МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка комп'ютерної 3D-гри
на базі рушія Unity

Виконав студент групи КН-19і
спеціальності 122 Комп'ютерні науки
Бурик Ілля Олександрович

Керівник к.т.н., доцент
Фразе-Фразенко Олексій
Олексійович

Рецензент Копиченко І.Ю.,
регіональний координатор
Програми EGAP

Одеса 2021

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕОМ – електронно-обчислювальна машина;
ІС − інформаційна система
ІТ − інформаційні технології
ОС – обчислювальна система;
СП – система прогресії
ПЕОМ – персональна електронно-обчислювальна машина;
ПЗ – програмне забезпечення;
OSI – Open Systems Interconnection;
CGI – Common Gateway Interface;
СВ – системні вимоги
КОП – компонентно-орієнтований підхід
ООП – об'єктно-орієнтоване програмування
SH – survival horror

ВСТУП

За останні 30 років комп’ютерні відеоігри придбали неймовірну популярність у всьому світі. У всі часи для людства ігри були нероздільною частиною в розвитку та прогресі. Крім того, вони приносять різноманітність до нашого повсякденного життя .

Ринок відеоігор є найбільшим сегментом світового ринку цифрового контента. Прогнози різних аналітичних компаній пророкують подальше стрімкий розвиток ринку в найближчі два роки. Згідно з прогнозами J'son and Partners Consulting, обсяг світового ринку ігор до 2021 року складе майже 130 мільярдів доларів Сполучених Штатів Америки (США), демонструючи середні щорічні темпи зростання в період 2016-2021 років на рівні 5,4%. Російський ринок відеоігор не відстає за темпами розвитку від світового, а в деякі роки навіть обганяє за обсягами виручки ринок кіноіндустрії.

Незважаючи на зміщення сегментів ринку відеоігор в сторону мобільних розробок, комп'ютерні ігри не втрачають свою популярність, так як аудиторія комп'ютерних ігор є ядром всієї ігрової аудиторії.

Якщо говорити про жанрові уподобання користувачів в одиночних іграх, то жанр action є одним з найпопулярніших. Серед його піджанрів в свою чергу найпопулярнішими є horror і shooter. Багато людей прагнуть випробувати небезпеку і відчути прилив адреналіну, а саме ці піджанри здатні викликати емоції тривоги, страху, напруженості. Незважаючи на те, що дані емоції вважаються негативними, медики вважають, що поєднання цих емоцій і усвідомлення того, що в реальності людині нічого не загрожує, адже вся дія відбувається на екрані, створює відчуття задоволення і щастя.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

Для розуміння комп'ютерних ігор нам необхідно зрозуміти, що з себе представляють ігри взагалі. Що таке «гра»?

Гра – це вид інтерактивних розваг, де гравець приймає на себе безпосередню роль в процесі. Саме завдяки інтерактивності люди отримають настільки великий спектр емоцій та досвід, які неможливо отримати ні в якому іншому медіа.

1.1 Дослідження предметної області

На сьогоднішній день комп'ютерні ігри міцно увійшли в щоденний дозвілля людей. Ігри допомагають людині пережити досвід, який він не зможе отримати в реальному житті: битися з драконом, подорожувати по світах і епохах, відчути сплеск адреналіну, уникаючи небезпечні для здоров'я і життя захоплення. Ринок відеоігор - один з найдинамічніших ринків інформаційний технологій (IT-ринків) в світі. За даними аналітичного агентства Newzoo (Newzoo's 2017 Global Games Market Report), за останні п'ять років доходи від продажів ігор в світі збільшилися на 56% за підсумками 2017 року і далі доходи будуть лише зростати.

Для початку розглянемо основні терміни і поняття з області 3D комп'ютерних ігор. Гра - це інтерактивна структура ендогенного (внутрішнього) значення, яка вимагає від гравця боротьби (подолання перешкод) для досягнення цілей

Комп'ютерна гра (відеогра) - гра, відтворена на комп'ютері (ігрові консолі), в якій зображення, які відображаються на екрані, реагують на натискання клавіш на клавіатурі комп'ютера або взаємодія з джойстиками на геймпаді.

3D гра - комп'ютерна гра, візуальний простір якої складається з тривимірних об'єктів, які обчислюються в трьох напрямках.

Жанр - рід творів в межах будь-якого мистецтва, що відрізняється особливими, тільки йому властивими сюжетними і стилістичними ознаками

Сеттінг - час і місце, в рамках яких відбувається дія книги, фільму, гри і т. д.

Розробка комп'ютерних ігор - це процес створення комп'ютерних ігор. Розробкою може займатися як одна людина, так і команда розробників. Чисельність штату команди залежить від складності проекту і його бюджету, чим більше проект і його бюджет, тим крупніше команда, яка працює над ним. Команда з розробки комп'ютерних ігор зазвичай складається з геймдизайнера, художника, програміста, дизайнера рівнів, звукорежисера і тестувальника.

Геймдизайнер - фахівець, який розробляє зміст і правила ігрового процесу, він краще за інших розуміє, як буде виглядати кінцевий продукт. У великих проектах може бути кілька.

Геймдизайнер: один головний і кілька підлеглих, кожен з яких відповідає за окрему частину гри (ігрову механіку, персонажів і т.д.).

Художник - фахівець, який оформляє візуальний стиль гри. Даний фахівець повинен володіти навичками роботи з 2D і 3D графікою або можна наймати різних художників для різних цілей. У роботу з 2D графікою входять створення концепт-артов, розробка користувальницького інтерфейсу, текстури, якщо гра виконана повністю в 2D графіку, то художник так само створює задні фони, спрайт. У роботу 3D художника входить створення моделей персонажів, оточення, анімація моделей (у великих проектах для цього може бути виділений окремий фахівець).

Програміст - фахівець, який реалізує ігрові механіки, задумані геймдизайнером (геймплей), налаштовує обробку сигналів, надходять з різних пристроїв введення (клавіатура, миша, геймпад), займається розробкою ігрового движка - ядра гри, розробляє сценарії поведінки штучного інтелекту, і інше. Тобто програміст об'єднує в єдине всі складові елементи гри і додає цим елементам інтерактивність.

Дизайнер рівнів - фахівець, який займається розробкою ігрових рівнів, не тільки їх зовнішнім виглядом, але і постановкою завдань, місій і перешкод, які треба подолати для просування по грі. Присутній в великих проектах і є складовою частиною команди. У маленьких проектах його роль виконує основний геймдизайнер і програміст.

Звукорежисер - технічний фахівець, який відповідає за музичне та звукове супровід в грі, звичайно є найманим робітником на даний конкретний проект, а не постійним членом команди.

Тестувальник - фахівець, який перевіряє гру на наявність помилок, гра повинна працювати так, як це було задумано, всі скрипти спрацьовувати чітко і в потрібний момент. Необхідно перевірити працездатність всіх функцій, сумісність гри з різними комп'ютерами, операційними системами або платформами, якщо гра замислювалася, як кроссплатформенна.

Розробка гри зазвичай включає в себе чотири основні етапи:

підготовка до виробництва, виробництво, випуск продукту і його підтримка. Розглянемо докладніше кожен етап.

На першому етапі готується передпроектна документація, розробляється концепція гри і дизайн персонажів, створюється прототип, вибираються засоби для реалізації проекту, складається план роботи над проектом.10

На етапі виробництва розробники реалізують складений раніше план, виробляють контент для наповнення гри, програмують механіки, реалізують «Вертикальний зріз» - мінімально можлива повноцінна версія гри, т. е. в даній версії гри може бракувати будь-яких дрібних деталей, бути не до кінця опрацьована графіка, але повинен бути повністю реалізований ігровий процес, це може бути одна повністю пророблена локація, якщо вона включає в себе втілення всіх фішок гри.

Етап випуску являє собою повністю завершену гру, яка надходить у продаж, на цьому етапі важливо грамотно піднести гру, підігріваючи до неї інтерес, щоб отримати максимальний прибуток. Під час підтримки збирається інформація від користувачів гри і при виникненні помилок випускаються патчі

(від англійського patch — латочка) для їх виправлення, так як не всі помилки можна обчислити в момент тестування гри, для онлайн ігор на даному етапі випускаються оновлення.

Зараз відеоігри є частиною популярної культури. Перед розробниками постає питання, як адаптувати прийоми класичного мистецтва в унікальному інтерактивному продукті? Відеоігри покладаються на ті принципи дизайну (перспективу, форму, цінність і т. д.), які класичні художники використовували для створення ілюзії, що полотно є вікном в уявний світ. У цьому їх естетична цінність для візуального оповідання. Кріс Соларскі (автор книги Drawing Basics and Video Game Art: Classic to Cutting Edge Art Techniques for Winning Video Game Design) називає картини художників класичною композицією, а відеоігри-динамічною композицією. Він пише, що динамічна композиція розкривається, якщо взяти основні лінії і форми з класичного живопису і на їх основі створити карту місцевості. Лінії, які неявно простежуються при погляді на класичну картину, зараз стають шляхами, по яких можна подорожувати по тривимірній середовище.

Шляхи всередині середовища – це лише одна частина динамічної композиції. Щоб повністю зрозуміти динамічну композицію, потрібно враховувати п'ять елементів гри і їх відносини один до одного.

Форма персонажа. Зовнішній вигляд персонажа може багато розповісти не тільки про його характер і фізичні здібності, а й про рівень розвитку персонажа. В іграх, як в кіно і літературі, персонаж не повинен починати і закінчувати сюжет в одному і тому ж стані. Наприклад, в RPG персонаж починає гру в простій ганчір'яної одязі і з одним видом зброї або взагалі без нього, але по ходу розвитку персонажа його обмундирування змінюється на більш складне — обладунки і меч – це наочно показує гравцеві, що герой не стоїть на місці, а розвивається. Це ж відноситься і до емоцій, протягом сюжету гравець повинен відчувати зміни в персонажі як в особистості, розуміти зміни в його психіці. Це можна передати через монологи героя, його взаємодія з

іншими персонажами, його реакції на навколишнє середовище і події, що відбуваються.

Анімації персонажа. Велике значення останнім часом має лицьова анімація, вона дозволяє сильніше відчути емоції персонажів і чим реалістичніше виглядає персонаж і його міміка, тим сильніше емоційний зв'язок між гравцем і героєм, так як він починає сприйматися, як жива людина. Анімація допомагає гравцеві відстежувати стан персонажа, наприклад, в грі Resident Evil головний герой стає фізично ослабленим, коли його отруюють або травмують. Лінії руху відображають різні емоції: тонкі і динамічні (криві лінії), повільні і мирні (прямі стійки і горизонталі) і агресивний (кутовий). При розробці рухів персонажа важливо вибирати лінії, які доповнюють емоції, які повинен випробувати гравець. Навколишнє середовище. Оточення персонажа є ключовою частиною динамічної композиції, тому що навколишнє середовище зазвичай займає більшу частину візуального простору в кадрі. Під навколишнім середовищем так само маються на увазі вороги і другорядні персонажі. Через форми в навколишньому середовищі можна зрозуміти чи знаходиться персонаж в безпеці або йому щось загрожує. Коли навколишнє середовище і персонаж гармоніюють один з одним, то гравець відчуває стан спокою, відчуває домашній затишок, якщо персонаж і навколишнє середовище знаходяться в стані дисонансу, то відразу відчувається загроза, що нависла над персонажем.

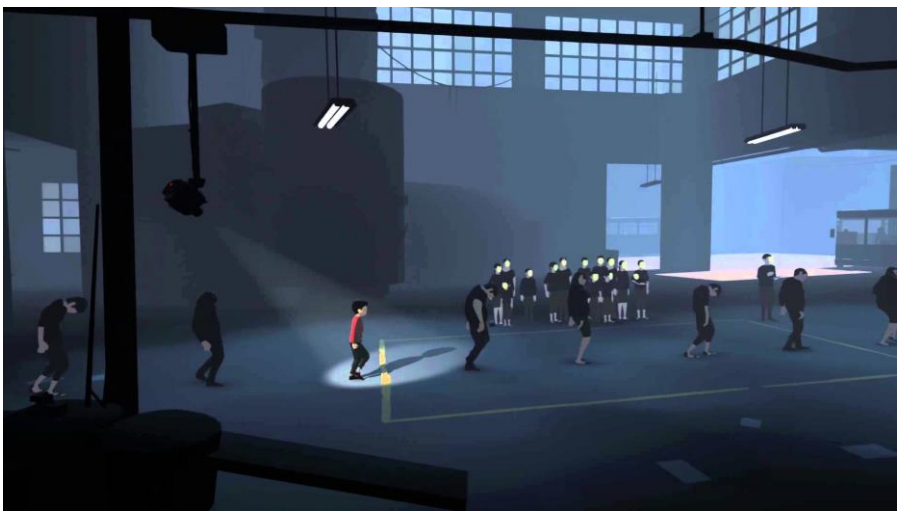

Рис. 1.1 – Скріншот гри Inside

Шляхи всередині гри. Пересування персонажа по локації дотримується певних форм для підтримки настрою гри: форми-спокій, гострі форми — динаміка і агресія. Звичайно, з поширенням ігор з відкритим світом у гравців з'явилося більше просто в переміщенні і дотримання даних форм в шляхах стає менш очевидним, проте дизайнери намагаються зберігати єдність форм навколишнього середовища і шляхів, по яких належить рухатися гравцеві.

Жести гравця. Цей аспект особливо важливий для ігор, які використовують контролери рухів у своєму ігровому процесі. Взаємозв'язок анімації персонажа на екрані і рухів гравця – це взаємодія, унікальне для відеоігор

Рис. 1.2 – Скриншот з гри Outlast

Вивчаючи комп'ютерні ігри, не можна акцентувати увагу тільки на їх естетичної складової, варто також розглянути такий термін, як технічне проектування комп'ютерних ігор. Технічне проектування передбачає розробку архітектури, написання технічного завдання і формальне планування виробничого циклу.

Архітектура комп'ютерної гри визначає з яких елементів складається гра і як ці елементи взаємодіють. Комп'ютерні ігри мають складну архітектуру і

до кожної потрібен індивідуальний підхід. Найбільш поширені це об'єктно-орієнтований (ООП) і компонентно-орієнтований підхід. В ООП об'єкти являють собою класи, від якого можуть успадковуватися інші класи або який сам успадковує когось. У КОП об'єкти – це Набори компонентів, де кожен компонент відповідає за свою функцію, наприклад, об'єкт "персонаж" складається з компонента управління персонажем, обробника зіткнень або взаємодії з іншими об'єктами, компонента анімації і так далі.

Для взаємодії різних об'єктів в грі використовуються машини станів (в україномовній літературі також використовується термін кінцевий автомат) і дерева поведінки. Логіка об'єктів в машині станів розбивається на поточний стан, дія, яка виконується в цьому стан, перехід між 2 станами і подія, яка викликає зміну станів, тобто активує перехід від одного стану в інше. Дерево поведінки-це ієрархічна деревоподібна структура, де в якості вузлів виступають блоки ігрової логіки (рисунок 4).

Дерево поведінок зазвичай складається з наступних вузлів: дія-функція, який повинна виконатися при відвідуванні даного вузла; умова для виконання, наступного за ним вузла; послідовність-виконує вкладені в неї вузли по порядку, поки який-небудь з них не завершиться невдачею або поки все не завершаться успіхом; селектор діє як послідовність, але завершується тільки при успішному проходженні всіх вкладень; ітератор використовується для проходження циклу задана кількість разів (рисунок 5).

Який саме підхід для реалізації взаємодії об'єктів вибрати залежить від конкретної гри, над якою ведеться робота. Можна також об'єднувати машину станів і дерево поведінок, так як реалізація дерева поведінок вважається більш складною для об'єктів, які отримують інформацію з поза (управляються гравцем або штучним інтелектом)

Рис. 1.3 – Машина станів

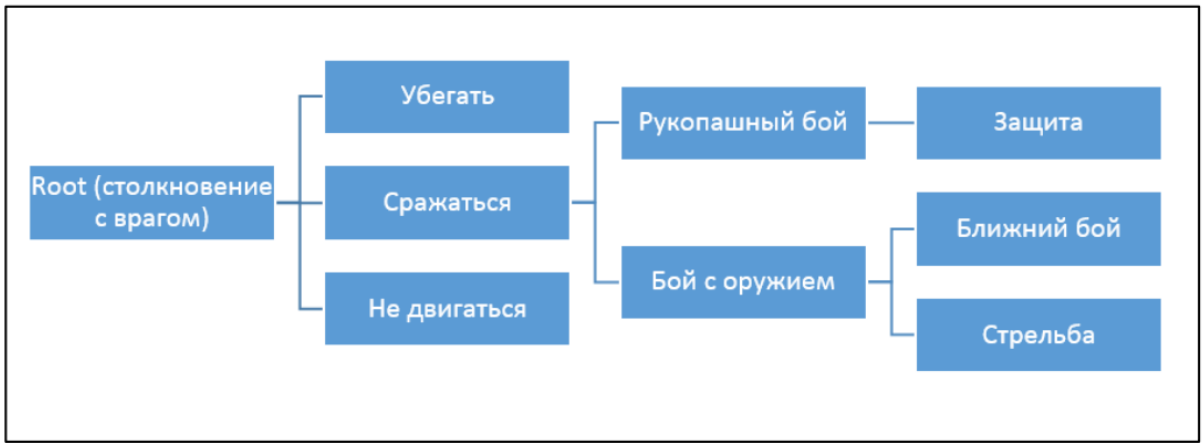

Рис. 1.4 – Дерево поведінки

Важливий етап технічного проектування гри-це підготовка документація. Особливу увагу варто приділяти проектній документації проектувальника, чим докладніше описані всі модулі, які необхідно втілити в грі, тим менше змін доведеться робити по ходу розробки, що заощадить і час, і гроші. Цей крок особливо важливий для великих команд, де над проектом

працює кілька програмістів і дизайнерів, в такому випадку проектною документацією ні в якому разі не можна нехтувати.

1.2 Огляд популярних ігор в жанрі survival horror

Жанр survival horror є досить популярним в середовищі відеоігор, особливо серед інді-розробників. Щоб не загубитися серед конкурентів і випустити цікавий продукт необхідно провести порівняльний аналіз найбільш популярних ігор в даному жанрі. Варто відзначити, що не во всіх іграх даного жанру присутня зброя і іноді від ворогів можна тільки тікати і ховатися, такі ігор розглядатися не будуть, хоч деякі і є дуже відомими, тому що в своєму проекті планується використання вогнепальної зброї для взаємодії з противниками. Для аналізу були обрані ігри, випущені протягом останніх 10 років: Siren: Blood Curse, Dying Light, Resident Evil 7, The Evil Within 2.

Siren: Blood Curse (Siren: New Translation) - ексклюзивна гра для PlayStation 3, розроблена sce Japan Studio на движку Havok і випущена Sony Computer Entertainment в 2008 році. Гра є переосмисленням першої частини серії Forbidden Siren. Графіку не можна назвати видатною навіть для 2008 року (рисунок 6). Текстури не відрізняються високою деталізацією, проте це не кидається в очі через темні локацій і слабке освітлення від ліхтаря, що грає тільки на руку атмосфері гри. Моделі людей і зомбі досить якісні і деталізовані, завдяки чому люди виглядають живими, а монстри моторошними. Управління просте, але іноді не дуже зручне. Даний невеликий недолік деякі гравці виділяють в гідність, як би дивно це не звучало. Управління давало відчуття, що головні герої прості люди, а не безсмертні супергерої, що тільки додавала почуття страху і тривоги. Гра представляє вид від третьої особи, камера знаходиться за спиною головного персонажа.

Рис. 1.5 – Геймплей Siren Blood Curse

У грі цікава подача сюжету-грати належить за декількох персонажів, всього сім чоловік, від імені кожного буде показати якийсь епізод історії, іноді дороги героїв перетинаються, що дозволяє подивитися на одну ситуацію з різних сторін. Геймплей і завдання будуть залежати від того, за кого належить грати в даний момент, наприклад, один з персонажів є десятирічна дівчинка і єдиний спосіб виживання для неї - це тікати від ворогів і не потрапляти їм на око.

Однією з найбільш цікавих особливостей гри є те, що можна дивитися на те, що відбувається очима ворога (зомбі) і одночасно продовжувати гру, в такі моменти екран ділиться на дві частини. Так само в грі є можливо перемикатися на вид від першої особи в моменти битв. До недоліків гри можна віднести те, що локації коридорні через що геймплей дуже лінійний, багато речей відразу відзначені на карті, що губить дослідницьку частину, після знаходження вогнепальної зброї вороги перестають настільки сильно лякати, так як знищуються буквально з двох пострілів

Рис. 1.6 – Скріншот з гри Siren

Dying Light – кроссплатформенна гра в жанрі survival horror і action RPG від першої особи з відкритим світом, розроблена студією Techland на движку Chrome Engine 6 і видана холдингом Warner Bros у 2015 році.

В ігровому процесі варто виділити активне використання паркуру при переміщенні в забудованому місті, через що на перший план швидше виходить жанр action. Битися потрібно не тільки з зомбі, але і з людьми за ресурси. У грі реалізована зміна дня і ночі, яка безпосередньо впливає на ігровий процес, вночі вороги більш агресивні і зустрічаються частіше і в більшій кількості. Деякі люди в грі просять допомоги головного героя, можна допомагати їм або ігнорувати, що теж позначиться на подальшому проходженні.

Також в грі присутній кооперативний режим і в цьому режимі можна грати не за людину, а за зомбі і полювати на інших гравців.

Рис. 1.7 – Dying Light

Гра має хорошу графіку, локації добре опрацьовані і деталізовані, що є великим плюсом особливо для відкритого світу. Текстури не замилені і добре виглядають, як на моделях персонажів, так і на навколишньому просторі. Моделі людей і зомбі також на високому рівні. Люди не схожі один на іншого, монстри цікаві і лякають. Хороша робота зі світлом в денний час, тіні динамічні, сонячне світло реалістичний як на вулиці, так і всередині приміщень

Resident Evil 2: Remake – кроссплатформена гра, розроблена на движку RE Engine і випущена компанією Capcom в 2019 році, є ремейком класичної частини в серії Resident Evil. Версія для PlayStation 4 підтримує режим віртуальної реальності через шолом PlayStation VR. Гра має вигляд від третьої особи і фотореалістичну графіку (рисунок 11). При створенні моделей людей і навколишніх предметів студія використовувала фотограмметрію, тобто загримована людина фотографувалася з різних сторін, і спеціальна програма створювала 3D модель на основі цих фотографій. Даний прийом, на жаль, не можна застосовувати для створення моделей монстрів, так як грим в даному випадку займе більше часу, ніж моделювання монстра в 3D редакторі.

Анімація як лицьові, так і анімації тіл живі і добре опрацьовані, тексту ри чіткі і реалістичні, грамотне динамічне освітлення тільки підкреслює всі плюси графіки.

Рис. 1.8 – Геймплей Resident Evil 2 Remake

На початку проходження гри гравець може вибрати одного з двох головних героїв, яким він почне проходити сюжетну кампанію. Друга частина сюжету гри проходиться другим персонажем відповідно. Ігровий процес за різних персонажів незначно відрізняється. Під час другого проходження шлях, пройдений персонажем, виявляється схожим, але ключові сюжетні події — іншими. Справжня кінцівка стає доступна після завершення обох сюжетних ліній. Сюжет в грі подається за допомогою кат-сцен, а також різних записок і заміток. Також гравцеві пропонується вибрати один з трьох рівнів складності, в залежності від якого змінюються одержуваний головним героєм шкоди, кількість боєприпасів, а також деякі інші ігрові механіки (наприклад, на останньому рівні складності, як і в оригінальній грі, процедура збереження гри вимагає від гравця наявність витрачаються «стрічок для друкарської машинки»).

У грі представлений різноманітний арсенал зброї, який представлений вогнепальною і метальною зброєю (наприклад, світлошумові і осколкові гранати), а також зброєю ближнього бою. Різні типи вогнепальної зброї вимагають різні види патронів. Патрони можуть бути знайдені на рівнях або створені з відповідних інгредієнтів. Також, вогнепальна зброя може бути модернізована. В якості зброї ближнього бою гравець може використовувати ніж, який з часом зношується і приходить в непридатність. Вся зброя, патрони та інші предмети зберігаються в інвентарі персонажа, який спочатку представлений 8 осередками і згодом може бути розширений за допомогою набедренних сумок, кожна з яких збільшує місткість на 2 осередки. Предмети з інвентарю можуть бути викинуті і втрачені назавжди або перекладені в один з ящиків.

Головному герою протистоять кілька видів супротивників: рядовими противниками виступають зомбі — люди і тварини. Рядові противники можуть бути уповільнені, оглушені на певний проміжок часу або вбиті. Крім звичайних супротивників протагоніста більшу частину сюжетної кампанії переслідує монстр Тиран, якого неможливо вбити. Як і в минулих частинах серії в грі присутні боси.

Ігровий процес пропонує не тільки сутички з зомбі і босами, але і дослідження рівнів, поєднане з вирішенням різних головоломок.

The Evil Within 2 – комп'ютерна гра в жанрі survival horror, розроблена компанією Tango Gameworks і випущена компанією Bethesda Softworks для платформ Microsoft Windows, PlayStation 4 і Xbox One 13 жовтня 2017 року. Продовжує історію гри The Evil Within. Пост керівника The Evil Within 2 займає Джон Йоханас (John Johnes), який працював над візуальними ефектами оригінальної гри, а також був постановником доповнень The Assignment і The Consequence. Шінджі Мікамі займає пост продюсера проекту.

Подібно його попереднику, The Evil Within 2 – гра в жанрі survival horror. Гравець бере на себе управління детективом Себастьяном Кастелланосом, який повинен спуститися в світ Юніона, щоб врятувати свою дочку, Лілі. У грі

є три режими складності, а саме звичайний, який рекомендує продюсер Шінджі Мікамі, виживання і кошмар. На початку гри персонажу надається комунікатор, який допомагає виділити цілі, ресурси і ворогів, що знаходяться в світі Юніона. У комунікаторі також будуть показані точки резонансу, в яких даються підказки щодо того, що сталося в світі гри. Гравці можуть вільно досліджувати область карти для завершення побічних завдань і пошуку ресурсів. Гравці можуть брати участь у прямій конфронтації з противниками, використовуючи зброю або ж вдаючись до скритності для безшумного усунення супротивників.

У грі є система крафта, в якій гравці можуть використовувати знайдені ресурси для виготовлення нових предметів, таких як боєприпаси. Також можна створювати предмети в будь-який час в самій грі, але для цього потрібно більше матеріалів, ніж на верстаті. Система кастомізації персонажа залишилася колишньою. Зелений гель, представлений в першій частині гри, може бути використаний для поліпшення здібностей Себастьяна, які діляться на п'ять різних гілок: здоров'я, скритність, бій, відновлення і атлетизм. Зброя в грі також можна покращувати за допомогою деталей, зібраних в ході вивчення Юніона

Рис. 1.9 – Evil Within 2

З проведеного аналізу можна зробити наступні висновки:

- в іграх жанру survival horror велику увагу потрібно приділити атмосфері, яка буде підтримувати в гравцеві почуття тривоги і страху;
- не варто зловживати так званими скримерами-прийом, використовуваний для виклику переляку за коштами гучних звуків, різких рухів, яскравих кольорів і так далі-краще взагалі відмовитися від даного прийому і сконцентруватися на створенні і підтримці почуття тривоги, так як скрімери швидко набридають — скидають напругу і через деякий час перестають лякати гравця;
- необхідно приділити увагу сюжету, так як ігор в даному жанрі на даний момент багато, із загальної маси його може виділити саме цікавий сюжет, так само грамотне обґрунтування мотивів головного героя, його характеру та історії дозволить викликати в гравця більше переживань за долю головного героя;
- присутність зброї або інших засобів і способів самозахисту є рисою жанру survival horror, яка дозволяє відрізнити даний жанр від дуже схожого на нього quest з елементом horror, де як правіло гравець може лише тікати від ворога або ворога немає взагалі, і тривога створюється лише за допомогою атмосфери і почуття невідомості;
- головною відмінною рисою жанру є мала кількість боєприпасів і аптечок, їжі і води (якщо таке передбачено геймплеєм), що є частиною виживання.

Перераховані вище висновки стали основними критеріями, на які йшов упор в розробці власної гри.

1.3 Аналіз програмного середовища розробки відеоігри

На даний момент будь-який бажаючий може самостійно почати розробляти ігри за допомогою різноманітних ігрових движків, якими

користуються не тільки початківці або інді-розробники, але і великі всесвітньо відомі компанії.

Ігровий движок – це програмне середовище, призначене для створення і розробки відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі. Движок гри включає в себе Візуалізатор, фізичний движок, звук, систему скриптів, анімацію, штучний інтелект, мережевий код, управління пам'яттю і багатопоточність .

Спочатку всі ігри створювалися з нуля, але в середині 1990-х років, після появи знаменитої гри Doom відбувся переворот у створенні ігор. Doom мав чітку архітектуру, яка легко ділилася на центральні компоненти (компонент для пересування, компонент для стрільби, компонент для обробника зіткнень і т. д.) і графічні ресурси, що утворюють ігровий світ. На основі такої архітектури стали створюватися ігрові движки, тепер компанії розробників могли використовувати один набір інструментарію для різних ігор, що скорочувало час і кошти в процесі розробки гри. У слідстві чого останнім часом ігри створюються або на загальнодоступних движках, або студії створюють движки на основі своїх раніше випущених ігор, допрацьовуючи їх в наслідку під тенденції технічного прогресу.

При створенні ігрового персонажа необхідно приділити велику увагу топології сітки моделі і побудови скелета для правильної анімації. Для початку розглянемо ці два терміни.

Топологія (від ін. - грец. τόπος-місце і λόγος-слово, вчення) – розділ математики, що вивчає явище безперервності, властивості просторів, які залишаються незмінними при безперервних деформаціях. Наприклад, зв'язність, ориентируемость. Є два основних типи інформації, необхідної для визначення полігональної сітки-геометрія і з'єднання, іншими словами, це набір точок в просторі (вершини) і набір відповідних зв'язків між точками (грані). Без інформації про зв'язки, полігональна сітка є неструктурованою, а тому і невизначеною. Введення набору граней – це крок, який, в кінцевому рахунку, актуалізує сітку і задає її характер з точки зору безперервності,

збіжності і зв'язності. Ця мережева структура і називається топологією простору. Іншими словами, топологія – це здатність сітки коректно реагувати на деформації, наприклад, при розробці міміки обличчя, одягу і рухів персонажа. Задля коректної поведінки сітки слід грамотно продумати і побудувати полігональну сітку персонажа .

Після топології варто приділити увагу скелету моделі. Він являє собою деревоподібну структуру кісток, в якій кожна подальша кістка з'єднана з попередньою, тобто повторює за нею руху і повороти з урахуванням ієрархії в скелеті. Таким чином, при русі окремої кістки рухаються і всі вершини, прив'язані до неї.

Ще один вид скелетної анімації – Анімація з ваговиками, яка являє собою більш просунутий варіант скелетної анімації, в ній кожна вершина моделі може бути пов'язана не з однією, а з декількома кістка. При цьому для кожної кістки визначається свою вагу, тобто величина вплив цієї кістки на переміщення вершини. Чим більше вага якоїсь кістки, тим сильніше вершина зміщується під її впливом.

2. ВИБІР СЕРДОВИЩА РОЗРОБКИ ТА ЖАНРУ

Складно уявити, що міг би існувати умовно безкоштовний движок, на якому реально створити комп'ютерну гру. Однак, цей движок існує. Він тривимірний, має нормальним IDE, вбудованої фізикою, аудіо-движком і прописаними можливостями мережевого мультиплейера. Движок Unity підтримує Windows, IOS, Android, операційні системи приставок Playstation, Xbox), на смартфонах і планшетах (iOS, Android, Windows Phone), на і Nintendo Wii.

Більшість інструментальних засобів для розробки ігор можна поділити на 3 группи:

1. Фреймворк - програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. Фреймворк відрізняється від поняття бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підпрограм близькою функціональності, не впливаючи на архітектуру програмного продукту і не накладаючи на неї ніяких обмежень. У той час як фреймворк диктує правила побудови архітектури додатку, задаючи на початковому етапі розробки поведінка за умовчанням, каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог.

2. Ігровий рушій - центральний програмний компонент комп'ютерних та відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі. Він забезпечує основні технології, спрощує розробку і часто дає грі можливість запускатися на декількох платформах, таких як ігрові консолі та настільні операційні системи, наприклад, GNU / Linux, Mac OS X і Microsoft Windows. Основну функціональність зазвичай забезпечує ігровий движок, що включає движок рендеринга («визуализатор»), фізичний движок, звук, систему скриптів, анімацію, штучний інтелект, мережевий код, управління пам'яттю і багатопоточність. Часто на процесі

розробки можна заощадити за рахунок повторного використання одного ігрового движка для створення безлічі різних ігор.

3. Конструктор ігор - програма, яка об'єднує в собі ігровий рушій і інтегроване середовище розробки, і, як правило, включає в себе редактор рівнів, що працює за принципом WYSIWYG. Такі програми значно спрощує

процес розробки ігор, роблячи його доступним любителям-непрограмістів, і можуть бути використані в початковому навчанні програмуванню

В роботі ми будемо досліджувати різноманітних представників всіх цих группи проте будемо приділяти більшу увагу на інструментам які мали реліз протяком останніх 5 років, оскільки світ мультимедія та ігрової індустрії дуже швидко оновлюется та прогресує, то набор інструментів 5-ти річної давності виявится неактуальним в поточних реаліях .

2.1 Ігровий рушій Unity

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

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація). Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні

ігрові об'єкти – тобто ті які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. У об'єктів з видимої геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить їх модель їх видимою.

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

При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна - буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, анімацію також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

У Unity вбудована підтримка мережі. Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі. Має вбудований генератор ландшафтів

Складно уявити, що Unity має можливості, які дозволяють зібрати блок команд в спеціальну версію для плагіна, який вбудовується в браузер. Таким чином, Ви зможете отримати у вікні браузера 3D картинку без заниження дозволу текстур і якості моделей.

Взагалі, є згадки про технології, які ставили собі за мету роботу з 3D в мережі Інтернет, зокрема ActiveWords і VRML. Але всі ці системи витіснив Flash (в доповненні з Java і Silverlight). Інші технології існують, але в даний момент, істотно пригнічені технологією Flash. Реліз третьої версії Unity насправді змусив звернути на цей движок увагу навіть гігантів, які розробляють ігри виключно в Flash. Так що ж всетаки становить суть і принципи движка Unity? Unity повноцінний ігровий движок, який

спрямований на те, що весь – це процес розробки гри (за винятком скриптинга і підготовки ігрових ресурсів) буде проводитися в окремому редакторі.

Розглянемо цей движок в порівнянні з UnrealEngine4.

Переваги Unity: – це IDE: поєднання редактора сцен (в комплексі загального редактора) з редактором ігрових об'єктів і редакторів скриптів. Додатково додаються генератори дерев і террейнов. Покращені можливості

скриптинга, а саме на відміну від вищевказаного движка, в Unity доступні три мови: JavaScript, C # і різновид Python's Boo – це кросплатформеність як уже згадувалося вище, підтримуються – це Windows, MacOS, Wii, iPhone, iPod, iPad, Android, PS3 і XBox), на смартфонах і планшетах (iOS, Android, Windows Phone), на 360, не всі з яких, звичайно, доступні у безкоштовній ліцензії. Ну і вебплагін, звичайно, забувати не варто.

Сучасний рівень графіки, здатний конкурувати з іншими двигунами. Unity, безумовно, програє UnrealEngine за кількістю реалізованих можливостей. Однак Unity володіє такими можливостями, як deferred25 освітлення, стандартний набір постпроцессінгових ефектів, SSAO, прискорена опрацювання лайтмапов. Гідним чином опрацьоване фізичний движок. Масштабованість і продуктивність. Більшу частину простих процесів движок обробляє на чудовому рівні.

Запуск будь-якої програми на Unity в веб-плагін. Невисока ціна за повну ліцензійну версію для великого веб-розробника.

Недоліки Unity:

– це закритість коду. Неможливість отримання вихідних кодів движка навіть за ліцензією.

– це неможливість доповнення фізики движка сторонніми можливостями.

Ви не зможете додати в движок сторонню фізику, або SpeedTree. Реальні мінуси складно визначити з першого погляду. Движок продуктивний, стабільний і легкий в застосуванні. У більшості нечисленних команд розробників комп'ютерних ігор основною проблемою часто ставав саме

движок. Досить складно писати з нуля єдиному програмісту в команді. Потрібен повноцінний безкоштовний движок, і потрібен він відразу, програміст починає шукати безкоштовні вирішенні (Ogre, Irrlicht). Ці двигуни не так вже й погані (Torchlight написаний на Ogre), але вони складні в освоєнні і вимагають не одного програміста, а цілої команди. Звичайно, можна звернутися до наборів типу GameMaker, але серйозну гру з його допомогою зібрати складно. Що стосується Unity, то в його випадку є вже завершений пайплайн, готовий рендерер, зібрані фізику, аудіо та мережеве взаємодія, багатомовність. Зовнішній вигляд (рис. 2.1):

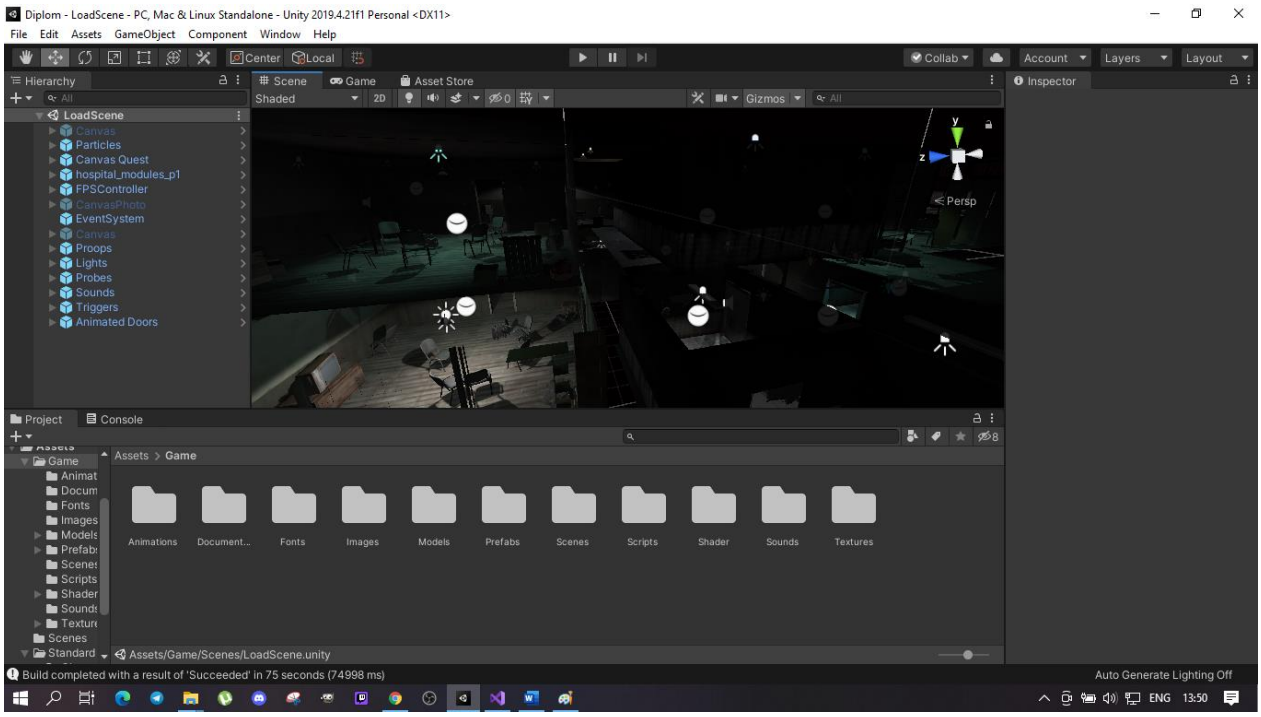

Рис. 2.1 – Рушій Unity

2.2 Мова програмування

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи.NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості,

винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхньоговикористання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

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

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

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

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах

виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

2.3 Жанр survival horror

Survival horror – жанр комп'ютерних та відео ігор. Якщо перевести його дослівно, то звучить він як «виживання в кошмарі». Він не такий древній і з'явився відносно недавно в 90-их.

Головною особливістю SH є те, що гравці, як правило, змушені пробиратися по похмурим і лякаючим локаціях населеними моторошними істотами. Способів захисту від ворогів не так багато як в шутрерах інших іграх, та й найчастіше вони не ефективні.

Здоров'я у головного героя так само мало, а поранення або маленький його запас призводить до каліцтв, персонаж починає накульгувати, екран червоніти і т. п.

Відмінні риси жанру:

- Атмосфера жаху – гравець ніби опиняється в страшному кошмарі, з якого нелегко вибратися. Все дуже похмуро і лякаюче. Дуже часто такі ігри роблять за подобою літератури або фільмів жахів.
- Складне управління-персонаж має скуті (частіше реалістичні) руху, погано бігає і б'ється, тим самим стаючи ще більш беззахисним.
- Брак боєприпасів і медикаментів – на відміну від інших жанрів, ігри Survival horror дуже сильно обмежують гравців в кількості життєво необхідних речей.

- Монстри та інші моторошні створення-противниками в таких іграх зазвичай бувають знамениті лякають створення, чий образ так само використовується в кіно і літературі (зомбі, вампіри, демони і т.д.).

Рис. 2.2 – Скріншот гри

3. ПРОЕКТУВАННЯ ГРИ «ESCAPE»

Незважаючи на приплив жіночої аудиторії в середу відеоігор, цільовим споживачем як і раніше є чоловіки (59 % гравців), віком від 18 до 35 років [2]. Любителі жанру horror зазвичай в іграх звертають увагу на атмосферу, наявність саспенса і сюжет, тому можна зробити висновок, що по моделі сегментації гравців за психологічними типами придуманої Річардом Аланом Бартлом професором Університету Ессекса, даний гравці є дослідниками (explorers). Їм цікаво вивчати ігровий світ і розкривати його таємниці, такі гравці люблять квести, цінують сюжет і захоплено вивчають механіки гри.

Проаналізувавши цільову аудиторію гри, було вирішено, що продукт повинен обов'язково мати сюжет, пропонувати гравцеві різні механіки (пошук предметів, стрільба, рішення головоломок), локації повинні бути влаштовані так, щоб по них можна було побродити, а не йти з одного пункту в інший за суворим маршрутом, для підтримки атмосфери потрібно використовувати звуковий супровід і правильно збудований світло.

3.1 Розробка концепції гри

Перше, що варто звернути увагу при розробці гри – це її концепції (ідея). Про що гра, які цілі стоять перед гравцями, який у ігри жанр, сеттинг і так далі. За час роботи над проектом концепт гри змінювався два рази. Спочатку гра планувалася, як невеликий horror, головною метою якого є розкриття історії, пошук записок, ключів. Лякати гравців повинна була атмосфера, скрімери і поява примари в кінці, тікаючи від якого потрібно було покинути будинок. Гра складалася б з однієї локації старого будинку.

Рис. 3.1 – Скріншот коридору з гри

Після вивчення планів різних будинків і котеджів, що задовольняють сюжет, був зроблений висновок, що дана локація буде занадто маленькою і гра вийде не цікавою. У слідстві чого, був трохи змінено початковий сценарій і задум. Гра все ще складалася з однієї локації, але будинок замінили на особняк. Тим не менш, геймплей залишався б досить мізерним і нудним. Тому для різноманітності механік в гру було вирішено додати вогнепальну зброю. Стало очевидно, що одного рівня для розкриття сюжету і механік мало. Гра повна зміна сценарію, в слідстві чого змінилися ворог і локації. Тепер гра набула рис жанру survival horror: наявність зброї, мізерна кількість припасів.

Рис.3.2 – Фотокамера

Незважаючи на неодноразову переробку ідеї гри, сеттінг залишався незмінним, так як гравці як правило відчувають себе затишно і комфортно в уже знайомому світі, події гри відбуваються в 21 столітті в Сполучених Штатах Америки. Жанр horror, а пізніше і survival horror, був обраний, тому це один з популярних жанрів, а так як на сьогоднішній день на ринку horror ігор представлено безліч ігор клонів таких проектів як Slender і Five Nights at Freddy's, то ігри відмінні від них по механіки привертають інтерес.

Рис. 3.3 – Головний хол з гри

Для того щоб урізноманітнити ігровий процес в гру було вирішено додати головоломки. Після невеликого опитування було вирішено робити головоломки у вигляді користувальницьких інтерфейсів.

Всього в грі п'ять головоломок:

- кодовий замок сейфа, потрібно знайти код захований в квартирі;
- кодовий замок на двері притулку, підказка в одній з кімнат
- притулок;
- головоломка, що складається з квадратів розміром 3x3 потрібно змінити колір всіх квадратів, при цьому натискання на один з них змінює колір
- відразу у декількох;
- головоломка на знаходження правильного шляху, складається з квадратів розміром 4x4, підказка в одній з кімнат притулку;

- головоломка складаються з квадратів, необхідно запалити правильну комбінацію.

3.2 Планування етапів розробки

Написання сценарію

На етапі розробки концепту гри був написаний чорновий сценарій, де був визначений головний герой, його цілі і місце дії. Тепер потрібно було розвинути цей концепт в повноцінний сценарій з репліками і діями персонажей.

На цьому етапі для пошуку фішок і ідей проглядалися фільми в жанрах horror, аналізувалися гри жанру survival horror. Так само вивчалися статті на тему написання сценарію і статті по прийомам, які використовуються в іграх і фільмах для виклику почуття страху і тривоги у глядача або гравця.

Сценарій був оформлений у вигляді таблиці в програмі Microsoft Excel, де були вказані основні події та репліки персонажа.

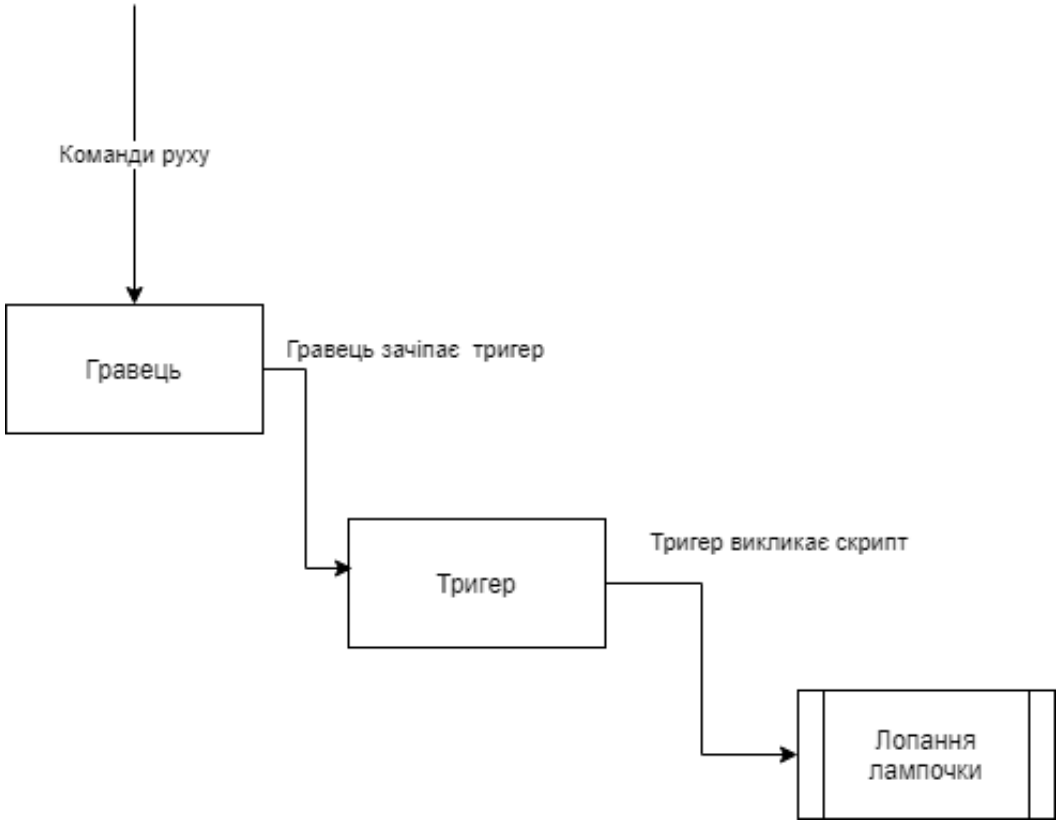

Рис. 3.4 – UML-діаграма тригера

Розробка моделей головного і другорядного персонажів

Персонажі створювався в програмі, призначеної для швидкого моделювання 3D моделей людей і зомбі - Mixamo Fuse. Дану програму можна придбати безкоштовно в інтернет-магазині Steam або купити нову версію Fuse CC в Adobe Creative Cloud. Програма не локалізована на російську мову, але має інтуїтивно зрозумілий інтерфейс, тому проблем з роботою не виникає. Створення персонажа схоже на створення персонажів в іграх, вибирається форма обличчя, тіла, рук, ніг, всі частини можна підкоригувати. Потім на модель надаватися одяг, взуття та зачіска, у цих об'єктів можна коригувати тільки текстуру (колір).

Рис. 3.5 – UML-діаграма базових можливостей персонажу

Розробка моделі ворога

Спочатку в якості ворога виступав привид жінки. Перші спроби його створення проходили так само в програмі Fuse, але в даній Програмою передбачено лише моделі живих людей і зомбі, тому модель створювалася в

програмі Daz 3D, в ній можна зробити людину болісно худим. Після в програмі Adobe Photoshop CC текстура шкіри видозмінювалася, щоб зробити її більш блідою і додати криваві бризки, рани і синці. Модель створювалася без волосся і одягу, вони прив'язувалися до моделі після в програмі 3Ds Max, там же у моделі віддалявся рідний скелет, так як він вступає в суперечності з алгоритмами mixamo.com

Моделювання локацій

Квартира і будинок дитячого притулку були змодельовані в програмі Archicad 21. Дана програма дозволяє швидко змоделювати будівлі і додати основні предмети інтер'єру та екстер'єру. Квартира була повністю змодельована і обставлена меблями в програмі Archicad 21. Деякі дрібні предмети інтер'єру скачували з Інтернету (сайти 3ddd.ru, turbosquid.com, free3d.com), щоб надати квартирі більш живий вигляд.

Будівля притулку було побудовано в програмі Archicad 21, після допрацьовувалося в програмі 3Ds Max, предмети інтер'єру так само бралися з бібліотеки Archicad 21 і скачували з Інтернета. Обидві моделі імпортувалися в 3Ds Max для підготовки їх перенесення в Unity. Необхідно було створити предметів колізії, за допомогою примітивів типу Box, налаштувати коректне відображення текстур, перевірити індекси полігонів, доопрацювати деякі моделі

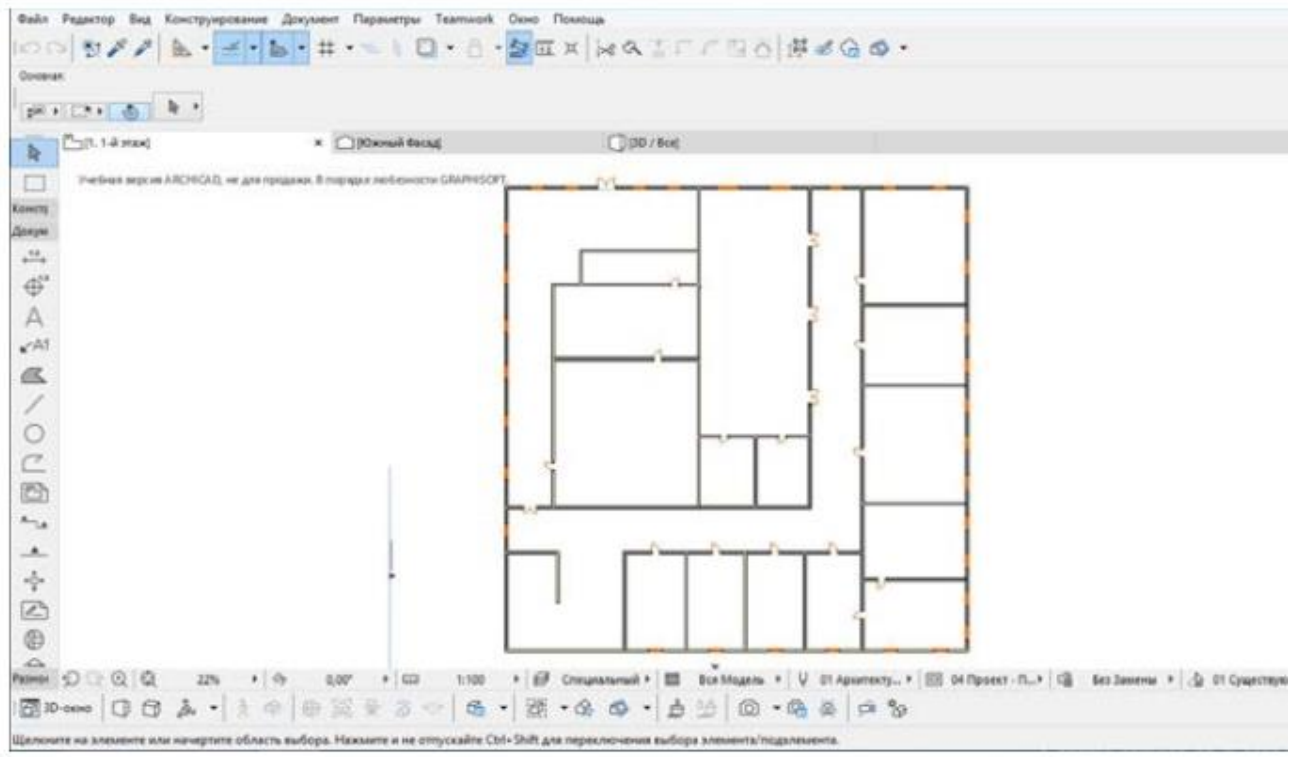

Рис.3.1 – Схема приміщення

3.3 Проектування Геймплею

Движок Unity дає можливо програмувати двома способами: на мові програмування C# або за допомогою візуального скриптової мови Package Manager.

Для розробки проекту був обраний візуальну мову Package Manager, так як по ньому більше корисної інформації на сайті движка Unity, він більш зрозумілий і швидко освоюється, мова C# має більше нюансів і раніше стикатися з ним не доводилося.

За допомогою Package Manager, як уже згадувалося раніше, програмувались головоломки, налаштовувалися матеріали, створювалася логіка взаємодії з деякими об'єктами і налаштовувалася машина станів для анімацій. Але найважливіший аспект - це програмування головного ігрового персонажа і штучного інтелекту ворога. Ворог у грі виконує мало функцій, тому його програмування не зайняло багато часу.

Функціонал ворога:

• переміщення по локації в межах заданого радіусу;

• якщо ворог бачить персонажа, то він біжить за ним;

• якщо персонаж потрапив в зону атаки, ворог атакує.

Реалізований функціонал ігрового персонажа (рисунок 40):

• переміщення;

• біг;

• включення / вимикання ліхтарика;

• стрільба з пістолета;

• прицілювання;

• взаємодія з предметами;

• система життя (зменшення і заповнення);

• вставання при бігу;

• система пересування в напівприсяді.

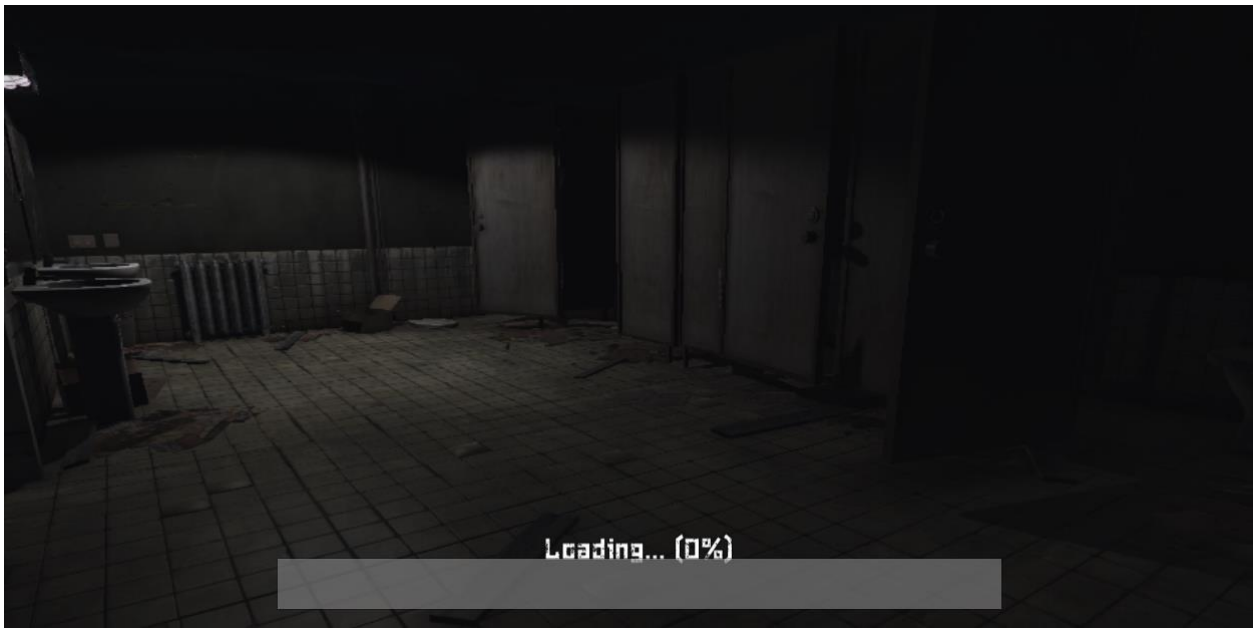

Рис. 3.2– Меню завантаження

За допомогою Package Manager програмувалися інтерфейси головного меню, титрів, підказок, меню налаштувань і меню паузи. У цих інтерфейсах кнопки мали обробник подій, який при натисканні на кнопку виконував будь-

яку дію, наприклад, якщо в інтерфейсі головного меню натиснути на кнопку Налаштування, то відкривався новий інтерфейс, що містить меню налаштувань.

Крім цього, програмувався пістолет, в його Package Manager налаштовувалася стрілянина, до цього звертається Package Manager головного героя при натисненні лівої кнопки миші для здійснення пострілу, при попаданні кулі в модель ворога у ворога віднімалося значення життя.

З важливих деталей можна відзначити так само Package Manager аптечок і магазинів з патронами. Аптечки зверталися до Package Manager головного героя і поповнювали значення життя, а патрони зверталися до Package Manager пістолета і поповнювали значення патронів.

Так само налаштовувалися події, що запускаються по тригеру. Персонаж заходив в зону тригера, після чого відбувалася певна дія. За допомогою тригерів в грі здійснюються збереження і відтворюються кат-сцени.

Рис. 3.3 – Меню

3.4 Архітектура системи і ієрархія класів

В якості основної парадигми програмування використовується об'єктно-орієнтований підхід [6]. ООП орієнтоване на розробку великих програмних комплексів. Об'єктно-орієнтоване проектування полягає в описі структури та поведінки проектованої системи, тобто, фактично, у відповіді на два основних питання:

- з яких частин складається система;
- в чому полягає відповідальність кожної з частин.

Виділення частин проводиться таким чином, щоб кожна мала мінімальний за обсягом і точно певний набір виконуваних функцій (обов'язків), і при цьому взаємодіяла з іншими частинами якомога менше.

Подальше уточнення призводить до виділення більш дрібних фрагментів опису. У міру деталізації опису та визначення відповідальності виявляються дані, які необхідно зберігати, наявність близьких по поведінці агентів, які стають кандидатами на реалізацію у вигляді класів з загальними предками. Після виділення компонентів і визначення інтерфейсів між ними реалізація кожного компонента може проводитися практично незалежно від інших (зрозуміло, при дотриманні відповідної технологічної дисципліни).

Велике значення має правильне побудова ієрархії класів. Одна з відомих проблем великих систем, побудованих за ООП-технології - так звана проблема крихкості базового класу. Вона полягає в тому, що на пізніх етапах розробки, коли ієрархія класів побудована і на її основі розроблено велику кількість коду, виявляється важко або навіть неможливо внести будь-які зміни в код базових класів ієрархії (від яких породжені всі або багато працюючих в системі класи). Навіть якщо вносяться зміни не торкнуться інтерфейс базового класу, зміна його поведінки може непередбачуваним чином позначитися на класах-нащадках. У разі великої системи розробник базового класу просто не в змозі передбачити наслідки змін, він навіть не знає про те,

як саме базовий клас використовується і від яких особливостей його поведінки залежить коректність роботи класів-нащадків.

Сукупність усіх підсистем, реалізованих за допомогою ООП і представлених у вигляді ієрархії класів, визначають внутрішню структуру ПП.

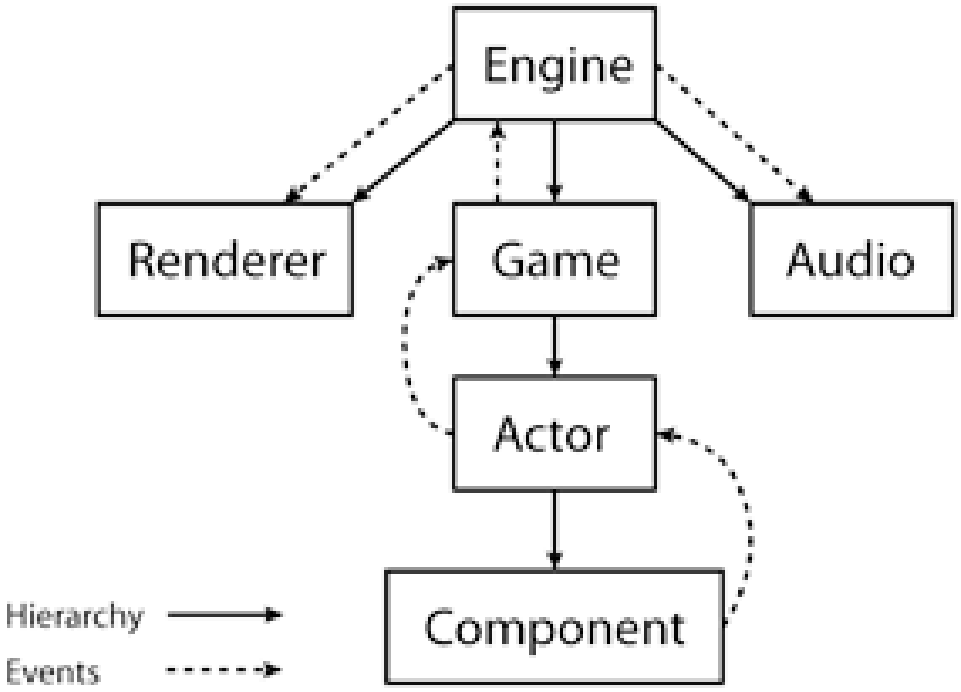

Рис. 3.4 – Uml діаграма ієрархій

4. РОЗРОБКА ТА РЕАЛІЗАЦІЯ

4.1 Реалізація механік гри

Вся логіка гри зібрана в методі update що викликається декілька разів за секунду. Обробка реакції гравця також реалізується в цьому методі. Використовуючи методи об‘єкту фізики ми можемо легко первірити чи зіштовхнулись, наклались 2 обьєкта, задати їм швидкість тощо. Також ми мажмо обробити реакцію на натиснення клавіш пермішення , пострілу та реакцію камери на переміщення гравця. Для обробки складних дій , аби зберігати читабельність коду, доцільно виносити їх в окремі функції та викликати в методі update або навіть створювати цілі класси для обробки складної логіки, виносити їх в окремі фали і підключити в html документі на початку.

Для підключення фізики до об‘єктів гри в методі create варто задати екземпляр об‘єкту фізики до об‘єкту гри та до властивості body ігрового об‘єкту. У Phaser існує 3 варіанта фізики Arcade,Ninja,P2 для різних типів ігор, в нашому випадку використовуємо варінт Arcade для ігор з видом згори.

Надалі налаштування поведінки кожного об‘єкту реалізужтся завдяки взаємодіє з властивістю body цих об‘єктів, тобто ми задаємо колізію, прискорення, швидкість , гравітацію тощо.

Для того щоб працювати з графічними елементами, перш за все варто їх завантажити в методі preload та назначити їм зручний аліас за допомогою якого ми будемо надалі до них звертатись.

Надалі в методі create мо можемо створити ігровий об‘єкт використовуючи аліас спрайту, задати йому початкове положення, розтянути, перевернути зображення, тощо. Також є можливість задати anchor - це точка відносно якої буде позиціонуватися об‘єкт на сцені, в термінах Phaser вона називається world. Початкові значення цієї точки (0,0) – це верхній лівий кут, значення anchor задається у відносних одиницях для прикладу точка (0.5,0.5) це

центр спрайту. Також є можливіст завантажити зображення зі спрайтовою анімацією та вказати розмір фреймів і фреймвор автоматично розділить його на мисив зображень.

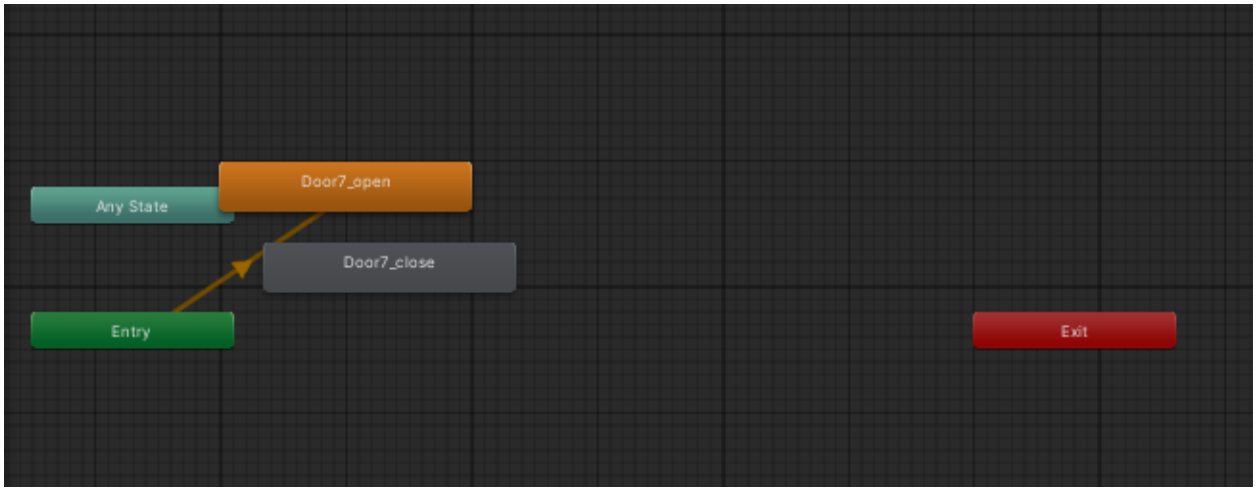

Рис.4.1 - Анімація

Геймплей будується виходячи з заданих правил і обмежень гри, а також індивідуального стилю поведінки гравця.

Механіка, геймплей, динаміка - як влаштований ігровий процес в іграх Існує безліч жанрообразующих комбінацій механік, які транслюються з гри в гру одного жанру. Наприклад, зв'язки: пересування + стрибок; подвійний стрибок; переміщення + прицілювання + постріл і так далі.

Геймдизайнер повинен розуміти, що у гравців будуть різні стилі поведінки (наприклад, психотипи Бартла) і потрібно підготувати геймплей таким чином, щоб він відповідав очікуванням обраної цільової аудиторії або задовольняв все розмаїття типів гравців.

Наприклад, в Deus Ex, геймплей спочатку задуманий під різні стилі проходження - активний (пряме зіткнення) і прихований (стелс). Гравці самі вибирають, який стиль їм підходить, і в будь-який момент можуть переключитися на інший вид геймплея - той, який для них краще в певній ситуації або за власним бажанням заради отримання нового досвіду.

Особливий досвід гравця є не тільки набір примітивних ментальних зв'язків, які формують системи правил і доступних механік, а й емоційно-чуттєву сторону сприйняття - осмислений досвід, який трансформується через внутрішню особистісну систему координат людини.

Геймплей по ходу гри може змінитися не тільки через зміну стратегії поведінки гравця, але і під впливом різноманітних вбудованих тригерів, які створюють для гравця нові стану гри і відповідно змінюють його поведінку. Наприклад, в грі Pac-Man тимчасовим зміною стану є збір бонусу (квасолини) в лабіринті, від чого миттєво змінюється механіка поведінки і параметри супротивників, як і параметри персонажа гравця.

Геймплей являє собою шаблони взаємодій ігрових механік і гравця в різних ігрових ситуаціях, а також перемикання станів цих взаємодій під дією тригерів.

Але для повноти картини напрошується ще один компонент, який впливає на формування досвіду гравця на довгій дистанції шляхом зміни обмежень в грі і параметрів ігрових механік. Цей компонент - ігрова динаміка.

Ігрова динаміка - це по суті оркестровка геймплея.

Уявіть, що ваші ігрові механіки - це музиканти з різними інструментами. Геймплей - весь оркестр цілком на чолі з гравцем-диригентом, а ігрова динаміка - ноти, які формують ту саму мелодію, унікальний почерк інтерактивності гри на всьому її протязі. У цю метафору також добре лягають типові жанрові геймплейні зв'язки - як окремі партії або фрагменти мелодії гри.

Ритм

Оркестровка - координація і управління складними системами і службами, розподілена в часі.

Термін співзвучний аналогічного в IT-сфері

- тривалість циклів ігрових механік і їх комплексність в рамках одного циклу.

Ритм може здаватися обмеженнями простору - наприклад складністю ландшафту ігрового світу, а також складністю поведінки ігрових об'єктів, з якими відбувається взаємодія.

Темп

Визначає швидкість розвитку геймплея в часі, швидкість відтворення всіх ігрових циклів.

Темп може контролюватися стисненням і розширенням простору, а також кількістю і швидкістю поведінки об'єктів взаємодії.

Тональність

- настройка параметрів ігрових об'єктів і функцій взаємодії між ними в певні періоди часу гри.

Мова йде про зміну звичних парадигм взаємодії гравця з ігровими об'єктами в потрібний автору момент. Така зміна може бути реалізовано на догоду історії або в залежності від умов і правил ігрового світу в той чи інший момент часу або місця в ігровому просторі.

Приклади. Поранення знижує маневреність, видимість і швидкість пересування. В певній галузі простору знижена видимість через туман. Видача гравцеві здібностей і інструментів на певній ділянці гри змінює формат поведінки гравця (гравіпушкі в Half Life 2).

Ігрова динаміка може бути простою або складною.

Для ігор, де немає сенсу створювати складну оркестровку геймплея в прогресії, використовують просту структуру. Наприклад, в Space Invaders динаміка гри змінюється лише за рахунок темпу, постійно прискорюючи всі процеси, - це вимагає від гравця швидшої реакції на кожному наступному рівні.

Для ігор, де важлива роль відведена сюжету, потрібно більш складна «мелодія» взаємодії як результат оркестровки геймплея - взаємодії повинні бути співзвучні історії, яку розповідає автор гри.

Комплексність - структурна і функціональна міра складності ігрової механіки

Це дозволяє найбільш повно передати досвід, закладений в черговому сюжетному повороті. Також це робить гру більш консистентної, фокусує гравця не просто на механічному вирішенні завдань і манчкінстве, а на осмисленні скоєних ним дій і подій, що відбуваються.

Іноді ламати звичний гравцеві шаблон поведінки корисно для історії - це і є зміна тональності в ігровій динаміці.

4.2 Збір сцени

Фінальний етап в розробці - це запаковування продукту. В Unity продукт можна запакувати для декількох платформ: Android, HTML5, Windows x32 і x64, Linux, iOS і tvOS. Так як гра призначена для комп'ютерів, а за статистикою більшість користувачів інтернет-магазину Steam (потенційні покупці) мають пристрої з операційною системою Windows x64, то для запаковування гри обраний саме цей формат. Перед тим як почати запаковування, необхідно в настройках вказати стартовий рівень, за замовчуванням в налаштуваннях стоїть порожній рівень, видалити з контенту зайві папки, якщо такі є, щоб не збільшувати розмір гри. Запаковування буде проходити у фоновому режимі, по закінченню пролунає звуковий сигнал.

Після цього гра буде представляти із себе додаток формату

* .Exe і кілька допоміжних папок (рисунок 43).

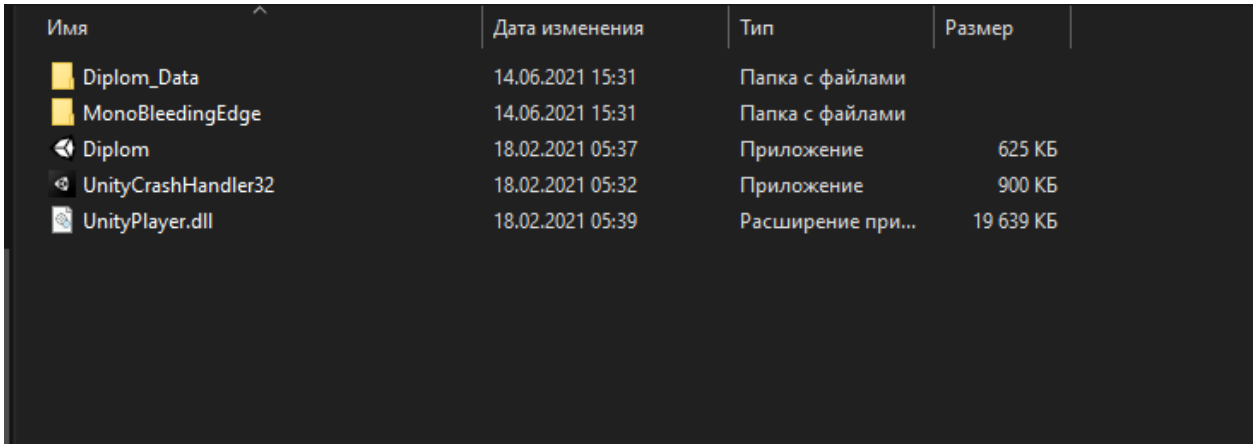

Рис.4.2 – Запакована гра

Після запаковування необхідно провести ще кілька тестів, так як деякі проблеми не виявити в движку. Після першого тесту був виявлений дефект з матеріалами трави і дерев (рисунок 44). Матеріали були загублені. Проблема вирішилася оновленням матеріалів у даних моделей.

Так само після запаковування гру можна переносити на інші комп'ютери з операційною системою Windows x64, щоб перевірити її працездатність. На сторонніх комп'ютерах гра запускається коректно, а значить можна її випускати.

4.3 Тестування

Даний етап протікав протягом всього процесу розробки, так як кожна нова функція тестувалася після її додавання і при виникненні помилок Blueprint виправлялися. Дані тести допомогли визначитися з методом патрулювання у ворога. Спочатку вороги переміщалися від однієї точки до іншої, але даний метод працював некоректно і іноді вороги не доходили до другої точки через складні маршрутів або застрявали на якийсь із точок.

Зміна методу на рандомне переміщення в межах радіусу показала себе як більш життєздатна, тому після повторних тестів було вирішено залишити её.

Так само дані тести допомогли визначитися з методом стрільби, показали, що важливі об'єкти варто підсвітити, так як ключі складно розгледіти в темряві.

Частина програмної системи, що забезпечує роботу інтерфейсу з користувачем - один з найбільш нетривіальних об'єктів для верифікації. Нетривіальність полягає в двоякому сприйнятті терміна "призначений для користувача інтерфейс".

З одного боку, призначений для користувача інтерфейс - частина програмної системи. Відповідно, на призначений для користувача інтерфейс пишуться функціональні і низькорівневі вимоги, за якими потім складаються

тест-вимоги і тест-плани. При цьому, як правило, вимоги визначають реакцію системи на кожен введення користувача (за допомогою клавіатури, миші або іншого пристрою введення) і вид інформаційних повідомлень системи, що виводяться на екран, принтер або інший пристрій виводу. При верифікації таких вимог мова йде про перевірку функціональної повноти призначеного для користувача інтерфейсу - наскільки реалізовані функції відповідають вимогам, чи коректно виводиться інформація на екран.

З іншого боку, призначений для користувача інтерфейс - "особа" системи, і від його продуманості залежить ефективність роботи користувача з системою. Фактори, що впливають на ефективність роботи, слабо піддаються формалізації у вигляді конкретних вимог до окремих елементів, проте повинні бути враховані у вигляді загальних рекомендацій і принципів побудови призначеного для користувача інтерфейсу програмної системи. Перевірка інтерфейсу на ефективність людино-машинного взаємодії отримала назву перевірки зручності використання (usability verification; в російськомовній літературі як переклад терміна usability часто використовують слово "практичність").

Функціональне тестування користувальницького інтерфейсу складається з п'яти фаз:

- аналіз вимог до призначеного для користувача інтерфейсу;

- розробка тест-вимог і тест-планів для перевірки користувальницького інтерфейсу;

- виконання тестових прикладів і збір інформації про виконання тестів;

- визначення повноти покриття призначеного для користувача інтерфейсу вимогами;

- складання звітів про проблеми в разі розбіжності поведінки системи і вимог або в разі відсутності вимог на окремі інтерфейсні елементи.

Всі ці фази точно такі ж, як і в разі тестування будь-якого іншого компонента програмної системи. Відмінності полягають в трактуванні деяких

термінів в застосуванні до призначеного для користувача інтерфейсу і в особливостях автоматизованого збору інформації на кожній фазі.

Так, тест-плани для перевірки користувальницького інтерфейсу, як правило, представляють собою сценарії, які описують дії користувача при роботі з системою. Сценарії можуть бути записані або на природній мові, або на формальній мові будь-якої системи автоматизації призначеного для користувача інтерфейсу. Виконання тестів при цьому виробляється або оператором в ручному режимі, або системою, яка емулює поведінку оператора.

При зборі інформації про виконання тестових прикладів зазвичай застосовуються технології аналізу виводяться на екран форм і їх елементів (в разі графічного інтерфейсу) або виводиться на екран тексту (в разі текстового), а не перевірка значень тих чи інших змінних, що встановлюються програмної системою.

Під повнотою покриття призначеного для користувача інтерфейсу розуміється то, що в результаті виконання всіх тестових прикладів кожен інтерфейсний елемент був використаний хоча б один раз у всіх доступних режимах.

Звіти про проблеми в інтерфейсі можуть включати в себе як опису невідповідностей вимог і реальної поведінки системи, так і опису проблем у вимогах до призначеного для користувача інтерфейсу. Основне джерело проблем в цих вимогах - їх тестонепрігодность, викликана розпливчастістю формулювань і неконкретність.

Особливістю текстових квестових ігор на платформі QSP є те, що велика частина інтерфейсу надається самим програвачем / інтерпретатором Quest Soft Player, і завдання розробника полягає лише в красивому оформленні та в перевірці того, що всі дії виконують саме те, що повинні. Варто зауважити, що у великій грі потенціал помилок в цій області величезний, і пропускати це тестування все одно не можна.

4.4 Системні вимоги

Рекомендовані системні вимоги для запуску проекту

Назва	Значення
ОС	Windows 7, 8.1, 10
Процесор	Intel Core i5-2600, AMD FX-5600
Відеокарта	Nvidia GT920
Оперативна пам’ять	6 GB
Місце на диску	1.5 GB

Таблиця 1. – Системні вимоги

ВИСНОВКИ

Темпи розвитку ринку ігрової індустрії наочно показують актуальність тематики даної роботи. Відеоігри надійно закріпилися в життя сучасної людини будь то великі проекти для комп'ютерів або невеликі головоломки для смартфонів. Завдяки доступності ігрових движків і достатку інформації в Інтернеті будь-хто може почати розробляти відеоігри, що тільки посприяє розвитку даного сегмента ринку цифрового контента.

В рамках дипломної роботи була розроблена 3D гра в жанрі survival horror з видом від першої особи на движку Unity. Крім движка в роботі були використані програми Archicad 21 для моделювання будівель, використовуваних в локаціях гри, 3Ds Max для доопрацювання і конвертації 3D моделей, Adobe Photoshop CC для створення карт нормалей текстур.

В результаті роботи виконані наступні завдання:

• досліджені літературні та інтернет-джерела з питань розробки відеоігор, написання сценарію;

• створено концепт гри, обраний жанр, вид;

• написаний сценарій гри;

• проаналізовані популярні ігри в жанрі survival horror, випущені за останні 10 років;

• вивчена документація, що описує методи роботи з движком Unity;

• повністю реалізований запланований функціонал проекту в движку Unity.

Таким чином мета роботи досягнута, а завдання повністю виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Juul Jesper Half Real: Video Games between Real Rules and Fictional – це Worlds MIT Press. 2006. – це

2. Курячий Г. В. Операционная система UNIX. Курс лекций. Учебное пособие; Интернет-университет информационных технологий – це Москва, 2004. – це 288 с.

3. Магда Ю. С. UNIX для студента; БХВ-Петербург – це Москва, 2007. – е 480с.

4. Мартемьянов Ю. Ф. Операционные системы. Концепции построения и обеспечения безопасности; Горячая Линия / Яковлев А. В., Яковлев А. В. – це Телеком – це , 2011. 338 с – це

5. Мертенс Петер Интегрированная обработка информации. Операционные системы в промышленности; Финансы и статистика , – це 2007. 424 с. – це

6. Назаров С. В Операционные системы. Практикум для бакалавров / Гудыно Л. П., Кириченко А. А. КноРус – це Москва, 2012. – це 376 с.

7. Таненбаум, Эндрю Современные операционные системы. – це С. 1040. Це Издательский дом «Питер», 2007. ISBN 978 5 318 00299 1.

8. Чернов Р. П. М.: [б. и.], 2007 (Центр полиграфических услуг – це "Радуга")

9. Шеховцов В.А. Операційні системи К.; Видавнича група BHV.2005. ISBN 966 552 157 8