МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Проектирование и разработка видеоигры с использованием Unity

Обучающегося 4 курса
очной формы обучения
Выговского Евгения Игоревича

Руководитель выпускной квалификационной работы:
доктор педагогических наук, профессор
Абрамян Геннадий Владимирович

Рецензент:
Ученая степень *(при наличии)*, ученое звание *(при наличии)*, должность
Ф. И. О. *(указывается в именительном падеже)*

Санкт-Петербург
2022

Оглавление

ВВЕДЕНИЕ

9 сентября 2021 г. Правительство РФ утвердило план мероприятий («дорожную карту») по «созданию дополнительных условий для развития отрасли информационных технологий». Принятый пакет предусматривает меры по выравниванию условий ведения бизнеса в РФ для международных и отечественных IT-компаний.[1]

В марте 2022 г. специальным указом президента России были введены новые меры поддержки российской IT-отрасли. Цель указа – обеспечение ускоренного развития отрасли информационных технологий. В связи с чем IT-компаниям предоставляют ряд льгот. До конца 2024 г. разработчики ПО освобождены от уплаты налога на прибыль, а также от налогового, валютного и др. видов контроля. Важное направление господдержки IT-отрасли – выделение грантов на разработку и внедрение новых цифровых товаров и сервисов.[2]

Многие жители России и других стран тратят значительную часть свободного времени на видеоигры. К концу 2021 г. количество геймеров в мире превысило 3 млрд. Объем мирового рынка игровой индустрии оценивается в больше чем 170 млрд. долларов, а объем российского рынка по итогам 2021 года достиг 177,4 млрд. рублей. Россия занимает четвертое место по объему потребления игрового контента и уступает только США, Южной Корее и Китаю.

Большинство разработчиков игрового программного обеспечения имеют формальное образование, по крайней мере, степень бакалавра. Работодатели, как правило, отдают предпочтение кандидатам, прошедшим курсы повышения квалификации, поскольку они имеют более глубокую научную подготовку, необходимую для разработки игр.

Хотя обучение очное обучение в вузе является наиболее популярным способом получения образования программиста видеоигр, существует

возможность получить образование программиста игр, пройдя аккредитованную онлайн-программу. Популярными специализациями для разработчиков игр являются информатика и компьютерная инженерия. Курсы могут охватывать C#, C++, Perl, компьютерную 3D-графику, исчисление, игровые алгоритмы, объектно-ориентированный дизайн и основы сетей. Некоторые курсы также включают обучение работе с Unity, Unreal Engine и другими движками для видеоигр. Такие курсы также помогают развить у обучающегося критическое мышление и навыки решения проблем, а также способность проводить анализ качества и операций. Программы.

У 40% сотрудников игровых компаний (а это разработчики, гейм-дизайнеры, тестировщики и др.) образование никак не связано с их нынешней деятельностью. Наибольший процент сотрудников имеющих соответствующее образование имеют программисты (41%).[18]

Чтобы стать разработчиком игр, необходимо иметь хорошие навыки кодирования, для чего необходимы базовые знания математики и физики. Требуются математические навыки, чтобы создавать уравнения, которые заставят компьютерные системы работать определенным образом. Например, для программирования анимации необходимо знать исчисление, тригонометрию и линейную алгебру. Кроме того, знание физики может помочь понять такие понятия, как масса, инерция и термодинамика. Разработчику видеоигр требуется хорошо разбираться в аппаратных технологиях и C, C#, C++, Java, Python, Lua и других языках программирования, чтобы писать код, в котором будет реализовано все, что связано с игрой. Может оказаться полезным опыт использования MySQL, Oracle или MS SQL для разработки баз данных.

«Коммерсант» сообщает со ссылкой на документ, что в настоящее время Правительство РФ рассматривает возможность создания Фонд

поддержки игровых проектов для помощи разработчикам компьютерных, мобильных и онлайн-игр.[20]

Актуальной является **задача** проектирования и разработки видеоигры. В результате договоренности был создан стартап с целью разработки продуктов в сфере массовых развлечений.

Предметом исследования является непосредственно проектирование и разработка видеоигры.

Теоретическая значимость заключается в анализе существующих стандартов, методов и методологий проектирования и разработки программных проектов, а также изучение популярных средств и сред разработки видеоигр.

Практическая значимость заключается в непосредственном проектировании и разработке видеоигры с сопутствующими материалами и использованием среды разработки выбранной по ходу исследования.

Целью выпускной квалификационной работы: создание игрового приложения для мобильных платформ на базе iOS и Android.

Для достижения этой цели необходимо решить следующие задачи:

1. проанализировать существующие стандарты, методы и методологии проектирования и разработки программных проектов;
2. провести анализ популярных платформ разработки в реальном времени;
3. создать концептуальную модель видеоигры;
4. разработать дизайн-документ проекта видеоигры;
5. совершить программную реализацию видеоигры посредством Unity.
6. Результатом бакалаврской выпускной квалификационной работы является рабочий прототип игры, включающий в себя меню, демонстрационные уровни с рабочими механиками окружения и игровых персонажей.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ РАЗРАБОТКА ВИДЕОИГР. МЕТОДЫ И МЕТОДОЛОГИИ РАЗРАБОТКИ ВИДЕОИГР

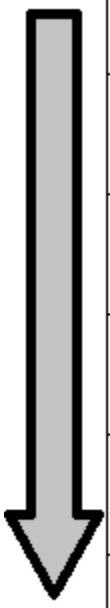
Видеоигра (англ. video game) — игра с использованием изображений, сгенерированных электронной аппаратурой. Другими словами, видеоигра является электронной игрой, которая базируется на взаимодействии человека и устройства посредством визуального интерфейса, например телевизора, монитора компьютера или телефона.[1]

Видеоигры можно классифицировать по основным признакам:

- жанр (RTS, RPG, FPS и т.д.);
- платформа (ПК, Android, Playstation и т.д.);
- количество игроков (одно- или многопользовательская игра);
- визуальное представление (2D, 3D, текстовая);
- И др.

О разработке игр или геймдеве (от англ. Game Development) невозможно судить отдельно от индустрии видеоигр в целом. Геймдев это лишь часть сложной системы, которая обеспечивает полный жизненный цикл (производства, распространения, потребления и поддержки) такого продукта, как видеоигра. На рисунке 1 предоставлена упрощенная схема игровой индустрии:

[1] https://ru.wikipedia.org/wiki/Видеоигра



Разработчик	Создают игру
Издатель	Ответственны за маркетинг и продвижение игры
Дистрибьютор	Организует логистику и доставку игры в розницу и/или потребителям
Платформа	Консоль/машина/формат для которого(-ой) предназначена игра
Продавец	Точка приобретения товара потребителем
Потребитель	Аудитория, покупающая игру

Рисунок 1

Современные проекты разработки игр полагаются на специализированные инструменты для физики, графики и создания интерфейса, чтобы абстрагироваться от общих игровых функций и снизить стоимость разработки.

В любой области знаний или предметной области решаемые проблемы часто имеют много общего. Методы, специфичные для предметной области, основаны на общих чертах между приложениями в аналогичной области для повышения качества и эффективности работы в этой области. Интеграция существующих знаний в предметной области с процессом разработки программного обеспечения делает возможными такие улучшения. Многие из этих методов повторно используют существующие решения общих проблем в своей области. Другие позволяют программистам и экспертам в предметной области разрабатывать новые приложения на языке предметной области, а не на языке компьютеров. В любом случае инструменты предметной области абстрагируются от общих черт своей области, позволяя

разработчикам выражать и контролировать концепции и процедуры, которые варьируются от приложения к приложению.

Проблема, однажды решенная, не требует решения снова. Это руководящий принцип библиотек. Существующие решения хорошо изученных распространенных проблем в предметной области можно сгруппировать и инкапсулировать в библиотеке. Используя библиотеки, разработчики могут сэкономить время и усилия, связанные с изобретением велосипеда, и вместо этого могут положиться на экспертные решения общих задач.

1.1 Игровой движок (Game Engine)

Игровой движок — это среда разработки программного обеспечения, также называемая «игровой архитектурой» или «платформой (фреймворком) игры», с настройками и конфигурациями, которые оптимизируют и упрощают разработку видеоигр на различных языках программирования. Игровой движок может включать в себя механизм рендеринга 2D- или 3D-графики, совместимый с различными форматами импорта, физический движок, имитирующий действия в реальном мире, искусственный интеллект (ИИ), автоматически реагирующий на действия игрока, звуковой движок, управляющий звуковыми эффектами, анимационный движок и множество других функций.

Ранние видеоигры разрабатывались с собственными движками рендеринга, каждый из которых был специально разработан для одной игры. Со временем игровые движки превратились из проприетарных собственных движков в коммерчески разработанные движки, широкодоступные сегодня. Разработчики игр, могут упростить и ускорить процесс разработки игр,

используя разработанные игровые движки для создания новых игр или распространения существующих игр на дополнительные платформы.

1.1.1 Unity

Unity[2] (unity в переводе с англ. — «единство», произносится как «ю́нити») — межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.[3]

Игровой движок Unity, разрабатываемый с 2005 года, стал одним из основных продуктов индустрии инди-игр. С постоянными обновлениями и новыми важными функциями, такими как Unity Reflect, которые добавляются каждый год, поддержка движка невероятна. Этот движок не только хорошо подходит как для 2D-, так и для 3D-игр любого типа, но также является популярным выбором для создания игр виртуальной реальности и разработки дополненной реальности благодаря тому, что многие компании и разработчики создают удобные SDK для движка.

Помимо этого, у Unity также есть большое сообщество с активным магазином ассетов[4] (Asset Store), где у вас под рукой есть как бесплатные, так и платные ресурсы. Поскольку это такой надежный движок, он так же бесплатен для разработчиков, зарабатывающих менее 100 тысяч долларов в год, это отличный выбор для новичков, независимо от того, что они хотят создавать.

---

[1] https://unity.com/ru

[3] https://ru.wikipedia.org/wiki/Unity_(игровой_движок)

[4] Игровой ассет (Game Asset) - цифровой объект, преимущественно состоящий из однотипных данных, неделимая сущность, которая представляет часть игрового контента и обладает некими свойствами.

При этом, если вы хотите построить целую студию видеоигр на основе Unity, лицензии могут быть дорогостоящими, хотя они имеют больше функций. Кроме того, Unity может быть тяжелее для вашей системы, если вы запускаете некоторые из технических демонстраций более высокого уровня, чтобы раскрыть все возможности движка. Наконец, стоит упомянуть, что, поскольку Unity так часто обновляется, можно легко пропустить новые функции или найти старые, поскольку пользовательский интерфейс и система доступа к ним могут измениться.

Преимущества Unity:

- бесплатен для личного пользования и для разработчиков с доходом менее 100 тыс. долларов;
- отлично подходит для 2D и 3D игр;
- мощная поддержка разработки мобильных игр;
- доступность VR и AR SDK;
- Магазин ассетов с множеством бесплатных ресурсов.

Недостатки Unity:

- дорогие лицензии для профессионалов;
- для демонстраций передовых технологий требуются более совершенные компьютеры;
- много изменений пользовательского интерфейса.

Примеры игр, созданных с помощью Unity:

- Rust;
- Genshin Impact;
- Hollow Knight;
- Hearthstone;
- The Forest;

1.1.2 Unreal Engine

Unreal Engine — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх.[5]

Unreal Engine обладает мощными возможностям с освещением, шейдерами и многим другим, что делает его одним из самых популярных движков для высокобюджетных разработок, существующих сегодня. Учитывая его широкое использование в этом секторе, движок был специально разработан для решения множества сложных задач более эффективно, чем другие движки.

Однако Unreal Engine лучше подходит для крупных проектов и проектов, над которыми работает большая команда. Кроме того, поскольку программа тяжела с точки зрения графики, для нее требуется более мощный компьютер по сравнению с другими движками, такими как Unity. Помимо этого, также следует отметить, что, хотя Unreal Engine может создавать 2D-игры в дополнение к 3D-играм, этот движок не обязательно лучше всего подходит для этой задачи.

Преимущества:

- отлично подходит для высококачественной графики;
- более производительный, чем другие движки;
- лучший выбор для виртуальной реальности;
- визуальное проектирование для непрограммистов;

[5] https://ru.wikipedia.org/wiki/Unreal_Engine

- большой магазин бесплатных ассетов.

Недостатки:

- не лучший вариант для простых или сольных проектов;
- высококачественная графика требует более мощных компьютеров;
- лучше для 3D, чем для 2D игр.

Примеры игр, созданных с помощью Unreal Engine:

- Mass Effect;
- Gears of War;
- Fortnite;
- Sea of Thieves;
- Mortal Kombat 11.

1.1.3 GameMaker Studio

GameMaker – очень удобный выбор для тех, у кого нет опыта программирования, поскольку он в основном использует собственный язык визуальных сценариев с перетаскиванием, чтобы пользователи с любым уровнем навыков могли создавать игры. Тем не менее, для тех, кто предпочитает кодирование, он также предлагает свой язык (GameMaker Language) для программирования пользовательских поведений, которые выходят за рамки того, что может охватывать визуальное программирование. В целом, движок очень удобен для новичков и открывает возможности разработки игр практически для всех. Однако GameMaker Studio предназначен специально для 2D-игр. Хотя у него есть ограниченные 3D-возможности, он не может сравниться с тем, что могут сделать Unity или Unreal Engine.

Преимущества:

- поддерживается на множестве платформ;
- простое программирование методом перетаскивания;
- прост в освоении.

Недостатки:

- ориентирован на 2D игры;
- ограниченный функционал;

Примеры игр, созданных с помощью GameMaker Studio:

- Katana ZERO;
- Undertale;
- Hotline Miami.

1.1.3 Defold

Defold — это совершенно бесплатный игровой движок для разработки консольных, настольных, мобильных и веб-игр. Исходный код доступен на GitHub с удобной для разработчиков лицензией. Редактор Defold работает в Windows, Linux и macOS и включает в себя редактор кода, отладчик, профилировщик и расширенные редакторы сцен и пользовательского интерфейса. Логика игры написана на Lua с возможностью использования собственного кода для расширения движка дополнительными функциями. Defold используется все большим числом разработчиков для создания коммерческих хитов, а также игр для геймджемов и в школах для обучения разработке игр. Defold известен своей простотой использования, его хвалят за техническую документацию и дружелюбное сообщество разработчиков.[6]

Преимущества:

- поддержка большого количества платформ;

[6] https://defold.com/about/

- простое программирование с помощью Lua;
- прост в освоении;
- полностью бесплатен;
- ориентирован на мобильные платформы.

Недостатки:

- ориентирован на 2D игры;
- ограниченный функционал;
- небольшое сообщество;
- сложно найти обучающие материалы.

Примеры игр, созданных с помощью Defold:

- Plague Lords;
- Blastlands;
- JUUMP!

1.2 Методы и модели разработки видеоигр.

Разработка видеоигр – это процесс разработки программного обеспечения, т.к. по своей сути видеоигра – хоть и специфическое, но все же программное обеспечение (с графикой, звуком, программированием, проработкой игровых механик и др). Зачастую начинающие разработчики видеоигр зачастую упускают из виду формальные методологии, что негативно сказывается на всей разработке.

Различают следующие этапы разработки игры:

- Планирование: тип игры, ключевые особенности, мир, персонажи, целевая аудитория и др.;
- Подготовка к производству (pre-production): технологические возможности, раннее прототипирование, составление дизайн-документа и др.;

- Производство (production): моделирование, проектирование, создание аудио- и визуальных эффектов, разработка игровых механик и др.;
- Тестирование: исправление ошибок, «полировка» всех аспектов видеигры, выявление «проблемных» частей игрового процесса;
- Предзапуск (pre-Launch): альфа/бета-тестирование;
- Запуск: публикация готового продукта;
- Послепроизводственный этап (post-production): исправление ошибок, не выявленных на стадии тестирования, выпуск обновлений и пользовательская поддержка.

Существует много устоявшихся методологий разработки программного продукта. Выбор зависит от специфики проекта, системы бюджетирования, предпочтений руководства.

Разработка игр с неправильной/плохой методологией (или вовсе с ее отсутствием) наиболее вероятно будет полна сопутствующих проблем и ошибок. Планирование разработки важно также для проекта, в котором команда разработки состоит из одного человека.

1.2.1 Waterfall (каскадная модель или «водопад»)

Водопадная модель разработки игр — одна из старейших и широко используемых моделей разработки игр в Индии и опытных разработчиков игр. Он использует прямой и последовательный процесс, такой как водопад. Методологии этой модели помогают разработчикам игр постоянно двигаться вперед, следуя прошлым разработкам. Разработчики просто сосредотачиваются на текущем процессе разработки и, завершив его, продвигаются вперед. Без завершения текущего этапа разработчики не

переходят к следующему этапу. В «водопаде» обычно выделяют 5 этапов проекта (Рисунок 2):

- проектирование;
- планирование (дизайн);
- разработка (реализация, кодирование);
- тестирование;
- эксплуатация и сопровождение (поддержка).

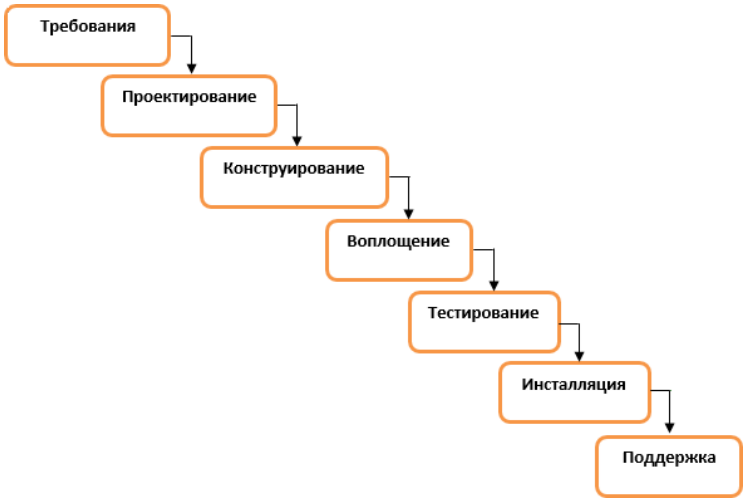


Рисунок 2

К плюсам данного метода можно отнести легкость управления проектом. Благодаря заданной строгости последовательности выполнения этапов разработка проходит быстро, а стоимость и сроки заранее известны.

Минусом же является то, что данная модель будет давать отличный результат лишь в проектах с заранее определенными требованиями, отсутствует гибкость. То есть, метод не подразумевает изменений по ходу разработки. Например, тестирование производится только после полного окончания разработки и не может быть реализовано по ее ходу.

Пример: CD Project – польская компания, занимающаяся изданием и разработкой видео игр, использовала данную модель как основную для разработки своих проектов. Однако их игра Cyberpunk 2077, выпущенная в декабре 2020 года, была полна ошибок и проблем с производительностью, из-за чего игра стала предметом за кранчи, маркетинговых методы и

управление ожиданиями от выпуска игры. Игра также занимает центральное место в двух коллективных исках, поданных инвесторами, которые утверждают, что компания сделала ложные или вводящие в заблуждение заявления о Cyberpunk 2077. Всё это стало следствием неэффективного управления разработкой игры. Позднее компания сделала заявление о изменении стратегии компании. Изменения включают в себя преобразование студии и изменение методологии разработки видеоигр в пользу гибкой (Agile) системы разработки.[26]

1.2.2 Итеративная (итерационная) модель

В этой модели разработчики игр могут начать свой процесс с реализации набора требований к программному обеспечению. Кроме того, эта реализация может раскрыть другие требования процесса. Для этого не нужны полные требования и спецификации программного обеспечения. Эта модель может быть разделена в основном на пять частей. Он включает в себя планирование, анализ и проектирование, реализацию, тестирование и, наконец, этап эволюции. Эти этапы очень важны для успешного завершения проектов. (Рисунок 3)

Итеративная модель представляет из себя итеративное выполнение этапов жизненного цикла ПО. Проект при этом подходе на каждой итерации проходит повторяющийся цикл PDCA.

Рисунок 3

Цикл Деминга (PDCA - Plan-Do-Check-Act) представляет собой четырехэтапный итеративный метод, используемый для решения проблем и улучшения организационных процессов.

Plan: на этом этапе исследуется текущая ситуация, для полного понимания характера решаемой проблемы. Разработали план и основа для работы, а также указываются желаемые результаты и результаты.

Do: определяется реальная проблему путем анализа данных, определения и реализации плана решения. Цикл PDCA фокусируется на небольших постепенных изменениях, которые помогают улучшить процессы с минимальным нарушением.

Check: отслеживание эффекта от плана внедрения и при необходимости нахождение контрмер для дальнейшего улучшения решения. Требуется проверка во время реализации, чтобы убедиться, что цели проекта достигаются. Рекомендуется провести вторую проверку по завершении, чтобы учесть успехи и неудачи, а также внести коррективы в будущем на основе извлеченных результатов.

Act: имплементация решений и рекомендаций. Принятие решения, эффективно ли решение для дальнейшей интеграции его в стандартные рабочие практики, либо откажитесь отказ от него.

Итеративная модель зачастую используется не для всей разработки видеоигры, а для некоторых его частей, например:

- пользовательский интерфейс;
- игровые механики;
- графический дизайн;
- архитектура;
- др.

1.2.3 V-образная (V-shape) модель

Модель разработки игр V-Shape (рисунок 4) — один из самых эффективных процессов разработки игр. Инди-разработчкики и игровые

студии используют методологию этой модели для разработки потрясающих игр. Это также последовательная модель, которая имеет пошаговые этапы. Эта модель также широко называется моделью верификации и валидации. Особенность этой V-образной модели заключается в том, что она разделена на три части: первая — проверка, вторая — проверка, а третья — кодирование. Этап верификации включает анализ требований, проектирование системы, проектирование архитектуры и проектирование модуля. Этап валидации включает модульное тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование. С другой стороны, раздел кодирования соединяет обе части друг с другом. В целом это отличная модель разработки видеоигр, которые всесторонне используются инди-разработчиками и игровыми студиями по всему миру.

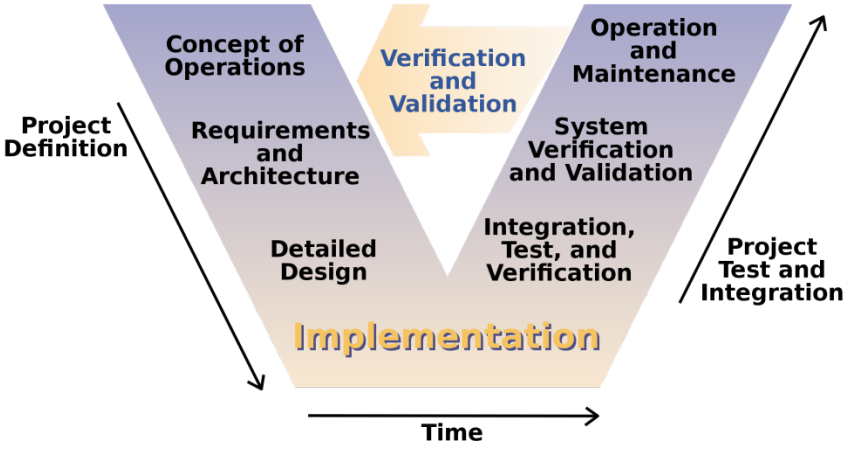


Рисунок 4

1.2.4 Спиральная модель

Спиральная модель – тип итеративной модели разработки. Обычно используется в проектах с высоким риском.

Жизненный цикл данной модели представляет собой спираль, где каждый цикл этой спирали указывает на этап разработки. В модели может

иметься любое количество циклов в зависимости от сложности и характера проекта. Каждый цикл имеет 4 области (квадранта)(Рисунок 5):

I. используется для определения целей, альтернативных решений и ограничений;
II. анализ рисков и оценка альтернатив;
III. разработка и тестирование;
IV. планировка следующей итерации.

На любом этапе цикла, если кто-то из команды считает, что риск слишком велик, есть условие прервать цикл. Такая оценка может выполняться внутренним членом команды или внешним клиентом.

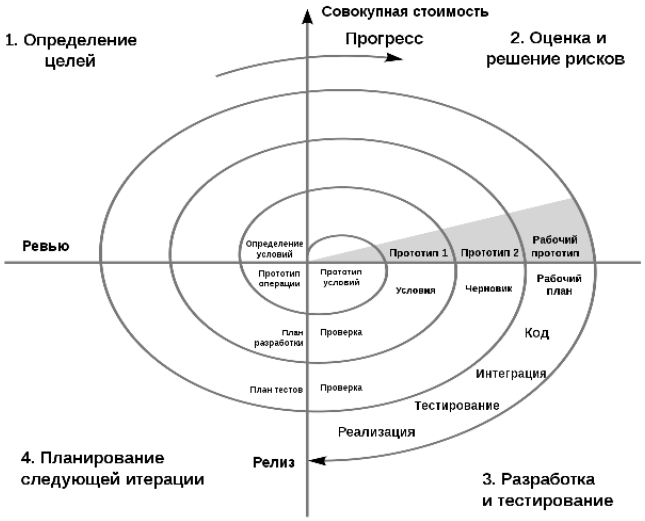


Рисунок 5

Преимущество спиральной модели заключается в том, что она позволяет просто добавлять элементы продуктов, если они доступны или известны. Это гарантирует, что не возникнет конфликта с предыдущими требованиями и дизайном. Этот метод надежен для подходов с несколькими сборками и выпусками программного обеспечения, что позволяет упорядоченно переходить к обслуживанию. Следующим положительным аспектом этого метода является то, что модель спирали требует раннего вовлечения пользователя в процесс разработки системы. Наоборот, действительно строгое управление применяется для того, чтобы продукты были закончены, и, следовательно, существует риск закручивания спирали в

неопределенный цикл. Поэтому очень важно создавать и успешно использовать продукт с дисциплиной изменений и степенью обработки запросов на изменение.

1.2.5 Agile

Agile (Гибкая разработка ПО) – обобщающий термин для целого ряда подходов и практик, основанных на ценностях Манифеста гибкой разработки программного обеспечения[7] и принципах, лежащих в его основе.

12 принципов, которые составляют Agile Methodology, можно поделить на 4 главные идеи:

- приоритет людей и общения над инструментами и процессами;
- приоритет работающего продукта над полной документацией;
- приоритет сотрудничества с заказчиков над утверждением контракта;
- приоритет готовности меняться над следованием первоначально созданному плану.

Agile является и итеративным, и инкрементальным. Согласно его принципам, проект разбивается на модули, которые затем собираются в готовый прототип. Следовательно, инициация и планирование проводятся для всего проекта, а последующие этапы (разработка, тестирование и пр.) проводятся для каждого модуля (инкремента) отдельно и итерационно. Agile подходит для нацеленных на длительный жизненный цикл больших проектов, в процессе реализации которых требования к продукту могут меняться.

По сути Agile не является ни методологией, ни моделью. Его можно рассматривать как набор идей и принципов для реализации проектов. А уже

[7] http://agilemanifesto.org/iso/ru/manifesto.html

на основе этих принципов и практик были разработаны различные гибкие методы (фреймворки/frameworks): Scrum, Kanban и др. Эти методологии достаточно сильно отличаются друг от друга, однако все они придерживаются принципов Agile.

Главным преимуществом гибкой модели (по сравнению с каскадной) является гибкость и адаптивность. В условиях современного рынка это важнейший аргумент при создании программных проектов, что и делает Agile такой популярной в применении компаниями.

1.2.5.1 SCRUM

SCRUM – гибкая методология управления задачами и проектами. Принципы SCRUM заключается в командном подходе, коротких итерациях и непрерывном улучшении в процессе работы. Хоть данная методология и применима в решениях любых задач (от производства до бытовых проблем), наиболее популярна она в сфере IT.

Не смотря на то, что Scrum заимствует принципы Agile, данная методология использует свои методы и тактики управления проектами. «Agile – философия, а SCRUM – методология. И хотя SCRUM – Agile, Agile – это не SCRUM».

В рамках SCRUM в центре проекта стоит команда. Зачастую может отсутствовать менеджер проекта, следовательно, работа основывается на самоорганизации и самоуправлении. Данный подход эффективен в команде опытных и мотивированных специалистов.

Следуя принципам Agile, SCRUM разбивает проект на части. Далее эти части приоритезируются менеджером проекта (если он отсутствует, в команде обычно присутствует представитель заказчика). Команда разработчиков обычно состоит из 5-9 человек. Самые приоритетные части

отбираются для выполнения в «спринте» - итерации в SCRUM, длящиеся от 2 до 4 недель. В конце спринта предоставляется готовый инкремент продукта, после чего команда приступает к следующей итерации. Перед началом каждого спринта происходит переоценка невыполненных частей проекта и внесение изменений в их содержание. В этом процессе участвует команда и SCRUM Master (лидер команды проекта).

На рисунке 6 представлена типичная схема работы по методологии SCRUM:

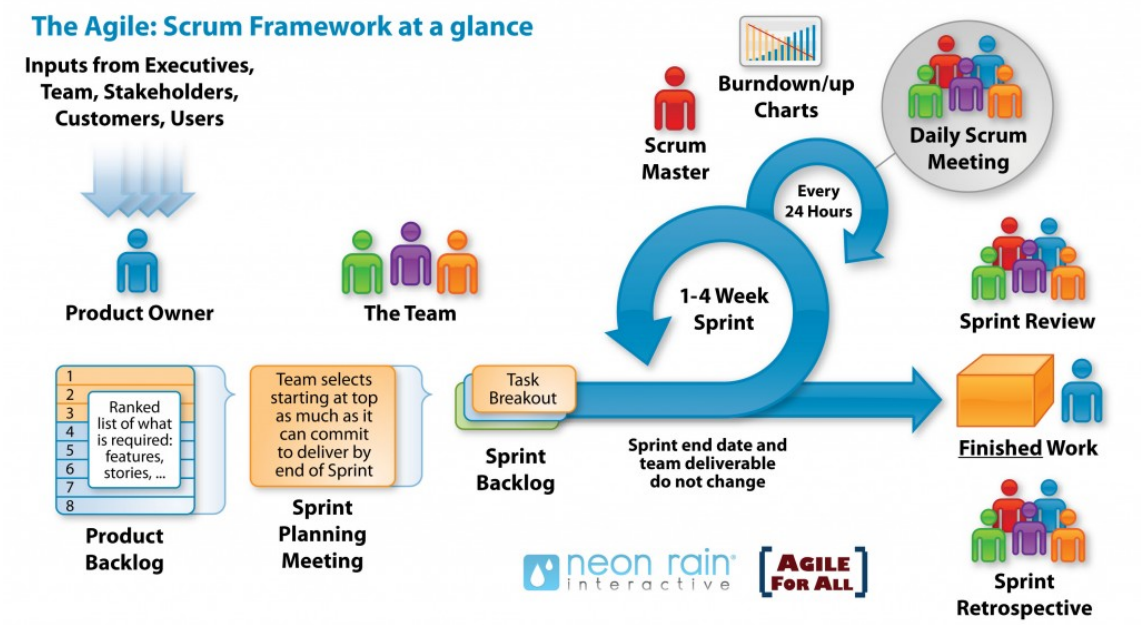


Рисунок 6

Методологии свойственны преимущества и недостатки Agile.

Плюсы методологии:

- спринты, благодаря которым команда может справиться с большими сложными проектами в меньшие сроки;
- динамичность – работа разбивается на 30-дневные циклы с еженедельными собраниями, вследствие чего разработка и внесение изменений проходят динамично;

- командная работа – по причине зачастую отсутствия менеджера проекта, работа команды основывается на самоорганизации, члены которой четче понимают свои задачи и вовлечены в проект.

Минусы методологии:

- недостаточная гибкость для больших команд;
- требовательность к компетенциям членов команды;
- требовательность к командной работе.

SCRUM наиболее эффективен для применения в опытных, дисциплинированных и мотивированных командах.

1.2.5.2 Kanban

Kanban – метод управления разработкой, основанный на принципе «точно в срок», который способствует равномерному распределению нагрузки между членами команды разработки.

Если SCRUM – фреймворк с довольно жесткими правилами и принципами, то в случае Kanban – это метод, в котором можно использовать все практики, или их часть, а можно не использовать их вовсе.

Kanban в наше время имеет большую популярность в сфере разработки программного обеспечения. Методология основывается на 9 ценностях :

- прозрачность;
- баланс;
- сотрудничество;
- клиентоориентированность;
- поток;
- лидерство;
- понимание;
- согласие;

- уважение.

Основные принципы:

- начать с того, что есть сейчас;
- договориться об эволюционном развитии;
- поощрять лидерство на всех уровнях.

Суть методологии заключается в разбиении плана работы на отдельные этапы (прим. Рисунок 7). Для этого используется доска Kanban (может использоваться как физическая, так и цифровая версия), которая делится на несколько столбцов. Столбцы обозначают этапы разработки. Ключевая ценность Kanban – сохранение последовательности выполнения этапов (поток). Далее проект делится на части (задачи), названия которых записываются на Kanban-карточки, после чего их крепят в начало доски. Доска визуально отражает все задачи проекта и этапы их разработки.

Визуализация помогает проанализировать проект в целом, скорректировать отдельные его части. Завершить проект точно в срок реально, если правильно распределять нагрузку команды.

Рисунок 7

1.2.6 Проблемы применения методологий в разработке при создании видеоигр

Исследовав блоги и статьи различных видеоигровых разработчиков, можно сделать вывод, что водопадная модель - ненавистный процесс работы

для разработчика, особенно в играх, потому что он рассматривает проект как набор определенностей, а разработку как акт перевода. Реальность большинства игровых проектов такова, что они часто пытаются обнаружить работающую динамику игры, что по своей сути является небрежным и экспериментальным процессом. Таким образом, большинство разработчиков предпочитают работать в гибкой (формальной или неформальной) манере. Однако большинство контрактов между разработчиками и издателями сформулированы в виде квази-водопада, предписывая вехи, такие как документы по дизайну игры, и результаты, такие как активы или функциональность, в соответствии с графиком. Причина, по которой они делают это таким образом, заключается в том, что водопад имеет тенденцию создавать иллюзию уверенности и является самым простым методом для объяснения на руководящем уровне. Однако реальность такой работы — расточительство, отмененные проекты и кранч.

Большинство игровых компаний используют гибридные методы методологий, в чистом виде они практически не встречаются.[27]

1.3 Итеративное проектирование видеоигр

Итеративный процесс проектирования делится на 4 этапа:

- анализ;
- проектирование;
- реализация;
- тестирование.

Анализ проводится с целью ответить на вопросы для «кого проектируется игра?», «какими ресурсами обладает разработчик?», «какие уже существуют похожие проекты?», «как быстрее получить рабочий прототип?»

Проектирование – этап планирования проекта, направленный на разработку общей концепции игры, этапов разработки, создание документации, описывающей все задачи и графики их выполнения.

Реализация – воплощение проекта, разработка. Цель данного этапа – как можно быстрее перейти от концепции к рабочему прототипу игры. Создание небольшого фрагмента перед тестированием зачастую помогает обратить внимание на мелкие ошибки и недоработки, нежели отслеживание их при реализации больших фрагментов.

Тестирование. Поиск ошибок, их документирование, выслеживание недоработок игровых механик. Эффективнее будет иметь несколько тестеровщиков для нахождения большего количества ошибок и недочетов, а также для большего количества отзывов о проекте.

1.4 Unity

1.4.1 Интерфейс редактора Unity.

На рисунке 8 представлена графическая оболочка редактора:

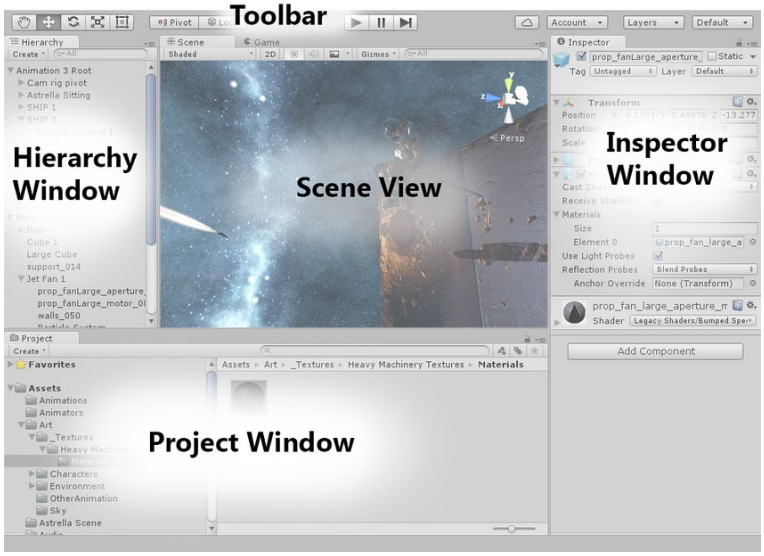


Рисунок 8

- **Обозреватель проекта (Project Window)** *(*рисунок 9*)*

Окно проекта отображает библиотеку ассетов, доступных для использования в проекте. Когда импортируются ассеты в проект, они появляются здесь.

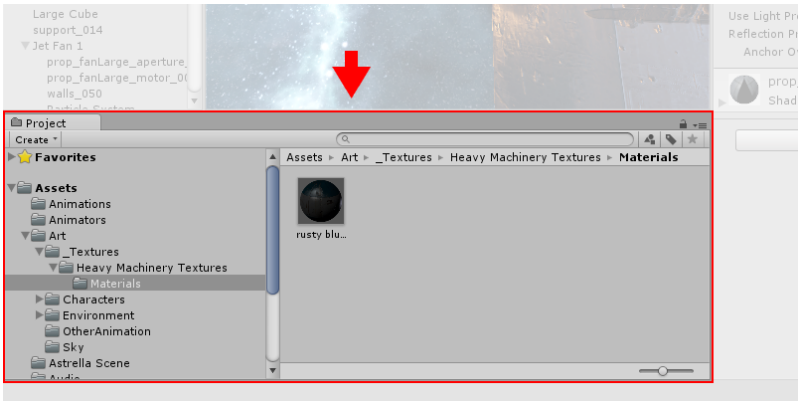


Рисунок 9

Панель в левой части окна отображает структуру папок проекта в иерархическом виде. Над списком структуры находится раздел Избранное (Favorites), в котором можно хранить ссылки на часто используемые ассеты для быстрого доступа к ним. В Избранном также можно сохранять поисковые запросы.

Окно проекта имеет мощные возможности поиска, особо полезные в большом или незнакомом проекте. Поиск будет фильтровать ассеты согласно тексту, введенному в поле для поиска (находится в верхней правой части обозревателя).

- **Окно сцены (Scene View)** (рисунок 10*)*

Окно «Сцена» позволяет визуально перемещаться по сцене и редактировать ее. Вид сцены может отображать трехмерную или двухмерную перспективу, в зависимости от типа проекта, над которым ведется работа.

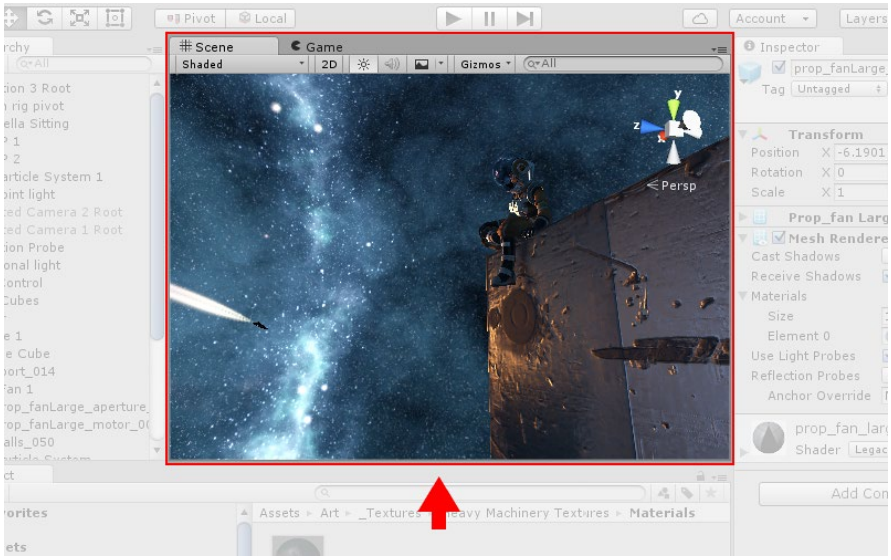


Рисунок 10

- **Игровое окно (Game View)** (рисунок 11*)*

Игровое окно показывает то, как будет видеть проект пользователь. Это отображение финальной, опубликованной игры. Требуется использовать одну или более камер для контроля того, что игрок фактически увидит когда запустит готовый проект.

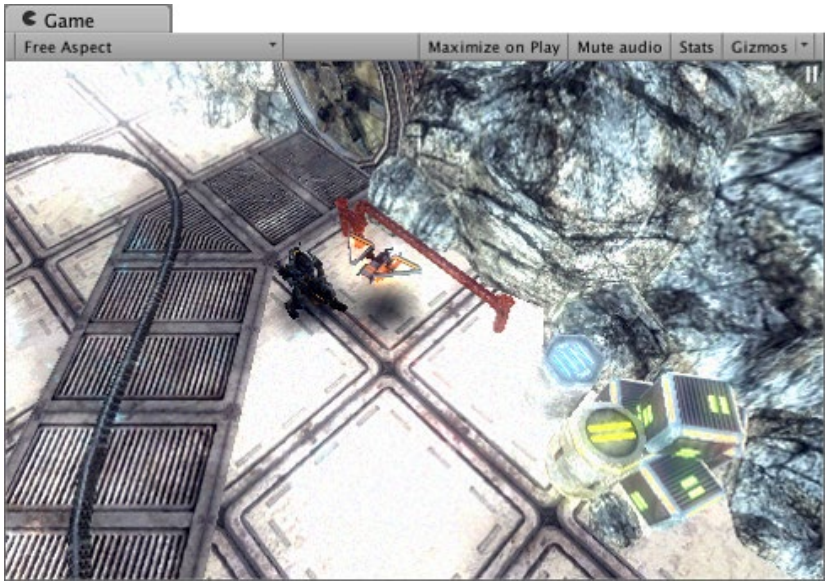


Рисунок 11

- **Окно Иерархии (Hierarchy Window)** *(*рисунок 12*)*

Окно иерархии представляет собой иерархическое текстовое представление каждого объекта в сцене. У каждого элемента сцены есть запись в иерархии, поэтому два окна неразрывно связаны. Иерархия раскрывает структуру того, как объекты связаны друг с другом.

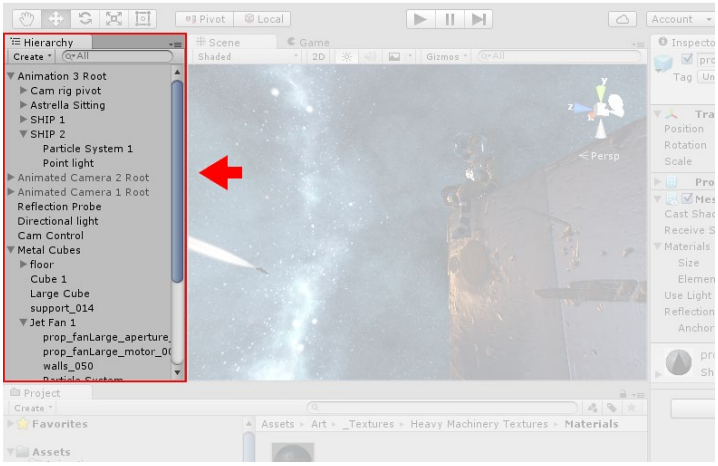


Рисунок 12

- **Окно инспектора (Inspector Window)**

Окно инспектора позволяет просматривать и редактировать все свойства выбранного объекта. Поскольку разные типы объектов имеют разные наборы свойств, макет и содержимое окна инспектора будут различаться.

- **Панель инструментов (Toolbar)**

Панель инструментов обеспечивает доступ к наиболее важным рабочим функциям. Слева он содержит основные инструменты для управления видом сцены и объектами внутри него. В центре находятся элементы управления воспроизведением, паузой и шагом. Кнопки справа обеспечивают доступ к облачным службам Unity и учетной записи Unity, за которыми следует меню видимости слоев и, наконец, меню макета редактора (которое предоставляет несколько альтернативных макетов для окон редактора и позволяет сохранять свои собственные настройки). Панель инструментов — это не окно, и это единственная часть интерфейса Unity, которую нельзя изменить.

1.4.2 Программирование в Unity

В Unity скрипты можно использовать для разработки практически любого элемента игры (игровой мир, сюжет игры, персонажи, музыка. визуальные эффекты и др.) или интерактивного контента с графикой реального времени. Редактор поддерживает скрипты, созданные в соответствии с одним из двух основных подходов: традиционным и широко использующимся объектно-ориентированным подходом и информационно-ориентированным подходом.

Объектно-ориентированное проектирование – это процесс планирования системы взаимодействующих объектов с целью решения программной проблемы.

Информационно-ориентированное программирование – парадигма, которая устраняет обычную сложность, возникающую при объединении данных и кода в объекты и классы. В ИОП хранятся данные приложения в постоянных общих структурах данных, отделенных от кода программы. Для манипулирования данными без их изменения используются функции общего назначения. Этот подход избавляет приложения от ошибок, связанных с состоянием, и значительно упрощает понимание и поддержку кода.

Unity поддерживает C#, стандартный в отрасли язык программирования, в некоторой степени похожий на Java или C++. C# легче в изучении, например по сравнению с C++. К тому же, он относится к категории языков «с управлением памятью», то есть он автоматически распределяет память, устраняет утечки и т.д. Принцип работы игрового движка Unity строится на трёх блоках:

- объекты GameObject;
- компоненты;
- переменные.

GameObject — важное понятие в редакторе. Каждый объект в проекте это GameObject: от персонажей и собираемых предметов до света, камеры и спецэффектов.

GameObject не может делать что-либо сам по себе — требуется придать ему свойства, перед тем как он сможет стать персонажем, средой или спецэффектом.

GameObjects являются строительными блоками для сцен в Unity и действуют как контейнер для функциональных компонентов, которые определяют, как GameObject выглядит и что GameObject делает. В сценариях класс GameObject предоставляет набор методов, которые позволяют работать с ними в коде, включая поиск, установление соединений и отправку сообщений между игровыми объектами, а также добавление или удаление компонентов, прикрепленных к игровому объекту, и установку значений, относящихся к их статус на сцене. Чтобы придать игровому объекту свойства, необходимые для того, чтобы он стал светом, деревом или камерой, требуется добавить к нему компоненты. В зависимости от того, какой объект требуется создать, в GameObject добавляются различные комбинации компонентов.

Пустой GameObject содержит имя ("GameObject"), тег ("Untagged") и слой ("Default"). Он также содержит компонент Transform.

Компоненты определяют поведение игровых объектов, к которым они прикреплены, и управляют ими.

Компонент - базовый класс для всего, что связано с GameObject. Программный код никогда не создает компонент напрямую. Вместо этого создается сценарий для GameObject.

Компонент Transform имеет решающее значение для всех игровых объектов, поэтому у каждого игрового объекта есть один, но игровые объекты также могут содержать другие компоненты. Каждая сцена по

умолчанию имеет GameObject, отвечающий за основную камеру. Он состоит из нескольких компонентов.

Глядя на инспектор игрового объекта Main Camera, можно увидеть, что он содержит дополнительные компоненты. В частности, компонент камеры, GUILayer, Flare Layer и Audio Listener. Все эти компоненты обеспечивают функциональность этого GameObject.

Среда разработки Unity располагает обширной библиотекой компонентов, обладающих разными свойствами, параметрами и функционалом, но зачастую разработчик сталкивается с необходимостью создания собственных для реализации нужных ему алгоритмов. Их можно создать с помощью скриптов. Скрипты позволяют запускать игровые события, изменять свойства компонентов с течением времени и реагировать на действия игрока.

Среда выполнения скриптов в Unity поддерживает множество функций C# и средств отладки, доступных в C# 6.0 и более поздних версиях. Это также обеспечивает среду разработки для C#, дополняющую новые функции C#. В настоящее время Unity поставляется с Visual Studio (интегрированная среда разработки, созданная Microsoft и используемая для разработки различных типов программного обеспечения).

Unity имеет две системы программирования (scripting backend[8]): Mono и IL2CPP (промежуточный язык для C++), каждый из которых использует другую технику компиляции:

- Mono использует JIT-компиляцию и компилирует код по запросу во время выполнения.
- IL2CPP использует предварительную компиляцию (AOT) и компилирует все ваше приложение перед его запуском.

[8] Фреймворк, который обеспечивает выполнение сценариев в Unity.

1.4.3 Скриптинг

Скрипты сообщают GameObjects, как вести себя; именно скрипты и компоненты, прикрепленные к GameObjects, и то, как они взаимодействуют друг с другом, создают игровой процесс. Написание сценариев в Unity отличается от чистого программирования. В Unity не нужно создавать код, который запускает приложение, потому что Unity самостоятельно. Вместо этого разработчик сосредотачивается на игровом процессе в скриптах (сценариях). Unity работает в большом цикле. Движок считывает все данные, находящиеся в игровой сцене. Например, он считывает свет, сетки, поведение и обрабатывает всю эту информацию самостоятельно.

API (Application Programming Interface) означает программный интерфейс приложения, что по сути означает способ взаимодействия с частью программного обеспечения через код. Любое обеспечение, с которым нужно взаимодействовать, имеет API.

Unity Scripting API — это способ программного взаимодействия с игровым движком и редактором Unity. У каждого объекта, который добавляется в сцену, есть класс, и у этого класса есть API, которым можно управлять с помощью кода.

При создании скрипта в редакторе, создается файл, на рисунке 13 отображен открытый созданный файл:

NewBehaviourScript.cs

Прочие файлы

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class NewBehaviourScript : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
19
```

Рисунок 13

Скрипт устанавливает связь с внутренней работой Unity, реализуя класс, производный от встроенного класса с именем MonoBehaviour. При каждом прикреплении компонента скрипта к игровому объекту, создается новый экземпляр объекта, определенного программой. Имя класса берется из имени, которое было указано при создании файла. Имя класса и имя файла должны совпадать, чтобы дать возможность прикрепить компонент к игровому объекту.

Самое важной в методе – это две функции, определенные внутри класса. Функция **Update** – место для размещения кода, который будет обрабатываться каждый кадр для игрового объекта. Это может включать в себя движение, инициацию действия, реакцию на ввод пользователя, в основном все, что необходимо обрабатывать во время игры. Чтобы функция могла выполнять свою задачу, рекомендуется устанавливать переменные, читать преференции и устанавливать связи с другими игровыми объектами до того, как произойдет какое-либо игровое событие. Функция **Start** вызывается до того, как начинается игровой процесс (т.е. до первого вызова функции Update), и это лучшее место для любой инициализации.

1.4.4 Классы

Классы – это заготовки объектов. Классы в Unity – контейнер для переменных и функций, это способ определения пользовательского типа данных с помощью процесса, называемого инкапсуляцией, который объединяет связанные данные и методы.

Они представляют собой организационный инструмент, известный как объектно-ориентированное программирование (ООП). Один из принципов ООП состоит в том, чтобы разделить сценарии на насколько сценариев, каждый из которых берет на себя одну роль или классы ответственности, поэтому рекомендуется, чтобы он был посвящен одной задаче.

Список наиболее важных и часто используемых встроенных в Unity классов, которые можно использовать при написании сценариев:

- GameObject. Представляет тип объектов, которые могут существовать в сцене;
- MonoBehaviour. базовый класс, от которого наследуется каждый скрипт Unity по умолчанию;
- Object. базовый класс для всех объектов, на которые Unity может ссылаться в редакторе;
- Transform. Предоставляет вам различные способы работы с положением, вращением и масштабированием GameObject с помощью скрипта, а также его иерархическую связь с родительскими и дочерними GameObjects;
- Vectors. Классы для выражения и управления 2D, 3D и 4D точками, линиями и направлениями;
- Quaternion. Класс, представляющий абсолютный или относительный поворот и предоставляющий методы для их создания и управления ими;

- ScriptableObject. Контейнер данных, который можно использовать для сохранения больших объемов данных;
- Time. Позволяет измерять и контролировать время, а также управлять частотой кадров вашего проекта;
- Mathf. Набор общих математических функций, включая тригонометрические, логарифмические и другие функции, обычно используемые в играх и разработке приложений;
- Random. Предоставляет вам простые способы генерации различных часто используемых типов случайных значений;
- Debug. позволяет визуализировать информацию в редакторе, которая может помочь вам понять или исследовать, что происходит в вашем проекте во время его выполнения;
- Gizmos и Handles. позволяет рисовать линии и фигуры в представлении «Сцена». Интерактивный взгляд на мир, который вы создаете. Вы используете Scene View для выбора и размещения пейзажей, персонажей, камер, источников света и всех других типов игровых объектов.

Выводы по главе 1

1. Проведен анализ предметной области разработки видеоигр.
2. Рассмотрены наиболее популярные средства разработки видеоигр (Unity, Unreal Engine, Defold, GameMaker Studio).
3. Проведен анализ наиболее популярных методов и методологий разработки программных продуктов, а также их применение в геймдеве.
4. Была рассмотрена среда разработки Unity как средство для разработки видеоигры. Рассмотрены основные спецификации; интерфейс, состоящий из окон проекта, сцены, иерархии, инспектора и панели инструментов; принцип работы, заключающийся в компонентно-ориентированном программировании; базовый функционал игровых объектов, компонентов и сценариев (скриптов).

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВИДЕОИГРЫ

В рамках проектирования разработки была выбрана гибридная модель проектирования и разработки с использованием методологии SCRUM ввиду наилучшей эффективности для небольшой команды разработки. Команда разработчиков включает 3 человек, отвечающих за разные компетенции (дизайн, звук, программирование, анимация).

Ввиду использования SCRUM разработчикам не требуется детально и четко описывать все функции и особенности будущего продукта. Задачи и функционал программы можно изменить между спринтами.

В ходе проектирования была составлена технологическая дорожная карта с использованием Google Таблицы (Приложение 1), составлен первый бэклог продукта (журнал требований продукта, список возможностей) (Приложение 2) с приоритетностью и сложностью задач.

2.1 Этап планирования проекта

На данном этапе осуществляется определение ряда сущностей и взаимосвязей, предназначенных для точного представления видеоигры

2.1.1 Концептуальная модель видеоигры

Концептуальная модель – первый и самый важный аспект создания игрового проекта. Основная цель этой модели – способствовать эффективному описанию видеоигры, эффективную организацию этих описаний.

При описании концептуальной модели были сформулированы требования к продукту:

- Жанр: MOBA;
- Режим игры: одиночный и многопользовательский;
- Графика: 3D;
- Камера: вид сверху;
- Цель: 1 – остаться последним уцелевшим на арене; 2 – добежать до финиша как можно раньше остальных игроков;
- Мир: фэнтези;
- Игровая валюта: кристаллы и слитки;
- Задания: ежедневные и еженедельные;
- Персонажи: набор персонажей разных рас с уникальными способностями и атрибутами;
- Средство разработки: Unity;
- Платформа: iOS, Android;
- Язык программирования: C#.

UML-диаграмма (рисунок 17)

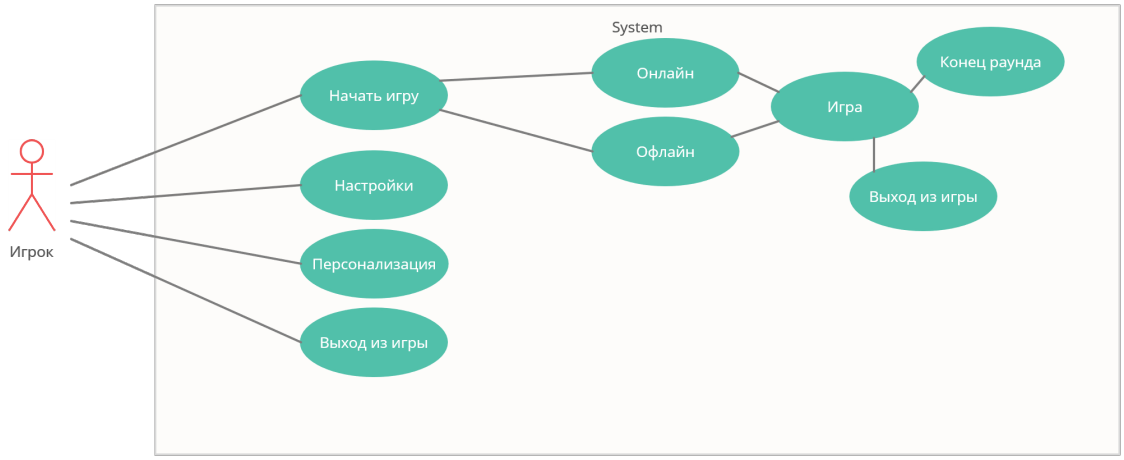


Рисунок 14 - Диаграмма прецедентов

2.1.2 Дизайн-документ видеоигры

Дизайн-документ (англ. game design document, часто используется аббревиатура «диздок») — это детальное описание разрабатываемой компьютерной игры.

Фрагмент дизайн-документа:

Описание игры

Основная задача игрока – выжить. В игре предусматриваются два игровых режима: арена и гонка.

В первом режиме игроки сражаются на круглой арене, которая в течение игрового раунда постепенно сужается. За пределами арены находится область, которая может оказаться смертельной, если игроки задержатся на ней надолго. Игроки вольны использовать стиль игры на свой вкус: могут играть агрессивно, пытаясь вытолкнуть противника за пределы арены, либо добить его, используя способности игрового персонажа; либо же играть оборонительно, уворачиваясь от атак противников и пытаться держаться в пределах арены.

Второй режим предлагает условия, в которых игрокам нельзя находиться на одном месте. Уровень представляет собой произвольную «трассу», по которой бегут игроки, состоящую из мелких блоков. При контакте с любым персонажем блок вскоре исчезает. По истечении времени на уровне появляется «финиш», добежав до которого игрок гарантирует себе выживание. Гонка предполагает игру в несколько раундов, по окончании которых подсчитываются очки каждого игрока за каждый «забег». Очки начисляются в зависимости от того, каким по счету прибежал на «финиш» игрок. Если игрок не добежал до финиша, или выбыл из раунда досрочно, первый не получает очков. В данном режиме также предусмотрено использование способностей, уникальных для каждого персонажа.

Сравнение и предпосылки создания

Игра “WQR” имеет, с одной стороны, некоторые оригинальные решения в жанре, но в то же время концепция игры использует следующие свойства выбранных образцов:

Fall guys: концепция «Царь горы» — это PvP-режим (игрок против игрока), где команды и одиночные игроки сражаются за место в рейтинге. Цель участников состязания — провести лучшую победную серию и прийти первым к финишной черте.

Brawl stars: цель игрока — продвигаться по игровой дороге, участвовать в боях с другими игроками, а также открывать и улучшать новых игровых персонажей с уникальными способностями и характеристиками.

Геймплей игры сосредоточен на том, чтобы в одиночку, командой из двоих человек или в кооперативе из троих или пяти человек победить команду других игроков, или противника под руководством ИИ, в разнообразных игровых режимах. Игроки могут выбрать персонажей, каждый из которых имеет свои навыки и суперспособность.

Warlock (Warlock Brawl): контролируйте свое местоположение, используйте магию и быстрые рефлексы, чтобы расправиться с противниками. Победите своих врагов и выживите.

Функциональная спецификация

Игровой мир и время

Действие происходит в вымышленном средневековом фентезийном мире. В игре присутствуют 4 расы: люди, орки, гномы, эльфы.

Жизненная и боевая модель

Каждый персонаж обладает определенным уровнем здоровья, которое может уменьшаться под действием атаки противника и/или нахождения в определенной зоне. Текущее здоровье персонажа отображается на экране.

Движение

Персонаж управляется под действием игрока с помощью джойстика в нижней левой части экрана смартфона.

Способности

Каждый игровой персонаж обладает уникальной способностью, при использовании которой тратится энергия. Кнопка активации способности находится в нижней правой стороне экрана смартфона. При нажатии на кнопку, игрок должен, не отпуская палец с экрана, выбрать направление использования способности.

Интерфейс пользователя

На рисунках 15, 16 схематически изображены основные экраны приложения. На рисунке 17 изображена схема, по которой организовано игровое меню

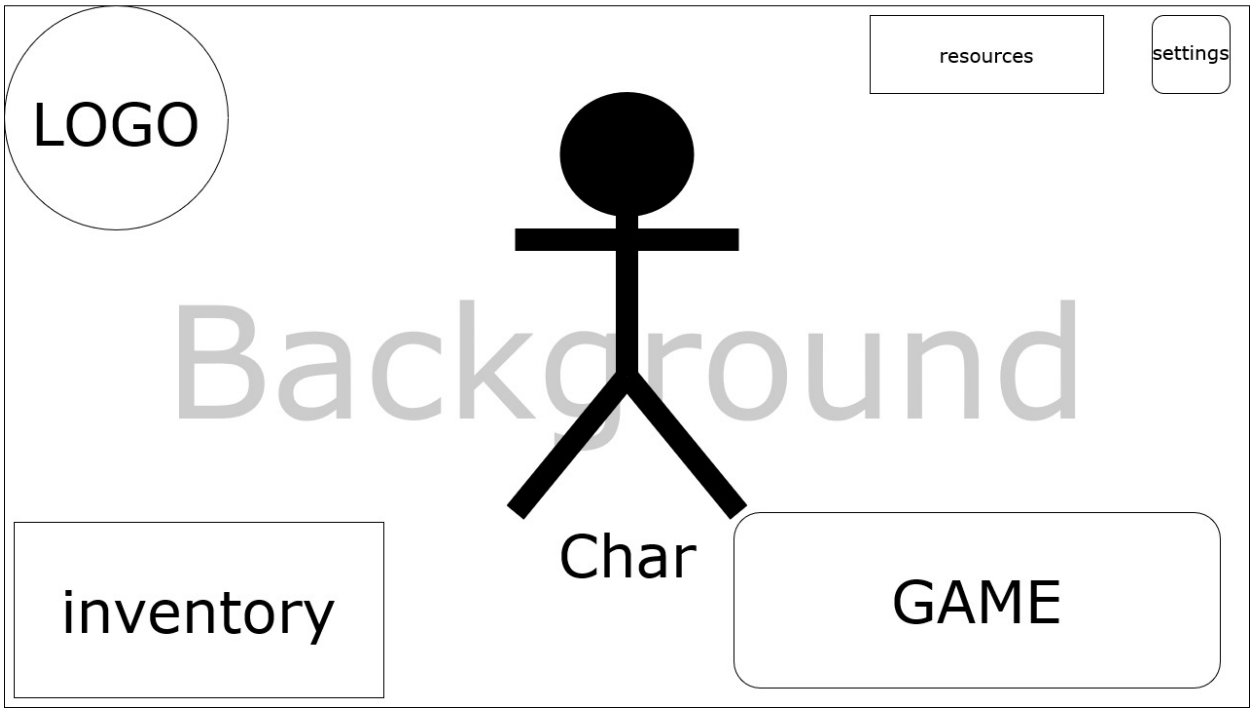


Рисунок 15 - главное меню

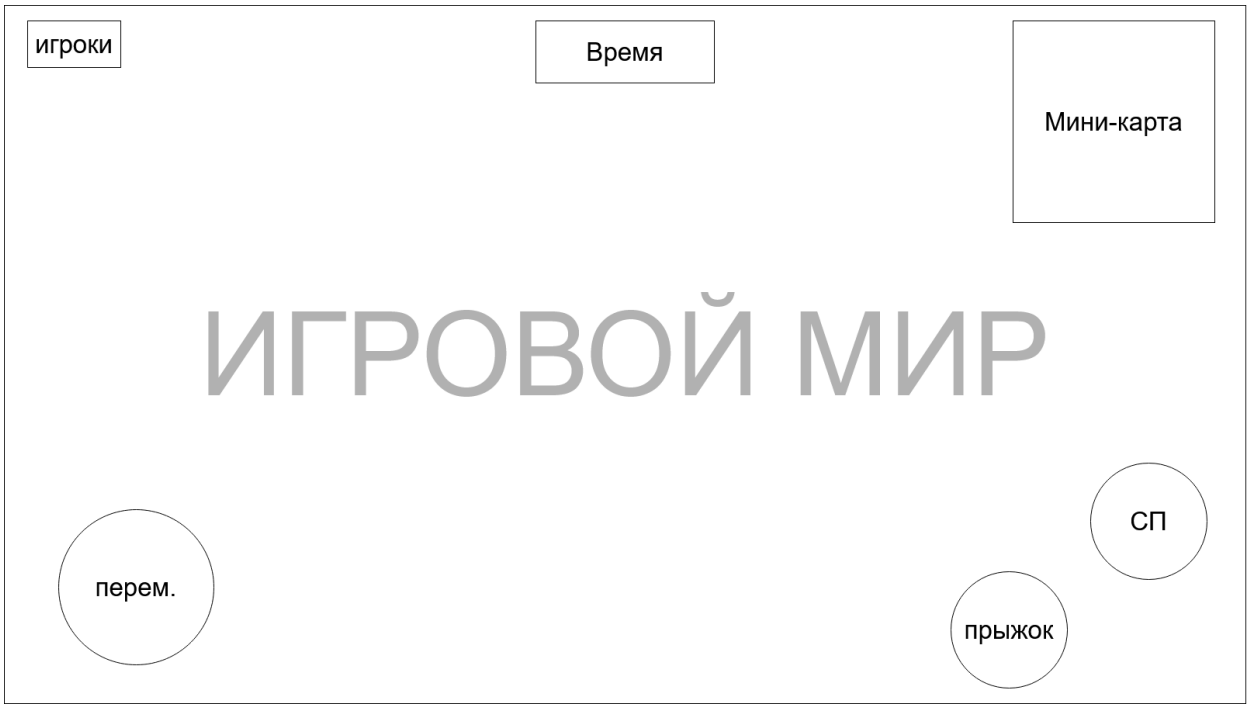


Рисунок 16 – игровой экран

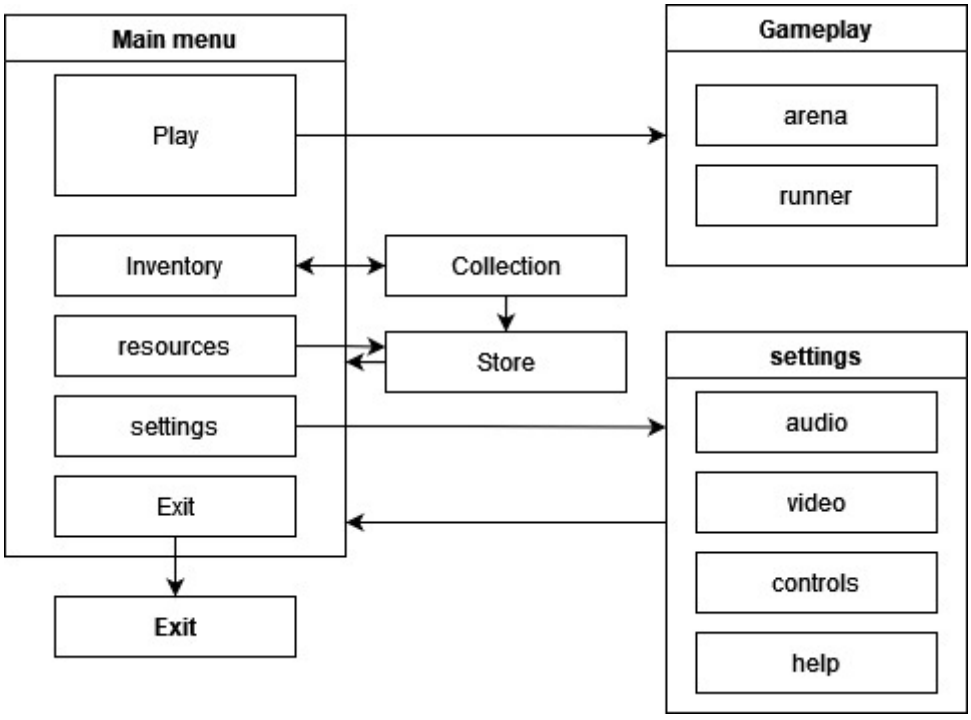


Рисунок 17 – структура главного меню

2.1.3 Выбор средства разработки

В качестве средства разработки была выбрана платформа разработки в реальном времени Unity, ввиду его простоты, гибкости и универсальности и других преимуществ, описанных в главе 1. Unity имеет низкий порог входа и отлично подходит для разработки в небольшой команде или команды, состоящей из одного разработчика. В сравнении с Unreal Engine, Unity использует язык C# (C++ для Unreal), более строгий и имеющий меньший порог входа по сравнению с C++.

2.2 Этап pre-production

2.2.1 Создание прототипа в Unity

На стадии подготовки к разработке был разработан прототип видеоигры, включающий следующие элементы:

- тестовый уровень;
- механика перемещения персонажа;
- механика использования способностей персонажа;
- механика здоровья персонажа и получение последним урона.

Реализованные сценарии представлены в приложениях 3-5.

2.3 Этап production

На стадии разработки продукта были затронуты следующие элементы:

- разработка уровня режима «арена»;
- разработан тестовый уровень режима «раннер»;
- разработан прототип интерфейса, описанный на этапе планирования продукта.

2.3.1 Создание уровня режима «Арена»

В рамках разработки уровня режима «арена» были реализованы следующие механики окружающей среды:

- сужение боевой платформы;
- нанесение урона персонажам, находящимися за пределами боевой платформы.

Реализованный код представлен в приложениях 6 и 7 соответственно.

На рисунке 18 изображена краткая диаграмма зависимостей объектов на Сцене:

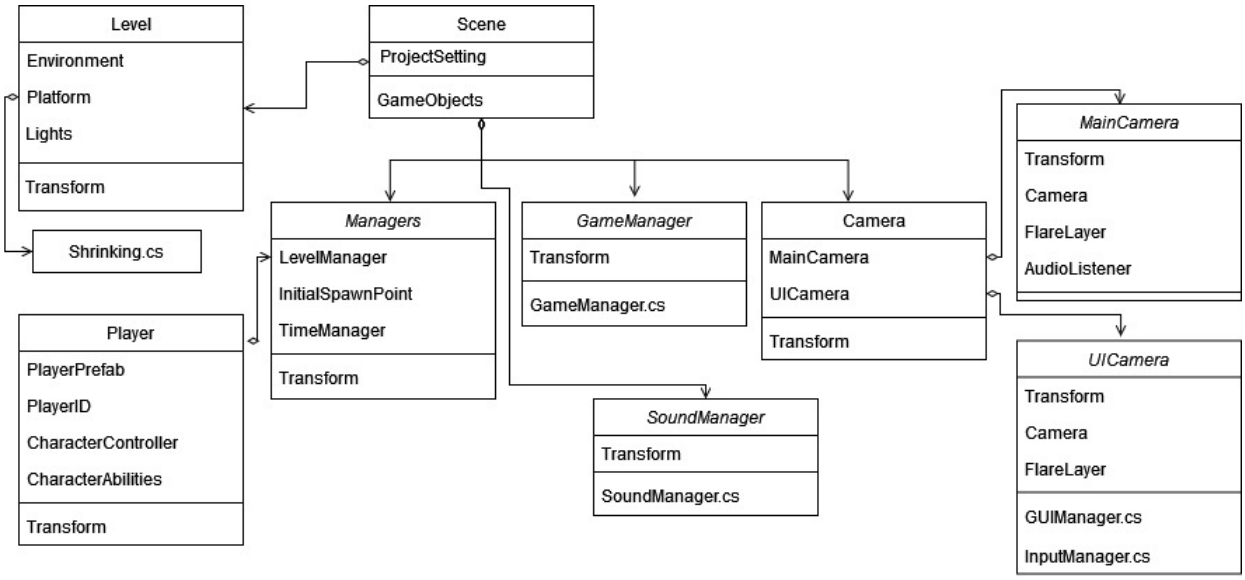


Рисунок 18

2.3.2 Создание уровня режима «Раннер»

В рамках разработки уровня режима «раннер» были реализованы следующие механики окружающей среды:

- удаление платформ при столкновении персонажа с первыми;
- генерация точки финиша на случайном месте на игровом уровне.

Реализованный код представлен в приложениях 8 и 9 соответственно.

На рисунке 19 изображена краткая диаграмма зависимостей объектов на Сцене:

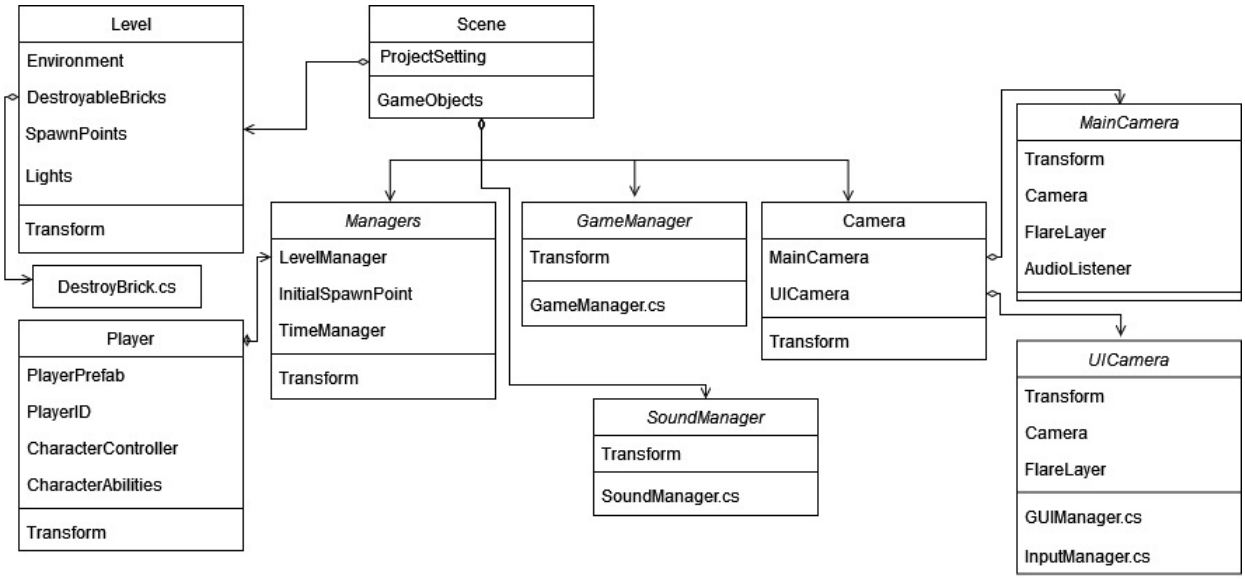


Рисунок 19

2.3.3 Разработка пользовательского интерфейса и меню

Реализованный код пользовательского интерфейса продемонстрирован в приложении 10

Выводы по главе 2

1. Пройден этап планирования видеоигры: создан концепт игры, выбрано средство разработки, предоставлен дизайн документ для дальнейшей разработки.
2. Разработан прототип игры в среде разработки Unity, демонстрирующий рабочие механики управления персонажем и его воздействия с окружающей игровой средой.
3. Разработаны макеты игровых уровней двух режимов игры, реализованы сценарные особенности уровней «арена» и «раннер».
4. Разработан пользовательский интерфейс.

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной выпускной квалификационной работы был проделан анализ предметной области в сфере игровой индустрии, изучены популярные методы и методологии разработки программных продуктов, а также проектирование и разработка видеоигры в среде разработки Unity. Результатом работы является рабочий прототип, включающий себя 2 основных режима игры с рабочими основными механиками, интерфейс и меню.

Для решения этой задачи было сделано следующее:

- Проведен анализ средств разработки видеоигр;
- Исследованы методы и методологии разработки программных продуктов;
- Изучен интерфейс и базовый функционал игрового движка Unity;
- Пройден этап планирования, включающий в себя:
    - Сформирована концептуальная модель игры;
    - Спроектирован дизайн-документ;
    - Построена технологическая дорожная карта проекта;
    - Составлен бэклог продукта;
- На стадии разработки созданы 2 уровня-прототипа, которые станут основой для полноценного готового коммерческого проекта.

Таким образом, поставленные задачи полностью выполнены, а следовательно, цель выпускной квалификационной работы выполнена.

СЛОВАРЬ ТЕРМИНОВ

Геймдев – разработка видеоигр.

Геймплей (gameplay) — игровой процесс в компьютерной игре.

Концепт-документ (концепт) – это краткое и ёмкое описание концепции (идеи) игры, то есть, максимально сжатый документ, в котором рассказывается о том, какой будет игра, чем она будет интересна и как она должна выглядеть после разработки.

Дизайн-документ (GDD, Game Design Document) – это описание игры максимально полное. Он позволяет разработчикам игры составить «план дальнейших действий» по воплощению задуманного проекта в проект реальный.

MOBA (Multiplayer Online Battle Arena) – многопользовательская боевая арена онлайн, то есть, боевая арена для многих игроков.

Скрипт (сценарий) — это последовательность действий, описанных с помощью скриптового языка программирования для автоматического выполнения определенных задач.

Интерфейс — форма, метод организации взаимодействия между отдельными системами; совокупность средств и правил, обеспечивающих взаимодействие отдельных систем.

Интерфейс пользователя — система средств для взаимодействия пользователя с системой, реализованная с помощью графических изображений и манипулирование программой через них.

Фэнтези — жанр современного искусства, разновидность фантастики. Фэнтези основывается на использовании мифологических и сказочных мотивов в современном виде.

Однопользовательская игра или одиночная игра́ — режим компьютерной игры, во время которого с ней через устройства ввода-вывода взаимодействует один человек.

Многопользовательская игра, или мультиплеер — режим компьютерной игры, во время которого играет больше одного человека.

Ежедневные/еженедельные задания - механика, которая дает игроку возможность выполнить конечное число определенных действий, которые обновляются после заданного промежутка времени.

БИБЛИОГРАФИЯ

1. Акт правительства Российской Федерации "План мероприятий ("дорожная карта") "Создание дополнительных условий для развития отрасли информационных технологий"" от 09.09.2021
2. Указ Президента Российской Федерации "«О мерах по обеспечению ускоренного развития отрасли информационных технологий в Российской Федерации»" от 2.03.2022
3. Аджич Гойко Impact Mapping. Как повысить эффективность программных продуктов и проектов по их разработке. Альпина Паблишер, 2017. - 88 с.
4. Арсеньев Ю. Н., Давыдова Т. Ю. Управление проектами, программами: учебник: в 2 томах. Том 1. Методология проектов . - Москва, Берлин: Директ-Медиа, 2021. - 473 с.
5. Дж. Г. Бонд Unity и C#. Геймдев от идеи до реализации. - 2 изд. - СПб: Питер, 2019. - 928 с.
6. Евдокимов П. В. C#. Практическое руководство. - СПб.: Наука и техника, 2022. - 416 с.
7. Ехлаков Ю. П. Управление программными проектами: учебник. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 217 с.
8. Иванова Г.С. Технология программирования. - Москва: КноРус, 2021. - 333 с.
9. Корнилов А. В. UNITY. Полное руководство. - 2-е изд. - М.: Букмастер, 2021. - 336 с.
10. Кушнер Дэвид Повелители DOOM. Как два парня создали культовый шутер и раскачали индустрию видеоигр. - М.: Бомбора, 2021. - 496 с.
11. Ларкович С. Н. Справочник UNITY. Кратко, быстро, под рукой. - СПб.: Наука и техника, 2020. - 288 с.

12. Савченко А. Игра как бизнес. От мечты до релиза. - М.: Бомбора, 2022. - 336 с.

13. Торн А. Искусство создания сценариев в Unity. - М.: ДМК Пресс, 2019. - 360 с.

14. Уточкин В. Н., Сахнов К. С. Хочу в геймдев! Основы игровой разработки для начинающих. - М.: Бомбора, 2022. - 224 с.

15. Ферроне Харрисон Изучаем C# через разработку игр на Unity. - 5-е изд. - СПб.: Прогресс книга, 2022. - 400 с.

16. Хокинг Дж. Unity — в действии. Мультиплатформенная разработка на C#. — 2 изд. — СПб: Питер, 2016. — 336 с.

17. Шелл Джесси Геймдизайн. Как создать игру, в которую будут играть все. - М.: Альпина Паблишер, 2022. - 640 с.

18. Исследование: какую роль образование играет в геймдеве? // App2Top URL: https://app2top.ru/analytics/kakuyu-rol-obrazovanie-igraet-v-gejmdeve-153176.html (дата обращения: 01.05.2022).

19. Компьютерные и видеоигры (российский рынок) // TAdviser URL: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8_%D0%B2%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D1%8B_(%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D1%8B%D0%BD%D0%BE%D0%BA) (дата обращения: 12.03.2022).

20. Патриоты седьмого уровня // Коммерсантъ URL: https://www.kommersant.ru/doc/4783772 (дата обращения: 01.05.2022).

21. Управление программными проектами: процессы, инструменты, методики // habr.com URL: https://habr.com/ru/post/286896

22. Управление программными проектами: процессы, инструменты, методики // project.dovidnyk.info URL:

https://project.dovidnyk.info/index.php/home/tehnologiyarazrabotkiprogrammnogoobespecheniya/22-upravlenieproektom

23. Aversa D., Dickinson C. Unity Game Optimization: Enhance and extend the performance of all aspects of your Unity games. - 3-е изд. - Бирмингем: Packt Publishing, 2019. - 404 с.

24. Rogers S. Level Up! The Guide to Great Video Game Design. - 2-е изд. - M.: John Wiley & Sons Limited, 2014. - 560 с.

25. Best Game Engines for 2022 – Which Should You Use? // GameDev Academy URL: https://gamedevacademy.org/best-game-engines/ (дата обращения: 01.03.2022).

26. CD PROJEKT updates its strategy // CDProject URL: https://www.cdprojekt.com/en/media/news/cd-projekt-updates-its-strategy/ (дата обращения: 01.05.2022).

27. Gaming & Esports: Media's Next Paradigm SHIFT // BCG URL: https://www.bcg.com/2021/gaming-and-esports-sector-are-the-next-shift-in-media (дата обращения: 01.03.2022).

28. The Definitive Guide to Project Management Methodologies // www.workamajig.com URL: https://www.workamajig.com/blog/project-management-methodologies

29. Unity (игровой движок) // wikipedia.org URL: https://ru.wikipedia.org/wiki/Unity_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)

30. Unity Tutorials: Learn how to bring your vision to life with Unity // YouTube URL: https://www.youtube.com/c/unity/playlists (дата обращения: 29.08.2021).

31. Unity User Manual // docs.unity3d.com URL: https://docs.unity3d.com/Manual/index.html

ПРИЛОЖЕНИЯ