**1 АНАЛИТИЧЕСКИЙ ОБЗОР СФЕРЫ РАЗРАБОТКИ ИГРОВЫХ ПРИЛОЖЕНИЙ**

**1.1 Описание предметной области**

По степени влияния на потребителей и вовлеченности их в интерактивное окружение, предлагаемое видеоиграми, этот сегмент уже давно выделяется среди других видов развлечений.

Разработку игр невозможно рассматривать обособленно от индустрии компьютерных игр в целом. Непосредственно создание игр – это только часть комплексной «экосистемы», обеспечивающей полный жизненный цикл производства, распространения и потребления таких сложных продуктов, как компьютерные игры.

В структуре современной игровой индустрии можно выделить следующие уровни:

– платформы;

– игровые движки;

– разработка видеоигр;

– издание и оперирование;

– популяризация и потребление.

Платформы – аппаратно-программные системы, позволяющие запускать интерактивные игровые приложения. Среди основных видов можно выделить следующие:

– персональные компьютеры на базе *Windows*, *Mac/OS X* или *Linux*;

– игровые консоли (специализированные устройства для игр, *Xbox One*, *PlayStation 4*, *Nintendo Wii U*;

– мобильные устройства (*iOS*, *Android*, *Windows*).

Игровые движки – программная прослойка между платформой и собственно кодом игры. Использование готового игрового движка позволяет существенно упростить разработку новых игр, удешевить их производство и существенно сократить время до запуска. Кроме того, современные игровые движки обеспечивают кроссплатформенность создаваемых продуктов. Из наиболее продвинутых движков можно выделить: *Unity* *3D*, *Unreal* *Development Kit*, *CryENGINE* *3 Free SDK*.

Разработка игр. Большое количество компаний и независимых команд занимаются созданием компьютерных игр. В разработке участвует специалисты разных профессий: программисты, гейм-дизайнеры, художники, *QA* специалисты и др. К разработке крупных коммерческих игровых продуктов привлекаются большие профессиональные команды. Стоимость разработки подобных проектов может составлять десятки миллионов долларов. Однако вполне успешные игровые проекты могут воплощаться и небольшими командами энтузиастов. Этому способствует присутствие на рынке большого количества открытых и распространенных платформ, качественных и практически бесплатных движков, площадок по привлечению «народных» инвестиций (краудфандинг) и доступных каналов распространения.

Издание и оперирование игр. Распространением игр или оперированием (в случае с *MMO*) занимаются, как правило, не сами разработчики, а издатели. При этом издатели (или операторы) локализуют игры, взаимодействуют с владельцами платформ, проводят маркетинговые компании, разворачивают инфраструктуру, обеспечивают техническую и информационную поддержку выпускаемым играм. Для средних и небольших игровых продуктов данный уровень практически не доступен. Такие продукты, как правило, сами разработчики выводят на рынок, напрямую взаимодействуя с платформами.

Популяризация. Специализированные средства массовой информации всегда являлись мощным каналом донесения информации до пользователей. Сейчас наиболее эффективным и широко представленным направлением СМИ являются информационные сайты, посвященные игровой тематике. Игровые журналы, долгое время выступавшие главным источником информации об играх, в настоящее время уступили свое место интернет-ресурсам. Специализированные выставки все еще остаются важным информационными площадками для игровой индустрии (*E3*, *GDC*, *Gamescom*, *ИгроМир*, *КРИ*, *DevGamm*). Прямое общение прессы и игроков с разработчиками, обмен опытом между участниками рынка, новые контакты – вот то, что предлагают конференции и выставки в концентрированной форме. Еще один важный канал донесения полезной информации до игроков – это ТВ-передачи, идущие как в формате классического телевещания, так и на множестве каналов видео-контента.

Игроки – это основной источник прибыли для игровых продуктов. Но в современном мире наиболее активные игроки стали существенной движущей силой в популяризации игр и отчасти в расширении контента.

**1.2 Жанр аркадного фермерского симулятора**

Жанр игр про ферму включает в себя уникальные особенности и подходы к разработке. Создание игры в этом жанре требует не только технических навыков, но и понимания аспектов, характерных для фермерской тематики. Некоторые ключевые этапы разработки фермерской игры включают в себя концепцию, дизайн, программирование, тестирование и выпуск игры.

Разработчики фермерских игр должны учитывать разнообразные аспекты, включая управление фермой, выращивание культур, разведение скота, улучшение оборудования и многое другое. Важно создать баланс между реализмом и увлекательным игровым процессом, чтобы игроки могли насладиться аутентичным опытом фермерской деятельности.

Фермерские игры обычно имеют вид сверху вниз и позволяют игрокам управлять всеми аспектами фермерского хозяйства. Игроки могут посеять и ухаживать за растениями, строить и улучшать фермерские постройки, обрабатывать поля, управлять скотом и многое другое. Цель игры может варьироваться от создания прибыльного фермерского хозяйства до выживания в форс-мажорных обстоятельствах.

Разработка игр в жанре фермерских симуляторов требует специфических навыков и инструментов. Этот жанр игр подразумевает создание виртуальных ферм, где игроки могут управлять аспектами фермерского хозяйства, от посевов до ухода за животными.

Создание фермерской игры начинается с концепции, где разработчики определяют основные механики и особенности игрового процесса. Затем идет проектирование элементов игры, включая графику, интерфейс, уровни сложности и систему управления. Программирование игры включает в себя создание логики игровых механик, взаимодействия объектов на ферме и реализацию анимаций.

Для разработки фермерских игр часто используются специализированные игровые движки, такие как *Unity* или *Godot*. Эти инструменты предоставляют разработчикам возможность создавать детальные *2D* и *3D* модели ферм, животных, растений, а также реализовывать сложные игровые системы, например, учет погодных условий или сезонности.

Фермерские симуляторы предлагают игрокам широкий спектр деятельности, включая посев и уход за культурами, разведение и уход за животными, строительство и расширение фермерских построек, покупку нового оборудования и техники. Важным аспектом игры является реализм фермерской деятельности, отраженный в визуальных и звуковых эффектах, а также в поведении растений и животных.

Игры в жанре фермерских симуляторов часто представлены в виде сверху вниз, что позволяет игрокам получить обзор на всю ферму и ее окрестности. Важным аспектом таких игр является создание уникальной атмосферы и стиля фермы, чтобы игроки могли погрузиться в виртуальный мир сельского хозяйства.

Фермерские игры предлагают игрокам разнообразие сценариев и возможностей развития. Игроки могут выбирать, какие культуры выращивать, какими методами ухаживать за животными, как распределять ресурсы и многое другое. Это позволяет каждой игре быть уникальной и дать разработчикам возможность создать идеальную ферму по своему вкусу.

**1.3 Аналитический обзор представителей игр жанра аркадного фермерского симулятора**

В этом подразделе представлен аналитический обзор наиболее значимых и популярных представителей жанра аркадного фермерского симулятора. Цель этого обзора – выявить ключевые черты и механики, которые способствовали успеху данных игр, а также понять, какие аспекты их дизайна и игрового процесса привлекают игроков. Рассмотрение успешных примеров позволит выделить лучшие практики и тенденции в жанре, что станет основой для дальнейшего развития собственной игры.

Серия игр *Farming Simulator* представляет собой значительное явление в жанре аркадных фермерских симуляторов. Начав своё существование в 2008 году, эта серия продолжает выпускаться и по сей день, демонстрируя высокий уровень популярности среди игроков. Основная концепция игр этой серии заключается в симуляции сельскохозяйственного труда, где игроку предоставляется возможность заниматься разнообразными видами деятельности, включая земледелие, животноводство, лесоводство и другие аспекты фермерской жизни.

*Farming Simulator* выделяется своей реалистичностью и широким спектром возможностей. Игроки могут выбирать разные направления в сельском хозяйстве, будь то выращивание сельскохозяйственных культур или разведение животных, а также приобретать и использовать разнообразное оборудование. Игра предлагает детализированную симуляцию, в которой каждое действие требует тщательного планирования и выполнения.

Обучение и интерфейс. Для новичков в игре предусмотрены обучающие задания, которые помогают освоить основные механики игры и понять, как эффективно управлять фермой. Ветераны серии, в свою очередь, могут использовать свой накопленный опыт для оптимизации процессов и максимизации урожайности.

Разнообразие деятельности. Игрокам предоставляется широкий спектр занятий, начиная от вспашки и сева полей и заканчивая уходом за животными и лесозаготовками. В игре представлено множество культур для выращивания, таких как пшеница, ячмень, кукуруза, картофель и другие. Каждая культура требует особого подхода к уходу и удобрению.

Экономические аспекты. Помимо земледелия, игроки также могут заниматься торговлей, покупая и продавая оборудование, а также продукцию своей фермы. Важным элементом игрового процесса является логистика, включающая перевозку урожая и управление запасами. В последних версиях игры были добавлены поезда, которые значительно облегчают транспортировку, но требуют дополнительных затрат и навыков управления.

Лесное хозяйство. В версии *Farming Simulator 15* была введена возможность заниматься лесоводством, что добавляет ещё один интересный элемент в игровой процесс. Этот аспект игры также требует специфических знаний и навыков для успешного ведения бизнеса.

Игры серии *Farming Simulator* учат игроков основам фермерства, предлагая погружение в процесс и понимание всех тонкостей сельскохозяйственного труда. Эти игры не только развлекают, но и предоставляют возможность получить полезные навыки, которые могут быть применимы и в реальной жизни.

Таким образом, серия *Farming Simulator* является не просто развлекательным продуктом, но и обучающим инструментом, который помогает игрокам лучше понять и оценить труд фермеров, предоставляя глубокий и увлекательный игровой опыт.

На рисунке 1.1 представлен кадр из *Farming Simulator*.



Рисунок 1.1 Кадр из *Farming Simulator*

Игра *Stardew Valley*, созданная и разработанная всего одним человеком, Эриком Бароном, является значимым представителем жанра аркадных фермерских симуляторов. Вышедшая в 2016 году, она быстро завоевала популярность благодаря своему уникальному сочетанию различных игровых механик и богатого контента. Игра сочетает элементы фермерства, ролевых игр и симуляторов жизни, создавая уникальный и увлекательный игровой опыт.

Сюжет и персонажи. В основе сюжета лежит история персонажа, который получает в наследство ферму от своего дедушки. Игрок может настроить внешность и пол персонажа, однако эти параметры не влияют на игровой процесс. Главный герой свободен в своих действиях: можно заниматься фермерством, исследовать пещеры, общаться с местными жителями или выполнять квесты.

Игровая свобода. *Stardew Valley* предоставляет игрокам значительную свободу выбора. Игроки могут решать, каким образом управлять фермой, какие культуры выращивать, как ухаживать за животными и какие задания выполнять. В игре присутствуют элементы дэйтинг-симулятора, позволяющие строить отношения с другими персонажами и даже вступать в брак.

На рисунке 1.2 представлен кадр из игры *Stardew Valley*.



Рисунок 1.2 – Кадр из игры *Stardew Valley*

Ролевая система и развитие навыков. В отличие от многих других фермерских симуляторов, *Stardew Valley* включает ролевую систему, в которой игроки могут прокачивать навыки своего персонажа. Каждый вид деятельности, будь то использование мотыг или лейки, приносит очки опыта, которые улучшают соответствующие навыки и открывают новые возможности.

Сезонность и планирование. Игра симулирует смену времён года, что влияет на доступность различных культур и видов деятельности. Летом можно выращивать определенные растения и собирать мед, а зимой заниматься консервацией продуктов. Важно учитывать прогноз погоды и планировать свои действия, чтобы максимально эффективно использовать ресурсы и время.

Стратегическое планирование. Успешное ведение фермы требует тщательного планирования. Игрокам нужно учитывать бюджет, выбирать и подготавливать участки для посадки, а также защищать урожай от вредителей с помощью пугал. В игре существует ограничение по времени и энергии персонажа, что добавляет элемент стратегического планирования в повседневные задачи.

Многопользовательский режим. Игра поддерживается и развивается своим разработчиком, и в последние годы была добавлена возможность многопользовательской игры.

*Stardew Valley* предлагает глубокий и захватывающий игровой опыт, сочетая элементы фермерства, ролевой игры и симулятора жизни. Игра позволяет игрокам чувствовать себя настоящими профессиональными садоводами, наслаждаясь процессом и достигнутыми результатами. Благодаря своему уникальному подходу к жанру и постоянной поддержке со стороны разработчика, *Stardew Valley* продолжает оставаться одной из самых популярных и любимых игр в жанре аркадных фермерских симуляторов.

**1.4 Игровой движок *Unity***

*Unity* – это профессиональный игровой движок, позволяющий создавать видеоигры для различных платформ.

Любой игровой движок предоставляет множество функциональных возможностей, которые задействуются в различных играх. Реализованная на конкретном движке игра получает все функциональные возможности, к которым добавляются ее собственные игровые ресурсы и код игрового сценария.

Приложение *Unity* предлагает моделирование физических сред, карты нормалей, преграждение окружающего света в экранном пространстве (*Screen* Space *Ambient* *Occlusion*, *SSAO*), динамические тени. Список можно продолжать долго. Подобные наборы функциональных возможностей есть во многих игровых движках, но *Unity* обладает двумя основными преимуществами над другими передовыми инструментами разработки игр. Это крайне производительный визуальный рабочий процесс и сильная межплатформенная поддержка. Визуальный рабочий процесс – достаточно уникальная вещь, выделяющая *Unity* из большинства сред разработки игр. Альтернативные инструменты разработки зачастую представляют собой набор разрозненных фрагментов, требующих контроля, а в некоторых случаях библиотеки, для работы с которыми нужно настраивать собственную интегрированную среду разработки (*Integrated Development Environment, IDE*), цепочку сборки и прочее в этом роде. В *Unity* же рабочий процесс привязан к тщательно продуманному визуальному редактору. Именно в нем будут компоновать сцены будущей игры, связывая игровые ресурсы и код в интерактивные объекты. *Unity* позволяет быстро и рационально создавать профессиональные игры, обеспечивая невиданную продуктивность разработчиков и предоставляя в их распоряжение исчерпывающий список самых современных технологий в области видеоигр.

Редактор особенно удобен для процессов с последовательным улучшением, например, циклов создания прототипов или тестирования. Даже после запуска игры остается возможность модифицировать в нем объекты и двигать элементы сцены. Настраивать можно и сам редактор. Для этого применяются сценарии, добавляющие к интерфейсу новые функциональные особенности и элементы меню.

На рисунке 1.3 представлен интерфейс игрового движка Unity.



Рисунок 1.3 – интерфейс движка *Unity*

Дополнением к производительности, которую обеспечивает редактор, служит сильная межплатформенная поддержка набора инструментов *Unity*. В данном случае это словосочетание подразумевает не только места развертывания (игру можно развернуть на персональном компьютере, в интернете, на мобильном устройстве или на консоли), но и инструменты разработки (игры создаются на машинах, работающих под управлением как *Windows*, так и *Mac OS*). Эта независимость от платформы явилась результатом того, что изначально приложение *Unity* предназначалось исключительно для компьютеров *Mac*, а позднее было перенесено на машины с операционными системами семейства *Windows*. Первая версия появилась в 2005 году, а к настоящему моменту вышли уже пять основных версий (с множеством небольших, но частых обновлений). Изначально разработка и развертка поддерживались только для машин *Mac*, но через несколько месяцев вышло обновление, позволяющее работать и на машинах с Windows. В следующих версиях добавлялись все новые платформы развертывания, например межплатформенный веб-плеер в 2006-м, *iPhone* в 2008- м, *Android* в 2010-м и даже такие игровые консоли, как *Xbox* и *PlayStation*. Позднее появилась возможность развертки в *WebGL* – новом фреймворке для трехмерной графики в веб-браузерах. Немногие игровые движки поддерживают такое количество целевых платформ развертывания, и ни в одном из них развертка на разных платформах не осуществляется настолько просто. Дополнением к этим основным достоинствам идет и третье, менее бросающееся в глаза преимущество в виде модульной системы компонентов, которая используется для конструирования игровых объектов. «Компоненты» в такой системе представляют собой комбинируемые пакеты функциональных элементов, поэтому объекты создаются как наборы компонентов, а не как жесткая иерархия классов. В результате получается альтернативный (и обычно более гибкий) подход к объектно-ориентированному программированию, в котором игровые объекты создаются путем объединения, а не наследования.

Оба подхода схематично показаны на рисунке 2.1.

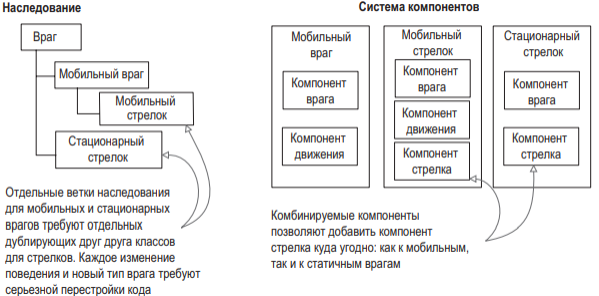


Рисунок 1.4 – Сравнение наследования с компонентной системой

Каждое изменение поведения и новый тип врага требуют серьезной перестройки кода. Комбинируемые компоненты позволяют добавить компонент стрелка куда угодно: как к мобильным, так и к статичным врагам. В компонентной системе объект существует в горизонтальной иерархии, поэтому объекты состоят из наборов компонентов, а не из иерархической структуры с наследованием, в которой разные объекты оказываются на разных ветках дерева. Разумеется, ничто не мешает написать код, реализующий вашу собственную компонентную систему, но в *Unity* уже существует вполне надежный вариант такой системы, органично встроенный в визуальный редактор. Эта система дает возможность не только управлять компонентами программным образом, но и соединять и разрывать связи между ними в редакторе. Разумеется, возможности не ограничиваются составлением объектов из готовых деталей; в своем коде вы можете воспользоваться наследованием и всеми наработанными на его базе шаблонами проектирования [5, c.17].

**1.5 Игровой движок *Unreal Engine***

*Unreal* *Engine* (*UE*) – игровой движок, разрабатываемый и поддерживаемый компанией *Epic Games*.

Движком называют рабочую среду, позволяющую управлять всей системой элементов, из которых состоит игра.

Сегодня движок *Unreal* *Engine* активно применяется для разработки простых казуальных игр для смартфонов и планшетов, а также для создания полноценных высокобюджетных игр, рассчитанных на массовую аудиторию (их называют *ААА*-проектами). При этом не потребуется самостоятельно писать код, т. к. система визуального создания скриптов *Blueprints* *Visual Scripting* значительно упрощает задачу. Если же разработчик желает прописать игровую логику вручную, он может использовать язык программирования *С++*.

5 апреля 2022 года *Epic* *Games* порадовала пользователей, представив обновленный движок *Unreal Engine 5*, анонсированный два года назад. Среди главных фишек – максимум фотореализма, увеличенная производительность и новый интерфейс.

*Unreal* *Engin* остается популярным более 20 лет, т. к. обладает следующими достоинствами:

– широкий функционал;

– визуальное программирование;

– бесплатная лицензия;

– возможность создать кросс-платформер;

– большая база пользователей.

*Epic Games* решила дать разработчикам больше, чем простой инструмент – в *UE* пользователи могут начать работу даже без узкоспециализированных знаний в области языков программирования.

На рисунке 1.4 представлен интерфейс *Unreal Engine*.

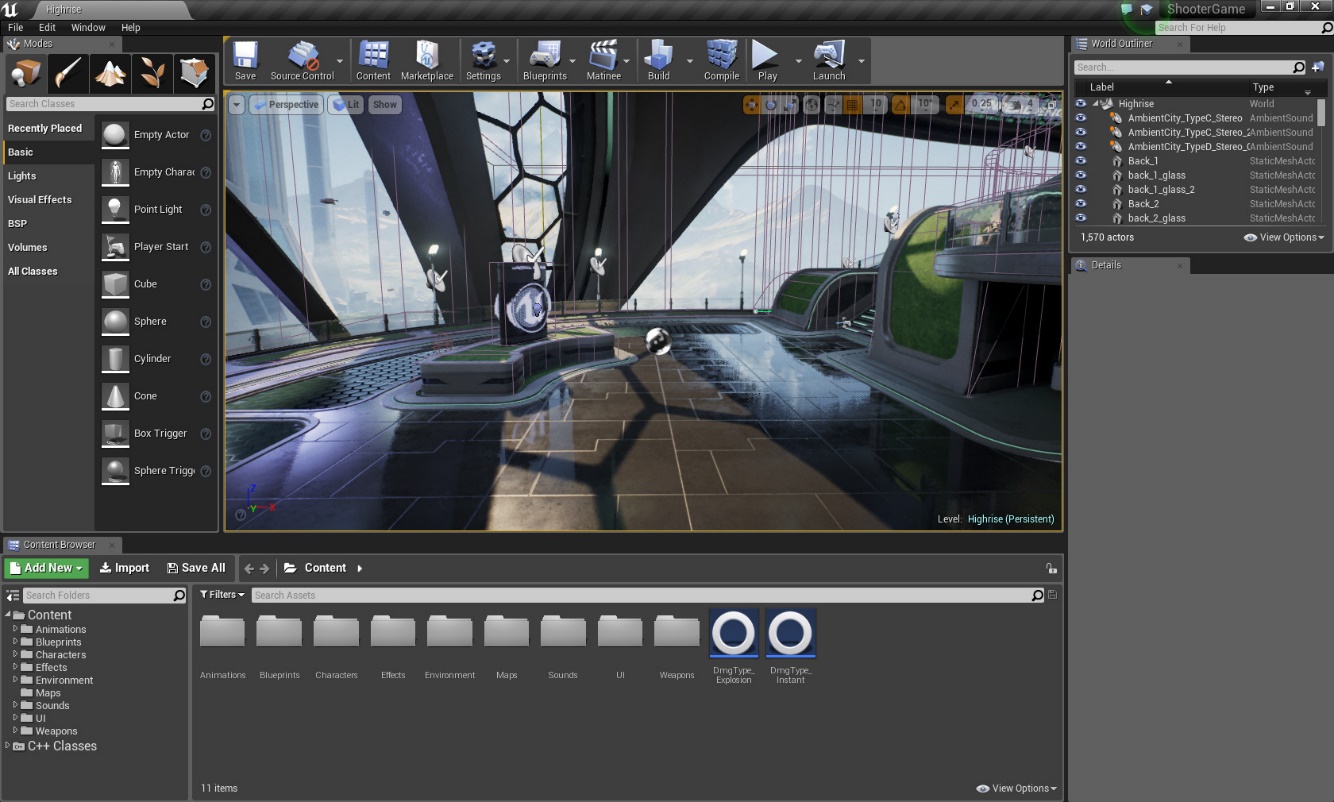


Рисунок 1.4 – интерфейс *Unreal Engine*

Для тех, кто далек от *кодинга*, корпорация предложила простую и удобную в использовании систему *Blueprints* *Visual* *Scripting*. С ее помощью можно легко создать прототип любой игры, имея минимум теоретических знаний. Конечно, умение работать с функциональным и объектно-ориентированным программированием будет плюсом, но начать разработку геймплея в *UE* можно и без него.

*Blueprints* значительно проще для понимания и использования, чем *С++*, при этом их функции и возможности в большинстве случаев схожи. Однако иногда все же придется прибегнуть к кодингу: для произведения сложных математических расчетов, изменения исходного кода самого движка *UE* и ряда базовых классов проекта.

В игровом мире существуют объекты с уникальными оттенками, фактурами и физическими свойствами. В движке *UE* внешний вид зависит от настроек материалов. Цвет, прозрачность, блеск – задать можно практически любые параметры. При работе над игрой в *UE* материалы можно наносить на любые объекты, вплоть до мелких частиц. Отметим, что речь идет не просто о настройке текстур: материалы открывают более широкие возможности. К примеру, можно создавать необычные визуальные эффекты, причем *UE* позволяет делать это прямо в процессе игры.

Пользовательский интерфейс. Игроку важно не только видеть действия своего персонажа и карту, на которой он находится, но и иметь текстовую информацию, а также сведения о количестве очков, пунктах здоровья, инвентаре и т. д. С этой целью разработчики тщательно продумывают пользовательский интерфейс (*User Interface, UI*). В движке *Unreal* для создания *UI* применяется *Unreal* *Motion Graphics* (*UMG*). Он позволяет выстраивать интуитивно понятный *UI*, выводить на экран необходимую пользователю информацию, а также менять положение кнопок и текстовых меток.

Анимация. Персонаж любой современной игры подвижен и гибок, умеет бегать и прыгать. Все это возможно благодаря анимированию. В *UE* начинающие разработчики могут импортировать уже готовые мэши со скелетами персонажей и настройки анимации. Неопытных пользователей, которые желают познакомиться с ПО поближе, приятно удивит *Animation Blueprint* – скрипт, который значительно упрощает работу по созданию паттернов движений персонажа без использования кодинга.

Звук. Для полного погружения в игру недостаточно просто собрать саундтрек из десятка файлов – музыку следует подобрать по тематикам сцен, настроить уровень ее громкости, прописать и расставить по нужным местам диалоги персонажей. В *UE* можно по-разному настраивать звуковые эффекты, зацикливать музыку и модулировать тон при каждом новом воспроизведении, а также работать с несколькими эффектами одновременно. За последнее отвечает ассет *Sound Cue*.

Система частиц. Данный компонент необходим для создания визуальных эффектов. Взрывы, брызги, искры, туман, снегопад или дождь – в *UE* все это можно создать, используя систему *Cascade*.

Искусственный интеллект. В компьютерной игре существуют не только главные, но и второстепенные персонажи. Искусственный интеллект (ИИ) отвечает за их решения (увидеть действие и среагировать). Настроить ИИ в *UE* можно, используя так называемые деревья поведения, *Behavior* *Trees*. В простые схемы закладываются алгоритмы действий и принятия решений. Здесь не только новичкам, но и профессионалам будет удобнее работать в *Blueprints* *Visual Scripting*, ведь все деревья визуально напоминают простые блок-схемы. Выстроить их гораздо быстрее и проще, чем писать длинный код.

**1.6 Сравнение *Unity* и *Unreal Engine***

Первая область сравнения – редакторы для создания уровней, которые очень похожи. В них есть браузеры контента для ассетов, скриптов и других файлов проекта. Игровые объекты можно перетаскивать в область сцены и, таким образом, добавлять в её иерархию.

Объекты в редакторе сцены изменяются с помощью инструментов перемещения, поворота и масштабирования – они похожи в обоих движках. Свойства *Unity*-объектов отображаются в *Inspector*, а *UE4* – в части *Details*. *Jayanam* также сравнивает возможности *Unity* *Prefabs* c *Blueprints*.

В обоих движках есть статические меши *(static meshes*) – их можно двигать, поворачивать и масштабировать – и скелетные меши *(skeletal meshes*) – геометрические объекты, привязанные к костям скелета и используемые для анимирования персонажей. Их можно создавать в программах вроде *Blender* или *Maya*.

Анимации, включённые для скелетных мешей, также можно импортировать. В *Unity* они прикрепляются к импортированному объекту, как клипы анимации (*animation clips*), а в *UE4* называются последовательностями анимации (*animation sequences*). В первом движения управляются с помощью контроллеров анимации (*animation* *controllers*), а во втором – по тому же принципу действуют анимационные *Blueprints*.

В обоих движках есть стейт-машины, определяющие переходы из одного состояния ассета в другое.

В *UE4* система называется *Persona*, а в *Unity* – *Mecanim*. В них возможно применение скелетных мешей одного скелета к другим, но в *Unity* это в основном используется для анимирования гуманоидов. В *UE4* анимации можно редактировать, в *Unity* – практически нет, особенно плохо дело обстоит с движениями гуманоидов. Движки не подходят для профессионального анимирования персонажей – лучше использовать программы вроде *Blender* или *Maya*, а результат импортировать в виде *FBX*-файлов. Прикреплённый к объектам материал добавляется в проект, но его свойства вроде шейдера или текстур придётся применять вручную [7].

Для этого в *Unity* нужно задать материалу шейдер и добавить в его слоты текстуры – карты шероховатостей, нормалей или диффузии. Собственные шейдеры придётся писать самостоятельно или с помощью сторонних инструментов вроде *Shader* *Forge* или *ASE*. А в *UE4* встроен очень мощный редактор материалов, основанный, как и система *Blueprints*, на нодах.

Для программирования в *UE4* используется язык *C++*, который не все любят из-за сложности и продолжительности компилирования. Однако *Jayanam* считает, что у движка понятный *API* и приемлемый период компиляции. В *UE4* очень мощная и проработанная система визуального скриптования – *Blueprints*, с помощью которой можно достичь практически тех же результатов, что и *c C++*.

*Unity 5* поддерживает языки *C#* и *UnityScript*. *API* и его концепт очень похож на аналог из *UE4*. При использовании управляемого языка вроде C#, программист не обязан использовать указатели (*pointers*), компилирование происходит быстро. В *Unity* нет системы визуального скриптования, и чтобы использовать что-то подобное, разработчик вынужден покупать сторонние дополнения вроде *Playmaker*.

Для *2D*-разработки в *Unity* есть великолепные инструменты – *sprite creator*, *sprite editor* и *sprite packer*. *UE4* также поддерживает спрайты в *Paper 2d*, но решения из *Unity* мощнее, кроме того, в последнем есть отдельный физический движок для *2d*-объектов.

В *UE4* встроен *постпроцессинг*. К сцене можно применять *bloom*-эффект, тонирование и антиалиасинг как глобально, так и к отдельным её частям (при помощи компонента *PostProcessVolume*).

В *Unity* есть стек постпроцессинга, который можно скачать из магазина ассетов движка. Система менее гибкая, чем в *UE4* – эффекты применяются только стеком или скриптами к камере.

*Sequencer* в *UE4* можно использовать для создания синематиков. Это мощный инструмент, работающий по принципу добавления объектов на временную шкалу. К настоящему моменту в *Unity* 5.6 нет системы для синематиков, но *timeline*-редактор добавили в *Unity* 2017.

Для реализации приложения был выбрал игровой движок *Unity*, т.к. он обладает наибольшим инструментарием и информационной базой, которые в один момент являются удобными и простыми [6].

1.7 Требования к проектируемому программному обеспечению

Целью разработки является создание программного обеспечения (ПО) для аркадного фермерского симулятора, который предоставит пользователю возможность управлять виртуальной фермой, заниматься сельскохозяйственными работами, взаимодействовать с другими персонажами и развивать свои навыки в различных аспектах фермерской деятельности.

Программное обеспечение должно включать следующие функциональные возможности:

– cоздание нового игрового состояния: возможность выбрать название нового игрового состояния;

– загрузка выбранного игрового состояния: возможность выбора сохраненного игрового состояния;

– управление фермой: посадка и уход за различными культурами, управление ресурсами и инвентарем;

– социальное взаимодействие: общение с *NPC* (неигровыми персонажами);

– экономическая система: покупка и продажа продуктов, управление бюджетом фермы;

– атмосферное составляющее: смена дня и ночи;

Игровые механики включают:  
1. посадка и сбор урожая:

– механика вспашки земли;

– посадка семян различных культур;

– полив растений;

– сбор урожая по мере созревания;

2. механика загрузки и сохранения игрового состояния;

3. общение с *Npc*;

4. взаимодействие с активным, вспомогательным инвентарем, а также возможность хранения предметов в сундуке или сундуках;

5. экономика:

– продажа продукции;

– покупка семян;

6. смена дня и ночи;

7. возможности настройки игровой среды:

– смена разрешения экрана;

– смена полноэкранного режима на оконный и наоборот;

– громкости музыки и звуковых эффектов;

– смена управления;

8. сохранение и загрузка настроек игровой среды.

Графические и визуальные элементы включают следующие пункты:

1. стиль графики: 2.5*D* графика (создание 2*D*-спрайтов на основе 3*D* моделей с картами освещения и нормалей для достижения глубины и детализации, яркие насыщенные цвета для создания уютной и привлекательной атмосферы;

2. Персонажи: спрайты персонажей с анимацией движения;

3. Окружение: спрайты растительности, зданий и объектов;

4. Интерфейс пользователя: интуитивно понятный и легко читаемый интерфейс;

5. Эффекты и анимация:

– анимации различных действий персонажа(сбор урожая, вспашка, полив);

– визуальные эффекты для улучшения восприятия;

6. Звуковые эффекты и музыка:  
 – эффекты для действий;

– фоновая музыка, создающая атмосферу спокойствия и умиротворенности.

Входные данные:  
 – данные о игровом состоянии (игровая валюта);

– команды от пользователя (например, действия по посадке, сбору урожая).

Условия эксплуатации:

– программное обеспечение должно работать на платформах *Windows, macOS* и *Linux*.

– поддержка разрешений экрана от *1024x768* и выше;

– требования к минимальной производительности: 4 ГБ оперативной памяти, процессор с частотой 2 ГГц, видеокарта с поддержкой *OpenGL* 3.0;

– обеспечение возможности сохранения и загрузки игрового процесса.

Требования к надежности и быстродействию:

– программное обеспечение должно обеспечивать стабильную работу без сбоев и утечек памяти;

– быстродействие должно обеспечивать плавный игровой процесс с частотой не менее 30 кадров в секунду.

В качестве игрового движка был выбран *Unity*, так как он обеспечивает мощные инструменты для разработки 2*D* игр, поддержку множества платформ и имеет обширное сообщество разработчиков. Среда разработки *Visual Studio*, интегрированная в *Unity*, для удобства написания и отладки кода.

Инструментарий для разработки графики был выбран в пользу *Blender* для создания 3*D* моделей и *Adobe Photoshop* с *Aseprite* для обработки графических элементов.

Система контроля версий *Git* для отслеживания и сохранения изменений в процессе работы над проектом.

Таким образом, разработка программного обеспечения будет осуществляться с использованием проверенных и современных технологий, обеспечивающих надежность, быстродействие и удобство работы как для разработчиков, так и для конечных пользователей.