Language Infrastructure (CLI).

C # був розроблений Андерсом Хейлсбергом, а його командою розробників в даний час керує Мадс Торгерсен. Остання версія - 8.0, випущена в 2019 році разом з Visual Studio 2019 версії 16.3.

Під час розробки .NET Framework бібліотеки класів спочатку були написані з використанням компілятора системи керованого коду під назвою «Simple Managed C» (SMC). У січні 1999 року Андерс Хейлсберг сформував команду для створення нової мови під назвою Cool, який позначав «C-подібна об'єктно-орієнтована мова». Microsoft розглядала збереження назви «Cool» в якості остаточної назви мови, але вирішила не робити цього з причин, пов'язаних з товарними знаками. До того часу, коли проект .NET був публічно оголошено на конференції професійних розробників в липні 2000 року, мову був перейменований в C #, а бібліотеки класів і середовище виконання ASP.NET були портований на C #.

Хейлсберг - головний дизайнер і головний архітектор C # в Microsoft, раніше займався проектуванням Turbo Pascal, Embarcadero Delphi (раніше CodeGear Delphi, Inprise Delphi і Borland Delphi) і Visual J ++. У своїх інтерв'ю і технічних статтях він заявив, що недоліки в більшості основних мов програмування (наприклад, C ++, Java, Delphi і Smalltalk) призвели до основ Common Language Runtime (CLR), що, в свою чергу, призвело до розробки сама мова C #.

Джеймс Гослінг, який створив мову програмування Java в 1994 році, і Білл Джой, співзасновник Sun Microsystems, творця Java, назвали C # «імітацією» Java; Далі Гослінг сказав, що «C # - це свого роду Java з надійністю, продуктивністю і безпекою видалений». Клаус Крефт і Анджеліка Лангер (автори книги про потоках C ++) заявили у своєму блозі, що «Java і C #» це майже ідентичні мови програмування. Нудне повторення без інновацій », « Навряд чи хто-небудь буде стверджувати, що Java або C # є революційними мовами програмування, які змінили спосіб написання програм », а« C # багато запозичив у Java - і навпаки Тепер, коли C # підтримує упаковки і розпаковування, у нас буде дуже схожа функція в Java. "У липні 2000 року Хейлсберг сказав, що C #" не є клоном Java "і" набагато ближче до C ++ "по своєму дизайну.

З часу випуску C # 2.0 в листопаді 2005 року мови C # і Java розвивалися за все більш і більш розбіжним траєкторіях, ставши двома зовсім різними мовами. Одним з перших основних відхилень стало додавання узагальнень до обох мов з абсолютно різними реалізаціями. C # використовує reification для надання «першокласних» універсальних об'єктів, які можна використовувати як будь-який інший клас, з генерацією коду, що виконується під час завантаження класу. Крім того, в C #, додано кілька основних функцій для програмування функціонального стилю, кульмінацією яких є розширення LINQ, випущені в C # 3.0, і підтримуюча його структура лямбда-виразів, методів розширення і анонімних типів. Ці функції дозволяють програмістам C # використовувати методи функціонального програмування, такі як замикання, коли це вигідно для їх застосування. Розширення LINQ і функціональний імпорт допомагають розробникам скоротити обсяг стандартного коду, який включається в загальні завдання, такі як запити до бази даних, аналіз файлу XML або пошук в структурі даних, зміщуючи акцент на реальну логіку програми, щоб поліпшити читаність і ремонтопридатність.

По своєму дизайну C # є мовою програмування, який безпосередньо відображає основну інфраструктуру спільної мови (CLI). Більшість його внутрішніх типів відповідають типам значень, реалізованих середовищем CLI. Однак специфікація мови не містить вимог до генерації коду компілятора, тобто не вказує, що компілятор C # має орієнтуватися на середовище виконання спільної мови, або генерувати загальний проміжний мова (CIL), або генерувати будь-який інший конкретний формат. Теоретично, компілятор C # може генерувати машинний код, як традиційні компілятори C ++ та Fortran.

C # підтримує строго типізовані оголошення неявних змінних з ключовим словом var, а також неявно типізовані масиви з ключовим словом new [], за яким слід ініціалізатор колекції.

C # підтримує суворий логічний тип даних, bool. Оператори, які приймають умови, такі як while і if, вимагають вирази типу, що реалізує оператор true, такого як логічний тип. Хоча C ++ також має логічний тип, він може вільно перетворюватися в цілі і з цілих чисел, а вирази, наприклад, якщо (a) вимагають тільки, щоб a був перетворений в bool, дозволяючи a бути int або покажчиком. C # забороняє цей підхід «цілочисельне значення істина або брехня» на тій підставі, що примус програмістів використовувати вирази, які повертають саме bool, може запобігти певні типи помилок програмування, таких як if (a = b) (використання присвоювання = замість рівності ==).

C # більш безпечний для типів, ніж C ++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявного перетворення між логічними значеннями і цілими числами, а також між членами перерахування і цілими числами (за винятком літерала 0, який може бути неявно перетворений в будь-який перераховується тип).

Будь кероване перетворення повинне бути чітко позначено як явне або неявне, на відміну від конструкторів копіювання C ++ та операторів перетворення, які за замовчуванням неявні.

C # має явну підтримку коваріації і контраваріантності в універсальних типах, на відміну від C ++, який має деяку ступінь підтримки контраваріантності просто завдяки семантиці повертаються типів віртуальних методи.

Учасники перерахування розміщуються у своїй області видимості.

Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні і функції.

Локальні змінні не можуть приховувати змінні вміщує блоку, на відміну від C та C ++.

C # має підтримку строго типізованих покажчиків на функції через ключове слово делегат. Як і псевдо-C ++ фреймворк Qt, в C # є семантика, зокрема навколишнє події стилю публікації-підписки, хоча C # використовує для цього делегати.

Керована пам'ять не може бути звільнена; замість цього він автоматично збирається. Збірка сміття вирішує проблему витоків пам'яті, позбавляючи програміста від відповідальності за звільнення пам'яті, яка більше не потрібна.

На відміну від C ++, C # не підтримує множинне успадкування, хоча клас може реалізовувати будь-яку кількість інтерфейсів. Це було дизайнерське рішення провідного архітектора мови, щоб уникнути ускладнення і спростити архітектурні вимоги у всьому CLI. При реалізації декількох інтерфейсів, які містять метод з однаковою сигнатурою, і. тобто два методу з одним і тим же ім'ям, що приймають параметри одного й того ж типу в одному і тому ж порядку, C # дозволяє реалізовувати кожен метод в залежності від того, через який інтерфейс викликається цей метод, або, як Java, дозволяє реалізувати метод один раз і мати один виклик за викликом через будь-який з інтерфейсів класу.

Однак, на відміну від Java, C # підтримує перевантаження операторів. Тільки найбільш часто перевантажені оператори в C ++ можуть бути перевантажені в C #.

C # має можливість використовувати LINQ через .NET Framework. Розробник може запросити будь-який об'єкт IEnumerable <T>, документи XML, набір даних ADO.NET і бази даних SQL. [60] Використання LINQ в C # дає такі переваги, як підтримка Intellisense, потужні можливості фільтрації, безпека типів з можливістю перевірки помилок компіляції і узгодженість даних для запиту з різних джерел. Є кілька різних мовних структур, які можна використовувати з C # з LINQ, і вони є виразами запитів, лямбда-виразами, анонімними типами, неявно типізованими змінними, методами розширення і ініціалізаторами об'єктів

У серпні 2001 року корпорації Microsoft, Hewlett-Packard і Intel Corporation виступили співавторами в уявленні специфікації для C #, а також інфраструктури спільної мови (CLI) в організацію по стандартизації Ecma International. У грудні 2001 року ECMA випустила специфікацію мови ECMA-334 C #. C # став стандартом ISO у 2003 році (ISO / IEC 23270: 2003 Інформаційні технології. Мови програмування - C #). ECMA раніше прийняла еквівалентні специфікації як друге видання C # в грудні 2002 року.

У червні 2005 року ECMA схвалив видання 3 специфікації C # і оновило ECMA-334. Доповнення включали в себе часткові класи, анонімні методи, нульові типи і універсальні шаблони (щось схоже на шаблони C ++).

У липні 2005 року ECMA представила в ІСО / МЕК JTC 1 за допомогою прискореного процесу останнього стандарти та відповідні ТЗ. Цей процес зазвичай займає 6-9 місяців.

Визначення мови C # і CLI стандартизовані у відповідності зі стандартами ISO і Ecma, які забезпечують розумну та недискримінаційну ліцензійну захист від патентних претензій.

Microsoft погодилася не пред'являти позов розробникам програмного забезпечення з відкритим вихідним кодом за порушення патентів в некомерційних проектах в частині структури, охоплюваній OSP. Microsoft також погодилася не застосовувати патенти на продукти Novell щодо платять клієнтів Novell, за винятком списку продуктів, в яких явно не згадується C#.NET або реалізація Novell .NET (The Mono Project). Тим не менш, Novell стверджує, що Mono не порушує жодних патентів Microsoft. Microsoft також уклала конкретну угоду про відмову у захисту патентних прав, пов'язаних з плагіном браузера Moonlight, який залежить від Mono, якщо він отриманий через Novell

Microsoft очолює розробку еталонного компілятор C # з відкритим вихідним кодом і набору інструментів, що раніше носили кодова назва "Roslyn". Компілятор, який повністю написаний на керованому коді (C #), був відкритий, а функціональність відображена як API. Це дозволяє розробникам створювати інструменти рефакторінгу та діагностики.

Інші компілятори C # (деякі з яких включають реалізацію інфраструктури спільної мови і бібліотек класів .NET):

• Проект Mono надає компілятор C # з відкритим вихідним кодом, повну реалізацію загальномовної інфраструктури з відкритим вихідним кодом, включаючи необхідні бібліотеки інфраструктури, як вони зазначені в специфікації ECMA, і майже повну реалізацію пропрієтарних бібліотек класів Microsoft .NET до .NET 3.5. Починаючи з Mono 2.6, планів по впровадженню WPF не існує; WF планується до більш пізнього випуску; і є лише часткові реалізації LINQ to SQL і WCF.

• Проект DotGNU (в даний час припинений) також надав компілятор C # з відкритим вихідним кодом, майже повну реалізацію інфраструктури спільної мови, включаючи необхідні бібліотеки інфраструктури, як вони зазначені в специфікації ECMA, і підмножина деяких залишилися пропрієтарних класів Microsoft .NET. бібліотеки .NET 2.0 (не документовані або не включені в специфікацію ECMA, але включені в стандартний дистрибутив Microsoft .NET Framework).• Загальна мовна інфраструктура загального ресурсу Microsoft під кодовою назвою «Rotor» забезпечує реалізацію спільного джерела середовища CLR і компілятора C #, ліцензованого тільки для освітніх і дослідницьких цілей, а також підмножина необхідних бібліотек інфраструктури Common Language Infrastructure в специфікації ECMA (вгору в C # 2.0 і підтримується тільки в Windows XP).

C # підтримує сильно набрані неявні декларації змінних із ключовим словом var та неявно набрані масиви з ключовим словом new [], за яким слідує ініціалізатор колекції.

C # підтримує строгий логічний тип даних bool. Для операторів, які приймають умови, наприклад, while і if, потрібен вираз типу, що реалізує істинний оператор, наприклад, логічний тип. Хоча C ++ також має булевий тип, його можна вільно перетворювати в цілі числа та з них, а такі вирази, як if (a), вимагають лише того, щоб a конвертувався в bool, дозволяючи a бути int або покажчиком. C # забороняє цей підхід "цілочисельне значення істинне або хибне" на тій підставі, що примушування програмістів використовувати вирази, які повертають точно bool, може запобігти певним типам помилок програмування, наприклад if (a = b) (використання assignment = замість рівності ==).

C # є більш безпечним для типу, ніж C ++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Ніяких неявних перетворень не відбувається між логічними і цілими числами, а також між членами перерахування та цілими числами (за винятком літералу 0, який можна неявно перетворити на будь-який перерахований тип). Будь-яке визначене користувачем перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C ++ та операторів перетворення, які обидва є неявними за замовчуванням.

C # має явну підтримку коваріації та контраваріації у загальних типах, на відміну від C ++, який має певний ступінь підтримки контраваріації просто через семантику типів повернення у віртуальних методах.

Члени перерахування розміщуються у власному обсязі.

Мова C # не дозволяє використовувати глобальні змінні або функції. Усі методи та члени повинні бути оголошені в класах. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

Локальні змінні не можуть затінювати змінні блоку, що охоплює, на відміну від C та C ++.

Метод у C # - це член класу, який можна викликати як функцію (послідовність вказівок), а не просто можливість зберігання вартості властивості класу. Як і в інших синтаксично подібних мовах, таких як C ++ та ANSI C, підпис методу - це декларація, що містить у порядку: будь-які необов'язкові ключові слова доступності (наприклад, приватні), явну специфікацію його типу повернення (наприклад, int, або ключове слово void, якщо значення не повертається), ім'я методу і, нарешті, послідовність специфікацій параметрів, розділених комами, що складається з типу параметра, його офіційного імені та, за бажанням, значення за замовчуванням, яке використовується, коли надається. Деякі конкретні типи методів, такі як ті, які просто отримують або встановлюють властивість класу шляхом повернення значення або присвоєння, не вимагають повного підпису, але в загальному випадку визначення класу включає повну декларацію підпису його методів.

Як і C ++, і на відміну від Java, програмісти C # повинні використовувати ключове слово модифікатор області віртуального, щоб дозволити перевизначення методів підкласами. [61]

Методи розширення в C # дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати в цьому об'єкті та його похідних.

Динамічний тип забезпечує прив'язку методів часу виконання, дозволяючи виклики методів, подібних до JavaScript, та композицію об'єктів під час виконання.

C # підтримує сильно введені покажчики функцій через делегат ключового слова. Подібно до псевдо-сигналу та слота Qt фреймворку, C # має семантику, яка оточує події стилю опублікувати-підписатись, хоча C # використовує для цього делегатів.

C # пропонує схожі на Java синхронізовані виклики методів через атрибут [MethodImpl (MethodImplOptions.Synchronized)] та має підтримку взаємовиключних блокувань за допомогою блокування ключового слова.

C# була обрана основною мовою розробки даного проекту з огляду на усе вищеописане, а саме — на функціонал, який підтримує дана мова, операційні системи, які підтримуються та простота вивчення.

Загалом — функціонал C# найкраще підходить під дану розробку, за рахунок простої реалізації основних принципів ООП та підтримки Unity.

2.2. Середовище розробки Visual Studio

Microsoft Visual Studio - це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп’ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків. Visual Studio використовує платформи Microsoft для розробки програмного забезпечення, такі як API Windows, Windows Forms, Foundation Presentation Foundation, Windows Store та Microsoft Silverlight. Він може створювати як власний код, так і керований код.

Visual Studio включає редактор коду, що підтримує IntelliSense (компонент доповнення коду), а також рефакторинг коду. Інтегрований налагоджувач працює як налагоджувач на рівні джерела, так і налагоджувач на рівні машини. Інші вбудовані інструменти включають кодовий профілер, конструктор для побудови програм GUI, веб-дизайнер, дизайнер класів та дизайнер схем бази даних. Він приймає плагіни, які покращують функціональність майже на кожному рівні - включаючи додавання підтримки для систем керування джерелами (наприклад, Subversion і Git) та додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для мов, що задаються доменом або набори інструментів для інших аспектів розробки програмного забезпечення життєвий цикл (як клієнт Azure DevOps: Team Explorer).

Visual Studio підтримує 36 різних мов програмування та дозволяє редактору коду та налагоджувачу підтримувати (в різній мірі) майже будь-яку мову програмування за умови існування послуги, що залежить від мови. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML та CSS. Підтримка інших мов, таких як Python, Ruby, Node.js та M серед інших, доступна через плагіни. Java (і J #) підтримувалися в минулому.

Найбільш основне видання Visual Studio, спільноти, доступне безкоштовно. Гасло для видання Visual Studio Community - "Безкоштовне повнофункціональне IDE для студентів, відкритих джерел та індивідуальних розробників".
На даний момент підтримується версія Visual Studio - 2019 рік.

Visual Studio не підтримує жодної мови програмування, рішення чи інструменту внутрішньо; натомість це дозволяє підключати функціональність, кодовану як VSPackage. Після встановлення функціональність доступна як Сервіс. IDE надає три послуги: SVsSolution, яка надає можливість перераховувати проекти та рішення; SVsUIShell, який забезпечує функцію вікон та інтерфейсу користувача (включаючи вкладки, панелі інструментів та вікна інструментів); та SVsShell, яка займається реєстрацією VSPackages. Крім того, IDE також відповідає за координацію та забезпечення зв'язку між службами. Всі редактори, дизайнери, типи проектів та інші інструменти реалізовані як VSPackages. Visual Studio використовує COM для доступу до VSPackages. SDK Visual Studio також включає в себе керовану рамку пакетів (MPF), яка є набором керованих обгортків навколо COM-інтерфейсів, які дозволяють писати пакети будь-якою мовою, сумісною з CLI. Однак MPF не забезпечує всю функціональність, що піддається інтерфейсам Visual Studio COM. Потім послуги можуть бути використані для створення інших пакетів, які додають функціональність Visual Studio IDE.

Підтримка мов програмування додається за допомогою спеціального VSPackage, який називається Language Service. Мовна служба визначає різні інтерфейси, які може реалізувати реалізація VSPackage, щоб додати підтримку різних функціональних можливостей. Функції, які можна додати таким чином, включають забарвлення синтаксису, завершення оператора, відповідність дужок, підказки інформації про параметри, списки членів та маркери помилок для складання фону. Якщо інтерфейс буде реалізований, функціонал буде доступний для мови. Мовні послуги реалізуються на мовній основі. Реалізації можуть повторно використовувати код з аналізатора або компілятора для мови. Мовні сервіси можуть бути реалізовані як у рідному коді, так і керованому коді. Для нативного коду можуть бути використані або рідні інтерфейси COM, або Babel Framework (частина Visual Studio SDK). Для керованого коду MPF включає обгортки для написання сервісів керованої мови.

Visual Studio не включає вбудовану підтримку управління джерелами, але вона визначає два альтернативних способи інтеграції систем управління джерелами з IDE. VSPackage управління джерелом може забезпечити власний індивідуальний інтерфейс користувача. Навпаки, плагін управління джерелом за допомогою MSSCCI (Microsoft Source Code Control Interface) надає набір функцій, які використовуються для реалізації різних функцій управління джерелом, зі стандартним інтерфейсом користувача Visual Studio. MSSCCI вперше використовувався для інтеграції Visual SourceSafe з Visual Studio 6.0, але згодом був відкритий через SDK Visual Studio. Visual Studio .NET 2002 використовував MSSCCI 1.1, а Visual Studio .NET 2003 використовували MSSCCI 1.2. Visual Studio 2005, 2008 та 2010 використовує MSSCCI версії 1.3, яка додає підтримку перейменування та видалення розповсюдження, а також асинхронного відкриття.

Visual Studio підтримує запуск декількох екземплярів середовища (кожен зі своїм набором VSPackages). Екземпляри використовують різні вулики реєстру (див. Визначення MSDN терміна "вулик реєстру" у сенсі, що використовується тут) для зберігання конфігураційного стану та диференціюються їх AppId (ідентифікатор програми). Екземпляри запускаються специфічним для AppId .exe, який вибирає AppId, встановлює кореневий вулик та запускає IDE. VSP-пакети, зареєстровані для одного AppId, інтегруються з іншими VSP-пакетами для цього AppId. Різні випуски продуктів Visual Studio створені за допомогою різних додатків. Продукти випуску Visual Studio Express встановлюються разом із власними AppIds, але продукти Standard, Professional та Team Suite мають однаковий AppId. Отже, видання Express можна встановити поряд з іншими виданнями, на відміну від інших видань, які оновлюють ту саму установку. Професійне видання включає в себе набір VSPackages у стандартному виданні, а командний набір включає суперсет VSPackages в обох інших виданнях. Система AppId використовується Visual Studio Shell в Visual Studio 2008.

Дане середовище розробки було обрано по декількох критеріях:

- Підтримка мови C#
- Інтуїтивно зрозумілий інтерфейс

Так як усі критерії задовільнено — середовищем розробки було обрано саме Visual Studio 2019.

2.3. Ігровий двигун Unity

Unity - це крос-платформний ігровий двигун, розроблений Unity Technologies, вперше анонсований і випущений в червні 2005 року на всесвітній конференції розробників Apple Inc. як ексклюзивний ігровий движок для Mac OS X. Станом на 2018 рік двигун був розширений для підтримки більш ніж 25 платформ. Механізм може бути використаний для створення тривимірних, двовимірних, ігор у віртуальну реальність та доповнену реальність, а також моделювання та іншого досвіду. Двигун був прийнятий на виробництво за межами відеоігор, таких як кіно, автомобілебудування, архітектура, машинобудування та будівництво.

З моменту запуску було випущено кілька основних версій Unity. Остання стабільна версія, 2020.1.12, вийшла в листопаді 2020 року.

Ігровий движок Unity був запущений в 2005 році, маючи на меті "демократизувати" розробку ігор, зробивши її доступною для більшої кількості розробників. Наступного року Unity був визнаний віце-місцем у категорії «Найкраще використання графіки Mac OS X» на Apple Design Awards від Apple Inc. Спочатку Unity було випущено для Mac OS X, пізніше додана підтримка Microsoft Windows та веб-браузерів.

Unity 2.0 вийшов у 2007 році з приблизно 50 новими функціями. Випуск включав оптимізований двигун місцевості для детальних 3D-середовищ, динамічні тіні в режимі реального часу, спрямовані ліхтарі та прожектори, відтворення відео та інші функції. Випуск також додав функції, завдяки яким розробники могли легше співпрацювати. Він включав мережевий рівень для розробників для створення багатокористувацьких ігор на основі протоколу користувача датаграми, що пропонує переклад мережевих адрес, а також державні синхронізації та віддалених процедурних дзвінків.

Коли Apple запустила свій App Store у 2008 році, Unity швидко додала підтримку iPhone. Протягом декількох років двигун був безперечним на iPhone, і він став добре відомим розробникам ігор для iOS.

Unity 3.0, запущений у вересні 2010 року, має функції, що розширюють графічні можливості двигуна для настільних комп’ютерів та відеоігор. На додаток до підтримки Android, Unity 3 включав інтеграцію інструменту Beast Lightmap від Illuminate Labs, відкладеного рендерингу, вбудованого редактора дерев, власного візуалізації шрифтів, автоматичного УФ-відображення та аудіофільтрів, серед іншого.

У 2012 році VentureBeat писав: "Нечисленні компанії внесли свій внесок у потік ігор, що виробляються самостійно, як Unity Technologies. Понад 1,3 мільйона розробників використовують його інструменти для створення графіки gee-whiz на своїх консолях iOS, Android , ПК та веб-ігри. Unity хоче стати двигуном для мультиплатформенних ігор, період ". Опитування журналу Game Developer у травні 2012 року вказало Unity як головний ігровий движок для мобільних платформ. У листопаді 2012 року Unity Technologies представила Unity 4.0. Ця версія додала підтримку DirectX 11 та Adobe Flash, нові інструменти анімації під назвою Mecanim та доступ до попереднього перегляду Linux.

Facebook інтегрував комплект розробки програмного забезпечення для ігор, що використовують ігровий движок Unity, у 2013 році. Це включало інструменти, які дозволяли відстежувати рекламні кампанії та глибокі посилання, де користувачі отримували пряме посилання від публікацій у соціальних мережах до певних частин в іграх та простий обмін зображеннями в грі. У 2016 році Facebook розробив нову ігрову платформу для ПК з Unity. Unity забезпечував підтримку ігрових платформ Facebook, і розробники Unity могли швидше експортувати та публікувати ігри на Facebook.

The Verge сказав про випуск Unity 5 2015 року: "Unity розпочато з метою зробити розробку ігор загальнодоступною. Unity 5 - це довгоочікуваний крок до цього майбутнього". Завдяки Unity 5 двигун вдосконалився його освітлення та аудіо. За допомогою WebGL розробники Unity могли додавати свої ігри до сумісних веб-браузерів, не потребуючи плагінів для гравців. Unity 5.0 пропонував глобальне освітлення в режимі реального часу, попередній перегляд відображення світла, Unity Cloud, нову аудіосистему та фізичний движок Nvidia PhysX 3.3. П'яте покоління механізму Unity також представило ефекти Cinematic Image Effects, щоб ігри Unity виглядали менш загальними. Unity 5.6 додав нові ефекти освітлення та частинок, оновив загальну продуктивність двигуна та додав рідну підтримку Nintendo Switch, Facebook Gameroom, Google Daydream та графічний API Vulkan. Він представив відеоплеєр 4K, здатний запускати 360-градусні відео для віртуальної реальності. Однак деякі геймери критикували доступність Unity через великий обсяг швидко випускаються ігор, опублікованих на платформі розповсюдження Steam недосвідченими розробниками. Генеральний директор Джон Рікчієлло заявив в одному з інтерв'ю, що, на його думку, це є побічним ефектом успіху Unity у демократизації розвитку ігор: "Якби я мав свій спосіб, я хотів би бачити 50 мільйонів людей, які використовують Unity - хоча я не думаю, що ми скоро прибудемо туди. Я хотів би бачити, як їх використовують діти середніх шкіл та коледжів, люди поза основною галуззю. Я думаю, що сумно, що більшість людей є споживачами технологій, а не творцями. Світ краще місце, коли люди знають, як творити, а не просто споживати, і саме це ми намагаємось просувати ".

У грудні 2016 року компанія Unity Technologies оголосила, що змінить систему нумерації версій для Unity з ідентифікаторів на основі послідовностей на рік випуску, щоб узгодити версію з їх частішим кадентом випуску; Тому за Unity 5.6 пішов Unity 2017. Інструменти Unity 2017 включали механізм візуалізації графіки в режимі реального часу, класифікацію кольорів та світобудування, аналітику реальних операцій та звітність про ефективність. Unity 2017.2 підкреслив плани Unity Technologies, крім відеоігор. Це включало нові інструменти, такі як Timeline, що дозволяло розробникам перетягувати анімацію в ігри, та Cinemachine, розумну систему камер в іграх. Unity 2017.2 також інтегрував інструменти Autodesk 3DS Max і Maya в механізм Unity для спрощеного процесу обміну активами в процесі ітерації в грі.

У Unity 2018 представлений Scriptable Render Pipeline для розробників для створення високоякісної графіки. Це включало конвеєр високої чіткості рендеринга для консолі та ПК та легкий конвеєр рендерингу для мобільних пристроїв, віртуальної реальності, доповненої реальності та змішаної реальності . Unity 2018 також включав інструменти машинного навчання, такі як імітаційне навчання, завдяки чому ігри навчаються на реальних звичках гравців, підтримку Magic Leap та шаблони для нових розробників.

У 2019 році було додано нове посилання на мову Wolfram, що дозволило отримати доступ до функцій високого рівня мови Wolfram від Unity. Виклик об'єктів Unity з мови Wolfram також став можливим через бібліотеку UnityLink.

У червні 2020 року Unity представила студію змішаної та доповненої реальності (MARS), яка надає розробникам додаткову функціональність для генерації додатків доповненої реальності (AR) на основі правил.

Unity надає користувачам можливість створювати ігри та досвід як у 2D, так і в 3D, а движок пропонує основний API сценаріїв на C # як для редактора Unity у формі плагінів, так і для самих ігор, а також функцію перетягування. До того, як C # був основною мовою програмування, що використовувалася для движка, він раніше підтримував Boo, який був видалений випуском Unity 5 та версію JavaScript під назвою UnityScript, яка була припинена в серпні 2017 року після випуску Unity 2017.1, на користь C #.

У 2D-іграх Unity дозволяє імпортувати спрайти та вдосконалений 2D-світовий рендерінг. Для 3D-ігор Unity дозволяє визначити стиснення текстур, мипмапи та налаштування роздільної здатності для кожної платформи, яку підтримує ігровий движок, і забезпечує підтримку відображення ударних зображень, відображення відображення, відображення паралакса, оклюзія навколишнього середовища екрану (SSAO), динамічний тіні з використанням карт тіней, рендерингу до текстури та повноекранних ефектів післяобробки.

Єдність - це крос-платформний двигун. Редактор Unity підтримується на Windows, macOS та платформі Linux, тоді як сам движок в даний час підтримує створення ігор для більш ніж 25 різних платформ, включаючи мобільні, настільні, консольні та віртуальну реальністьПлатформи включають iOS, Android, Tizen, Windows, Універсальну платформу Windows, Mac, Linux, WebGL, PlayStation 4, PlayStation Vita, Xbox One, 3DS, Oculus Rift, Google Cardboard, Steam VR, PlayStation VR, Gear VR, Windows Mixed Reality , [Daydream, Android TV, Samsung Smart TV, tvOS, Nintendo Switch, Fire OS, Facebook Gameroom, Apple ARKit, Google ARCore, Vuforia, і Magic Leap.

Станом на 2018 рік Unity використовувався для створення приблизно половини мобільних ігор на ринку та 60 відсотків вмісту доповненої реальності та віртуальної реальності, включаючи приблизно 90 відсотків на нових платформах доповненої реальності, таких як Microsoft HoloLens, і 90 відсоток вмісту Samsung Gear VR. Технологія Unity є основою для більшості досвіду віртуальної реальності та доповненої реальності, і Fortune сказала, що Unity "домінує у бізнесі віртуальної реальності". Агенти машинного навчання Unity - це програмне забезпечення з відкритим кодом, завдяки якому платформа Unity підключається до програм машинного навчання, включаючи Google TensorFlow. Використовуючи спроби та помилки в Unity Machine Learning Agents, віртуальні персонажі використовують підкріплення для побудови креативних стратегій у реальних віртуальних пейзажах. Програмне забезпечення використовується для розробки роботів та самокерованих автомобілів.

Раніше Unity підтримував інші платформи, включаючи власний веб-програвач Unity, плагін веб-браузера. Однак він був припинений на користь WebGL. Починаючи з версії 5, Unity пропонує свій пакет WebGL, скомпільований до JavaScript, використовуючи двоступеневий мовний перекладач (C # на C ++ і, нарешті, JavaScript).

Unity - це набір за замовчуванням для розробки програмного забезпечення (SDK), який використовувався для відеоігрової консолі Wii U від Nintendo, а Nintendo додавала безкоштовну копію до кожної ліцензії розробника Wii U. Компанія Unity Technologies назвала це об'єднання сторонніх SDK "перш за все галуззю".

2.4. Платформа Windows

Microsoft Windows, яку зазвичай називають Windows, являє собою групу декількох власних сімейств графічних операційних систем, які всі розробляються та продаються компанією Microsoft. Кожна сім'я обслуговує певний сектор обчислювальної галузі. До активних сімейств Microsoft Windows належать Windows NT та Windows IoT; вони можуть охоплювати підсімейства (наприклад, Windows Server або Windows Embedded Compact) (Windows CE). До неіснуючих сімейств Microsoft Windows належать Windows 9x, Windows Mobile та Windows Phone.

Microsoft представила операційне середовище під назвою Windows 20 листопада 1985 року як графічну оболонку операційної системи для MS-DOS у відповідь на зростаючий інтерес до графічних користувальницьких інтерфейсів (GUI). [5] Microsoft Windows стала домінантою на світовому ринку персональних комп'ютерів (ПК) з часткою ринку понад 90%, випередивши Mac OS, яка була представлена в 1984 році. Apple сприйняла Windows як несправедливе посягання на їхні інновації у розробці графічного інтерфейсу, реалізовані на продуктах такі як Ліза та Макінтош (врешті-решт вирішено в суді на користь Microsoft у 1993 році). На ПК Windows як і раніше є найпопулярнішою операційною системою. Однак у 2014 році Microsoft визнала втрату більшості загального ринку операційних систем для Android [6] через значне зростання продажів смартфонів Android. У 2014 році кількість проданих пристроїв Windows було менше 25% від кількості проданих пристроїв Android. Однак це порівняння може бути не повністю актуальним, оскільки дві операційні системи традиційно націлені на різні платформи. Тим не менш, цифри для серверного використання Windows.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Розробка графічного інтерфейсу користувача

Запуск супроводжується запуском ігрового меню (рис. 3.1.)

Рис. 3.1. Меню

Гра починається з запуску першого рівня, гравець з’являється в крайньому лівому куті ігрового поля(рис. 3.2.)

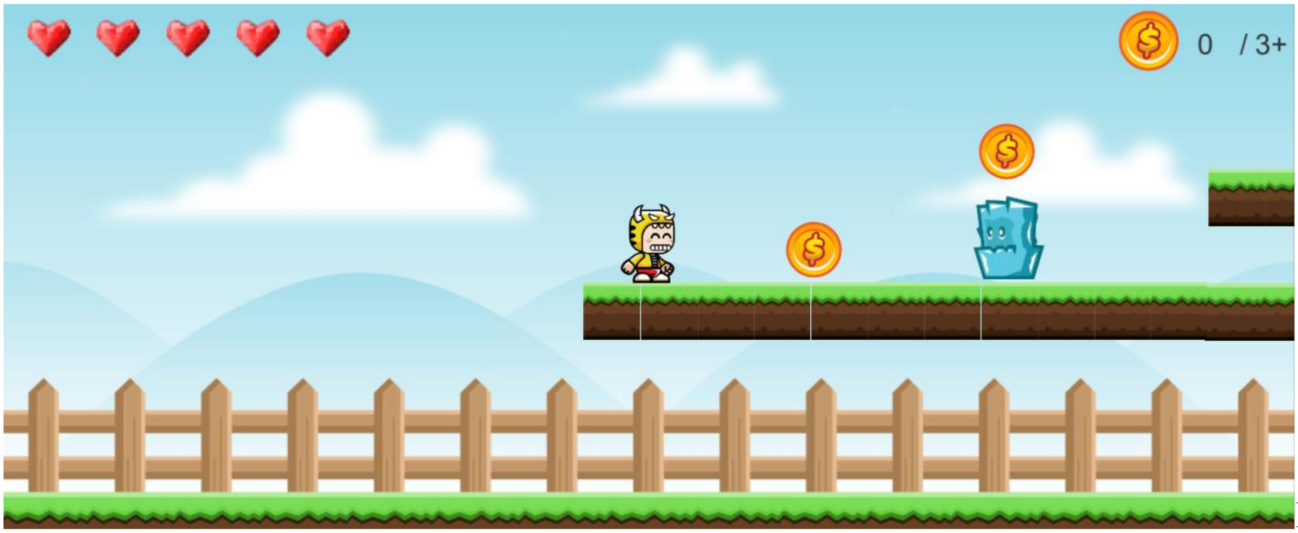


Рис. 3.2. Початок гри

Гра розроблена на базі рівнів, які збудовані і завантажені в гру, тобто без безперервної генерації. Основна задача гри — дістатися до кінця рівня, при цьому не померши, обминаючи усіх ворогів.

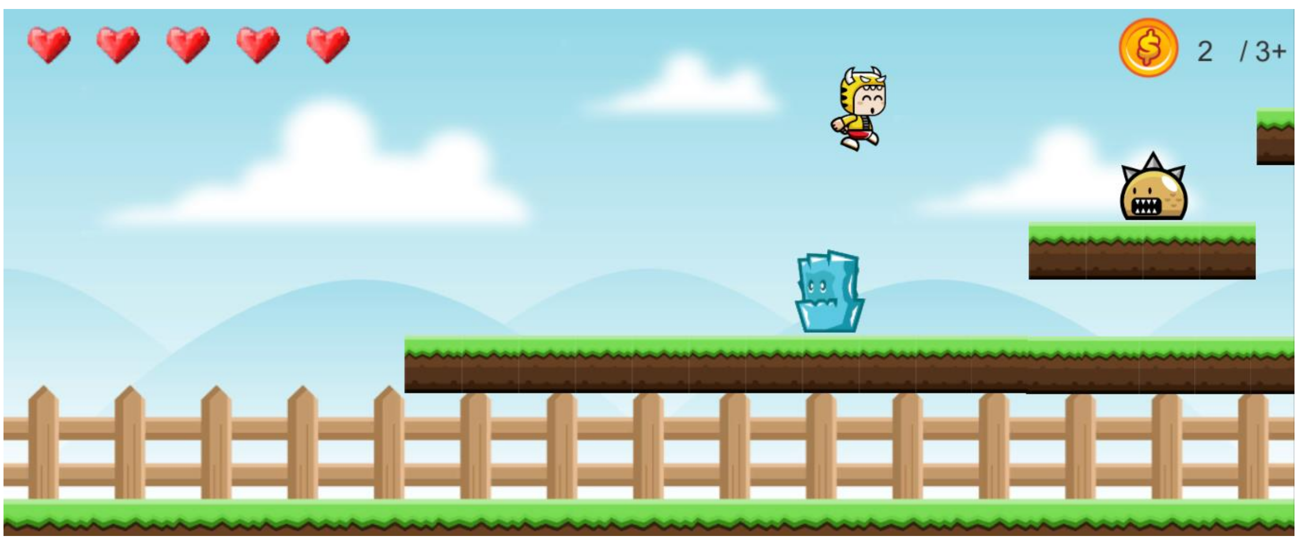


Рис. 3.3. Ігровий процес

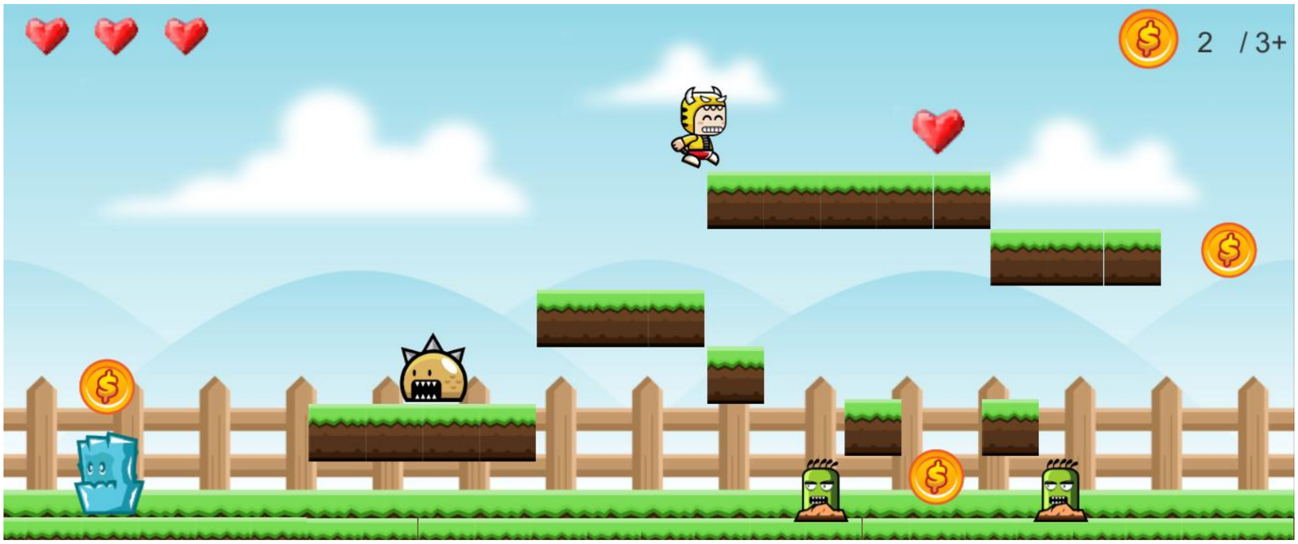


Рис. 3.4. Ігровий процес

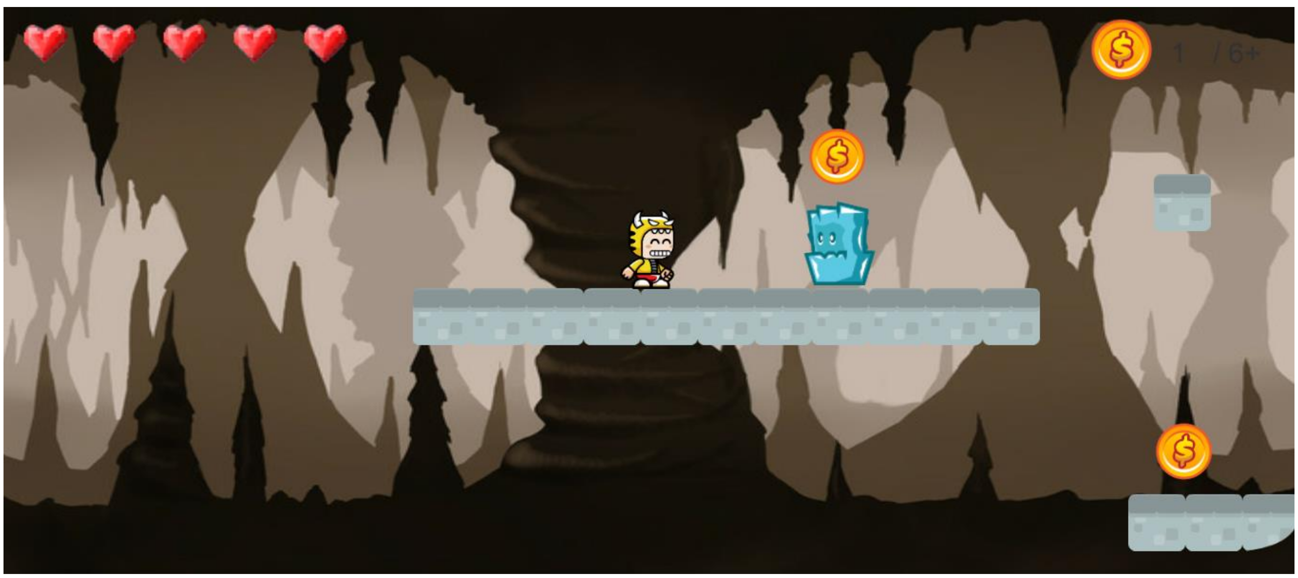


Рис. 3.5. Ігровий процес

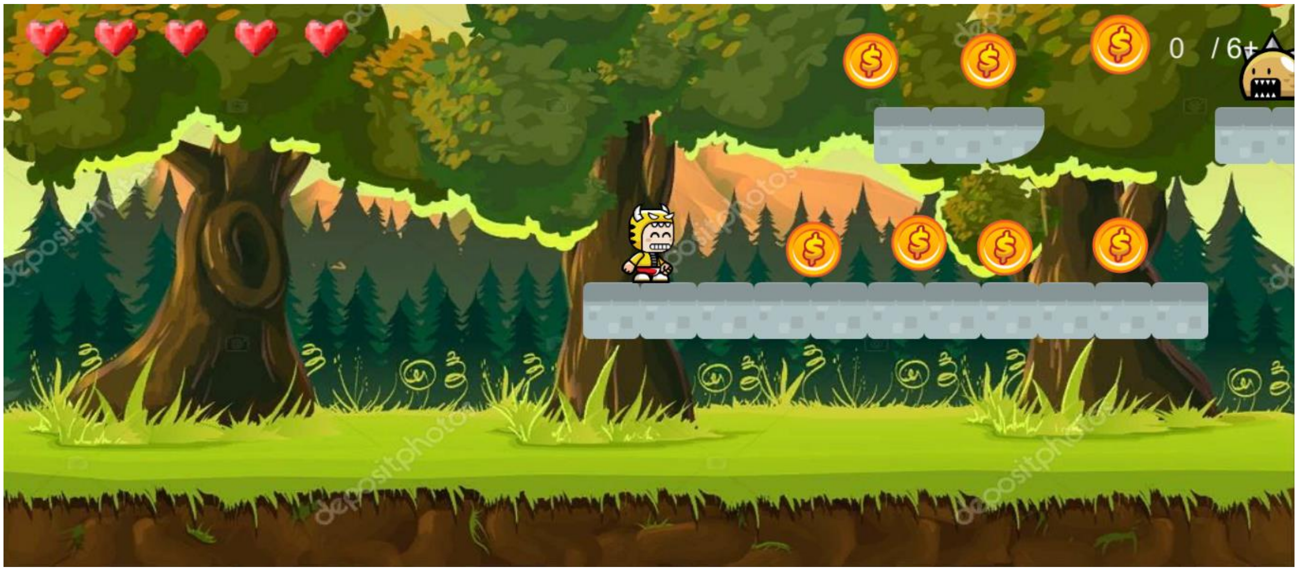


Рис. 3.6. Ігровий процес

3.2. Тестування розробленого програмного засобу

Реалізована відеогра відповідає таким вимогам:

– керування інтуїтивнозрозуміле для користувача;

– користувач може відрізнити безпечні об’єкти та пастки;

– рівні є різноманітними та з прогресуючою складністю;

– рівні можливо пройти;

– під час тестування не було знайдено місць, з гравець не зможе вийти;

– кожен об’єкт має чіткі границі, які відповідають візуальним розмірам зображення об’єкту;

– на площі порожнього фону немає невидимих об’єктів, через які не може пройти персонаж.

В процесі тестування було виявлено та виправлено такі недоліки:

– персонаж міг перестрибнути коллайдер кінця рівня, впасти в прірву після блоку фінішу та програти.

ВИСНОВКИ

У дипломній роботі виконано розробку розважальної гри Great Walk на мобільні пристрої у жанрі «платформер». В процесі виконання даної роботи проаналізовано принципи створення ігор на базі Unity 3D з використанням мови C# для створення ігрового контенту.

Під час аналізу здійснено пошук інформації в тематичній літературі та документації, відбулося вивчення можливостей ігрового рушія. Досліджено сучасний ринок подібних ігор та визначено їх переваги і недоліки.

На етапі моделювання розроблено концепт гри та поставлені вимоги до її функціоналу, що використовувалися при написанні сценаріїв.

У процесі проектування створено універсальні компоненти для розробки ігор, за допомогою яких можна керувати базовим функціоналом ігрового додатку. Також створено методи для роботи з візуалізацією контенту, які можуть використовуватись повторно і слугувати шаблоном для створення нових рівнів, чи навіть ігор.

Для досягнення поставленої мети проаналізовано актуальність розроблюваного ПЗ, розроблено функціональні вимоги до застосунку, досліджено методи плавного переходу між різними станами заданих об'єктів, створено модель та архітектуру ігрового застосунку.

Робота пройшла апробацію на конференції, теза якої опублікована: РОЗРОБКА ГРИ В UNITY3D// Розробка ігор. Збірник тез. – К.: ДУТ, 2021. — С.115–116, м. Київ, 12.02.2021.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Богданов Е.В. К вопросу о специфике аудиовизуального перевода в России и Финляндии [Электронный ресурс]. – Режим Доступа : http://old.petrsu.ru/Faculties/Balfin/EVBogdanov_2011.html
2. Горшкова В. Е. Переклад в кіно / Горшкова Е. В. – Іркутськ. : ІДУ, 2006. – 276 с.
3. Козуляев А.В. Обучение динамически эквивалентному переводу аудиовизуальных произведений: опыт разработки и освоения инновационных методик в рамках школы аудиовизуального перевода / Козуляев А.В. – М. : ВПУ, 2013. – 143 с.
4. Найда Ю. К науке переводить / Найда Ю. – Ереван. : Лингва, 2007. –231 с.
5. Найда Ю. Наука перевода / Найда Ю. – М. : Наука, 1970. –147 с.
6. Прунч Э. Пути развития западного переводоведения. От языковой асимметрии к политической / Прунч Э. – М. : Р.Валент, 2015. – 511 с.
7. Райс К. Классификация текстов и методы перевода. Вопросы теории перевода в зарубежной лингвистике / Райс К. – М. : Международные отношения, 1978. – 228 с.
8. Раренко М.Б. Основные понятия англоязычного переводоведения / Раренко М.Б. – М. : ИНИОН, 2011. – 250 с.
9. Решетньова М. Ф. Особливості перекладу фільмів з субтитрами / Решетньова М. Ф. – К. : Світ, 2006. – 144 с.
10.Толковый переводоведческий словарь / [под ред. Л.Л. Нелюбина]. – 3-е изд., перераб.. – М. : Флинта, 2003. – 742 с.
11.Шерешевский Л. Особенности локализации программного обеспечения на примере SCADA-системы WinCC / Шерешевский Л. – Астрахань. : АСУ, 2004. – 147 с.
12.Anderman G. Audiovisual Translation: Language Transfer on Screen/ Anderman G. – Belfast. : Palgrave Macmillan, 2009, – 272 p. 74

13.Anthony P. Localization: On its nature, virtues and dangers / Anthony P. – Siettle. : Science, 2005. – 341 p.

14.Averdieck E. Kinderleben oder Karl und Marie / Averdieck E. – Deutchland. : GrossBuch, 1856. – 275 p.

15.Bartolomé, A. I. H. New trends in audiovisual translation: The latest challenging modes / Bartolomé, A. I. H. – Milan. : A Journal of English and American Studies , 2005. – 104 p.

16.Bernal-Merino M. Á. On the Translation of Video Games / Bernal-Merino M. Á. – Barcelona. : JoSTrans, The Journal of Specialised Translation, 2006. – 216 p.

17.Chesterman A. Memes of Translation: The Spread of Ideas in Translation Theory / . Chesterman A. – Amsterdam. : John Benjamins Publishing,1997. – 363 p.

18.Coleridge S. T. Biographia Literaria / Coleridge S. T. – London. : British Library, 1817. – 321 p.

19.Consalvo, M. Console video games and global corporations: creating a hybrid culture [Electronic Resourse]. – Mode of access : http://dx.doi.org/10.1177/1461444806059921

20.Costales, F. A. Exploring translation strategies in video game localisation / Costales, F. A. – Milan. : Tantor Media, 2012–408 p.

21.Di Marco, F. Cultural Localization: Orientation and Disorientation in Japanese Video Games [Electronic Resource]. – Mode of access : http://www.raco.cat/index.php/tradumatica/article/viewfile/75765/96195

22.Diaz-Cintas J. Introduction The didactics of audiovisual translation / DiazCintas J. – Amsterdam. : John Benjamins Publishing Company, 2008. –189 p.

23.Esselink B. A Practical Guide to Localization / Esselink B. – Amsterdam. : John Benjamins Publishing Company, 2000. – 421 p.

24.Francisco R. Cyberculture and New Media / Francisco R. – NY. : Oxford University Press , 2009. – 176 p. 75

25.Gambier Y. Rapid and Radical Changes in Translation and Translation Studies International [Электронный ресурс]. – Режим Доступа : http://ijoc.org/index.php/ijoc/article/viewFile/3824/1570

26.Globalization and localization [Electronic resource]. – Mode of access : (www.gala-global.org).
27.Heimburg, E. Localizing MMORPGS, Perspectives on Localization / Heimburg, E. – Amsterdam. : John Benjamins Publishing Company, 2006. – 241 p.
28.Jakobson R. On linguistic aspects of translation / Jakobson R. – London. : Gregg Press, 1959. – 341 p.
29.Klinberg G. Children's Fiction in the Hands of the Translators / Klinberg G. – Munchen. : CWK Gleerup, 1986. – 481 p.
30.Mangiron C. Game localisation: unleashing imagination with «restricted» translation [Electronic resource]. – Mode of access : www.jostrans.org/issue06/art_ohagan.php
31.Maxwell-Chandler H. The Game Localization Handbook / Maxwell-Chandler H. – NY. : Charles River Media, 2005. – 348 p.
32.Mayoral R. Concept of constrained translation. non-linguistic perspectives of translation / Mayoral R. – Boston. : Meta, 1988. – 591 p.
33.O’Hagan M. Game Localization: Translating for the global digital entertainment industry / O’Hagan M. – Amsterdam. : John Benjamins Press, 2014. – 256 p.
34.O’Hagan M. Video games as a new domain for translation research / O’Hagan M. – Amsterdam. : John Benjamins Press, 2013. – 243 p.
35.Palumbo G. Key Terms in Translation Studies / Palumbo G. – London. : Continuum, 2009. – 317 p.
36.Paquin, R. Translator, Adapter, Screenwriter. Translating for the audiovisual / Paquin R. – London. : Translation Journal,1998. – 132 p.
37.Remael A. Audiovisual translation / Remael A. – Amsterdam. : John Benjamins Publishing Company, 2010. – 18 p. 76
38.Sainsbury L. The Postmodern Carnival of Children’s Literature: Necessary Playgrounds and Subversive Space in the Protean Body of Children’s Literature / Sainsbury L. – Surrey. : University Press of Surrey, 1998. – 234 p.

39.Sánchez, J. L. G. Playability: analysing user experience in video games. Behaviour & Information Technology [Electronic Resource]. – Mode of access : http://dx.doi.org/10.1080/0144929X.2012.710648
40.Schäler, R. Linguistic resources and localisation / Schäler, R. – Limerick. : John Benjamins Press, 2008. – 615 p.
41.Tan E. S. The game experience / Tan E. S. – Amsterdam. : Elsevier, 2008 – 415 p.
42.Thayer A. Localization of digital games: The process of blending for the global games market. Technical Communication / Thayer A. – Oxford. : Lancer Books, 2004. – 319 p.
43.Tonkin H. The Translator as Mediator of Culture / Tonkin H. – Amsterdam. : John Benjamins Publishing Company, 2010. – 318 p.
44.Top 100 countries by game revenues [Electronic resource]. – Mode of access : https://newzoo.com/insights/rankings/top-100-countries-by-game-revenues/
45.Translation. An Advanced Resource Book / Munday J. – NY. : Tuttle publishing, 2004. – 341 p.
46.Vieira E. «Liberating calibans»: Readings of antropofagia and Haroldo de Campos «poetics of transcreation» / Vieira E. – . : Routledge, 1999. – 317 p.
47.Wardrip-Fruin N. First Person: New Media as Story, Performance and Game / Wardrip-Fruin N. – Massachusetts. : MIT, 2004. – 354 p.
48.Wolfgang I. The Act of Reading / Wolfgang I. – London. : The John Hopkins University Press, 1978. – 398 p.
49.Wolfgang I. The Implied Reader / Wolfgang I. – London. : The John Hopkins University Press, 1974. – 421 p.

Додаток

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка гри „Great Walk” в жанрі 2D платформер з використанням ігрового двигуна UNITY та мови програмування c#

Виконав: студент 4 курсу групи ПД-42
Черниш Володимир Володимирович
Керівник роботи: к.т.н., доцент
Негоденко Олена Василівна

Київ – 2021

Актуальність

За останнє десятиліття у світі виник новий напрям – це комп'ютерні ігри. На сьогоднішній день ринок розваг є одним з найприбутковіших. З моменту початку інформаційно-технічної революції світ стрімко рухається в майбутнє, створюючи все більш досконалі комп'ютерні системи, щоб полегшити життя людини, а так само зайняти його дозвілля, на це й розрахований данний проект.

2

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** розробити ігровий додаток, з можливістю його використання на мобільній платформі.
- **Об'єкт дослідження** процес розробки 2Д ігрового додатку у жанрі платформер.
- **Предмет дослідження** методи інформаційних технологій розробки програмного забезпечення для мобільних пристроїв.

АНАЛОГИ

- *Broforce*
- *Terraria*
- *Beep*
- *Battleblock Theater*

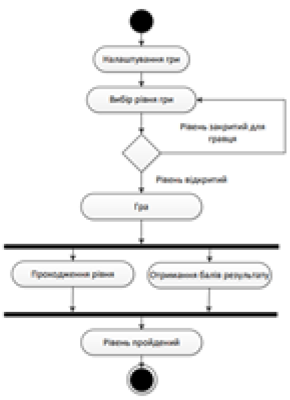
ТЕХНІЧНІ ЗАВДАННЯ

1. Провести аналіз схожих робіт.
2. Дослідити середовище розробки Unity3d та мову C#.
3. Розробити гру для проходження тестування.
4. Провести тестування.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА



Діаграма діяльності

7

КОМПОНЕНТИ

8

Ігровий процес

ВИСНОВКИ

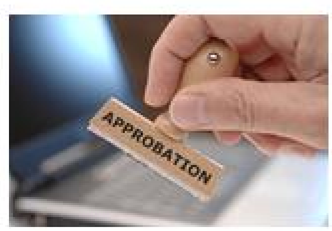
В роботі було проведено аналіз схожих робіт.

В процесі роботи над програмним додатком дипломного проекту, було розроблено та протестовано гру «Great Walk» в жанрі «2Д платформер» на платформі Unity.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Робота пройшла апробацію на конференції, теза якої опублікована:

РОЗРОБКА ГРИ В UNITY3D// Розробка ігор. Збірник тез. – К.: ДУТ, 2021. — С.115–116, м. Київ, 12.02.2021.



ДЯКУЮ ЗА УВАГУ!