МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К. И. Сатпаева

Институт автоматики и информационных технологий

Кафедра "Программная инженерия"

Салмаси Джаббар Арифович

Исследование современных технологий создания компьютерных игр и разработка видеоигры в жанре Adventure на платформе Unreal Engine

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К. И. Сатпаева

Институт автоматики и информационных технологий

Кафедра "Программная инженерия"

ДОПУЩЕН К ЗАЩИТЕ
Заведующая кафедрой ПИ
канд. физ-мат. наук, профессор
_____ А.Н. Молдагулова
«_23_» _____05_____ 2022 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: “Исследование современных технологий создания компьютерных игр и разработка видеоигры в жанре Adventure на платформе Unreal Engine”

по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнил Салмаси Д.А.

Рецензент
Доктор PhD
_____ К.А. Алибаева
“18” _____мая_____ 2022 г

Научный руководитель
Магистр технических наук
_____ Д.Е. Айжулов
“20” _____мая_____ 2022 г

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К. И. Сатпаева

Институт автоматики и информационных технологий

Кафедра "Программная инженерия"

УТВЕРЖДАЮ
Заведующая кафедрой ПИ
канд. физ-мат. наук, профессор
_____ А.Н. Молдагулова
«23» 05 2022 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Салмаси Джаббару Арифовичу*
Тема: *Исследование современных технологий создания компьютерных игр и разработка видеоигры в жанре Adventure на платформе Unreal Engine*
Утверждена *приказом проректора по академической работе* № 489-п/о от "24" 12 *2021 г.* Срок сдачи законченного проекта "23" 05 *2022 г.*
Исходные данные к дипломному проекту:
Перечень подлежащих разработке в дипломном проекте вопросов:
а) исследование рынка игровой индустрии
б) изучение современных технологий разработки видеоигр
в) проектирование компьютерной игры
г) реализация и тестирование программы
Перечень графического материала (с точным указанием обязательных чертежей): *представлены* 35 *слайдов презентации.*
Рекомендуемая основная литература: *из 42 наименований.*

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области	16.01.2022	**Выполнено**
2. Разработка технического задания на проектирование видеоигры	20.01.2022	**Выполнено**
3. Исследование и выбор современных технологий и инструментов для разработки видеоигры	12.02.2022	**Выполнено**
4. Разработка игры “Похищение”	25.02.2022	**Выполнено**
5. Реализация компьютерной игры, внутриигрового интерфейса, разработка видеоигрового управления	30.03.2022	**Выполнено**
6. Тестирование конечной программы	1.04.2022	**Выполнено**
7. Написание пояснительной записки к дипломному проекту	29.04.2022	**Выполнено**

Подписи

консультантов и нормоконтролера на законченный дипломный проект с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	Магистр технических наук, лектор Марғұлан Қ.	20.05.22г.	
Нормоконтролер	Доктор Ph.D., ассоциированный профессор Жекамбаева М.Н.	23.05.22	

Научный руководитель _____ Д.Е. Айжулов

Задание принял к исполнению обучающийся Салмаси Д.А. Салмаси

Дата "17" 11 2022 г.

АНОТАЦИЯ

Настоящий проект предназначен для разработки видеоигры в жанре Adventure на платформе Unreal Engine. В рамках дипломного проекта была разработана приключенческая игра “Похищение” в жанре визуального романа, на игровом движке Unreal Engine для персонального компьютера. Проект является игровой адаптацией графического романа Manuro (E. Quaireau), M.C. (M. Chevalier) “Captive”.

При работе с Unreal Engine использовалась среда разработки Visual Studio с применением высокоуровневого языка программирования общего назначения C++. А также встроенная UE система визуального скриптинга Blueprints.

Зачастую визуальные новеллы представляют собой книжные истории в игровом исполнении, с выбором или без выбора вовсе. Данный проект предлагает большую интерактивность. Игровые механики внутриигрового процесса:

- свободное передвижение;
- общение с персонажами;
- отыскивание предметов;
- разгадывание шифров;
- изучение улик;
- участие в стычках и погонях;
- принятие решений в нелинейном сюжете со множеством концовок.

АНДАПТА

Бұл жоба Unreal Engine платформасында шытырман оқиғалы жанрдағы бейне ойынды дамытуға арналған. Бітіру жобасы аясында «Похищение» шытырман оқиғалы ойыны жеке компьютерге арналған Unreal Engine ойын қозғалтқышында көрнекі роман жанрында әзірленді. Жоба Manuro (E. Quaireau), M.C. (M. Chevalier) “Captive” графикалық романының ойынға бейімделуі.

Unreal Engine-мен жұмыс істегенде, Visual Studio әзірлеу ортасы C++ жоғары деңгейлі жалпы мақсаттағы бағдарламалау тілі арқылы пайдаланылды. Сондай-ақ кірістірілген UE визуалды сценарий жүйесі Blueprints.

Көбінесе визуалды романдар ойын түрінде, таңдаусыз немесе таңдаусыз кітап әңгімелері болып табылады. Бұл жоба керемет интерактивті ұсынады. Ойын ішіндегі процестің ойын механикасы:

- еркін қозғалыс;
- кейіпкерлермен қарым-қатынас;
- элементтерді табу;
- шифрларды шешу;
- дәлелдемелерді зерттеу;
- қақтығыстар мен қуғындарға қатысу;
- бірнеше соңы сызықты емес әңгімеде шешім қабылдау.

ABSTRACT

This project is intended to develop a video game in the Adventure genre on the Unreal Engine platform. As part of the graduation project, the adventure game “Похищение” was developed in the genre of a visual novel, on the Unreal Engine game engine for a personal computer. The project is a game adaptation of the graphic novel by Manuro (E. Quaireau), M.C. (M. Chevalier) "Captive".

When working with Unreal Engine, the Visual Studio development environment was used using the high-level general-purpose programming language C++. As well as the built-in UE visual scripting system Blueprints.

Often, visual novels are book stories in a playful way, with or without choice at all. This project offers great interactivity. Game mechanics of the in-game process:

- free movement;
- communication with characters;
- finding items;
- deciphering ciphers;
- study of evidence;
- participation in skirmishes and chases;
- making decisions in a non-linear story with multiple endings.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня видеоигры - эта крупная массовая индустрия, которая за весьма короткий срок приобрела гигантское влияние наравне с музыкой и кино, став мощнейшим феноменом поп-культуры и крупнейшей формой искусства. За последние годы индустрия игр приносит больше выручки по сравнению с книгами и превышает доходы, чем кинематограф и музыка вместе взятые [16]. Таким образом создание видеоигр является актуальной темой в данный момент.

На данный момент индустрия разработки видеоигр не так развита в Казахстане [18], как в некоторых других странах СНГ. Например, компания “ 4A Games”, находящаяся в Украине или “Wargaming” в Белоруссии. Поэтому проект предназначен для привлечения внимания к важности развития игровой индустрии в нашей стране.

Игровая индустрия – это крупный рынок, в котором существует большая конкуренция, создающая преграды к разработке игр, что займут свою нишу, и в связи с тем, что имеется высокий спрос к жанру Adventure, целью проекта было выбрано исследование современных технологий создания компьютерных игр и разработка видеоигры в жанре Adventure на платформе Unreal Engine.

Главная ценность реализованной игры в её кардинальном отличии от графического романа, на котором она основана. В книге читатель во время выбора сюжетного действия случайно может преждевременно узнать важную информацию о сюжете, увидеть несколько концовок (на одной странице показаны 2–5 возможных развитий сюжета). То игра в свою очередь, полностью исключит случайные спойлеры, что предаст ей ценность в глазах игрока.

Исходя из этого методы исследования в данном дипломном проекте осуществлены при помощи анализа статистических данных рынка популярных жанров, платформ и актуальных технологий создания игр.

Пояснительная записка к дипломному проекту состоит из введения, основной части (исследовательский, технологический, проектный и экспериментальный разделы), заключения, списка использованной литературы и приложений.

В первом разделе проекта, “Исследовательский раздел”, проведены анализ и исследование игровой индустрии, актуальных данных игрового рынка.

Во втором разделе проекта, “Технологический раздел”, выполнено исследование современных и актуальных технологий разработки видеоигр.

Третий раздел проекта, “Проектная часть”, описывает разработку компьютерной игры “Похищение”.

Четвертый раздел, “Экспериментальный раздел”, посвящён реализации конечной программы.

В заключении описаны основные результаты, полученные при выполнении дипломного проекта.

Дипломный проект состоит из 59 страниц, 31 рисунок, 5 диаграмм, 2 приложений. В работе использовались 42 источника.

1. Исследовательский раздел

1.1 Цель проекта

Игровая индустрия — это большой и быстрорастущий рынок, в который ежегодно вливаются крупные инвестиции и в котором существует жесткая конкуренция, создающая преграды к разработке видеоигр, что займут свою нишу на рынке. Соответственно, возникает необходимость в исследовании современных средств создания игр, а в связи с тем, что наблюдается высокий спрос к жанру Adventure, целью проекта было выбрано исследование современных технологий создания компьютерных игр и разработка видеоигры в жанре Adventure на платформе Unreal Engine.

Следовательно, задачи моего проекта, следующие:

1. Провести анализ предметной области и исследовать рынок игровой индустрии, разработать техническое задание на проектирование видеоигры
2. Исследовать современные технологии и средства создания видеоигр
3. Разработать игру “Похищение”
4. Реализовать компьютерную игру, разработать внутриигровой интерфейс, видеоигровое управление
5. Провести тестирование конечной программы

1.2 Определения, термины и сокращения

В таблице 1.1 сформулированы все сокращения и термины, используемые в предметной области разрабатываемого проекта, и равным образом специфические термины, которые связаны с используемыми технологиями при разработке и программной реализацией проекта.

Таблица - 1.1 – Сокращения, термины и их определения

Сокращение/термин	Определение
Unreal Engine (UE)	Игровой движок, межплатформенная среда разработки компании Epic Games
Игровой движок	Программное обеспечение для разработки видеоигр
Visual Studio (VS)	Интегрированная среда разработки (IDE) программного обеспечения компании Microsoft
C++	Язык программирования общего назначения
Blueprints	Система визуального скриптинга, встроенная в UE
Скрипт (сценарий)	Программа/последовательность инструкций, интерпретируемая/выполняемая системой

Продолжение таблицы 1.1 - Сокращения, термины и их определения

Сокращение/термин	Определение
Спойлер	Важная преждевременно раскрытая сюжетная информация
Класс	В C++ описываются методы (функции) и данные, используемые объектом (переменная) класса
Adventure (квест, приключенческая игра)	Жанр видеоигр, представляющий интерактивную историю о персонаже, которым управляет игрок, с элементами исследовании мира и решения головоломок
GUI	Графический пользовательский интерфейс
DLC	Загружаемый контент
Патч	Программное средство для устранения технических проблем и багов
Баг	Программная ошибка
Рендеринг	Процесс получения изображения по модели при помощи программы
Монетизация	Стратегия увеличения прибыли
Уровни	Площадки мира игры представляющие, какие бы то ни было, локации: города, здания, леса и т. п.
Сеттинг	Обуславливает место, время и обстоятельства событий игры
Захват движений	Оцифровка реальных движений специальными датчиками
Комьюнити	Сообщество людей, объединенных общим интересом
Игровые механики (механики)	Правила определяющие и регулирующие действия и возможности игрока, и реакцию игры на них.
AAA (Triple-A)	Высокобюджетная игра
Инди	Независимая малобюджетная игра

1.3 Предметная область

1.3.1 Видеоигры: тогда и сейчас

Согласно Кембриджскому словарю, **видеоигра** – это игра, в которой игрок управляет движущимися изображениями на экране, нажимая кнопки [1]. Предшественниками видеоигр были компьютерные игры, а до них были электронные, на электронных устройства, не имеющие дисплея и компьютера с программой. Первоначально компьютеры были лишь у армии и университетов, и стоили огромных денег. Но даже тогда, новаторы разрабатывали игры.

1940 год – Эдвард Кондон, физик-ядерщик, представил на Всемирной выставке в Нью-Йорке Nimatron – игровой компьютер, представляющий электронно-релейную машину, являющийся адаптацией китайской игры ним. Суть игры заключалась в поочередном вытягивании предметов двумя игроками (человек и ИИ) из нескольких куч. Побеждал взявший последний предмет [2].
1947 год – Томас Голдсмит-младший и Эстл Рей Манн изобрели «Развлекательное устройство на основе ЭЛТ» - электронную игру в виде тира с выводом изображения на дисплей. Однако она не считается кандидатом на звание первой видеоигры, так как не работала на вычислительном устройстве. В связи со стоимостью и других факторов, не была выпущена на рынок [3].
1952 год – Сэнди Дуглас, профессор информатики в Кембридже, на основе ЭЛТ, разработал игру OXO – имитирующую крестики-нолики. Но, ее не могли распространить, так как компьютер EDSAC, для которого ее разработали, был уникален и располагался в библиотеке университета [4].

Хоть первые игры использовались в основном для демонстрации технических возможностей компьютеров тех лет, со временем программисты, пытались создать компьютерные игры в развлекательных целях.
1962 год – студенты MIT С. Рассел, У. Витенен и М. Грец, на передовом PDP-1, выпускают цифровую видеоигру **Spacewar!** – космосимулятор, где два игрока, управляющие космическими кораблями, маневрировали от гравитационного поля звезды, при этом отстреливаясь торпедами. Игрок мог использовать тягу и входить в гиперпространство, что позволяло переместиться в случайное место на экране [5]. Погибал тот, в кого попадала торпеда противника либо если корабль сталкивался со звездой/другим кораблем. Игра стала хитом, была на большинстве PDP-1. После вышла на аркадные автоматы и повлияла на создание множества игр-клонов. В последствии стала прародителем жанра Shoot ’em up (перестреляй их всех) [6]. Spacewar! сыграла важную роль в формировании игровой индустрии, вдохновив собой многих, что привело к переселению видеоигр из лабораторий в залы с автоматами, а затем в квартиры, стремясь превратить технологию в коммерциализированное развлечение.
1971 год – Х. Так и Б. Питтс, создают первый игровой автомат, с улучшенной версией Spacewar! – Galaxy Game. Если в первую играли лишь программисты [12], то Galaxy стала доступной любому. Автомат был коммерческим, а игра была первой, имеющей слот для монет [7]. 2 месяца спустя Nutting Associates, вместе с инженером Н. Бушнеллом, выпускает свою адаптацию Spacewar! – Computer Space, но она провалилась из-за сложного управления [8].
1972 год – выход первой игровой консоли **Magnavox Odyssey**, своему появлению обязанная Ральфу Баеру. Вместе с ней шли 12 игр [9] на картриджах – карты памяти, содержащие микросхемы ПЗУ, хранящие видеоигру. К тому же, для полного погружения, использовались накладки на экран телевизора, добавляющие элементы окружения. В последующие пять лет были проданы более 350,000 экземпляров [10].

Вдохновившись одной из игр Odyssey – пинг-понгом, **Нолан Бушнелл** основывает **Atari, Inc.**, в следующие десятилетия ставшая одной из

доминирующих игровых компаний, и предлагает концепцию игры А. Алькорну. Первый автомат Pong был установлен в местной забегаловке, но через две недели менеджер просил его убрать, говоря, что он сломан. Разобрав автомат, никаких поломок не обнаружили. Проблема была в монетоприемнике, он был переполнен. Тогда глава Atari, осознав, что в его руках золотая жила, решил сам издавать автоматы и отказался от помощи сторонних компаний [11].

Приобретя массовый успех, Pong привлекла внимание Баера, что привело к дальнейшим разбирательствам между Atari, Inc. и Magnavox. Но все это не помешало Нолану, и через год после массового производства и неимоверного успеха, другие крупные игровые компании начали создавать свои подделки на Pong. Так, японская компания SEGA решила выпустить аркадную игру Hockey TV, по сути являющаяся Pong с двумя наборами ракеток [12].

1977 год – с выходом приставок второго поколения, Atari выпускает 8-битную, Atari 2600, позже выпустившая знаменитую Shoot 'em up видеоигру Space Invaders, к 1982 году принесшая прибыль $3.8 млрд [13], а в общем больше $15 млрд (с учетом инфляции на 2021). Успех продаж произошел и с легендарными Donkey Kong и Pac-Man, ставшие самыми прибыльными играми в истории. Но, несмотря на успех Atari 2600 и продажи в 40 млн копий, компания производила низкие выплаты разработчикам и не упоминала их в играх. Уже через суд, было признано право любого разработчика, создавать проект для Atari 2600, что привело к заполнению рынка низкокачественными продуктами, выходом E.T. the Extra-Terrestrial в 1982, считающаяся одной из худших игр, крупнейшим провалом в индустрии, и к захоронению нераспроданных копий [14]. Все это повлияло к возникновению масштабного кризиса индустрии видеоигр 1983 года, когда разорилось большое количество американских игровых компаний.

Лишь спустя несколько лет, индустрия смогла восстановиться, благодаря распространению японской приставки Nintendo Entertainment System (NES).

1985 год – выход NES, ознаменовавший начало третьего поколения игровых систем. В том же году выходит Super Mario Bros., ставший в будущем одной из самых продаваемых игр в истории.

С 80-х видеоигры перестали быть хобби и просто игрушками. Приставки продавались миллионами копий по всему миру, по играм снимали фильмы, к примеру “Трон”, печатали журналы и т. д. Игры стали частью массовой культуры. А с последующими поколениями вышло множество проектов, повлиявших и изменивших индустрию. Но затрагивая даже небольшую часть, размер видеоигровых хроник в данном исследовании увеличится в разы.

1.3.2 Анализ рынка. Видеоигровая индустрия в цифрах

Для понимания, как устроен современный рынок игр, куда он движется – очень важно проанализировать, что популярно и перспективно в этой сфере. В

связи с этим, было проведено исследование рыночной статистики по продажам и актуальным платформам для видеоигр.

Прибыль игровой индустрии

- **2019 ≈ $144 млрд**;
- **2020 ≈ $174 млрд**;
- **2021 –** годовой доход индустрии составил рекордные ≈ **$180 млрд**.

Прибыль по платформам

- Мобильные игры ≈ **52%**;
- Персональные компьютеры ≈ **20%**;
- Консоли ≈ **28%**.

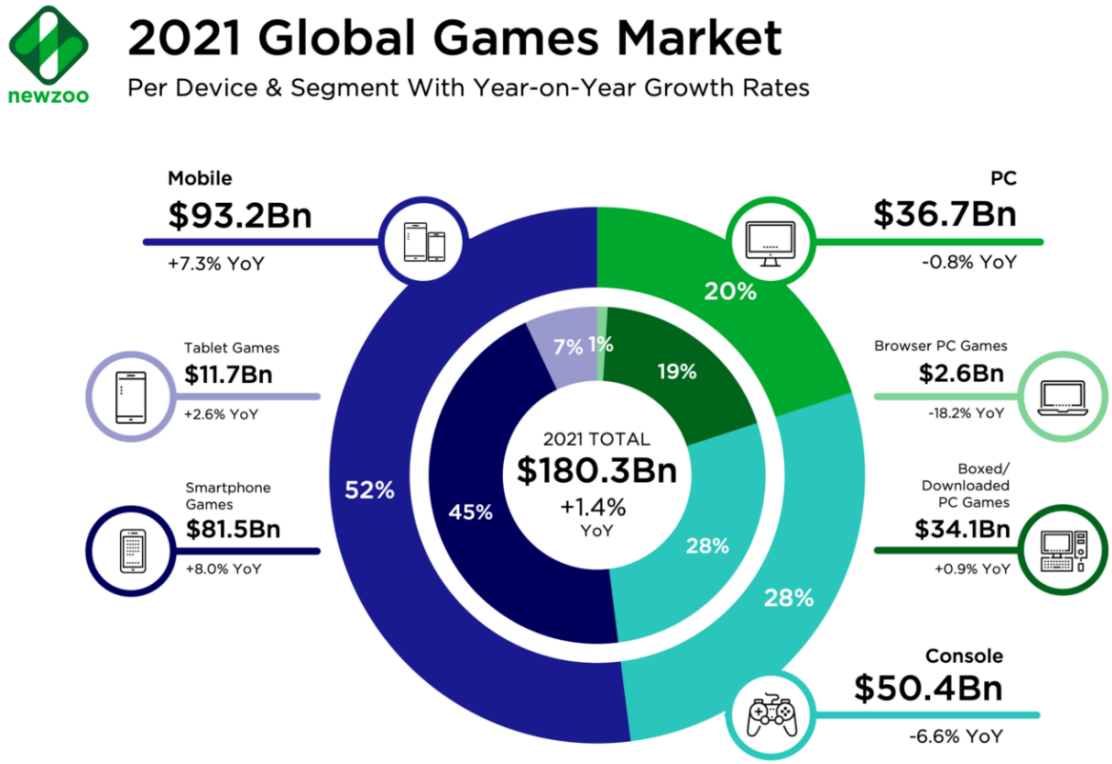

Рисунок - 1.1 – статистика прибыли игровой индустрии и прибыли по платформам 2021, предоставленная аналитической группой NewZoo [15]

Сравнивая доходы на 2020 год среди медиаиндустрии, годовая прибыль **киноиндустрии** составила лишь **$45 млрд**, что в 3.5 раза меньше игровой [16].

Статистика рынка по регионам на 2021 год [17]:

- Количество геймеров на 2022 год составляет ≈ **3 млрд** человек;
- Рынок **Азии ≈ 55%** доходов – в основном **Китай** и **Япония**. Из-за чего издатели ориентируются на данный рынок. К примеру, в играх компании Blizzard Entertainment, ежегодно празднуют китайский Новый год. В большинстве магазинах цифровой дистрибуции игр проводятся скидки;
- **Северная** и **Южная Америка, Европа ≈ 31%**;
- Россия на 11 месте среди стран, с доходами в ≈ $1.7 млрд;
- **Казахстан ≈ 47 место** [18].

Текущие низкие показатели привели к тому, что не все крупные разработчики локализируют свои игры для СНГ. К примеру, Psychonauts 2, изданная Xbox Game Studios и претендовавшая на лучшую игру года 2021 по версии престижной премии The Game Awards – не имеет русского языка.

Популярные жанры [19]

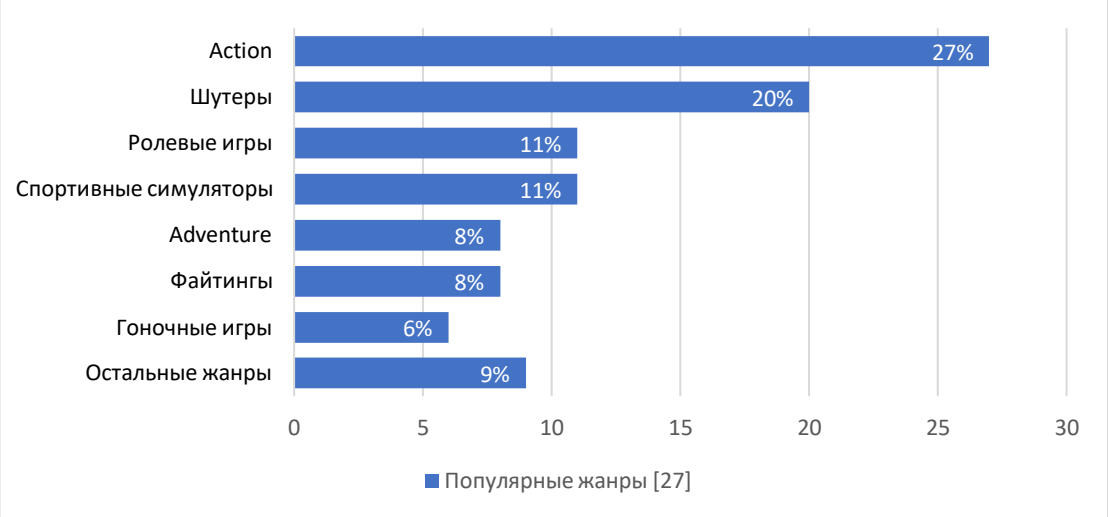

Диаграмма - 1.1 – статистика популярных жанров среди игр, предоставленная аналитической платформой Statista [19]

Самые прибыльные франшизы [20]

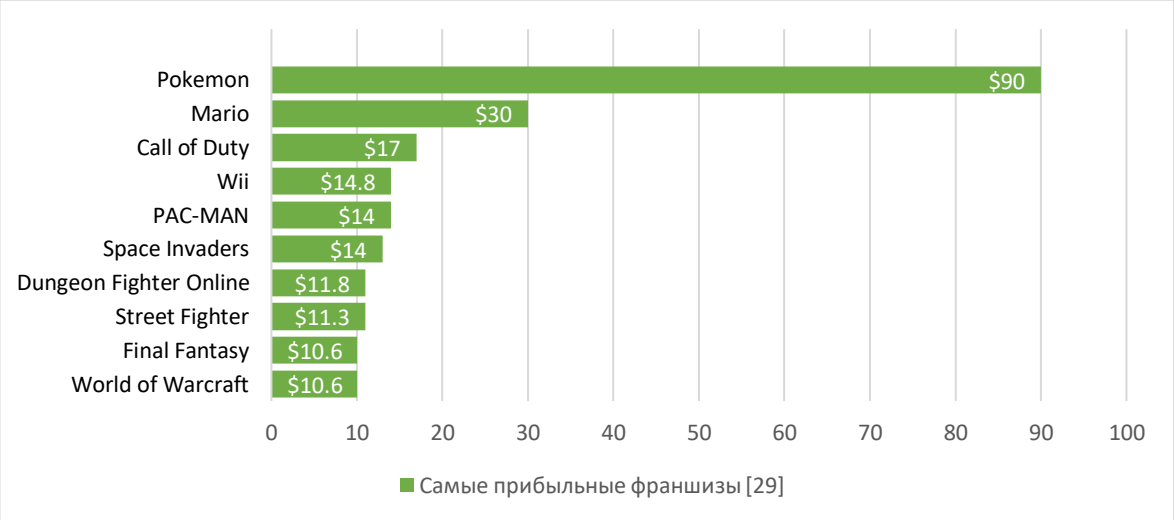

Диаграмма - 1.2 – самые прибыльные игровые франшизы, в $US млрд Статистика предоставлена аналитической платформой Statista [20]

В данном ТОПе все еще нет ни одного мобильного проекта, лишь консольные, компьютерные игры.

Самые продаваемые игры 2017–2022 годы [21]

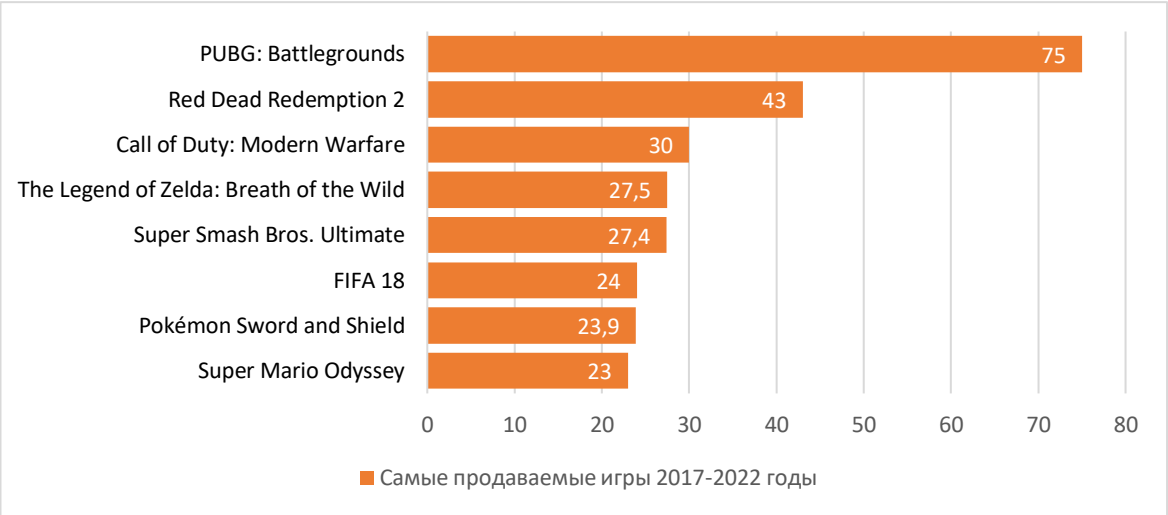

Диаграмма - 1.3 – самые продаваемые игры с 2017 по 2022 годы, в млн проданных копий. Статистика предоставлена интернет-энциклопедией Wikipedia [21]

Таким образом, рассматривая самые продаваемые игры последних пяти лет, игры однопользовательского жанра все еще сильно востребованы, приносят большие деньги, а доля консолей в последние годы возросла несмотря на то, что большую часть рынка составляют мобильные приложения.

Примерами успеха одиночных игр, служат проекты последних лет:

– Spider-Man – 3.3 млн проданных копий за 3 дня [22]
  87/100 баллов на игровой сайте-агрегаторе рецензий Metacritic [23];
– Red Dead Redemption 2 - $725 млн за 3 дня [24]
  97/100 баллов на игровой сайте-агрегаторе рецензий Metacritic [25];
– It Takes Two – в жанре action-adventure
  Лучшая игра года 2021 по версии The Game Awards [26].
  Исходя из вышеперечисленных данных, у игровых издателей нет особых причин отказываться от одиночных игр.

1.3.3 Квесты (Adventure games)

Затрагивая тему исследования, были рассмотрены игры жанра Adventure. Дефиниция “**квест, приключенческая игра”** – жанр видеоигр, один из основных, представляющий из себя интерактивную историю о персонаже, которым управляет игрок, основанной на крепком повествовании, исследовании мира и решениях различных задач/головоломок, визуальной эстетике и в которой бои не являются основной формой активности [27].

Как и первые игры в целом, так и первые квесты, из-за отсутствия компьютерной графики, были текстовыми, где всё повествование велось посредством текстовой информации.

1975 год –Уильям Краудер разработал **первый квест** в жанре Interactive fiction (интерактивная художественная литература) – **Colossal Cave Adventure**. Игра выводила на экран описания пещерных систем, а от игрока требовалось набрать команду куда идти, чтобы выбраться, попутно исследуя пещеру в поисках золотых сокровищ и решая проблемы используя найденные предметы. В пещере игрока поджидали смертельные опасности в виде дворфов и ям [27].

Первый квест, имеющий компьютерную графику, был **Mystery House** 1980 года, разработанный On-Line Systems, в будущем **Sierra On-Line**. Хоть игра представляла собой векторные статичные рисунки на чёрном фоне, она сделала прорыв в индустрии и разошлась тиражом в 10 тысяч копий [28].

1984 год – Sierra выпускает King's Quest, ставшая классикой жанра. В ней, в отличии от предыдущей игры компании, где локации состояли из обычных линий в пустоте, присутствовала красочная графика, появились деревья и реки, замки, по которым бродил герой. И хоть действия нужно было все еще набирать через команды, последствия уже можно было увидеть на экране. Именно в этой серии заложили основы жанра, где для того, чтобы двигаться по сюжету, нужно решать головоломки, внимательно исследовать локации, общаться с другими персонажами и взаимодействовать с разными предметами. Затем от той же Sierra On-Line вышла не менее значимая Space Quest.

С популярностью этих и других серий, в русском языке закрепилось слово "квест", как наименование для "адвенчур" [29]. А с появлением новых графических квестов произошел переход от обычного текстового к **point-and-click** интерфейсу, где для взаимодействия, нужно было навестись курсором мыши и щелкнуть кнопкой по определенной области на экране монитора.

В дальнейшем новым доминантом в среде квестов стал **LucasArts**, выпуская самые красивые и первоклассные игры на рынке [27]. Одной из которых были серия сатирических квестов **Sam&Max** о дуэте антропоморфного (обладающего человеческим образам) пса детектива Сэма и его зайцеобразного помощника Макса; брутальный **Full Throttle** с нетипичным сеттингом о байкерах; игра, которую ставят на вершину квестового искусства - **Grim Fandango**, благодаря ее уникальному визуальному стилю, отполированным механикам и трогательной истории о работнике департамента смерти в загробном мире, помогающий умершим в их последнем пути. Все перечисленные проекты получили высоченные оценки среди игроков и критиков, продались крупным тиражом и стали по-настоящему культовыми [30]. Таким образом квесты укрепили свои позиции в игровой индустрии.

Тем не менее приключенческие видеоигры меньше зависящие от технологий отображения, чем динамичные экшн-игры, в середине 2000-х стали получать меньше внимания прессы, а игроки перестали интересоваться ими, что привело к низким продажам и заблуждению о том, что жанр «мертв» [27].

В действительности квесты живы и хорошо себя чувствуют и остаются весьма популярными, так как этот жанр один из самых разветвленных, не прекращающий расти и развиваться. А со временем видоизменяясь, добирал в себя новые механики и эволюционировал, порождая новые жанры и поджанры. Например: Alone in the Dark – квест считающийся родоначальником survival horror (ужасы с элементами выживания); программист Майкл Той, под впечатлением от Colossal Cave Adventure, создал Rogue, породившую жанр Roguelike - основой которого, являются генерируемые случайным образом уровни [31]; вдобавок геймеры просто пристрастились к более зрелищным динамическим видеоиграм, вследствие чего adventure жанр стал менее популярен с приходом нового смешанного - Action-Adventure [27]. И те самые мегауспешные проекты последних лет, о которых писалось ранее в данном исследовании: It Takes Two, Red Dead Redemption 2, Marvel's Spider-Man – являются эталонными представителями жанра приключенческий боевик.

Одним из поджанров приключенческих видеоигр является **визуальный роман** (новелла) – в котором история демонстрируется за счет статичных изображений и текста. Хоть традиционно визуальная новелла преподносится в стиле комикса/манги (жанр зародился в Японии) сопровождаемая текстом, сегодня она, как и любые жанры может заимствовать игровые механики, стиль подачи из других: меньшее/большее количество интерактивности, 2D/3D, демонстрация статичных спрайтов или же возможность свободно передвигаться по миру игры и т. д., и т. п.

В 2010-х квесты переродились благодаря студии Telltale Games и их эпизодической серии о зомби-апокалипсисе The Walking Dead, созданной по мотивам комиксов “Ходячие мертвецы”. Формально это был всё тот же квест, однако игрался он иначе. Управление персонажем происходило не по щелчку мыши, а при помощи клавиш. Подбираемые предметы автоматически использовались в зависимости от контекста. И главное – это система решений и их последствий, где каждый выбор в диалогах или различных ситуациях влияют на ход истории (во всяком случае игра создает отличную иллюзию этого) не только в рамках одной игры, но даже в последующих частям одной серии, в зависимости от принятых ранее решений, сделанных игроком. Геймплей был сведен к минимуму, а существенную часть времени игрок принимает решения в интерактивных внутриигровых видео (кат-сценах). И данные решения, как и подход к приключенческим играм стали невероятно популярными. Более того, The Walking Dead: The Game получила награду игру года 2012 церемонии награждения Spike Video Game Awards [32].

В результате анализа квестов (adventure games) были выделены следующие характеризующие их качества:

- исследование;
- сбор/манипулирование предметами;
- решение головоломок;
- меньший упор на боевые и экшн элементы.

1.3.4 Разработка видеоигр

Этапы производства:

1) **концепт игры**
   a. геймдизайнер с командой разработчиков обозначают **жанр**, который станет вектором разработки всего проекта, т. к. от того какой жанр выбран, к примеру action, adventure или RPG - будут зависеть ключевые геймплейные **механики**, **игровой функционал** - вмещающий в себя логику мира игры, физику, условия win/loose, ИИ персонажей, интерактивность объектов игры, **характер** всего **игрового процесса** - от которого будет зависеть на какую аудиторию нацелена игра [33]
   b. выбор сеттинга [33], определяющий принадлежность игры к определенному типу виртуального мира и сюжетной теме. Примеры сеттингов: научная фантастика, киберпанк, стимпанк, средневековье, постапокалипсис и т. д.
   c. написание **concept документа** [33] – в котором набросаны начальные проработки всех аспектов игры, **vision документа** – описывающий игру, как конечный бизнес-продукт. Эти документы формируют геймдизайн-документ, с помощью которого, художники и команда получает представление о конечном продукте
   d. создание и презентация концепт-артов продюсерам/инвесторам
2) **прототипирование** [33]
   a. создание прототипа для проверки работы основных механик - будут ли они интересны игрокам, а от каких стоит отказаться
3) **средства** [33]
   a. программисты выбирают средства реализации: игровой движок, подсистемы, включающие движок рендеринга, анимаций, физики, звука, систему скриптов, ИИ, сетевой код и прочее.
4) **вертикальный срез**
   a. создание нескольких минут геймплея, части видеоигры, содержащей один/несколько игровых уровней, несколько моделей персонажей, и которая демонстрирует инвесторам/издателям как будет выглядеть полноценная часть проекта, его сеттинг, механики, графика
5) **наполнение контентом**
   a. разработчики настраивают управление, создают законы игровых объектов и их поведения, прорабатывают искусственный интеллект, который с учетом типа игры будет разным. Если стратегии требуют обширной работы над ним, то в более простом жанре, например в гоночном симуляторе, ИИ тоже будет проще, а в сетевом шутере может вовсе отсутствовать

b. левел-дизайнеры проектируют игровые уровни - придумывают интересные игровые ситуации для применения механик, увеличения интерактивности с окружением и раскрытия сюжета
c. художники по окружению разбрасывают по уровням объекты, декорации фоны и визуальные эффекты (свет, тени, горение огня, взрывы и т. д.) руководствуясь геймдизайном
d. дизайнеры, на основе концепт-артов, создают 2D/3D модели, состоящие из полигонов. Затем на модели накладывают заранее отрисованные текстуры, добавляют анимации [34], созданные вручную либо с помощью захвата движений
e. разработка GUI, мини-карты и меню игры

6) **сюжет**
   a. нарративный дизайнер (сценарист) с помощью различных игровых инструментов: **скриптовых событий** - происходящих при триггирировании игроком конкретных событий в случае совершения нужных действий или достижения определенных точек на уровне (нарративный дизайнер продумывает скрипты, а программисты реализовывают), **диалогов** - ведущихся между персонажами, **повествования через окружение** - включающее в себя интерьеры, обстановку локаций, различные записки/дневники, диктофонные записи и т. д., **кат-сцен** - созданных на движке или с помощью рендеринга - пишет сюжет игры, о героях обладающих своими характерами, проблемами; нарративно описывает правила мира и логику повествования
7) **звук**
   a. звукорежиссер записывает звуки для игры
   b. накладывается музыка, саундтрек, написанный композиторами и исполняемый музыкантами. Озвучиваются персонажи [33]
8) **создание** и **тестирование**
   a. объединение всех компонентов разработки и создание альфа-версии, в которую еще будут добавлять или убирать различные элементы [35]
   b. тестирование уровней, скриптов и механик. Исправление ошибок и багов
9) **продажа и поддержка после релиза**
   a. во время разработки издатель занимается маркетингом, рекламой
   b. к выходу налаживаются техподдержка и работа с комьюнити, оптимизируется конечная версию игры, геймдизайнер раздумывает над увеличением привязки монетизации
   c. после продаж работа над игрой не заканчивается - выполняется поддержка продукта в течение нескольких лет, выпуск патчей, добавление в игру DLC и проведение внутриигровых мероприятий/ивентов

2. Технологический раздел

2.1 Исследование современных технологий создания игр

Было проведено исследования современных технологий разработки видеоигр, среди которых, важнейшим инструментом при создании является игровой движок.

Сегодня, в индустрии игр, основными представителями движков, используемых разработчиками, являются Unreal Engine и Unity [36]. Важные отличительные различия между ними:

Таблица - 2.1 – Сравнение Unreal Engine и Unity [37]

Сравнение	Unreal Engine	Unity
Основная аудитория	Независимые разработчики и разработчики AAA игр	Инди-разработчики и разработчики-любители
Простота использования	Легкий уровень разработки используя Blueprint, и более сложный уровень используя C++	Средний уровень сложности используя C#
Открытый источник	Имеет открытый исходный код	Не является открытым
2D/3D	Полностью поддерживаются	Полностью поддерживаются
Искусственный интеллект	Продвинутый ИИ	Ограниченный ИИ
Платформы	Все актуальные платформы	Все актуальные платформы
Графические особенности	Расширенная графика, рендеринг, глобальное и объемное освещение, постобработка и редактор материалов, более мощные технологии и улучшенные анимации	Менее хорошая, чем в UE, рендеринг, глобальное освещение, постобработка и дополнительное объемное освещение через плагин

Резюмируя сравнительный анализ игровых движков, было принято решение использовать Unreal Engine для реализации видеоигры, исходя из следующий факторов:

- Помимо написания скриптов на языке программирования, частичная работа может быть выполнена, используя встроенную систему визуального скриптинга Blueprints;
- UE обладает большим инструментарием, в сравнении с Unity.

2.1.1 Unreal Engine

Для реализации проекта, был выбран игровой движок Unreal Engine, разработанный и поддерживаемый американской компанией Epic Games, представляющий из себя профессиональную межплатформенную среду разработки создания графических приложений, видеоигр для подавляющей части современных операционных систем: Windows PC, Linux, macOS; игровых консолей серий Xbox, PlayStation, Switch, Stadia, Steam Deck; мобильных устройств на базе Android, iOS; гарнитур виртуальной реальности [36].

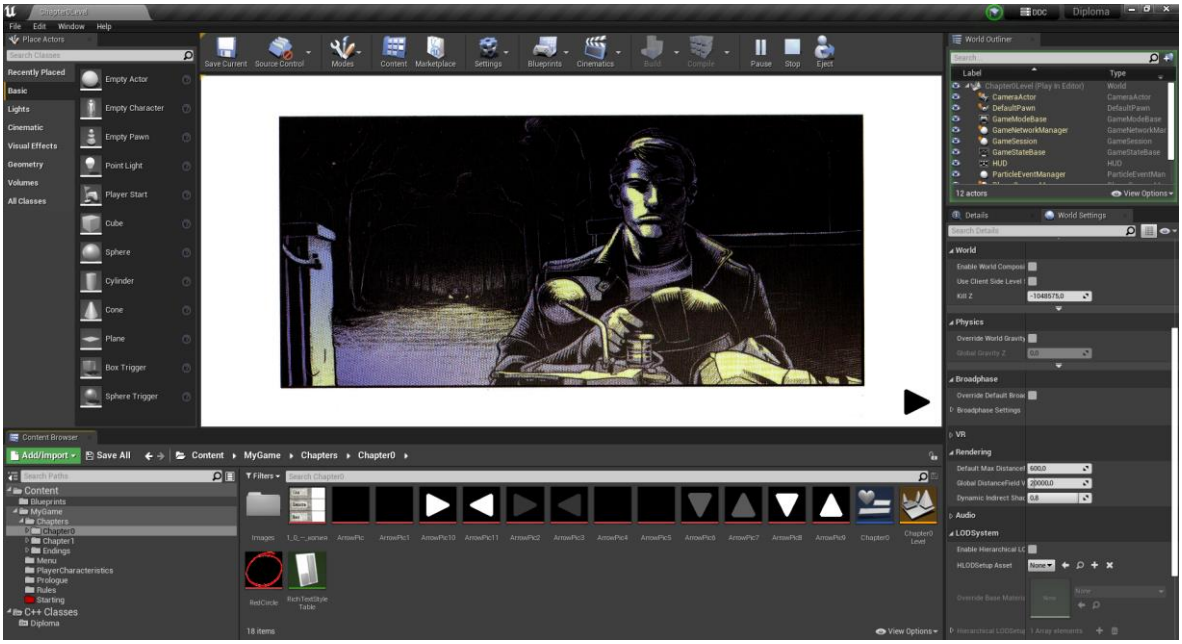

Рисунок - 2.1 – Интерфейс Unreal Engine

Редактор UE состоит из панелей: **Content Browser** – отображает файлы проекта**; Viewport** – обозревает уровень**; Toolbar** – содержит различные функции: компилирование, запуск программы и т. д.**; Modes** – здесь располагаются объекты: свет, объект персонажа, камера, эффекты и т. д.**; Worl Outliner** – отображает все объекты на текущем уровне**; Details** - демонстрирует свойства объектов.

На движке разрабатываются игры всевозможных жанров, как крупный AAA многопользовательский шутер/RPG (ролевая игра) в трехмерном пространстве, так и казуальный, массово ориентированный 2D инди симулятор.

UE распространяется на бесплатной основе, однако разработчики должны отчислять Epic Games денежные компенсации в размере 5% от выручки с продаж, при условии заработка свыше $1,000,000 [37].

В ходе реализации дипломного проекта использовалась последняя актуальная версия движка, а именно Unreal Engine 4.27.3.

При работе с Unreal Engine используется среда разработки Visual Studio.

2.1.2 Visual Studio и C++

Microsoft VS – это комплексная IDE для C++ и .NET разработчиков, она предоставляет набор функций и инструментов для усовершенствования всех этапов разработки программного обеспечения [38].

C++ - объектно-ориентированный язык программирования общего назначения, применяемый в разработке программного обеспечения, используемый в работе с серверами и в создании различных программ, приложений и видеоигр. Является одним из популярнейших языков программирования [39]. Именно на данном языке написан UE движок.

2.1.3 Blueprints

Помимо использования C++, скрипты в играх на Unreal Engine могут вспомогательно либо целиком писаться на встроенной системе визуального скриптинга Blueprints, состоящей из узлов(nodes), предоставляющая возможность не писать построчный код, а программировать визуально, перетаскивая узлы, задавая их свойства и соединяя их между собой. Это в свою очередь задает низкий порог вхождения для разработки и позволяет создавать видеоигры не только программистам, но и людям других специализаций: дизайнерам, художникам и др. [40] Узлы, представляют из себя такие объекты, как переменные, события (events), определяющие начало какой-либо логической последовательности, вызовы функций, операции управления потоком и т. д., которые могут быть использованы в графах чтобы определить функциональность конкретного графа и Blueprint, который содержит его.

2.1.4 Adobe Photoshop

Все спрайты игры будут обработаны/созданы в Photoshop версии 23.0.0. Photoshop - графический редактор компании Adobe, для создания изображений, артов, ретуши и улучшения изображений, веб-дизайна, видео и т. д.

3. Проектная часть

3.1 Разработка игры “Похищение”

В результате исследования, актуальных данных игрового рынка и актуальных технологий разработки, было принято решение создать видеоигру в жанре Adventure на платформе Unreal Engine, исходя из трех факторов:

1. Данный жанр входит в пятерку популярнейших
2. Одиночные игры не просто популярны, но продолжают приносить большие доходы
3. Человеческий фактор – разработчик просто любит квесты

Этапы разработки:

1) **концепт игры**
   a. adventure (приключенческая игра) видеоигра в жанре визуального романа игровые механики: физическое состояние главного героя определяется значением здоровья, которое будет уменьшаться при столкновениях с противниками (в зависимости от исхода боя), и увеличиваться благодаря найденным предметам; инвентарь предметов (игрок может носить с собой до 3-х предметов одновременно); также на протяжении прохождения, в мире героя важную роль играет время, значение которого, в текущий момент игры влияет на ход приключения; различные характеристики (сила, ловкость, воля) будут учитываться при стычках героя с противниками; игрок может свободно передвигаться и исследовать локации, находить потайные комнаты, общаться с персонажами, искать предметы, разгадывать шифры, изучать улики, участвовать в стычках и погонях и принимать решения в нелинейном сюжете в ряде различных концовок
   b. сеттинга игры будет представлен в виде комикса в мрачных тонах мистического детектива с элементами хоррора. Сюжет будет развиваться в современном времени
   c. проект будет реализован как игровая адаптация графического романа “Captive”
2) **средства**
   a. инструментов реализации выступит игровой движок Unreal Engine версии 4.27, используя C++ и Blueprints. Спрайты игры будут обработаны/созданы в Adobe Photoshop 2022
3) **наполнение контентом**
   a. мир игры наполнен различными персонажами, существами и предметами: главный герой / дочь героя / друзья и союзники героя / разного рода противники и звери / предметы для собирательства (фонарь, компас, ключ и т. д.)

b. меню игры предоставляет возможность начать и выйти из игры
c. меню паузы предоставляет возможность остановить игру, продолжить и выйти из неё
d. графический интерфейс игры демонстрирует таблицу характеристик персонажа: здоровье, сила, ловкость и воля; должен присутствовать инвентарь предметов персонажа. Переключение между сценами происходит по нажатию кнопок стрелок либо наводясь курсором мыши и щелкая по определенным областям на экране монитора.
При гибели персонажа на экране появляется кнопка с возможностью возвращения в меню

3) **сюжет**
a. дочь главного героя похитили и увезли в жуткий таинственный особняк. За неё требуют баснословный выкуп, но нет гарантии, что девочка останется жива даже после выплаты. Вооружившись пистолетом, герой прибывает в особняк, полный решимости спасти дочь и выяснить, кто стоит за похищением.
Нелинейный динамический сюжет со множеством концовок

4) **создание и тестирование**
a. реализация всех вышеописанных концептов и компонентов внутри игры
b. тестирование конечной программы

Диаграмма - 3.1 – Диаграмма последовательности видеоигры “Похищение”

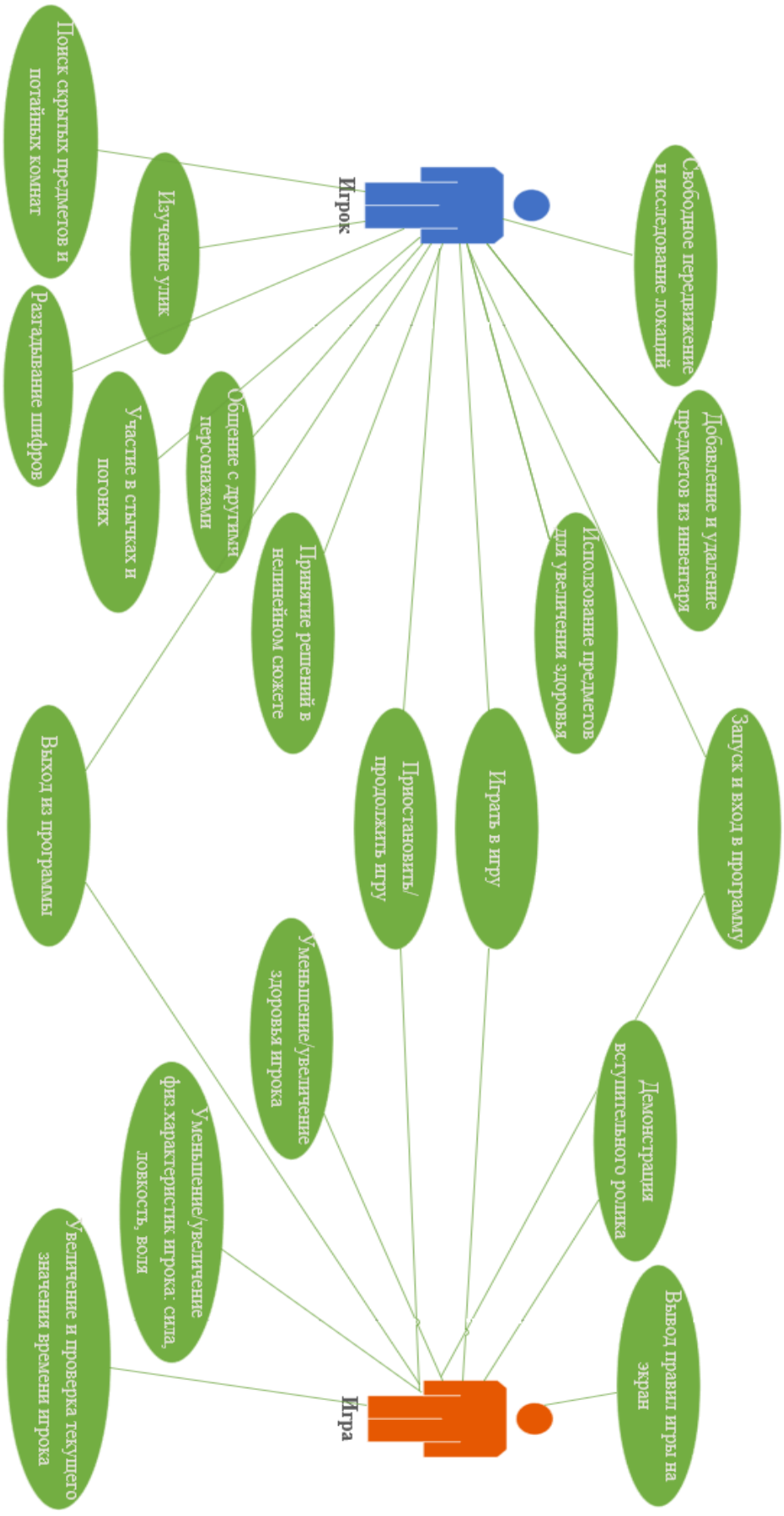

Диаграмма - 3.2 – Use case диаграмма видеоигры “Похищение”

4. Экспериментальный раздел

4.1 Реализация видеоигры “Похищение”

4.1.1 Заставка и пролог

При помощи раздела File в UE или нажатия левой кнопки мыши в папке с компонентами, был создан уровень игры (Level) и было задано ему имя, “StartingLevel”. Для использования графики/текста и отображения любой информации задействуется User Interface (UI), от которого наследуется Widget Blueprint, ему тоже задается имя – “StartingWidget”, в котором создаются элементы интерфейса.

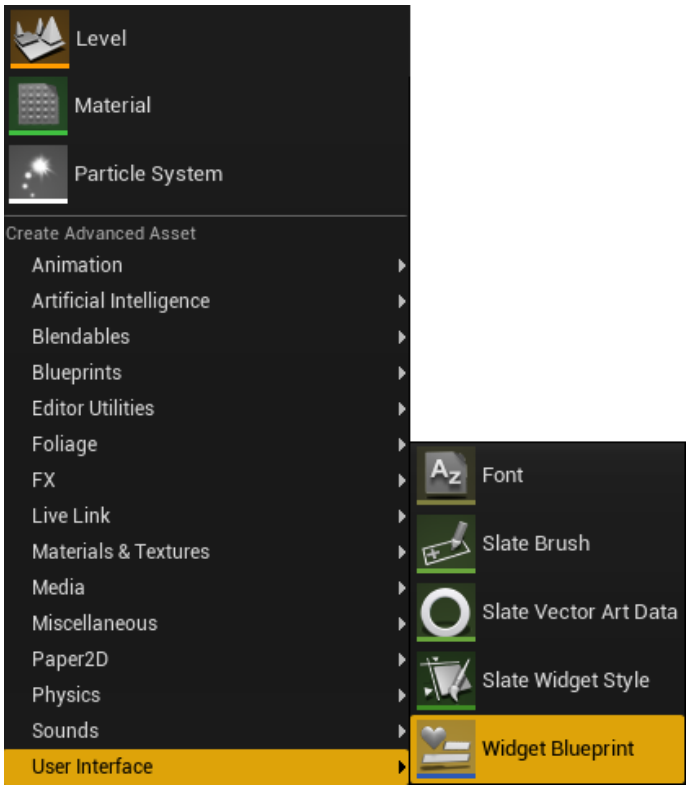

Рисунок - 4.1 – UI в Unreal Engine

Внутри Widget Blueprint по умолчанию располагаются 2 основные панели:

– Designer - здесь реализуется дизайн интерфейса, располагаются изображения, кнопки, текстовые поля ввода и вывода и другие элементы;
– Graph - где реализуется работа виджетов посредством Blueprints.

Далее в проекте была создана анимация, которая запускается после старта программы и демонстрирует название игры. Для этого в панели Graph в

“StartingWidget” используется функция “Play Animation”, определяющая время и скорость проигрывания вступительной заставки.

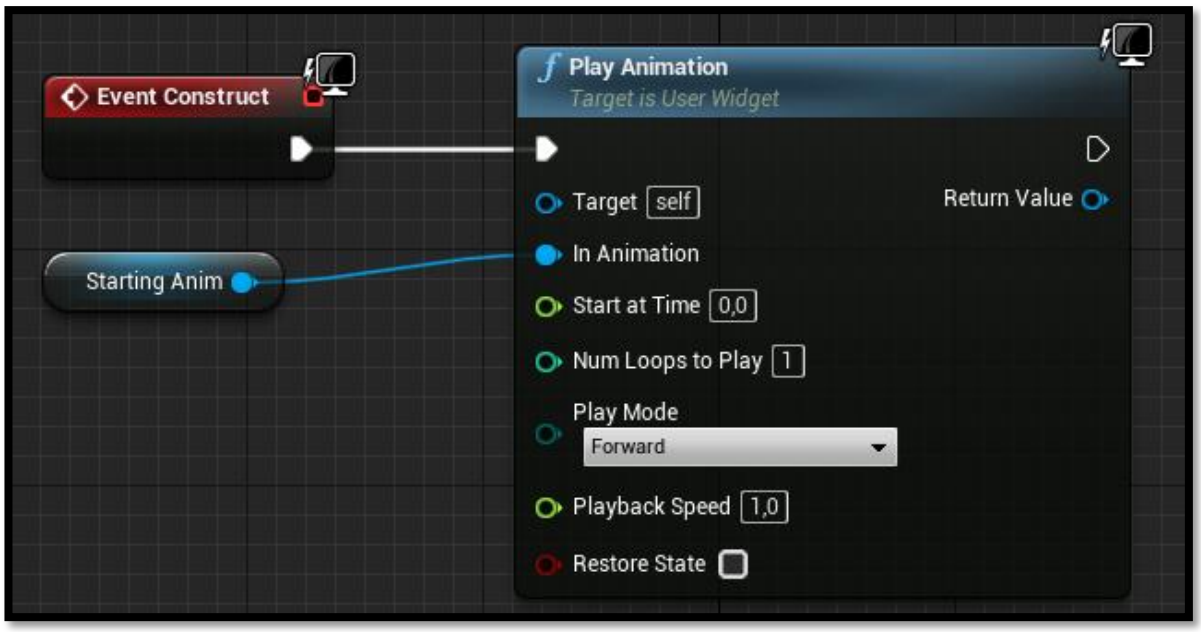

Рисунок - 4.2 – Blueprint узлы, осуществляющие исполнение анимации заставки игры

При запуске программы, на экран выведется пустой уровень, так как для отображения Widget Blueprint, внутри “StartingLevel” нужно создать событие, которое исполнит анимацию при старте уровня.

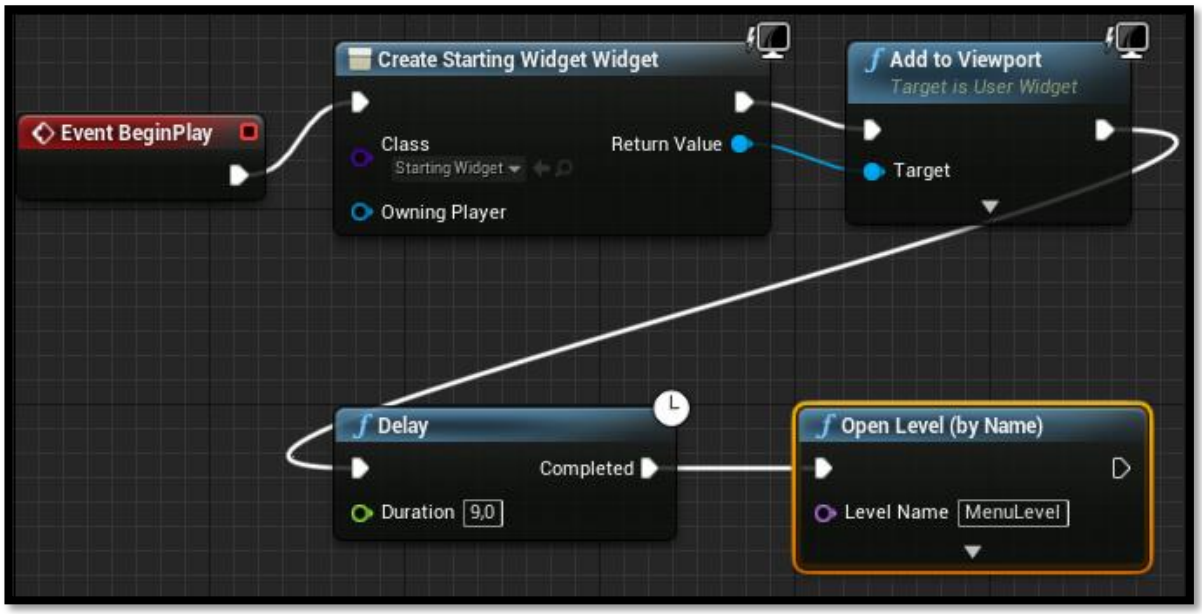

Рисунок - 4.3 - Blueprint узлы в панели Graph уровня игры, реализующие при старте уровня “StartingLevel” вывод на экран “StartingWidget” с задержкой в 9 секунд, а затем переход на уровень “MenuLevel”

В панели Designer любой созданный графический элемент будет являться дочерним к панели Canvas, отображающая любые графические элементы.

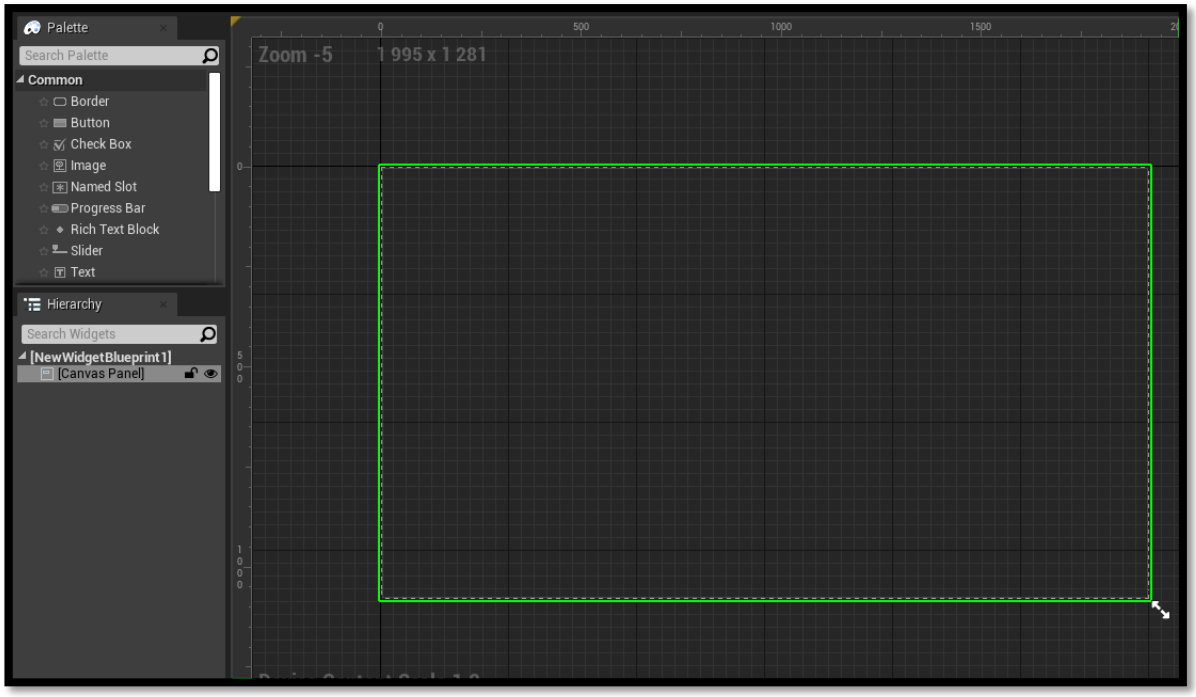

Рисунок - 4.4 – Canvas Panel

В Canvas вкладывается компонент Image, в свойства которого будет передан соответствующий спрайт.

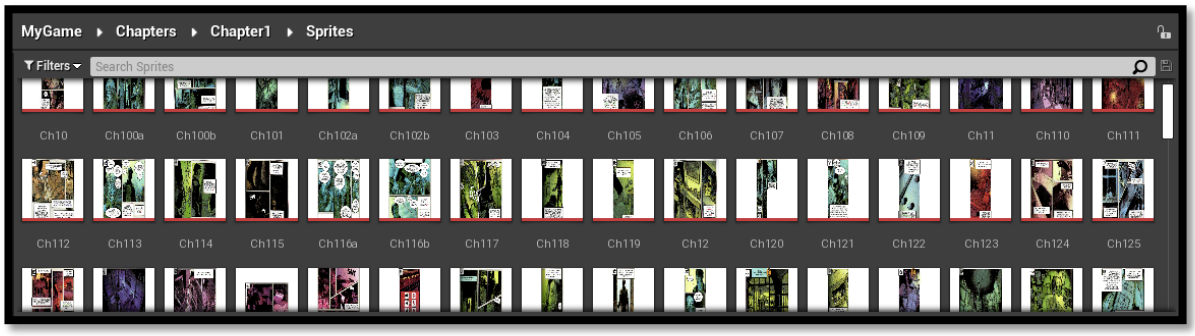

Рисунок - 4.5 – Папка проекта со спрайтами

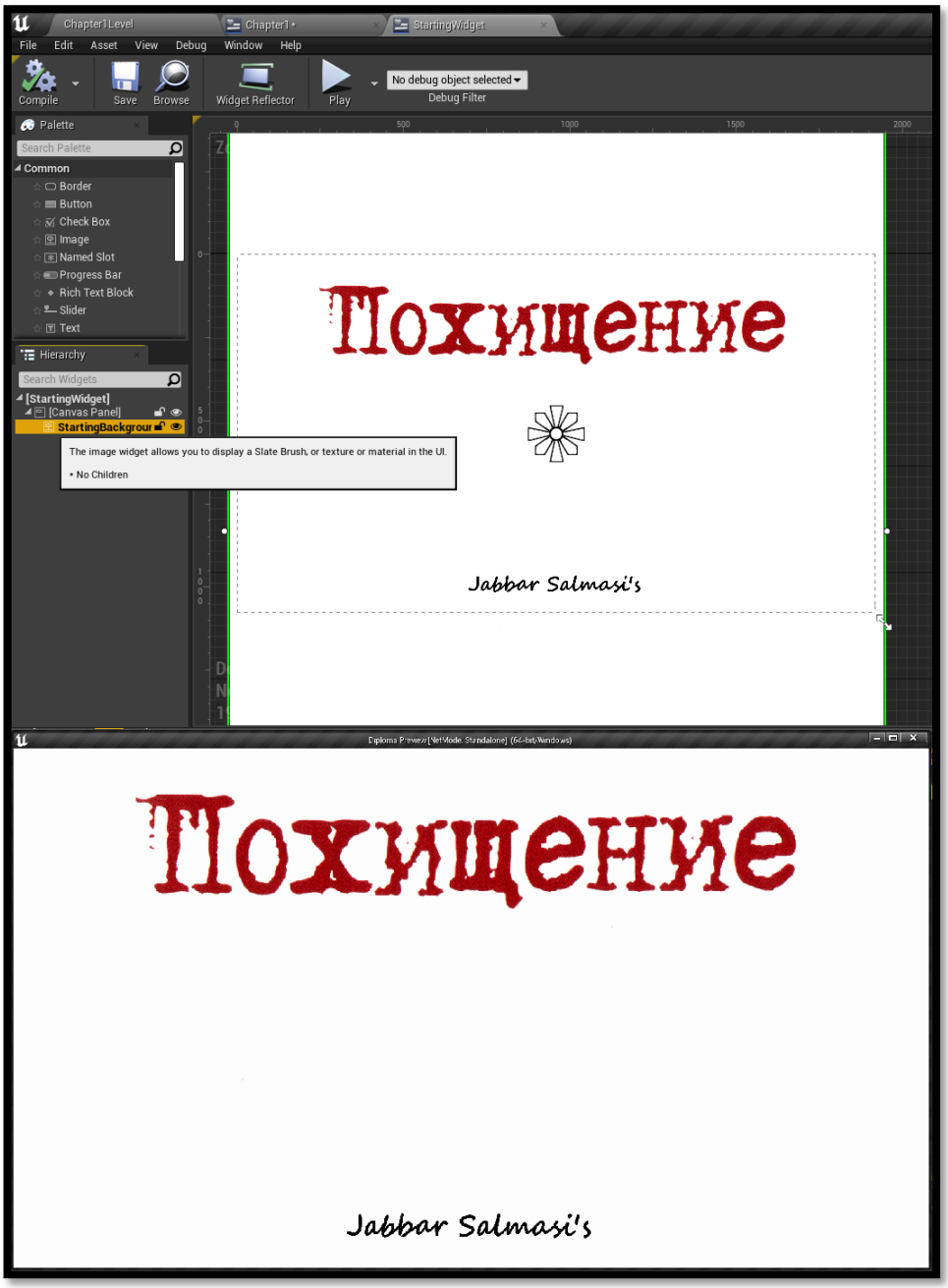

Рисунок - 4.6 – Вступительная заставка игры

Затем была разработана система узлов, запускающая заранее смонтированный ролик, представляющий собой пролог видеоигры, после нажатия кнопки “Начать игру”, которая будет располагаться в главном меню.

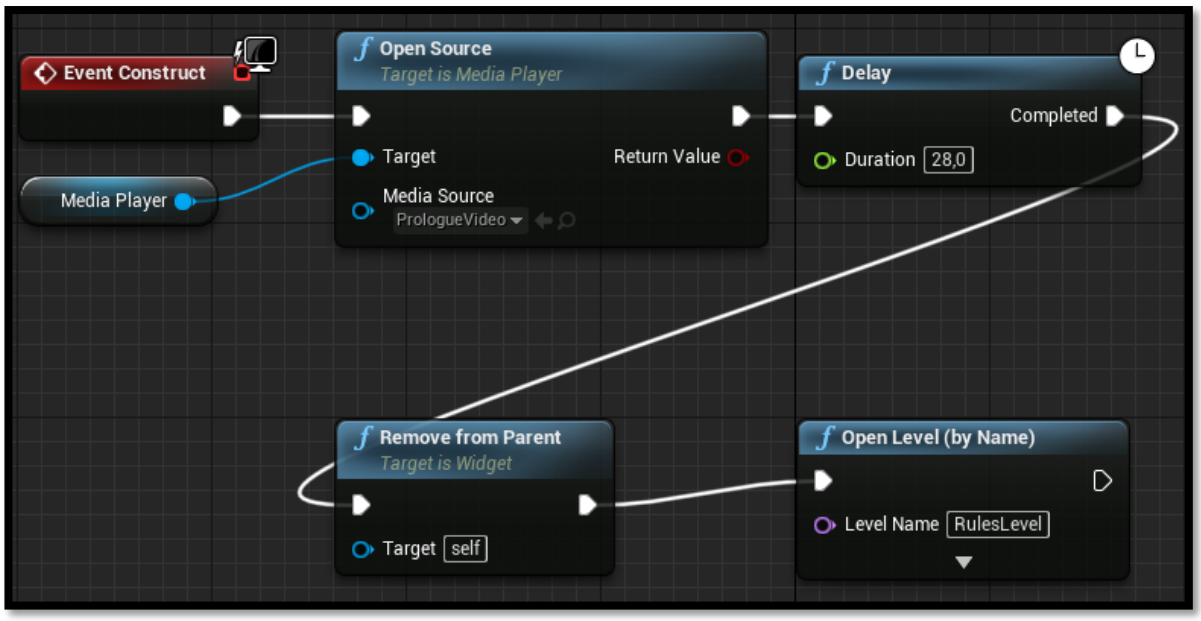

Рисунок - 4.7 – Blueprint узлы, реализующие событие, которое воспроизводит видеоролик “PrologueVideo” в течение 28 секунд, а затем исполняет переход на уровень “RulesLevel”

4.1.2 Меню и интерфейс игры

Аналогично в проекте был создан уровень “MenuLevel” и “MenuWidget”**.** Но работа кнопок в главном меню совершается при помощи C++ класса. Для этого в Unreal Engine в папке с классами был создан C++ класс “MenuWidget” (наследуемый от класса “UserWidget”), от которого наследовался Blueprint “MenuWidget”.

Тем самым, виджет “MenuWidget” созданный в Blueprint, представляющий из себя меню с кнопками, наследует функционал работы кнопок из одноименного C++ класса.

В Blueprint виджете в Canvas Panel были добавлены кнопки “Начать игру” и “Выход”, а также спрайт для служащий задним фоном меню.

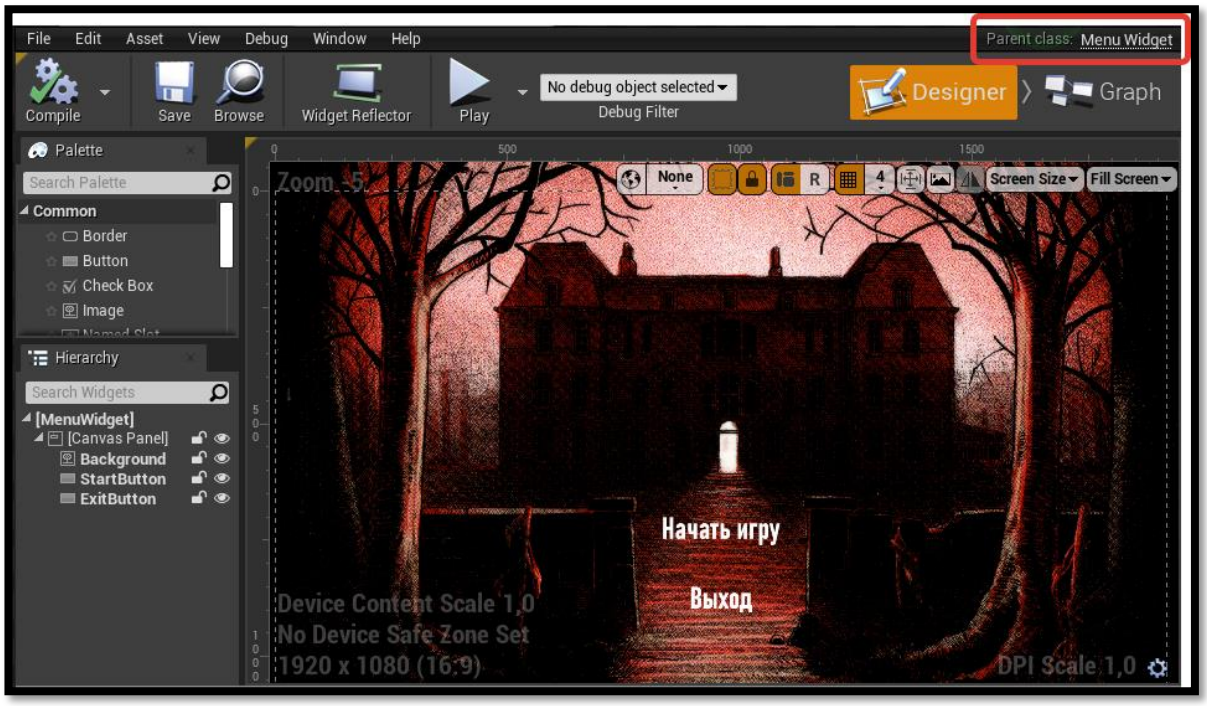

Рисунок - 4.8 – Демонстрация наследования Blueprint “MenuWidget” от C++ класса “MenuWidget” и компонентов Canvas панели

```cpp
#include "MenuWidget.h"
#include "Components/Button.h"

void UMenuWidget::NativeConstruct() {

    if (StartButton) {
        StartButton->OnClicked.AddDynamic(this, &UMenuWidget::StartButtonOnClicked);
    }

}
```

Рисунок - 4.9 – .cpp файл в котором реализуется условие, выполняемое при нажатии кнопки “StartButton”

```h
#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"
//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "MenuWidget.generated.h"

UCLASS()
class DIPLOMA_API UMenuWidget : public UUserWidget
{
    GENERATED_BODY()
protected:
    UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
    class UButton* StartButton;
    //StartButton - переменная, представляющая из себя кнопку в игре

    virtual void NativeConstruct() override;

    UFUNCTION()
        void StartButtonOnClicked() {
        //Создаем функцию, которая перенесет игрока на уровень с игрой, при нажатии кнопки
        //В данном случае кнопка "StartButton"
        UGameplayStatics::OpenLevel(this, "PrologueLevel");
        //UGameplayStatics - класс содержащий функции для геймплея
        //благодаря которому будем использовать метод OpenLevel,
        //служащий для перемещения на другой уровень.
        //в параметр которого передадим имя уровня, который нам нужен

    };

};
```

Рисунок - 4.10 - .h файл, содержащий функцию “StartButtonOnClicked”, запускающаяся при исполнении условия из .cpp файла и выполняющая начало игры, переходом к уровню “PrologueLevel”

```cpp
#include "MenuWidget.h"
#include "Components/Button.h"

void UMenuWidget::NativeConstruct() {

    if (ExitButton) {
        ExitButton->OnClicked.AddDynamic(this, &UMenuWidget::ExitButtonOnClicked);
    }

}
```

Рисунок - 4.11 – .cpp файл в котором реализуется условие, выполняемое при нажатии кнопки “ExitButton”

```h
#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"
//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "MenuWidget.generated.h"

UCLASS()
class DIPLOMA_API UMenuWidget : public UUserWidget
{
    GENERATED_BODY()

protected:
    UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
        class UButton* ExitButton;
    //ExitButton - переменная, представляющая из себя кнопку в игре

    virtual void NativeConstruct() override;

    UFUNCTION()
        void ExitButtonOnClicked() {
        UKismetSystemLibrary::QuitGame(this, nullptr, EQuitPreference::Quit,false);

    };
};
```

Рисунок - 4.12 - .h файл, содержащий функцию “ExitButtonOnClicked”, запускающаяся при исполнении условия из .cpp файла и выполняющая выход из игры

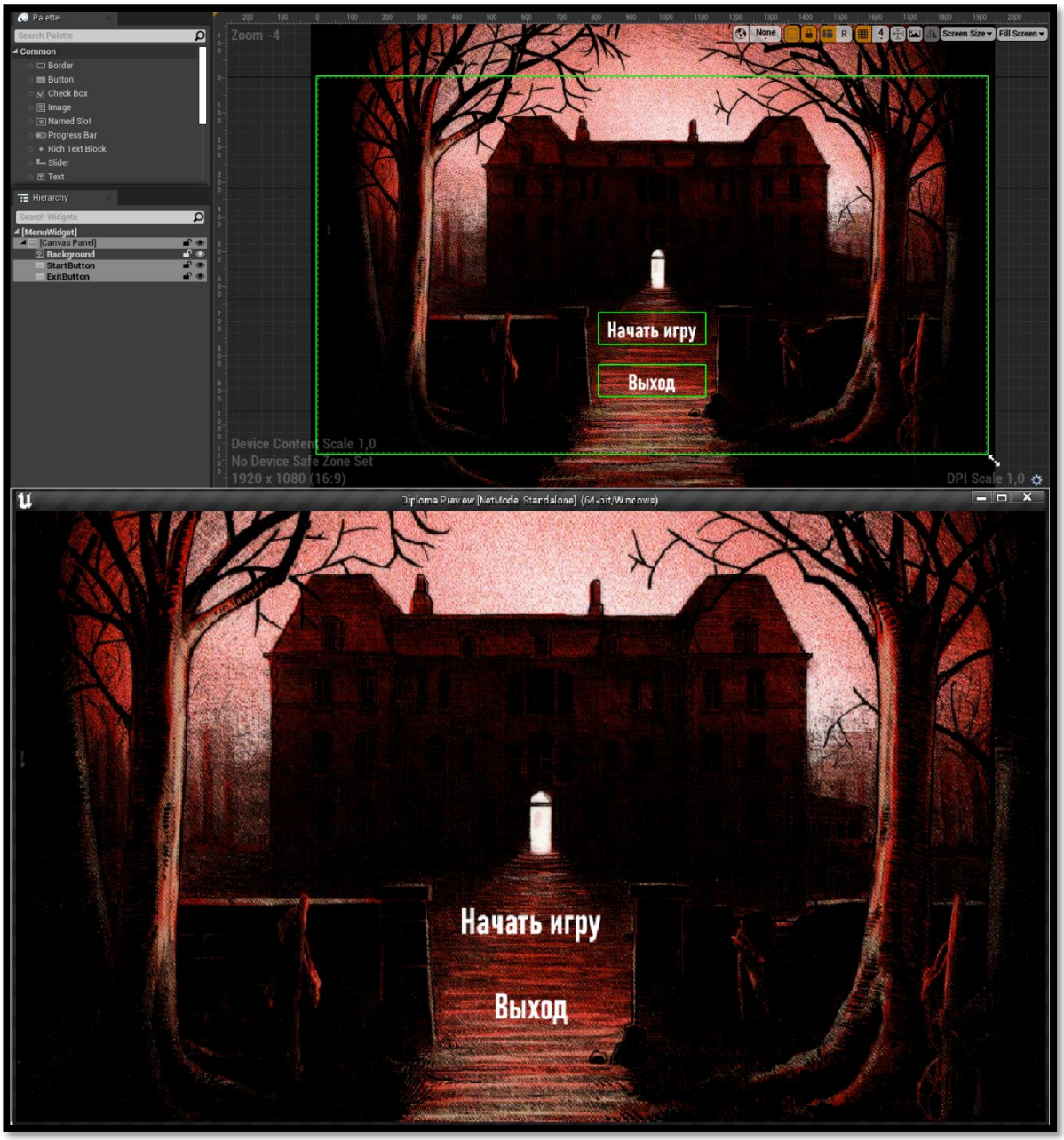

Рисунок - 4.13 – Главное меню игры

Далее в отдельном виджете и уровне был разработан интерфейс, который был затем наложен на игровые уровни.

Рисунок - 4.14 – Часть интерфейса, демонстрирующая таблицу характеристик персонажа

Рисунок - 4.15 - Часть интерфейса, демонстрирующая инвентарь предметов персонажа

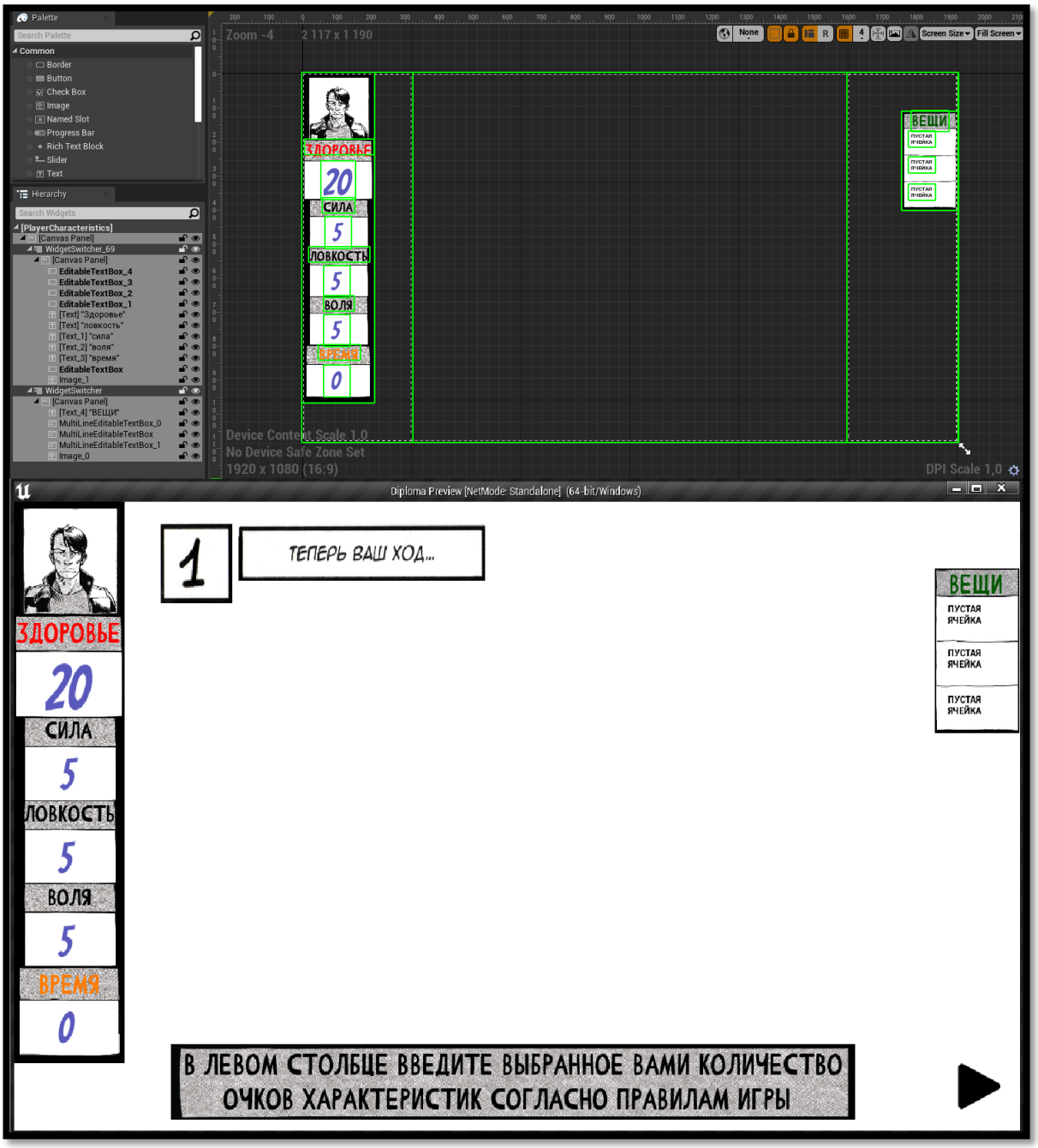

Рисунок - 4.16 – Интерфейс игры

После были созданы уровни содержащие сотни игровых локаций, сцен в виде спрайтов, по которым игрок будет перемещаться по нажатию кнопок стрелок либо наводясь курсором мыши и щелкая по определенным областям на экране монитора.

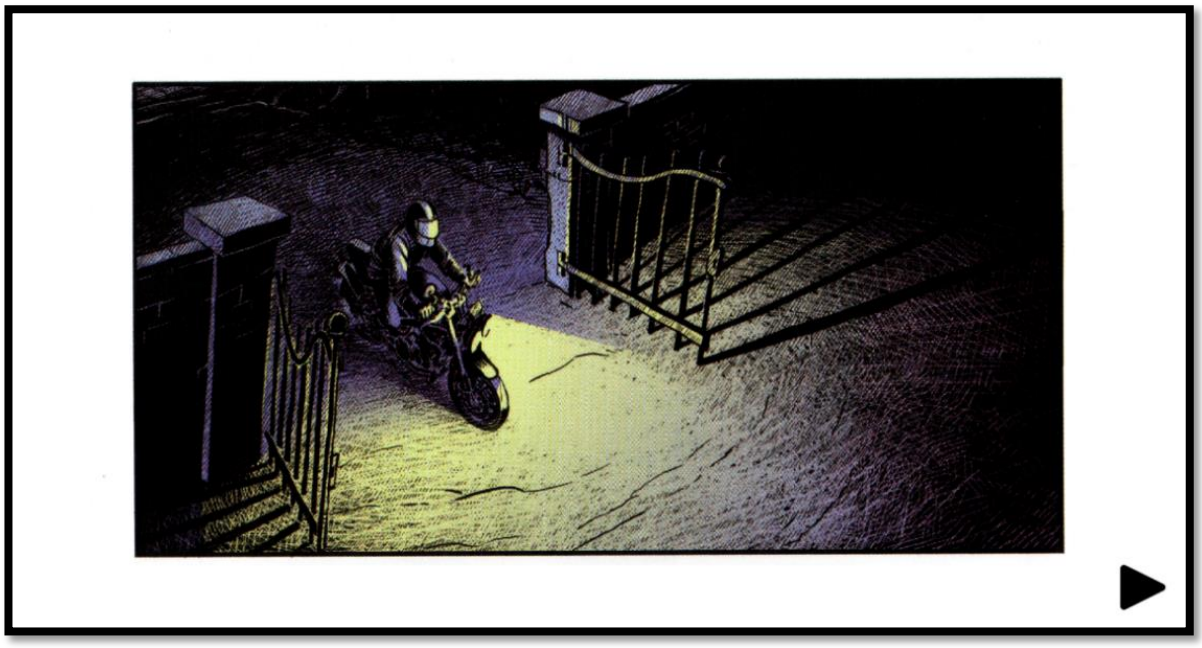

Рисунок - 4.17 - Переключение между сценами по нажатию кнопок (в данном случае стрелка в нижнем углу экрана)

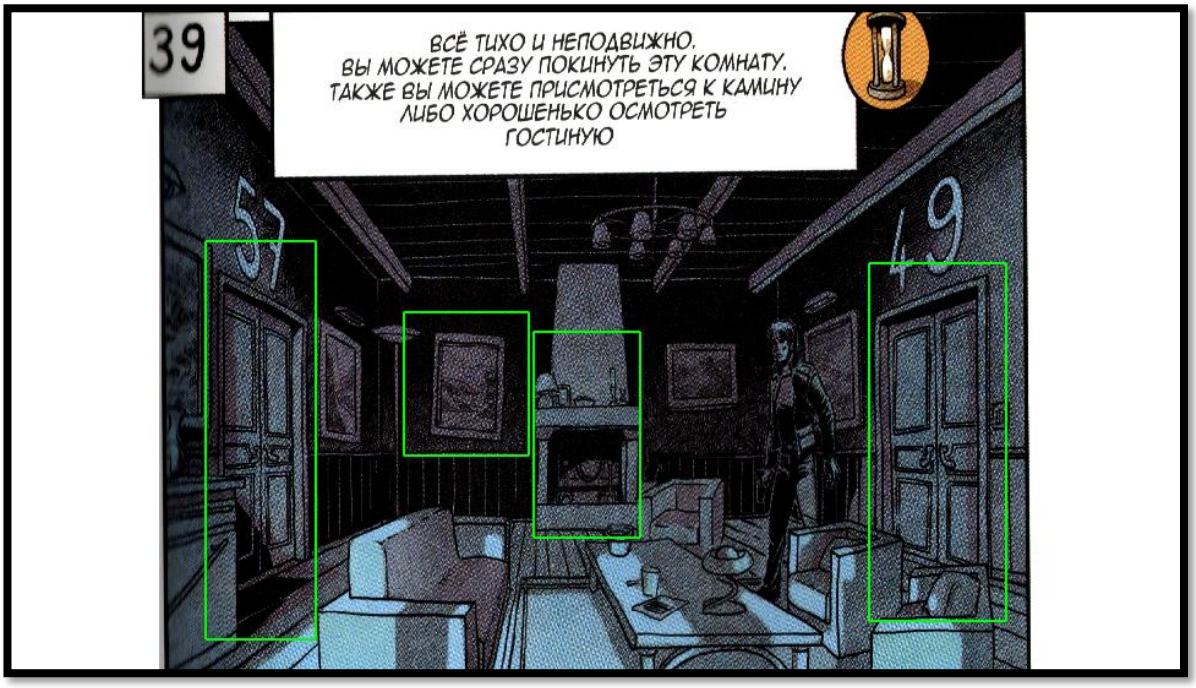

Рисунок - 4.18 – Скрытые области, при нажатии на которые производится переход между сценами и локациями

Чтобы игрок имел возможность выйти из программы в момент игры, было добавлено меню паузы. Для этого был создан отдельный виджет “PauseWidget”, в котором реализован дизайн меню паузы и указан уровень замыливания экрана. Были заданы узлы определяющие функции продолжить игру и покинуть.

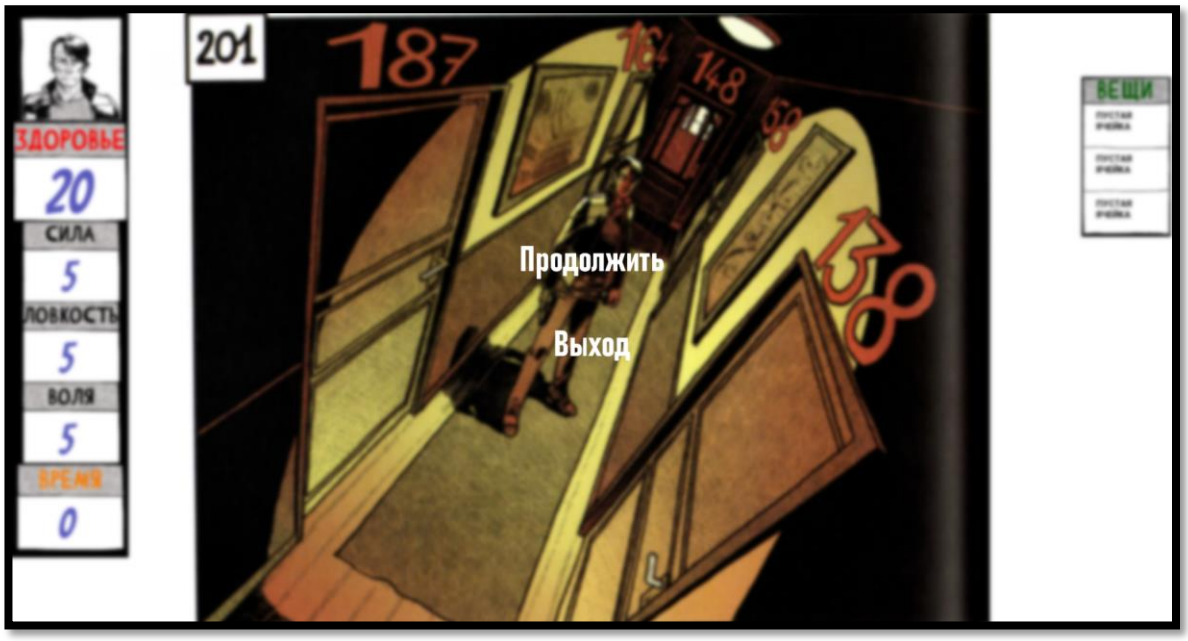

Рисунок - 4.19 – Меню паузы игры

В уровнях игры реализованы узлы для отображения меню паузы:

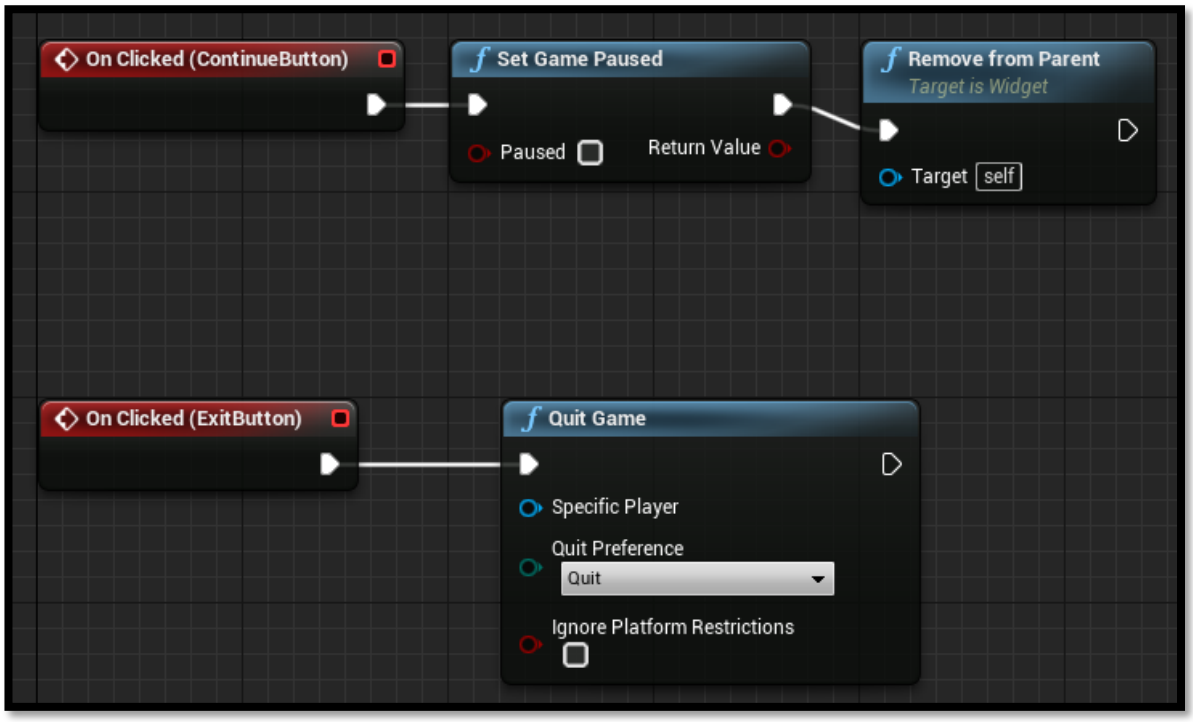

Рисунок - 4.20 - Blueprint узлы, реализующие функцию продолжения игры при нажатии кнопки “ContinueButton” и покинуть игру при нажатии “ExitButton”

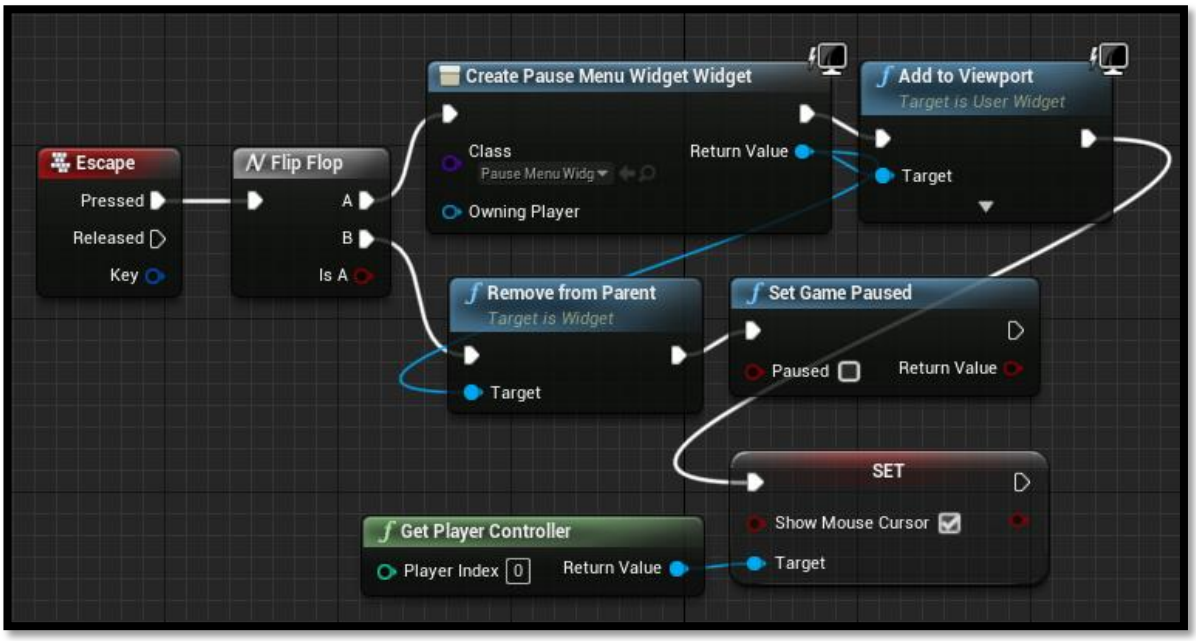

Рисунок - 4.21 - Blueprint узлы в панели Graph уровня игры, реализующие условие при нажатии клавиши “Escape”

Чтобы переходить между сценами и локациями по нажатию кнопок/областей на экране, используется узлы с параметрами индексации кадров.

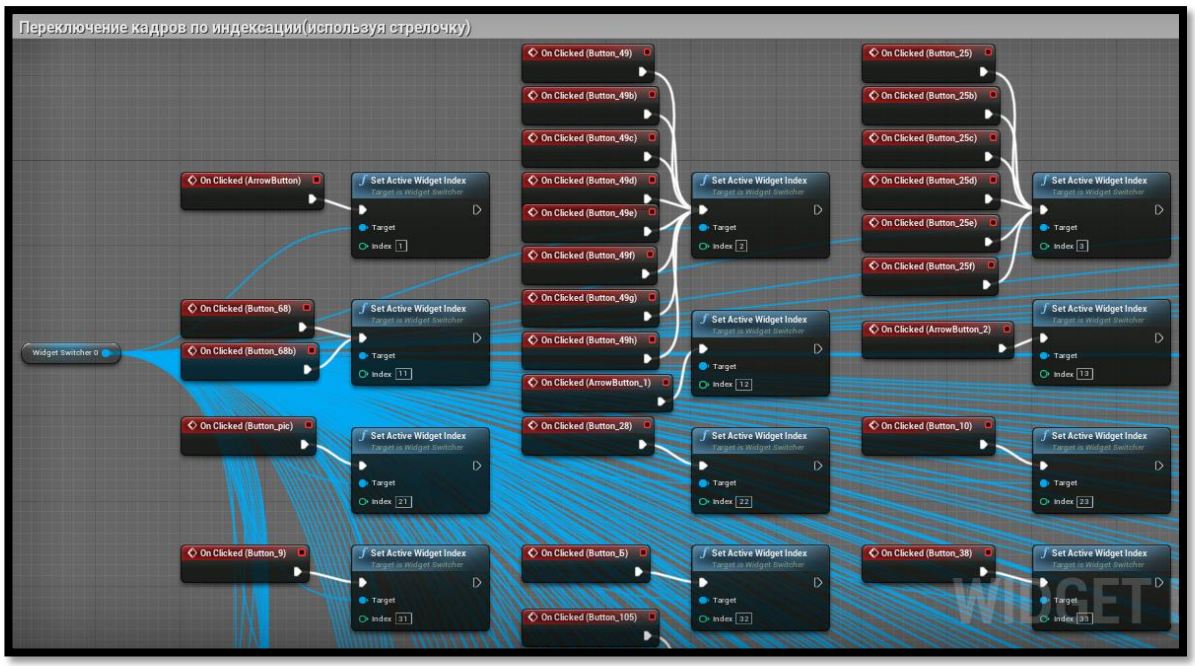

Рисунок - 4.22 – Пример Blueprint узлов, реализующих переключение между сценами/локациями используя индексы виджета

Для работы со сценами и локациями, используется панель Widget Switcher. В ней создаются дополнительные Canvas панели, содержащие спрайты, Text Box-ы и кнопки.

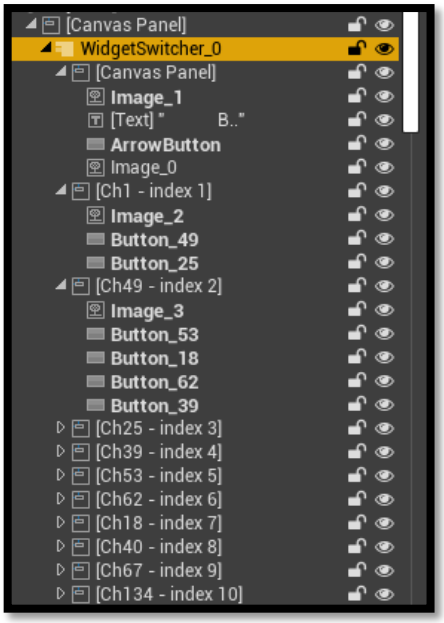

Рисунок - 4.23 – Иерархия Canvas Panel

После гибели героя, на экран выводится кнопка “Конец”, по нажатии которой, игрока перенесет в главное меню. Переход осуществляется, используя C++ класс.

Рисунок - 4.24 – Пример одной из концовок игры с выводом на экран кнопки “Конец”, позволяющая вернуться в главное меню.

```cpp
#include "DeathBySpiderWidget.h"
#include "Components/Button.h"

void UDeathBySpiderWidget::NativeConstruct() {

    if (EndAfterDeathBySpiderButton) {
        EndAfterDeathBySpiderButton->OnClicked.AddDynamic(this, &UDeathBySpiderWidget::EndAfterDeathBySpiderButtonOnClicked);
    }

}
```

Рисунок - 4.25 – Пример .cpp файла в котором реализуется условие, выполняемое при нажатии кнопки “EndAfterDeathBySpiderButton”, которая выводится на экран игры после смерти главного героя

```cpp
#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"
//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "DeathBySpiderWidget.generated.h"

UCLASS()
class DIPLOMA_API UDeathBySpiderWidget : public UUserWidget
{
    GENERATED_BODY()
protected:
    UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
        class UButton* EndAfterDeathBySpiderButton;
    //EndAfterDeathBySpiderButton - переменная, представляющая из себя кнопку в игре

    virtual void NativeConstruct() override;

    UFUNCTION()
        void EndAfterDeathBySpiderButtonOnClicked() {
        //Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
        //В данном случае кнопка "EndAfterDeathBySpiderButton"
        UGameplayStatics::OpenLevel(this, "MenuLevel");
        //UGameplayStatics - класс содержащий функции для геймплея
        //благодаря которому будем использовать метод OpenLevel, служащий для перемещения в меню игры
    };
};
```

Рисунок - 4.26 – Пример .h файла, содержащего функцию “EndAfterDeathBySpiderOnClicked”, которая выполняет переход в главное меню

4.1.3 Игровой процесс

Физическое состояние главного героя определяется значением здоровья, которое будет уменьшаться при столкновениях с противниками (в зависимости от исхода боя), и увеличиваться благодаря найденным предметам; инвентарь предметов (игрок может носить с собой до 3-х предметов одновременно); также на протяжении прохождения, в мире героя важную роль играет время, значение

которого, в текущий момент игры влияет на ход приключения; различные характеристики (сила, ловкость, воля) будут учитываться при стычках героя с противниками; игрок может свободно передвигаться и исследовать локации, находить потайные комнаты, общаться с персонажами, искать предметы, разгадывать шифры, изучать улики, участвовать в стычках и погонях и принимать сложный выбор в нелинейном сюжете со множеством концовок.

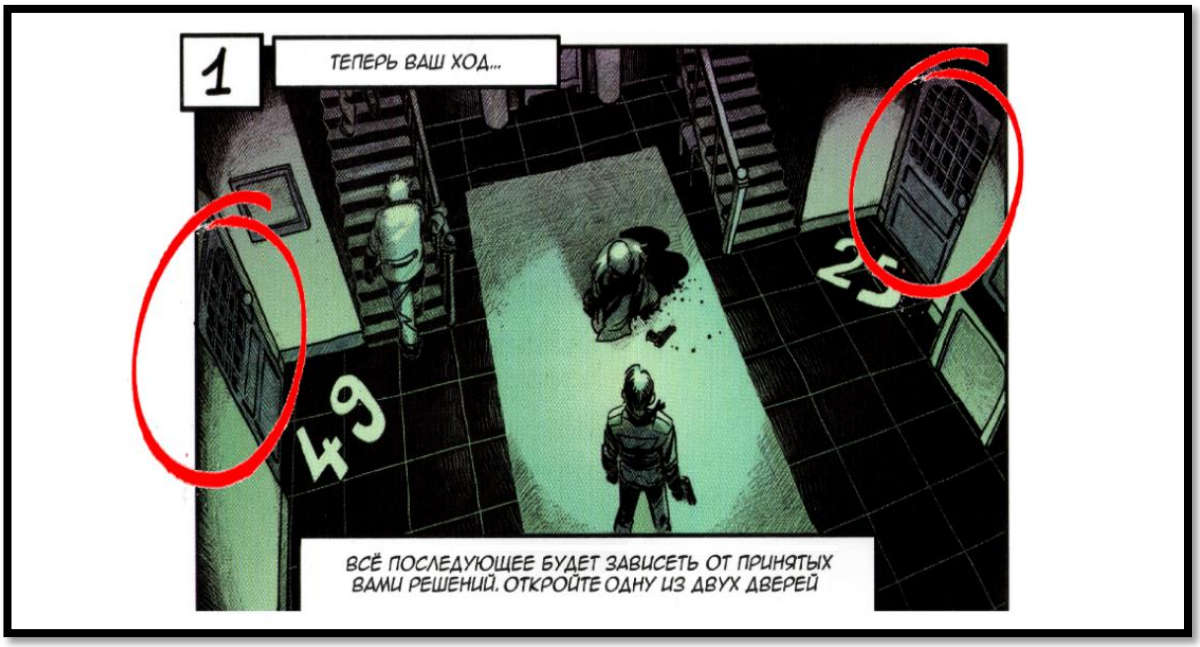

Рисунок - 4.27 - Игровой процесс выбора, стоящий перед игроком

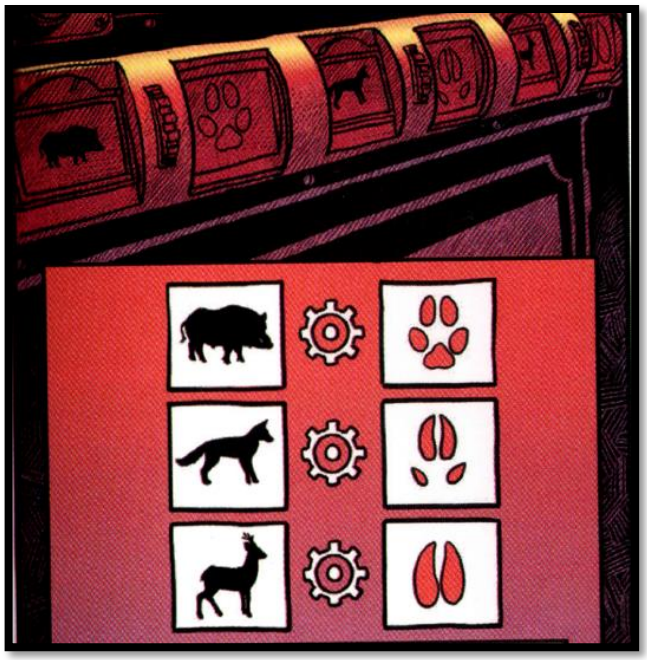

Рисунок - 4.28 – Кадр из игры, демонстрирующий возможность игрока решить одну из загадок с тайным кодом

Рисунок - 4.29 – Пример нескольких концовок игры

4.2 Тестирование программы

На всех этапах разработки компьютерной программы своевременно проводилось тестирование реализованных игровых механик, безошибочного функционирования интерфейса и управления видеоигры.

При окончании реализации проекта тестирование программы было успешно завершено.

4.3 Калькуляция проекта

Для реализации настоящего проекта было задействовано множество различных объектов.

Таблица - 4.1 – Объекты для реализации видеоигры

Элемент	Комментарий
Локации	более 100 с использованием спрайтов
Сцены	280
Скрипты	12 скриптов с общим количеством строк кода более 400
Заставка игры	
Прологовое и финальное видео игры	
Уровни	13
Виджеты	более 20
Дополнительный шрифт	1
Используемые внутриигровые предметы	16 с использованием спрайтов
Спрятанные записки	7 с использованием спрайтов
Спрайты	более 400 изображений

ЗАКЛЮЧЕНИЕ

Результатом дипломного проекта является разработка Adventure (приключенческая игра) видеоигры “Похищение” в жанре визуального романа, на игровом движке Unreal Engine для персонального компьютера.

В процессе исследования предметной области были проведёны исследование и анализ игровой индустрии, актуальных данных игрового рынка и современных технологий разработки видеоигр, а также сделан следующий вывод: индустрии видеоигр является гигантским и растущим в быстром темпе рынком, приносящим большие доходы. Геймеры — это огромный пласт общего населения нашей планеты, большая часть которых проживает в Китае, США и Японии. Игры для персональных компьютеров приблизительно стоят на одном уровне с консольными, но оба проигрывают мобильному рынку. Однако несмотря на это, видеоигры однопользовательского жанра для компьютеров и игровых приставок все еще востребованы и по статистике занимают первые места по продажам.

В процессе проектирования разрабатывались требования к компьютерной игре, а затем проводилась реализация самой программы.

Таким образом, все поставленные задачи данного проекта были решены, а цели успешно достигнуты.

Данная видеоигра является открытой для разработчиков и представляет собой законченный программный продукт, однако может быть доработана на основании новых технических требований. Например, к игровому процессу может быть добавлено музыкальное сопровождение.

В дальнейшем возможен выпуск игры в сервисах цифровой дистрибьюции компьютерных и мобильных игр.

Программный продукт не нуждается в особых требованиях к аппаратной/программной составляющей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Meaning of video game in English // Электронная версия на сайте https://dictionary.cambridge.org/dictionary/english/video-game.
2 Weiner C. Oral History Interviews - Edward Condon - Session II. 1968 // Электронная версия на сайте https://www.aip.org/history-programs/niels-bohr-library/oral-histories/4997-2.
3 Cohen D. S. Cathode-Ray Tube Amusement Device: The First Electronic Game. – 2019 // Электронная версия на сайте https://www.lifewire.com/cathode-ray-tube-amusement-device-729579.
4 Cohen D. S. OXO aka Noughts and Crosses - The First Video Game. – 2019 // Электронная версия на сайте https://www.lifewire.com/oxo-aka-noughts-and-crosses-729624.
5 Goodavage J.F. Space War!: A Computer Game Today, a Reality Tomorrow?, 1972. – С. 34-37, 92, 94 // Электронная версия на сайте http://www.kaleberg.com/spacewar/page1.html.
6 Ashcraft B. Arcade Mania: The Turbo-charged World of Japan's Game Centers, 2008. – С. 72.
7 Pitts B. The Galaxy Game // Электронная версия на сайте http://infolab.stanford.edu/pub/voy/museum/galaxy.html.
8 Mark J.P. Before the Crash: Early Video Game History, 2012. – С. 3.
9 1TL200: A MAGNAVOX ODYSSEY // Электронная версия на сайте https://videogamehistorian.wordpress.com/2015/11/16/1tl200-a-magnavox-odyssey/.
10 Baer R. Genesis: How the Home Video Games Industry Began // Электронная версия на сайте http://www.ralphbaer.com/how_video_games.htm.
11 Kent S. L. The Ultimate History of Video Games: From Pong to Pokémon and Beyond: the Story Behind the Craze that Touched Our Lives and Changed the World, 2001. – С. 43-45.
12 Hockey TV // Электронная версия на сайте https://segaretro.org/Hockey_TV.
13 Hansen D. Game On! Video Game History From Pong and Pac-Man to Mario, Minecraft and More - Нью-Йорк: MacMillan Publishing Group, LLC, 2016. – 368 с.
14 Owen S.G. E.T. cartridges found in infamous Atari landfill. 2014 // Электронная версия на сайте https://www.polygon.com/2014/4/26/5656282/atari-et-landfill-new-mexico-found-cartridges.
15 Wijman T. The Games Market and Beyond in 2021: The Year in Numbers. – 2021 // Электронная версия на сайте https://newzoo.com/insights/articles/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming/.

16 Guttmann A. Estimated media revenue worldwide in 2020, by category. – 2022 // Электронная версия на сайте https://www.statista.com/statistics/1132706/media-revenue-worldwide/.
17 Wijman T. The Games Market's Bright Future: Player Numbers Will Soar Past 3 Billion Towards 2024 as Yearly Revenues Exceed $200 Billion. – 2021 // Электронная версия на сайте https://newzoo.com/insights/articles/the-games-markets-bright-future-player-numbers-will-soar-past-3-billion-towards-2024-as-yearly-revenues-exceed-200-billion/.
18 Top 100 Countries/Markets by Game Revenues for 2018. – 2021 // Электронная версия на сайте https://www.resetera.com/threads/top-100-countries-markets-by-game-revenues-for-2018-newzoo.57251/.
19 Clement J. Video game sales in the United States in 2018, by genre. – 2022 // Электронная версия на сайте https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/.
20 Marcus L. The 50 Biggest Video Game Franchises by Total Revenue. – 2020 // Электронная версия на сайте https://www.visualcapitalist.com/50-biggest-video-game-franchises-revenue/.
21 List of best-selling video games // Электронная версия на сайте https://en.wikipedia.org/wiki/List_of_best-selling_video_games.
22 Sinclair B. Spider-Man sells 3.3 million in three days. – 2018 // Электронная версия на сайте https://www.gamesindustry.biz/articles/2018-09-20-spider-man-sells-3-3-million-in-three-days.
23 MARVEL'S SPIDER-MAN. – 2018 // Электронная версия на сайте https://www.metacritic.com/game/playstation-4/marvels-spider-man.
24 Wood A. Red Dead Redemption 2 sells $725 million in 3 days, the best opening weekend in entertainment history. – 2018 // Электронная версия на сайте https://www.gamesradar.com/red-dead-redemption-2-sells-dollar725-million-in-3-days-the-best-opening-weekend-in-entertainment-history/.
25 RED DEAD REDEMPTION 2. – 2018 // Электронная версия на сайте https://www.metacritic.com/game/playstation-4/red-dead-redemption-2.
26 GAME OF THE YEAR. – 2021 // Электронная версия на сайте https://thegameawards.com/nominees/game-of-the-year.
27 Rollings A., Adams E. Andrew Rollings and Ernest Adams on Game Design: New Riders Publishing, 2003. – 648 с.
28 Hi-Res Adventure #1: Mystery House // Электронная версия на сайте http://sierrachest.com/index.php?a=games&id=194&title=mystery-house&fld=general.
29 Печенкин П. Название жанра «квест» придумала журналистка Game.EXE. Или нет?. – 2021 // Электронная версия на сайте https://cyber.sports.ru/tribuna/blogs/gamesherald/2928575.html.
30 GRIM FANDANGO // Электронная версия на сайте https://www.metacritic.com/game/pc/grim-fandango.

31 Craddok D. L. Procedural Dungeons of Doom: The Making of Rogue – Chapter 1. – 2016 // Электронная версия на сайте https://episodiccontentmag.com/2016/06/03/rogue_chapter1/.
32 Spike TV Video Game Awards 2012: All the winners, the trailers and the news // Электронная версия на сайте https://www.polygon.com/2012/12/7/3741630/spike-video-game-awards-2012-trailers.
33 Bates B. Game Design: Thomson Course Technology, 2004. – 350 с.
34 Novak J. Game Development Essentials An Introduction, Third Edition: Cengage Learning, 2012. – 510 с.
35 Bethke E. Game development and production: Wordware Publishing, Inc., 2003. – 432 с.
36 The $120B Gaming Industry Is Being Built On The Backs Of These Two Engines // Электронная версия на сайте https://www.cbinsights.com/research/game-engines-growth-expert-intelligence/.
37 Unity vs. Unreal: Which Game Engine is Best For You? // Электронная версия на сайте https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you/#:~:text=The%20main%20difference%20between%20Unity,Unity%20editor%20and%20any%20plugins..
38 What platforms does UE4 support? // Электронная версия на сайте https://www.unrealengine.com/en-US/faq.
39 Unreal Engine End User License Agreement // Электронная версия на сайте https://www.unrealengine.com/en-US/eula/unreal.
40 It’s how you make software // Электронная версия на сайте https://visualstudio.microsoft.com/.
41 TIOBE Index for May 2022 // Электронная версия на сайте https://www.tiobe.com/tiobe-index/.
42 Totilo S. How Unreal Engine 4 Will Change The Next Games You Play. - 2012 // Электронная версия на сайте https://kotaku.com/how-unreal-engine-4-will-change-the-next-games-you-play-5916859.

Приложение А

Техническое задание

А.1 Техническое задание на разработку видеоигры в жанре Adventure на платформе Unreal Engine

Настоящее техническое задание распространяется на разработку Adventure (приключенческая игра) видеоигры “Похищение” в жанре визуального романа, на игровом движке Unreal Engine для персонального компьютера.

А.1.1 Основание для разработки

Программа разрабатывается на основании актуальности и высокой значимости создания видеоигр вследствие того, что индустрия игр — это большой и быстрорастущий рынок, приносящий выручки значительней, чем кинематограф и музыка вместе взятые. Ключевыми факторами выбора являются популярность жанра adventure и востребованность одиночных игр.

Проект будет разрабатываться на основании графического романа. Важной отличительной чертой является то, что, в книге читатель во время выбора сюжетного действия случайно может преждевременно узнать важную информацию о сюжете, увидеть несколько концовок (на одной странице показаны 2–5 возможных развитий сюжета). То игра в свою очередь, полностью исключит случайные спойлеры, что предаст ей ценность в глазах игрока (читателя).

А.1.2 Назначение

Разрабатываемая игра предназначена для привлечения внимания к важности развития индустрии видеоигр в Казахстане.

А.1.3 Требования к функциональным характеристикам

Так как данный проект представляет собой реализацию видеоигры, то предполагается только одна категория пользователя - игрок. В процессе работы

программы игрок непосредственно является участником игрового процесса и оказывает на него влияние.

Программа должна обеспечивать возможность выполнения следующих функций:

a) пользовательский интерфейс
– переходные сцены: заставка, пролог, финальная сцена
– главное меню и меню паузы игры
– графический интерфейс с возможностью редактирования характеристик главного персонажа: здоровье, уменьшающееся при столкновениях с противниками (в зависимости от исхода боя) и увеличивающееся благодаря найденным предметам; сила, ловкость, воля, учитывающиеся при стычках героя с противниками; время, значение которого, в текущий момент игры влияет на ход прохождения; а также осуществление использования внутриигрового инвентаря с возможностью хранения до трех предметов одновременно

b) внутриигровой функционал
– игрок будет иметь возможность поиска предметов
– перед началом прохождения игра предоставит пользователю правил игры
– больше 350 игровых локаций, сцен в виде спрайтов, по которым игрок сможет свободно перемещаться по нажатию кнопок стрелок или наводясь курсором мыши и щелкая по определенным областям на экране
– возможность исследования локаций и нахождение скрытых комнат
– возможность общения с другими персонажами
– возможность изучения улик и разгадывания шифров
– участие в боях и погонях
– реализация нелинейного сюжета со множеством концовок

А.1.4 Требования к надежности

При воспроизведении любой сюжетной концовки предусмотреть невозможность продолжения игры и вывод на экран кнопки возвращающую игрока в главное меню.

А.1.5 Требования к системной совместимости

– Операционная система: Windows 7 и выше;

Продолжение приложения А

- Процессор: Intel или AMD, 1.5 GHz и быстрее;
- Оперативная память: 2 GB RAM;
- Место на диске: 3 Gb
- Видеокарта: совместимая с DirectX 9.0c

Приложение Б

Текст программы

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "DiplomaGameMode.generated.h"

/**
 * The GameMode defines the game being played. It governs the game rules, scoring, what actors
 * are allowed to exist in this game type, and who may enter the game.
 *
 * This game mode just sets the default pawn to be the MyCharacter asset, which is a subclass of
DiplomaCharacter
 */
UCLASS(minimalapi)
class ADiplomaGameMode : public AGameModeBase
{
	GENERATED_BODY()
public:
	ADiplomaGameMode();
};


#include "MenuWidget.h"
#include "Components/Button.h"
void UMenuWidget::NativeConstruct() {
	if (StartButton) {
		StartButton->OnClicked.AddDynamic(this,
&UMenuWidget::StartButtonOnClicked);
	}
	if (ExitButton) {
		ExitButton->OnClicked.AddDynamic(this,
&UMenuWidget::ExitButtonOnClicked);
	}
}

#pragma once

#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать
UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "MenuWidget.generated.h"
```

Продолжение приложения Б

```
UCLASS()
class DIPLOMA_API UMenuWidget : public UUserWidget
{
	GENERATED_BODY()
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
	class UButton* StartButton;//StartButton - переменная, представляющая из себя кнопку в игре
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* ExitButton;//ExitButton - переменная, представляющая из себя кнопку в игре

	virtual void NativeConstruct() override;

	UFUNCTION()
			void StartButtonOnClicked() {//Создаем функцию, которая перенесет игрока на уровень с игрой, при нажатии кнопки
			//В данном случае кнопка "StartButton"
			UGameplayStatics::OpenLevel(this, "PrologueLevel");//UGameplayStatics - класс содержащий функции для геймплея
			//благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
			//в параметр которого передадим имя уровня, который нам нужен

	};
	UFUNCTION()
		void ExitButtonOnClicked() {
		UKismetSystemLibrary::QuitGame(this, nullptr, EQuitPreference::Quit,false);

	};
};

#include "StartChapter1Widget.h"
#include "Components/Button.h"
void UStartChapter1Widget::NativeConstruct() {
	if (Chapter1Button) {
		Chapter1Button->OnClicked.AddDynamic(this, &UStartChapter1Widget::StartChapter1OnClicked);
	}
}

#pragma once

#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать UGameplayStatics
```

Продолжение приложения Б

```
#include "Kismet/KismetSystemLibrary.h"
#include "StartChapter1Widget.generated.h"

UCLASS()
class DIPLOMA_API UStartChapter1Widget : public UUserWidget
{
	GENERATED_BODY()
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* Chapter1Button;//Chapter1Button - переменная, представляющая из себя кнопку в игре

	virtual void NativeConstruct() override;

	UFUNCTION()
		void StartChapter1OnClicked() {//Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
		//В данном случае кнопка "Chapter1Button"
		UGameplayStatics::OpenLevel(this, "Chapter1Level");//UGameplayStatics - класс содержащий функции для геймплея
		//благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
		//в параметр которого передадим имя уровня, который нам нужен
	};
};


#include "StartGameAfterRulesWidget.h"
#include "Components/Button.h"
void UStartGameAfterRulesWidget::NativeConstruct() {
	if (StartGameAfterRulesButton) {
		StartGameAfterRulesButton->OnClicked.AddDynamic(this, &UStartGameAfterRulesWidget::StartGameAfterRulesOnClicked);
	}
}

#pragma once

#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "StartGameAfterRulesWidget.generated.h"

UCLASS()
class DIPLOMA_API UStartGameAfterRulesWidget : public UUserWidget
{
```

Продолжение приложения Б

```
	GENERATED_BODY()
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* StartGameAfterRulesButton;//StartGameAfterRulesButton - переменная, представляющая из себя кнопку в игре

	virtual void NativeConstruct() override;

	UFUNCTION()
		void StartGameAfterRulesOnClicked() {//Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
		//В данном случае кнопка "StartGameAfterRulesButton"
		//И уровень на который эта кнопка перенесет игрока тоже имеет в названии число(49).
		//Каждая кнопка, функция выполняющая действия кнопок и название уровня, несут в себе чилсло - определяющее номер сцены книги
		UGameplayStatics::OpenLevel(this, "Chapter0Level");//UGameplayStatics - класс содержащий функции для геймплея
		//благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
		//в параметр которого передадим имя уровня, который нам нужен

	};
};


#include "DeathBySpiderWidget.h"
#include "Components/Button.h"
void UDeathBySpiderWidget::NativeConstruct() {
	if (EndAfterDeathBySpiderButton) {
		EndAfterDeathBySpiderButton->OnClicked.AddDynamic(this, &UDeathBySpiderWidget::EndAfterDeathBySpiderButtonOnClicked);
	}
}

#pragma once
#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "DeathBySpaceWidget.generated.h"

UCLASS()
class DIPLOMA_API UDeathBySpaceWidget : public UUserWidget
{
	GENERATED_BODY()
protected:
```

Продолжение приложения Б

```
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* EndAfterDeathBySpaceButton;//EndAfterDeathBySpaceButton - переменная, представляющая из себя кнопку в игре

	virtual void NativeConstruct() override;

	UFUNCTION()
		void EndAfterDeathBySpaceButtonOnClicked() {//Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
		//В данном случае кнопка "EndAfterDeathBySpaceButton"
		UGameplayStatics::OpenLevel(this, "MenuLevel");//UGameplayStatics - класс содержащий функции для геймплея
		//благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
		//в параметр которого передадим имя уровня, который нам нужен

	};
};


#include "DeathByDogWidget.h"
#include "Components/Button.h"
void UDeathByDogWidget::NativeConstruct() {
	if (EndAfterDeathByDogButton) {
		EndAfterDeathByDogButton->OnClicked.AddDynamic(this, &UDeathByDogWidget::EndAfterDeathByDogButtonOnClicked);
	}
}

#pragma once

#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "DeathByDogWidget.generated.h"

UCLASS()
class DIPLOMA_API UDeathByDogWidget : public UUserWidget
{
	GENERATED_BODY()
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* EndAfterDeathByDogButton;//EndAfterDeathByDogButton - переменная, представляющая из себя кнопку в игре

	virtual void NativeConstruct() override;
```

Продолжение приложения Б

```
	UFUNCTION()
		void EndAfterDeathByDogButtonOnClicked() {//Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
		//В данном случае кнопка "EndAfterDeathByDogButton"
		UGameplayStatics::OpenLevel(this, "MenuLevel");//UGameplayStatics - класс содержащий функции для геймплея
		//благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
		//в параметр которого передадим имя уровня, который нам нужен

	};
};


#pragma once
#include "CoreMinimal.h"
#include "Blueprint/UserWidget.h"
#include "Kismet/GameplayStatics.h"//из данного заголовочного файла, будем использовать UGameplayStatics
#include "Kismet/KismetSystemLibrary.h"
#include "DeathByBulletsWidget.generated.h"
UCLASS()
class DIPLOMA_API UDeathByBulletsWidget : public UUserWidget
{
	GENERATED_BODY()
protected:
	UPROPERTY(BlueprintReadWrite, meta = (BindWidget))
		class UButton* EndAfterDeathByBulletButton;//EndAfterDeathByBulletButton - переменная, представляющая из себя кнопку в игре
	virtual void NativeConstruct() override;
	UFUNCTION()
		void EndAfterDeathByBulletButtonOnClicked() {//Создаем функцию, которая перенесет игрока на другой уровень, при нажатии кнопки
		//В данном случае кнопка "EndAfterDeathByBulletButton"
		UGameplayStatics::OpenLevel(this, "MenuLevel");//UGameplayStatics - класс содержащий функции для геймплея //благодаря которому будем изпользовать метод OpenLevel, служащий для перемещения на другой уровень.
//в параметр которого передадим имя уровня, который нам нужен
	};
};
#include "DeathByBulletsWidget.h"
#include "Components/Button.h"
void UDeathByBulletWidget::NativeConstruct() {
	if (EndAfterDeathByBulletButton) {
		EndAfterDeathByBulletButton->OnClicked.AddDynamic(this, &UDeathByBulletWidget::EndAfterDeathByBulletButtonOnClicked);
	}}
```