Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

КОМПЬЮТЕРНАЯ ИГРА ИМИТИРУЮЩАЯ МОДЕЛЬ МАЛЫХ ЭКОНОМИЧЕСКИХ ГРУПП

Выпускная квалификационная работа
по направлению подготовки 09.03.02 Информационные системы
и технологии
профилю подготовки «Информационные технологии в медиаиндустрии»

Идентификационный номер ВКР: 303

Екатеринбург 2017

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующая кафедрой ИС
_____ Н. С. Толстова
«____» _____ 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

КОМПЬЮТЕРНАЯ ИГРА ИМИТИРУЮЩАЯ МОДЕЛЬ МАЛЫХ ЭКОНОМИЧЕСКИХ ГРУПП

Исполнитель:
обучающийся группы № ИТм-401 С. Э. Съедин
Руководитель:
ст. преподаватель И. А. Садчиков
Нормоконтролер: Б. А. Редькина

Екатеринбург 2017

АННОТАЦИЯ

Выпускная квалификационная работа состоит из игры «Garden Scramble» и пояснительной записки на 66 страницах, содержит 42 рисунка, 1 таблицу, 20 источников литературы.

Ключевые слова: игра, компьютерная игра, разработка, проект, инди-игра.

Съедин, С. Э. Компьютерная игра имитирующая модель малых экономических групп: выпускная квалификационная работа / С.Э. Съедин; Рос. гос. проф.-пед. ун-т, Ин-т инж. -пед. Образования, Каф. информ. систем и технологий. — Екатеринбург, 2017. — 66 с.

В работе рассмотрены вопросы разработки компьютерной игры.

Целью работы является создание компьютерной игры, имитирующей модель малых экономических групп. Для достижения цели были проанализированы современные тенденции в разработке компьютерных игр, исследована рыночная среда и компьютерные игры конкурентов. На основе проанализированных материалов и средств разработки, была разработана компьютерная игра «Garden Scramble».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня рынок видеоигр является одним из самых богатых в сфере развлечения. Каждый год создаются все более совершенные компьютерные системы, чтобы улучшить жизнь человека, а также занять его свободное время.

Компьютер стал незаменимым не только в сфере вычисления, но и является мощным инструментом для развлечения. Человек иногда даже не задумывается, что его смартфон так же является персональным компьютером. С ростом роли компьютера в жизни человека, он существенно влияет и на модель поведения человека. По последним исследованиям ученых средний возраст игрока компьютерных игр увеличился до 30 лет и выше.

Видеоигры делают жизнь человека ярче, насыщеннее и как следствие — эта экономическая сфера приносит огромные доходы разработчикам игр.

Особая роль в жизни человека отводится видеоиграм, первые из которых существовали уже в начале разработки компьютерной техники.

Видеоигры — это новый вид искусства, похожий на другие зрелищные жанры. Игры могут не только развлекать игрока, но и внести обучающий элемент, заставить игрока переживать, поднимать глобальные темы. Другими словами - игры - это современный вид искусства. Видеоигры дарят порой такие эмоции, которые не сравнить с фильмом или театральной постановкой.

Над играми работают большие коллективы программистов, сценарии к играм пишут ничуть не хуже, чем к фильмам. Видеоигры стали важной экономической составляющей. Прибыль от ААА — проектов приносит огромные деньги их создателям.

Самые нашумевшие ААА — проекты 2012 года выпуска, принесли их создателям в общей сумме 7 млрд. долларов за один год. Ниже приведена таблица (таблица 1) с количеством проданных копий игр, выпущенных в 2012 году [9].

Таблица 1 — Количество проданных копий AAA - проектов

№ п.п.	Название игры	Компания разработчик	Продано копий игры
1.	Assassin’s Creed III	Ubisoft	12 000 000 копий
2.	Borderlands 2	Gearbox Software	12 000 000 копий
3.	Call of Duty: Black Ops II	Treyarch	25 000 000 копий
4.	Diablo III	Blizzard Entertainment	За первые 24 часа было продано 3 500 000 копий, установив рекорд по скорости продаж среди ПК игр. Продано 30 000 000 копий
5.	Far Cry 3	Ubisoft Montreal	9 000 000 копий

Но в мире востребованы не только супертехнологичные видеоигры. Большое количество программистов, являются инди-разработчиками создающие игровые продукты, которые не обладают современной дорогой графикой и звуковым сопровождением, но обладают уникальным звуком и графикой, зачастую несут инновации в механике игр. В итоге они так же пользуются огромной популярностью и приносят разработчикам доходы.

В мире существует большое количество геймеров с различным опытом в сфере игровой индустрии. Есть любители игр, есть ностальгирующие игроки, поэтому на рынке востребованы проекты разной направленности.

Объект исследования: компьютерные игры.

Предмет исследования: разработка компьютерной игры.

Цель: разработка игры имитирующая модель рыночной экономики в условиях малых групп.

Задачи:

- изучить игровой движок Unity;
- изучить стадии разработки игр;
- проанализировать аналоги;
- разработка игровой механики;
- тестирование.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ и общая характеристика предметной области

Разработка компьютерных игр — процесс создания компьютерных игр [20].

Разработкой видеоигр занимается как один программист, так и команда. Часто коммерческие игры создаются командами разработчиков, нанятыми одной фирмой. Разработчики видеоигр могут специализироваться на создании игр для компьютера, игровых приставок или мобильных устройств. Часто разработка финансируется другой, более крупной фирмой — издателем. За частую именно издатель занимается распространением игры, издатель может так же нанять команду разработчиков, или разработчик самостоятельно распространяет игру, например, средствами цифровой дистрибуции Steam.

В состав команды разработчиков видеоигр требуются различные специальности:

- один или несколько продюсеров для наблюдения за производством;
- геймдизайнеры, или создатели;
- художники;
- звукооператоры;
- тестировщики.

Геймдизайнер — специалист, отвечающий за разработку дизайна игры (правила, содержание, игровой процесс, целостное видение игры). Роль геймдизайнера сопоставима с ролью режиссера в кино.

Художник — специалист, отвечающий за визуальную часть игры (спрайты, анимации, концепт-арты) [10].

Звукооператор — специалист, отвечающий за звуковое оформление игры (музыка, звуки). Без музыки и озвучки, видеоигра вряд ли будет успешна.

Тестировщик — специалист, отвечающий за поиск ошибок на стадии тестирования. Тестирование проекта является очень важным в процессе создания игр зачастую тестеровщиков нанимают из числа целевой аудитории проекта.

Разработку каждой игры можно разделить на несколько этапов:

- подготовительный этап (pre-production);
- производство (production);
- выпуск (reliese);
- поддержка.

Подготовительный этап — это первый этап работы над игрой. Задача разработчиков на этом этапе, разработать концепцию игры, дизайн персонажей, выбрать средства для реализации проекта (игровой движок), создать прототип игры, подготовить план, по которому будет создаваться игра и согласовать этот план с начальством, либо с компанией, которая планирует издавать игру. Как правило, все современные игры пишутся под конкретного издателя, который часто вкладывает в разработку немалые средства. На данном этапе командой создается план разработки игры, именуемый как «дизайн-документ». Дизайн-документ — это детальное описание разрабатываемой компьютерной игры. Игровой движок — это программное обеспечение отвечающие за обработку написанного кода и других компонентов игры в реальном времени. Движок дает возможность управлять основными технологиями, упрощает разработку видеоигры, так же дает возможность портирования игры на различные платформы, такие как игровые консоли и персональные компьютеры. Функционал, который дает игровой движок, позволяет визуализировать игру в режиме реального времени, обрабатывать физику объектов, воспроизводить звук, исполнять скрипты, воспроизводить анимацию, искусственный интеллект, исполнять сетевой код. Во время процесса разработки можно сэкономить большое количество денежных средств за счет использование одного движка повторно, но со временем они устаревают, поэтому выпускаются новые версии движков.

Производство — это ключевой этап в создании игры. Разработчики занимаются реализацией ранее созданного плана. Первоначальный план игры постоянно подвергается различным изменениям, изменения происходят каждый день и приходиться изменять все на ходу.

В ходе производства видеоигры — выполняется так называемый «вертикальный срез», зачастую это происходит у коммерческих проектов, к которым команда должна представить проект, называемый «прототип». То есть, к такому моменту должна быть готовая демоверсия, к другому первый уровень и так далее. Как правило, эти промежуточные результаты служат отличной рекламой новых игровых проектов демоверсии публикуют на игровых сайтах, геймеры «примеряют» к этим версиям возможности своего оборудования.

Выпуск — это этап разработки наступаемый после того как игра была окончательно закончена, протестирована и отлажена. Начинается издание игрового проекта. В этом этапе так же запускается маркетинговая компания по привлечению целевой аудитории, которая и будет покупать копии игры.

Поддержка — игры для ПК часто выходят с ошибками, все дело в том, что разработчикам вечно не хватает времени чтобы все как следует отладить. Поэтому всегда есть возможность устранить найденные ошибки и выпустить патч (от английского «patch» — заплатка) для игры. Зачастую это доходит до такого состояния, что разработчики, выпуская сырой продукт, патчат игру каждую неделю. Однако такое невозможно среди консольных игр, так как после выпуска разработчик не имеет возможности влиять на продукт, поэтому разработчик старается исправить все ошибки до стадии выпуска игры [16].

1.2 Анализ существующих разработок (выявление достоинств, недостатков, функциональных элементов)

Stardew Valley (рисунок 1) — это симулятор фермы с элементами RPG (отыгрыш роли), созданной Эриком Бароном и издательством Chucklefish Games. Stradew Valley была разработана под влиянием Harvest Moon. Игра разработана на игровом движке Gamemaker с помощью фреймворка Microsoft XNA, на языке C#. В начале игры, игрок создает своего персонажа, получая в наследство землю в маленьком городке, который называется Stardew Valley. На участке игрока находиться большое количество камней, травы, пней, деревьев, которые игрок должен расчистить, для того чтобы привести ферму в порядок.

Рисунок 1 — Игра Stardew Valley

Игрок может сажать различные растения используя семена, может сражаться с монстрами в шахте и добывая полезные ископаемые, может ловить рыбу в море, улучшать свои отношения с соседями, продавать/покупать различные товары, инструменты, растения и так далее (рисунок 2).

Рисунок 2 — Геймплей игры

В игре присутствует интерфейс, с которым игрок может взаимодействовать, экипируя на себя инструменты.

Достоинства:

- визуальная составляющая;
- механика игры;
- звуковая составляющая;
- сюжет;
- понятное управление.

Недостатки:

- нет рыночной экономики в срезе малых групп;
- отсутствует конкуренции среди жителей деревни (других фермеров);
- менее функциональный игровой движок.

Harvest Moon (рисунок 3) — японская видеоигра, симулирует жизнь игрока на ферме. Разработана геймдизайнером Ясухиро Вадой и студией Victor Interactive Software.

Игрок управляет фермером, занимается земледелием и животноводством, продает/покупает продукты, товары, строит и улучшает различные сооружения. Важно отметить, что помимо развития фермы игроку предоставляется возможность отыгрыша роли, героя окружают различные другие персонажи, и от общения с ними зависит успех в прохождении игры – игрок может жениться и завести своих детей, которые будут помогать ему работать на

участке. Так же происходит улучшение инструментов за счет навыка владения ими.

Рисунок 3 — Игра Harvest Moon

Игра начинается с того, что игрок получает во владение старую заброшенную ферму и теперь должен восстановить её практические с нуля. События происходят в реальном времени, в дневное время суток игрок должен успеть сделать все дела, которые он запланировал, поскольку работа ночью сбивает режим и отрицательно сказывается на его здоровье. Работая, персонаж тратит энергию, которую нужно восстанавливать с помощью горячих источников или приемом пищи, иначе фермер может потерять сознание и вынужден будет провести целый день в больнице. Ферма обычно находиться внутри деревни (рисунок 4), где живут другие персонажи, есть магазин, кузница, церковь, библиотека и прочие места.

Рисунок 4 — Деревня в Harvest Moon

В деревне находиться большое количество персонажей, которые имеют свое мировоззрение, заботы, распорядок дня, в светлое время они занимаются своими делами, вечером идут отдыхать в бар.

Достоинства:

- механика игры;
- звуковое сопровождение;
- сюжетная составляющая.

Недостатки:

- нет рыночной экономки в срезе малых групп;
- отсутствует конкуренции среди жителей деревни (других фермеров);
- старый игровой движок;
- достаточно старая игра, 1999 года выпуска.

Таким образом игры Harvest Moon и Stardew Valley можно называть идейными вдохновителями. Добавив своих новшеств в виде рыночной экономики и конкуренции среди жителей деревни. Можно получить достаточно интересную и захватывающую игру.

1.3 Анализ средств разработки и обоснование выбора технологии проектирования для всех элементов проекта

Проект включает в себя большое количество аспектов:

- программный код;
- визуальная часть (спрайты, анимации);
- звуковая часть (звуки, музыка);
- игровой движок.

Каждая разрабатываемая игра нуждается в игровом движке, поэтому это самая важная часть, в которой нужно выбрать лучший и подходящий игровой движок. В данный момент на рынке существует большое количество

игровых движков, но подходящие для проекта, это: Unreal Engine, Unity, Game Maker.

Unreal Engine (рисунок 5) — это игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первая игра, созданная на этом движке — Unreal, вышла в 1998 году. С тех пор различные версии движка были использованы в сотнях игр и других проектов. На данный момент этот движок можно назвать самым прогрессивным, но имеющим большой порог вхождения. Написан он на языке программирования C++, движок позволяет создавать для большинства ОС и платформ. Существует несколько версий движка: Unreal Engine 1, Unreal Engine 2, Unreal Engine 3, Unreal Engine 4. Четвертая версия является в данный момент самой последней и это первая бесплатная версия. Однако, разработчики, как и прежде, должны передавать 5% от прибыли игры компании Epic Games, но при условии, что доходы от игры составляют $3000 за квартал.

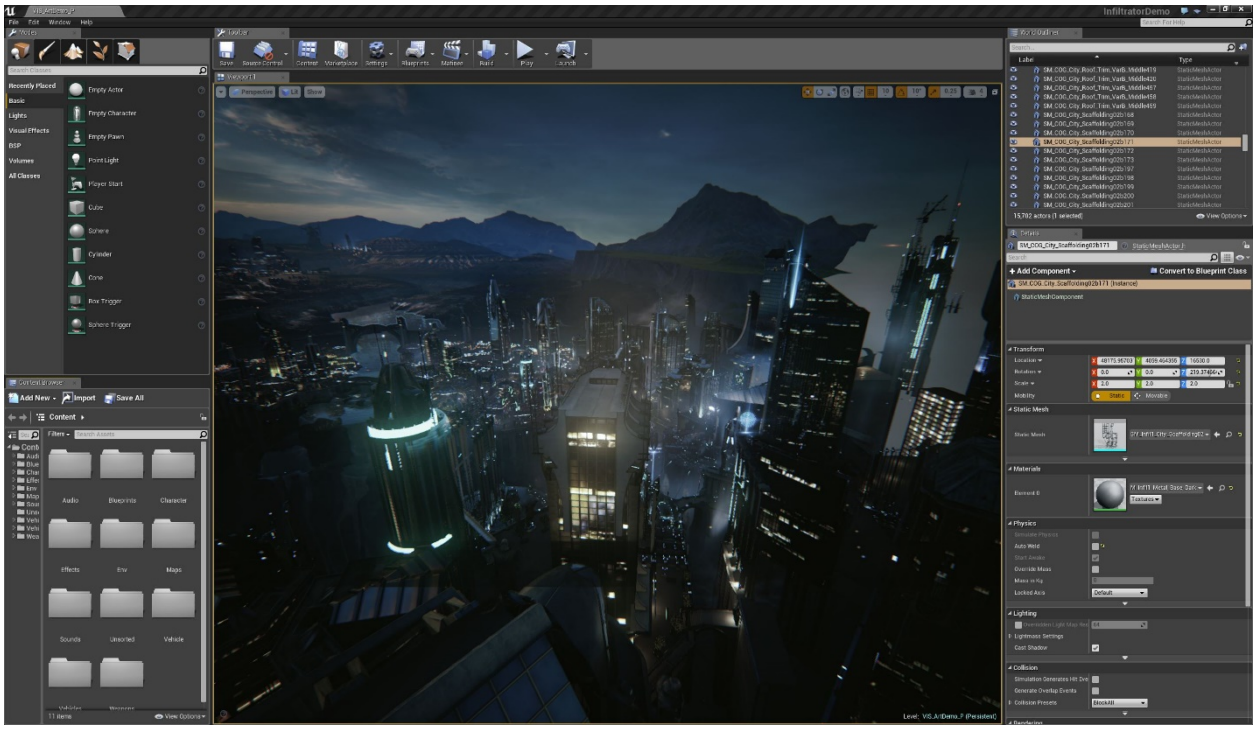

Рисунок 5 — Интерфейс игрового движка Unreal Engine 4

Game maker (рисунок 6) — один из самых известных конструкторов игр. Написан на языке программирования Delphi. Данная система рассчитана для создания двухмерных игр любого жанра. Низкий порог вхождения, по-

этому может быть рекомендован для изучения программирования и разработки игр. На данный момент последняя версия Game maker 8.1.

Создание игры на данном движке не требует знакомства с каким-либо языком программирования. Интерфейс объединяет в себе редакторы спрайтов, объектов, комнат, скриптов. В данный момент game maker распространяется на основе PTU («Pay to Use» - плати, чтобы использовать), Windows версию можно купить за 100$, так же можно использовать ограниченную версию программы, но возможность издательства игры ограничена. Скорость разработки даже при скромных знаниях и минимальной мотивации субъективно быстрее, чем на других движках. Установка и настройка для начинающих максимально проста и не требует особых знаний. Компиляция под другие платформы не требует смены кода игры и осуществляется одним кликом.

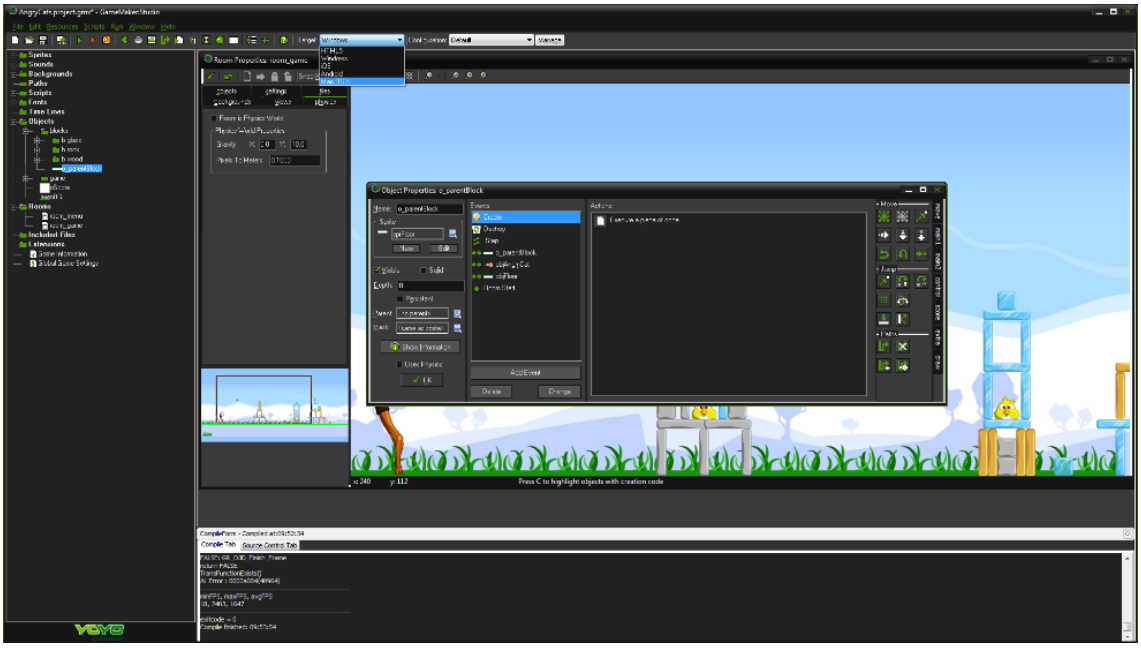

Рисунок 6 — Интерфейс игрового движка Game maker: Studio

Unity (рисунок 7) — это инструмент разработки двух- и трехмерных приложений и игр [18]. Созданные приложения и игры на Unity работают на большинстве операционных систем. Игровой движок имеет простой Drag&Drop интерфейс, который можно легко настроить под свои нужды. Состоит из различных окон, благодаря чему отладку игры можно производить прямо в редакторе. Движок поддерживает три языка программирования: C#,

JavaScript, Boo. Расчёты физик проводятся с помощью технологий PhysX от компании Nvidia. Средний порог вхождения. Этот игровой движок подходить идеально для тех, кто только начал разрабатывать компьютерные игры и имеет знания в языках программирования.

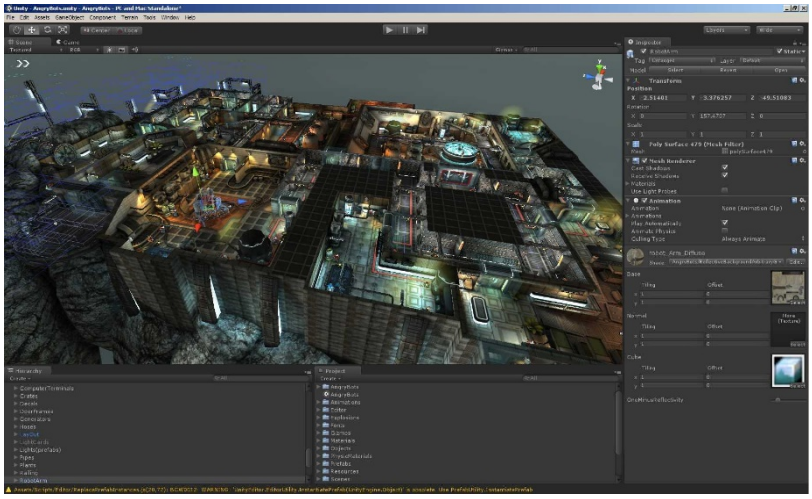

Рисунок 7 — Интерфейс игрового движка Unity 5.0

Так же в Unity присутствует инструментарий для совместной разработки, с помощью контроля версий. В данный момент последняя версия Unity 5.0. Есть возможность бесплатного пользования игровым движком, до тех пор, пока доход от опубликованной игры не превышает 100 000$ в год.

Для реализации проекта был выбран игровой движок Unity Engine.

Визуальная часть игры один из важнейших компонентов разрабатываемой игры, существует большое множество графических редакторов, но существует один уникальный который совмещает в себе все функции всех графических редакторов и достаточно легкий в изучении.

Photoshop (рисунок 8)— многофункциональный графический редактор, разрабатываемый фирмой Adobe Systems. В основном данный программный продукт работает с растровыми изображениями, однако имеет и векторные инструменты. Программа распространяется по системе PTU («Pay to Use» - плати, чтобы использовать). В проекте данная программа используется для рисования спрайтов персонажа, машин, мира, анимаций.

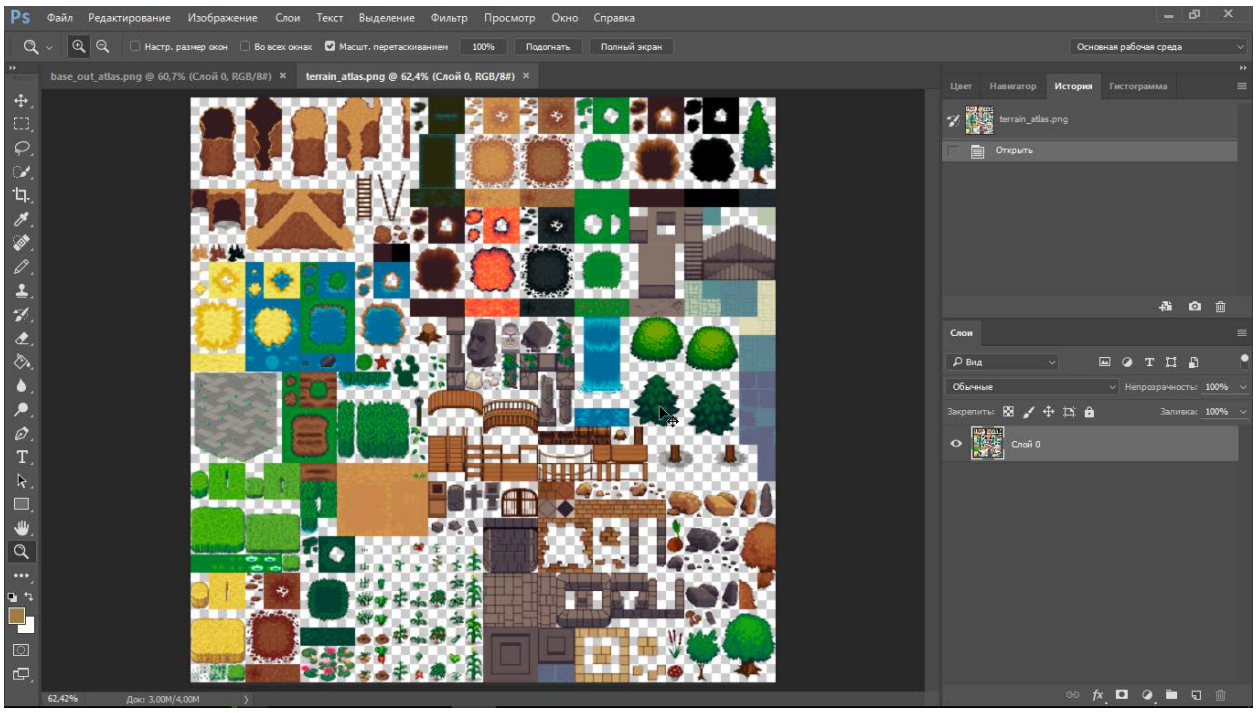

Рисунок 8 — Интерфейс графического редактора Adobe Photoshop

Проект использует двухмерную графику, чтобы легко создать мир можно использовать дополнительное программное обеспечение с интеграцией в игровой движок Unity. Этим обеспечением является Tiled2D.

Tile2D (рисунок 9) — это редактор карт, используемый для построения карты мира в двухмерной игре по средствам нанесения тайлов на каждый квадрат, таким образом получается мир игры [13].

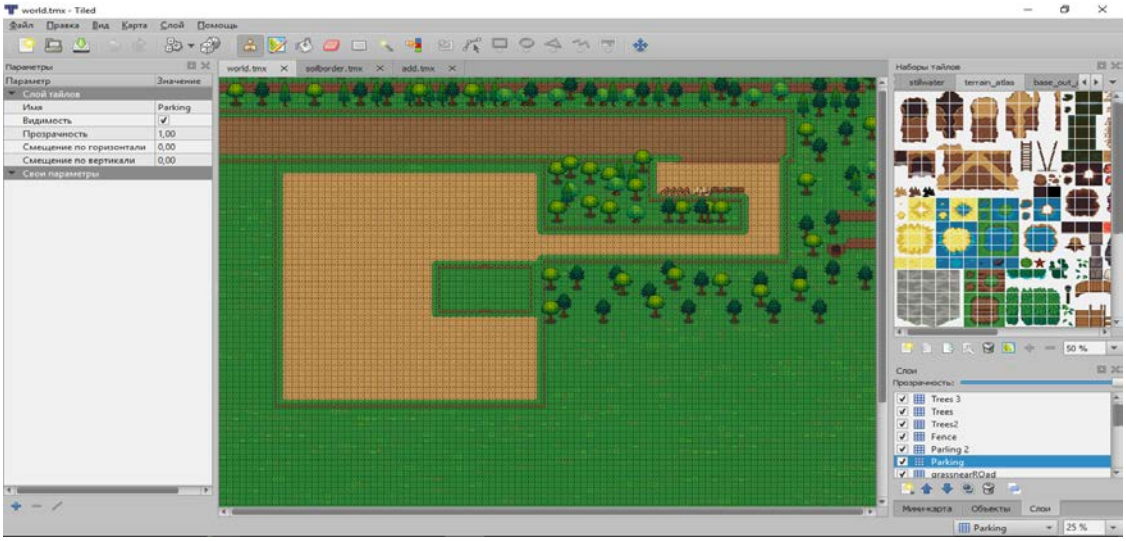

Рисунок 9 — Интерфейс Tiled2D

Для реализации программного кода используется Visual Studio 2015, который имеет возможность интеграции в среду Unity.

Visual Studio 2015 (рисунок 10) — программный продукт Microsoft, который включает в себя интегрированную среду разработки программного обеспечения и другие инструментальные средства [1]. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода [15]. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server) [14].

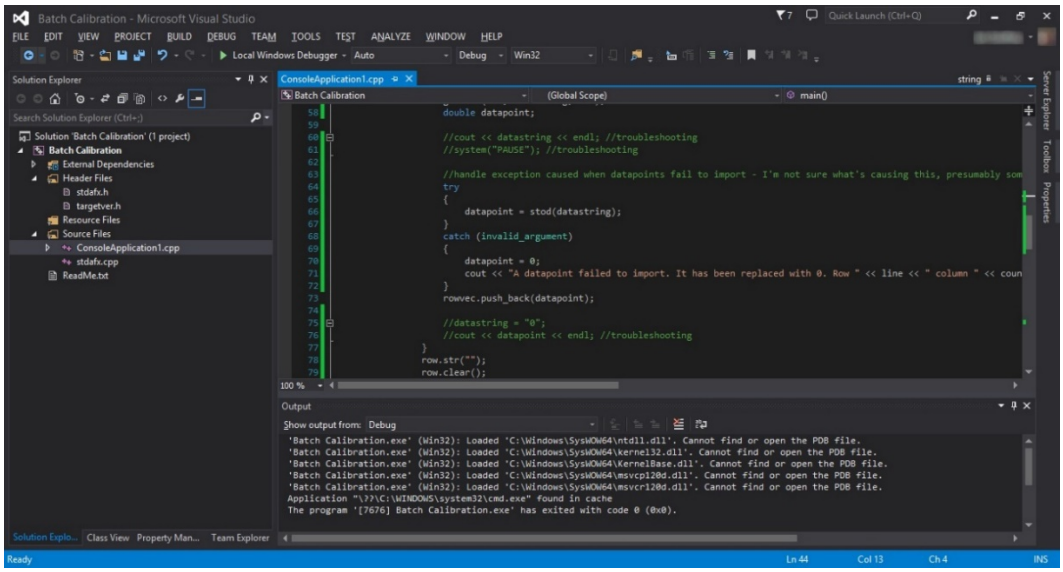

Рисунок 10 — Интерфейс Visual Studio 15

1.4 Общий алгоритм реализации проекта

- создание идеи игры;
- реализация системы инвентаря;
- создание базы данных предметов;
- реализация панели инструментов;
- реализация управления персонажа и его характеристик;
- создание спрайта персонажа и его анимации;
- реализация садовых грядок с растениями;
- реализация экономической системы, покупки/продажи товаров;
- создание остальных спрайтов (машины, автобуса, дома, карты мира и т.д.).

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Характеристика потенциальной аудитории потребителей проекта

Описание целевой аудитории важная часть любого проекта. Составление такого описания может помочь в понимании целевой аудитории и её нужд для разработчика. Поэтому для проекта была описана его целевая аудитория.

Общий уровень: подростки от 11 до 19 лет, отсутствует доход или имеют низкий уровень дохода. Жители как крупных, так и мелких городов. Занимаются учебой, работой.

Ключевые ценности: все свое свободное время предпочитают проводить за компьютером играя в игры, либо сидеть в интернете. Готовы пробовать/узнавать что-то новое.

Психографические характеристики: ведут пассивный образ жизни. Зачастую свое свободное время они проводят дома с семьей. По характеру совершения покупок — привычное покупательское поведение. Покупают как популярные игровые проекты, так и инди-разработки. Могут совершать импульсивные покупки. В покупках доверяют мнению друзей. Имеют хобби, но ему уделяют малую часть своего свободного времени.

Уровень товарной категории: покупать игры предпочитают сразу после выхода их на рынок, а за частую даже перед (пред заказ). Для осуществления выбора пользуются различными источниками информации: друзья, обзоры, отзывы. Покупают игры на платформах цифровой дистрибуции (Steam, Origin). Покупают игры известных разработчиков, но если отзывы или обзоры об игре хорошие, то они могут купить игру и у не известного разработчика.

Отношение к компании разработчику: если предыдущие проекты компании их зацепили, то они продолжают покупать их игры. Опыт общения с инди-разработчиками ограничивается на 1-2 случайных покупки.

Ориентированность работы. Самым ценным в деятельности человека, независимо от сферы его интересов, всегда является результат.

Эффективность результата можно оценивать по данным критериям:

- результат достигнут за время, которое планировалось;
- на достижение цели потрачено то количество ресурсов, которое планировалось;
- достигнут задуманный результат.

При этом для любого человека, который хочет работать продуктивно, важно знать, что не важно по какой причине произошел срыв сроков, а в первую очередь как выйти из данной ситуации и достигнуть результата.

Преимущества данного подхода:

- удобство диагностирования проблем, возникающих в ходе работы;
- быстрота устранения проблем;
- повышения уровня ответственности;
- повышения интереса к деятельности.

2.2 Постановка задачи проекта

2.2.1 Актуальность проекта

Начиная с двухтысячного года, в игровой индустрии появился новый жанр компьютерных игр, который называется инди-игры. Недавно этот жанр получил высокую популярность среди пользователей, в связи с более доступными средствами разработки и дистрибуции. Несмотря на высокую сложность и риск создания и сложность самостоятельной продажи, инди-игры очень популярные, а количество разработчиков игр данного жанра растет с каждым днем.

Название инди (от англ. Independent — независимый) дает понять каким образом разрабатываются данные игры. Точного определения таким играм еще нет, но инди-играми считаются те, которые созданы независимым разработчиком, а иногда и группой разработчиков [17].

Как правило, стартового бюджета у этих игр буквально нет, а если он и есть, то он катастрофически мал. Чаще всего деньги собирают с пожертвований на разных специальных сайтах (Kickstarter), поэтому создание проекта напрямую зависит от комьюнити — сообщества игроков, заинтересовавшихся в игре.

Инди-играм довольно сложно конкурировать с другими, особенно с высокобюджетными проектами, поэтому единственным шансом и главной фишкой многих разработчиков является инновация. Создатель может активно практиковать разные вещи, а если ему угодно, он может создать полностью бессмысленную и смешную игру.

Игры, в которые вкладывают много денег, просто не могут настолько сильно рисковать, в то время как инди даже не наказывает, а поощряет эти риски.

Инди-игры не очень популярны у нас, в сравнении с Европой и даже Азией. По большей части самым непроходимым барьером является языковой. Разработчики просто-напросто не способны создать перевод или локализацию на другой язык для своей игры.

Главной рекламной и соревновательной точкой среди разработчиков являются ежегодные встречи, как например Indie Game Jam, где лучшие Инди-игры представлены и показаны, а некоторые можно даже опробовать.

Отсутствие бюджета является не только проблемой, но и особенностью жанра. Дело в том, что разработчики могут очень быстро и просто создавать маленькие инди-игры, а комьюнити, дав оценку, может помочь советом, а иногда — даже выступить в роли волонтера и поддержать игру как своими деньгами, так и услугами.

Такие игры очень легко переделывать, дополнять или кардинально изменять в кротчайшие сроки за минимальное количество сил и средств.

Инди-игры очень сложно прорекламировать и продать, заинтересовать большую группу людей и удовлетворить её — ещё сложнее. Тем не менее, было создано очень много успешных игр, которые смогли победить даже высокобюджетных.

Самые популярные Инди игры:

- Hotline Miami;
- Yandere Simulator;
- Robocraft;
- Papers, Please;
- Black Mesa.

2.2.2 Цель и назначение проекта (модулей, сервисов, продуктов, приложений)

Цель проекта, внесение вклада в разработку компьютерных игр в жанре инди, создание инновационного геймплея, графики. Популяризация игр в жанре инди у игроков, развитие организаторских способностей, понимание игроками рыночной экономики в условиях малых групп, удовлетворения нужд целевой аудитории, составления портфолио, возможность продажи игры в системе цифровой дистрибуции. Для многих игроков инди-игры становятся хобби, которым они уделяют свое свободное время.

Инди-разработчик развивает самостоятельность. Таким образом разработчик полностью самостоятельно и никому не подчиняясь, реализует свой проект.

2.2.3 Функционал проекта, интерфейс проекта, эргономические и системные требования

Функционал. Модуль отвечающий за передвижение игрока, с помощью клавиш W, A, S, D или с помощью стрелочек. Модуль базы данных, в которой хранятся предметы, с которыми может взаимодействовать игрок, и их характеристики.

```
public List<Item> items = new List<Item>();
void Start ()
     {
        items.Add(new Item("Raw meat", 0, "Raw Meat. Don't eat
it!", 1,0, Item.ItemType.Food));
        items.Add(new Item("Cooked meat", 1,"Grilled Meat. That
so delicious!", 1,0, Item.ItemType.Food));
        items.Add(new  Item("Empty  bottle",  2,  "Empty  Bottle.
Don't break it!", 1,0, Item.ItemType.Head));
        items.Add(new Item("Branch",  3,  "Wooden branch!",  1,0,
Item.ItemType.Resource));
        items.Add(new  Item("Stone",  4,  "Common  stone!",  1,0,
Item.ItemType.Resource));
        items.Add(new Item("Stoneaxe", 5, "Can chop tree!", 1,0,
Item.ItemType.Tool));
        items.Add(new Item("Tomato seeds", 6, "You can plant it
to plowed area!", 10,2, Item.ItemType.Seeds));
        items.Add(new  Item("Tomato",  7,  "Fresh  Tomato.  You  can
sell it!", 1,4, Item.ItemType.Food));
        items.Add(new  Item("Wheat",  8,  "Wheat.  You  can  feed
chickens and cows!", 30,4, Item.ItemType.Fooder));
        items.Add(new Item("Wheat seeds", 9, "You can plant it
to plowed area!", 10,2, Item.ItemType.Seeds));
        items.Add(new Item("Pickaxe", 10, "Pickaxe be careful it
sharp!", 1,10, Item.ItemType.Tool));
        items.Add(new  Item("Watering  can",  11,  "Plants  grow
faster if you pour them!", 1,10, Item.ItemType.Tool));
        items.Add(new Item("Hoe", 12, "You can clean the weeds
by Hoe!", 1,10, Item.ItemType.Tool));
        items.Add(new  Item("Scythe",  13,  "You  can  harvest
wheat!", 1,10, Item.ItemType.Tool));
        items.Add(new  Item("Egg",  14,  "Ordinary  egg",  1,5,
Item.ItemType.Food));
      }
```

Модуль инвентаря для хранения, использования внутри игровых предметов таки как: семена, еда, овощи, товары, инструменты.

```
void Start () {
        int slotAmount = 0;
        int x = -90;
```

```
        int y = 90;
        database                    =                    GameOb-
ject.FindGameObjectWithTag("ItemDatabase").GetComponent<ItemData
base>();

            for(int i=1;i<5;i++)
        {
            for(int k=1;k<5;k++)
            {
                GameObject slot=(GameObject)Instantiate(slots);
                slot.GetComponent<SlotScript>().slotNumber      =
slotAmount;
                Slots.Add(slot);

                Items.Add(new Item());

slot.transform.SetParent(this.gameObject.transform,false);
                slot.name = "Slot" + i + "." + k;
                slot.GetComponent<RectTransform>().localPosition
= new Vector3(x, y, 0);
                x = x+60;
                if(k == 4)
                {
                    x = -90;
                    y = y -60;
                }
                slotAmount++;
            }
       }

     }
```

Модуль панели инструментов на которую игрок может вынести самые нужные вещи: семена, инструменты, еду, для быстрого доступа.

```
public void OnPointerDown(PointerEventData data)
    {
        inventory                   =                    GameOb-
ject.FindGameObjectWithTag("Inventory").GetComponent<Inventory>(
);
        if (inventory.draggingItem)
            {
                if (item.itemType != Item.ItemType.None)
                {
                    Item temp = item;
                    item = inventory.draggedItem;
                    inventory.draggedItem = temp;
                    inventory.ShowDraggedItem(temp, -1);
                }
                else
                {
                    item = inventory.draggedItem;
                    inventory.closeDraggedItem();
                }
```

```
            }
        }
        public void OnDrag(PointerEventData data)
        {
            inventory                       =                       GameOb-
ject.FindGameObjectWithTag("Inventory").GetComponent<Inventory>(
);
            if (inventory != null)
            {
                if (item.itemType != Item.ItemType.None)
                {
                    inventory.draggedItem = item;
                    inventory.ShowDraggedItem(item, -1);
                    item = new Item();
                }
            }
             }
```

С помощью инструментов можно взаимодействовать с миром игры, например, киркой можно уничтожать камни, которые находятся на участке главного героя, мотыгой полоть землю для дальнейшей засадки её семенами, использовать лейку для ускорения процесса проста растений. Возможность найма рабочих для автоматизации процесса и увеличения производственной мощи, рабочих можно назначать на различные модули фермы, на которых получая указания от игрока они будут работать за зарплату. Модуль задач для персонала, через который можно назначить рабочего на определенный модуль фермы и дать задачу на добычу каких-либо товаров. Модуль продажи/покупки товаров по средствам грузовой машины которая уезжает в город и возвращается с деньгами и товаром в зависимости от загруженности. Модуль информирования игрока о том какое место в деревне по капиталу занимает его ферма по средствам, утренний газеты или ноутбука находящемся в доме. Игроку представлена возможность пройти обучение, в котором игроку будет объяснена механика передвижения, использование инвентаря, панели инструментов, инструментов. Модуль задач предназначен для того чтобы игроку не стало скучно во время игры. Игроку предоставлена возможность выполнять задачи (квесты), за которые игрок будет получать различные награды.

Интерфейс. Создан интерфейс для взаимодействия игрока с различными модулями игры. Интерфейс инвентаря (рисунок 11) состоит из двух окон, одно для хранения предметов таких как: семена, товары, инструменты, второе для выбрасывания предметов на землю и их подбора с земли. Так же разработан интерфейс слотов инвентаря, панели инструментов, торговли и других модулей игры.

Рисунок 11 — Интерфейс инвентаря

Интерфейс панели инструментов (рисунок 12) используется для быстрого доступа к инструментам.

Рисунок 12 — Интерфейс панели инструментов

Интерфейс для отображения доступной энергии игрока (рисунок 13), которая используется для того чтобы совершать различные действия на ферме, для доступных финансов, которые предназначены для покупки товаров из города и отображающие конкурентоспособности среди других фермеров.

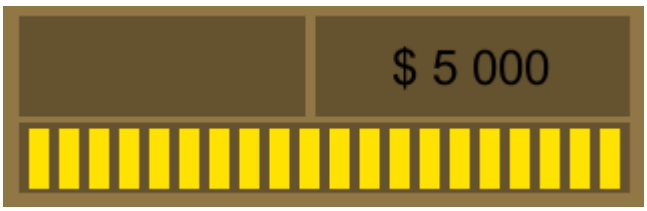

Рисунок 13 — Интерфейс характеристик персонажа

Интерфейс главного меню. Интерфейс менеджера грядок (рисунок 14), для возможности назначения персонала на грядки и автоматизации производства овощей.

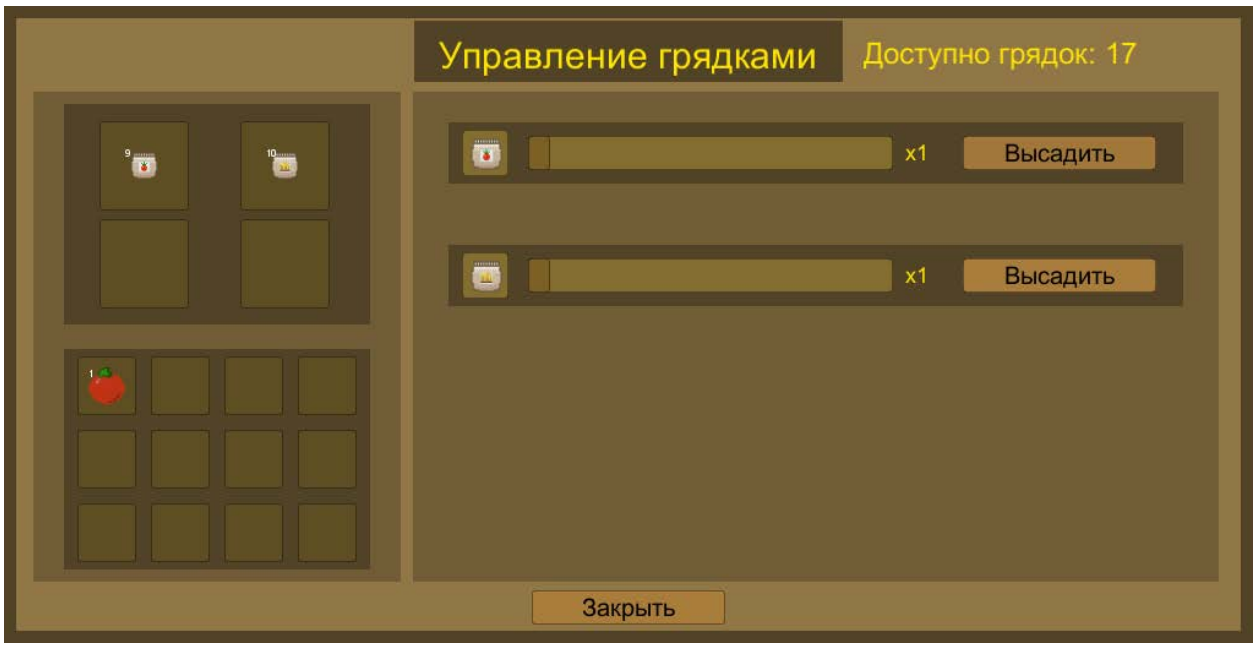

Рисунок 14 — Интерфейс управления грядками

Интерфейс грузовой машины (рисунок 15), которая отправляет товары игрока в город на продажу.

Рисунок 15 — Интерфейс грузовой машины

Интерфейс диалоговых окон (рисунок 16) с помощью который игрок обучается основным механикам игры таким как: передвижение, задачи, посадка растений, сон, характеристики персонажа, взаимодействие с предметами игрового мира.

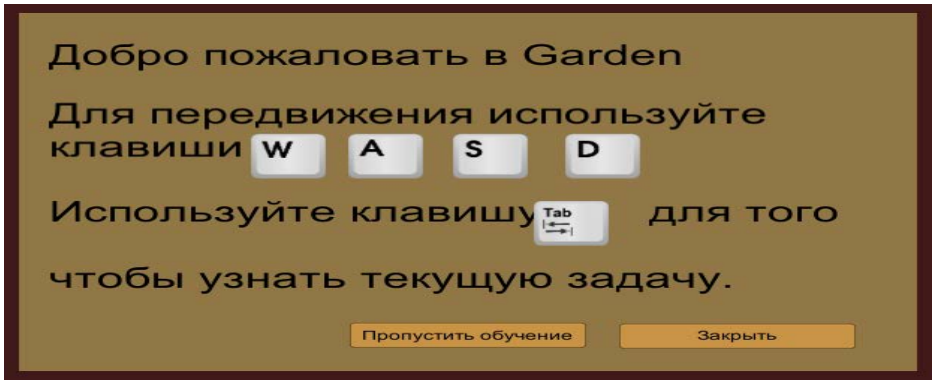

Рисунок 16 — Диалоговое окно туториала

Интерфейс задач для игрока (рисунок 17), выполняя которые он получает дополнительные награды.

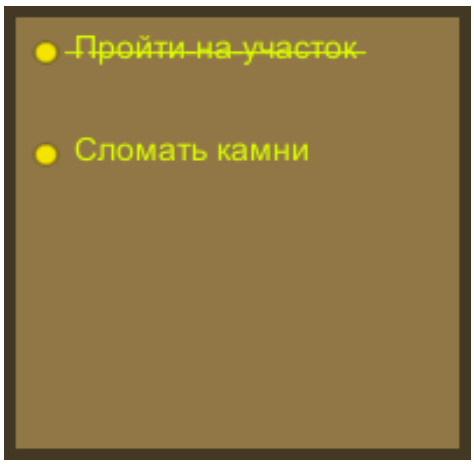

Рисунок 17 — Интерфейс задач игрока

2.2.4 Характеристики оборудования для реализации проекта (фотоаппарата, видеокамеры и т.д.)

Для реализации проекта был использован ноутбук с приведенными ниже характеристиками:

- процессор: Intel Core i3 4510U, частота процессора 1400 МГц, 2 ядра процессора;
- объем оперативной памяти: 8 Гб;
- тип памяти: DDR3L, частота 1600 МГц;
- Экран: 15.6”, максимальное разрешение экрана 1366x768;
- Дискретная и встроенная видео карта: AMD Radeon R5 M230 2048 Мб видеопамяти и Intel HD Graphics 4400.

2.3 Жизненный цикл проекта

Создание игры — это очень долгий и трудоемкий процесс, состоит этот процесс из самых разных этапов.

2.3.1 Этап концептирования

На первом шаге команда или один разработчик придумывают концепцию игры, и приводит начальную проработку игрового дизайна [9]. Главная цель данного этапа — определиться с целью. Что в итоге хочет получить разработчик. В командах данным этапом занимается руководитель проекта [8].

Жанр. Можно с самого начала представлять свой проект в мельчайших подробностях, а можно по ходу разработки додумывать сюжет, и стиль, и особенности игры. На этом этапе не так важна точность, но, как минимум нужно задать направление развитие игрового проекта. Жанр игры необходимо выбрать в самом начале концептирования, чтобы иметь понимание к чему стремиться. Выбранный жанр можно немного корректировать в ходе работы над проектом, но его сущность должна оставаться неизменной. Жанр — это своеобразный фундамент всей игры. Если внезапно захочется сменить жанр игры, то проще будет заново начать разработку игры, чем переделывать то, что уже было наработано.

Данный проект относиться к жанрам: симулятор.

Сеттинг. Разделение компьютерных игр на жанры весьма специфично и не походит на жанры фильмов или книг. Игровые жанры определяют лишь основные действия, которые будут совершаться игроком в процессе игры, тем самым они отвечают на вопрос «Что?». На вопросы «Где?» и «Когда?» отвечает другая основная характеристика игры — сеттинг.

Сеттинг — это принадлежность игры к какой-то сюжетной теме или к определённому виртуальному миру. Создание игры в популярном сеттинге обеспечивает её собственную популярность, да и игроки чувствуют себя уютно и комфортно в уже знакомом мире.

Сеттинг данного проекта — это симулятор жизни на ферме. Игроку достается в наследство фермерский участок, приехав на него главный герой думает, что его ждет спокойная, размеренная жизнь, но спустя несколько дней к нему заходит сосед, пригрозив главному герою, что он тут на долго не

задержится, главный герой понимает, что нужно доказать этому соседу обратное. Игрок начинает конкурировать с остальными фермерами по средствам повышения производства товаров, автоматизации производства, заработка за счет продажи товаров на рынке.

2.3.2 Этап написания программного кода для реализации модулей игры

После того как задана цель проекта, теперь можно приступить к написанию кода основных модулей игры (механик).

Игровая механика. Самая важная часть любой игры — игровая механика. Эта вещь зачастую ускользает от взгляда неопытных игроков. Подростки зачастую оценивают игры в большинстве случаев по качеству графики и не замечают, что красивые игры хоть и популярны, но их механики примитивны и скучны. Если представить игру в виде живого организма, то игровая механика будет являться его нервной системой и головным мозгом.

Основой всей механики являются объекты, с которыми может взаимодействовать игрок. Главный герой игры, компьютерные соперники, второстепенные персонажи (NPC), бонусы, подвижные объекты, декорации — все это игровые объекты со своими свойствами и возможностями.

Так же игровая механика определяет какими клавишами будет управляться главный герой или основной игровой объект, какое действие будет происходить после нажатия той или ной кнопки. Существует различные способы передвижения главного героя по уровням игры от самых простых клавишами клавиатуры, кликом мышки, до сложных, например, управление с помощью шлема виртуальной реальности. Сюда так же относиться поведение объектов и поведение врагов (искусственный интеллект).

В данном проекте реализовано множество механик. Основной механикой является поведение главного героя, его характеристики и возможности.

Игрок может передвигать главного героя по средствам нажатия на клавиши W, A, S, D или стрелочками. При этом обрабатывается нажатие на определенную клавишу клавиатуры и сменяется анимация спрайта персонажа в зависимости от хода направления персонажа.

```
if(isDead==false)
        {
            rigidbody.       velocity       =       new       Vec-
tor3(Input.GetAxisRaw("Horizontal") * speed * Time.deltaTime,
                Input.GetAxisRaw("Vertical")     *     speed     *
Time.deltaTime);
        }
        if (sleep.sleepPlayer == true)
        {
            isDead = true;
        }
        else
        {
            isDead = false;
      }
```

Так же спрайтам главного героя нужная анимация, чтобы игра не казалась однообразной, за это так же отвечает программный код, представленный ниже

```
if (rigidbody.velocity.x < 0f)
        {
            animator.SetFloat("Xdir", -1f);
        }
        if (rigidbody.velocity.x > 0f)
        {
            animator.SetFloat("Xdir", 1f);
        }
        if    (rigidbody.velocity.x    ==    0f    &&    rigid-
body.velocity.y==0f)
        {
            animator.SetFloat("Xdir", 0f);
            animator.SetFloat("Ydir", 0f);
        }
        if (rigidbody.velocity.y < 0f)
        {
            animator.SetFloat("Ydir", -1f);
        }
        if (rigidbody.velocity.y > 0f)
        {
            animator.SetFloat("Ydir", 1f);
        }
```

Чтобы создать предметы нужно создать базу данных где бы хранились они и их основные характеристики, изображения, названия, описания. В данном случае база данных создана по средствам коллекций List языка C#.

```
items.Add(new Item("Raw meat", 0, "Raw Meat. Don't eat it!", 1,0, Item.ItemType.Food));
        items.Add(new Item("Cooked meat", 1,"Grilled Meat. That so delicious!", 1,0, Item.ItemType.Food));
        items.Add(new Item("Empty bottle", 2, "Empty Bottle. Don't break it!", 1,0, Item.ItemType.Head));
        items.Add(new Item("Branch", 3, "Wooden branch!", 1,0, Item.ItemType.Resource));
        items.Add(new Item("Stone", 4, "Common stone!", 1,0, Item.ItemType.Resource));
        items.Add(new Item("Stoneaxe", 5, "Can chop tree!", 1,0, Item.ItemType.Tool));
        items.Add(new Item("Tomato seeds", 6, "You can plant it to plowed area!", 10,2, Item.ItemType.Seeds));
        items.Add(new Item("Tomato", 7, "Fresh Tomato. You can sell it!", 1,4, Item.ItemType.Food));
        items.Add(new Item("Wheat", 8, "Wheat. You can feed chickens and cows!", 30,4, Item.ItemType.Fooder));
        items.Add(new Item("Wheat seeds", 9, "You can plant it to plowed area!", 10,2, Item.ItemType.Seeds));
        items.Add(new Item("Pickaxe", 10, "Pickaxe be careful it sharp!", 1,10, Item.ItemType.Tool));
        items.Add(new Item("Watering can", 11, "Plants grow faster if you pour them!", 1,10, Item.ItemType.Tool));
        items.Add(new Item("Hoe", 12, "You can clean the weeds by Hoe!", 1,10, Item.ItemType.Tool));
        items.Add(new Item("Scythe", 13, "You can harvest wheat!", 1,10, Item.ItemType.Tool));
        items.Add(new Item("Egg", 14, "Ordinary egg", 1,5, Item.ItemType.Food));
```

Создание инвентаря для хранения товаров, инструментов и других игровых предметов. Количество ячеек создается по средствам кода создавая

префабы — это особый тип ассетов в Unity, позволяющий хранить игровой объект, который был создан в сцене, и при инициализации, позволяет переносить его на игровую сцену, так же слотам инвентаря задается уникальное имя, чтобы можно было взаимодействовать с каждым из них по отдельности.

```
int slotAmount = 0;
        int x = -90;
        int y = 90;
        database                    =                    GameOb-
ject.FindGameObjectWithTag("ItemDatabase").GetComponent<ItemData
base>();

            for(int i=1;i<5;i++)
        {
            for(int k=1;k<5;k++)
            {
                GameObject slot=(GameObject)Instantiate(slots);
                slot.GetComponent<SlotScript>().slotNumber     =
slotAmount;
                Slots.Add(slot);

                Items.Add(new Item());

slot.transform.SetParent(this.gameObject.transform,false);
                slot.name = "Slot" + i + "." + k;
                slot.GetComponent<RectTransform>().localPosition
= new Vector3(x, y, 0);
                x = x+60;
                if(k == 4)
                {
                    x = -90;
                    y = y -60;
                }
                slotAmount++;
            }
             }
```

После создания слотов, нужно описать их поведения при нажатии, при перетаскивании предметов и общее взаимодействие с ними. Каждый слот инвентаря может содержать в себе предмет и его характеристики поэтому нужно создать каждому слоту коллекцию предметов и связь с базой данных, чтобы можно было подгружать предметы в слоты из базы данных. Так же слоты обрабатывают нажатие на себя с предметом, находящимся в руке персонажа (мышке) и в зависимости от того находиться ли в данный момент предмет в слоте заменяет его, либо если слот пуст помещает туда предмет.

```
if (inventory.Items[slotNumber].ItemName == null && inventory.draggingItem)
        {
            inventory.Items[slotNumber] = inventory.draggedItem;
            inventory.closeDraggedItem();
        }
        else
            try
            {
                if (inventory.draggingItem)
                {

if(inventory.Items[slotNumber].ItemName==inventory.draggedItem.ItemName&&inventory.draggedItem.itemType==Item.ItemType.Seeds)
                    {
                        int temp = inventory.Items[slotNumber].itemValue;
                        inventory.Items[slotNumber]= new Item(inventory.Items[slotNumber].ItemName, inventory.Items[slotNumber].ItemID, inventory.Items[slotNumber].ItemDisc, inventory.Items[slotNumber].itemValue, inventory.Items[slotNumber].itemPrice, inventory.Items[slotNumber].itemType);
                        inventory.closeDraggedItem();
                    }
                    if (inventory.Items[slotNumber].ItemName != null)
                    {
                        inventory.Items[inventory.indexOfDraggedItem] = inventory.Items[slotNumber];
                        inventory.Items[slotNumber] = inventory.draggedItem;
                        inventory.closeDraggedItem();
                    }
                }
            }
                 catch { }
```

Так же нужно создать слоты для панели инструментов и описать их поведение при нажатии, при перетаскивании предметов и общее взаимодействие с ними.

```
if(item.itemType!=Item.ItemType.None)
        {
            itemAmount.enabled = false;
            transform.GetChild(0).GetComponent<Image>().enabled = true;
            transform.GetChild(0).GetComponent<Image>().sprite = item.ItemIcon;
```

```
            if   (item.itemType   ==   Item.ItemType.Seeds   ||
item.itemType   ==   Item.ItemType.Fooder||   item.itemType   ==
Item.ItemType.Food || item.itemType == Item.ItemType.Resource)
            {
                itemAmount.enabled = true;
                itemAmount.text = "" + item.itemValue;

            }
        }
        else
        {
            transform.GetChild(0).GetComponent<Image>().enabled
= false;
            itemAmount.enabled = false;
        }
        if (item.itemValue == 0)
        {
            item = new Item();
             }
```

Написание кода, описывающего параметры игрового персонажа: энергия, деньги. Энергия использоваться для совершения действий на ферме. Деньги используются для покупки товаров, семян, инструментов на рынке.

```
void cashspace()
    {
        string cashString;
        int charcount;
        cashString = cash.ToString();
        charcount = cashString.Length;
        if (charcount == 4)
        {
            cashString  =  cashString.Substring(0,  1)  +  "  "  +
cashString.Substring(1, 3);
            moneyText.text = "$ " + cashString;
        }
        if (charcount == 5)
        {
            cashString  =  cashString.Substring(0,  2)  +  "  "  +
cashString.Substring(2, 3);
            moneyText.text = "$ " + cashString;
        }
        if (charcount == 6)
        {
            cashString  =  cashString.Substring(0,  3)  +  "  "  +
cashString.Substring(3, 3);
            moneyText.text = "$ " + cashString;
        }
        if (charcount == 7)
        {
            cashString  =  cashString.Substring(0,  1)  +  "  "  +
cashString.Substring(1, 3) + " " + cashString.Substring(4, 3);
```

```
            moneyText.text = "$ " + cashString;
        }
        if (charcount == 8)
        {
            cashString  =  cashString.Substring(0,  2)  +  "  "  +
cashString.Substring(2, 3) + " " + cashString.Substring(5, 3);
            moneyText.text = "$ " + cashString;
        }
         }
void energyUi()
    {
        if (energy < 20)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(19).gameObject;
            cell.SetActive(false);
        }
        if (energy < 19)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(18).gameObject;
            cell.SetActive(false);
        }
        if (energy < 18)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(17).gameObject;
            cell.SetActive(false);
        }
        if (energy < 17)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(16).gameObject;
            cell.SetActive(false);
        }
        if (energy < 16)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(15).gameObject;
            cell.SetActive(false);
        }
        if (energy < 15)
        {
            GameObject          cell          =          GameOb-
ject.FindGameObjectWithTag("Canvas").transform.GetChild(11).GetC
hild(2).GetChild(14).gameObject;
            cell.SetActive(false);
```

```
        }
```

Так же нужно описать появление диалоговых окон обучения при срабатывании триггера.

```
if (camChange.camIsActive == true)
        {
            backwindow0.SetActive(true);
            tutorialWindow0.SetActive(true);
        }
        if (rockQuest.enablequest == true)
        {
            backwindow1.SetActive(true);
            tutorialWindow1.SetActive(true);
        }
        if(questComp.questIsDone)
        {
            tutorQuest = false;
        }
        if(rock.rocksAreDestroyed)
        {
            backwindow2.SetActive(true);
            tutorialWindow2.SetActive(true);
        }
        if(sleepTutorial.newDay==true)
        {
            backwindow3.SetActive(true);
            tutorialWindow3.SetActive(true);
            sleepTutorial.newDay = false;
        }
```

Так же нужно описать дополнительны задачи, выполняя которые игрок будет получать дополнительные награды.

```
IEnumerator rockQuestCoroutine()
    {
        float a = 0;
        while (a < 300)//500
        {
            a++;
            yield return new WaitForSeconds(0.01f);
        }
        a = 0;
        questText0.text = "Сломать камни";
        questText1.text = "";
        questMark1.SetActive(false);
        yield return new WaitForSeconds(5f);
        StopCoroutine("rockQuestCoroutine");
    }
    IEnumerator sleepQuestCoroutine()
    {
        float a = 0;
        while (a < 300)//500
```

```
        {
            a++;
            yield return new WaitForSeconds(0.01f);
        }
        a = 0;
        questText0.text = "Зайти в дом";
        questText1.text = "";
        questMark1.SetActive(false);
        yield return new WaitForSeconds(5f);
        StopCoroutine("sleepQuestCoroutine");
         }
```

Описание менеджера грядок, с помощью которого игрок может назначать рабочих на места работы, для автоматизации производства.

```
void buttonPlant()
    {
        plantButton0 = true;
        scrollBar0.interactable = false;
        if (seedSlots.transform.GetChild(0).GetComponent<SeedStorageScript>().item.ItemName == "Tomato seeds")
        {
            if (seedSlots.transform.GetChild(0).GetComponent<SeedStorageScript>().item.itemValue >= cellsToPlant)
            {
                plantTomato = true;
            }
        }

if(seedSlots.transform.GetChild(0).GetComponent<SeedStorageScript>().item.ItemName == "Wheat seeds")
        {
            if (seedSlots.transform.GetChild(0).GetComponent<SeedStorageScript>().item.itemValue >= cellsToPlant)
            {
                plantWheat = true;
            }
        }
         }
```

Описание грядок, которые взаимодействуют с менеджером грядок и так же взаимодействие с игроком и посадку растений с помощью семян.

```
if(gardenManager.plantButton0 == true&&seedStorageSlot0.item.itemValue>0)
        {
            plantWorker();
        }
        if(seedStorageSlot0.item.itemType==Item.ItemType.None&&gardenManager.plantButton0 == false)
```

```
        {
            gardenManager.scrollBar0.interactable = true;
        }
        if(seedStorageSlot0.item.itemValue<=0)
        {
            gardenManager.plantButton0 = false;
        }
    }
    void plantWorker()
    {
        if(gardenManager.plantTomato==true)
        {
            if (gardenManager.cellsToPlant == 1)
            {
                gar-
denPlow0.transform.GetChild(0).gameObject.SetActive(true);
                gardenCell0.cellIsActive = true;
                gardenCell0.plant_TomatoSpawn();
                StartCoroutine("destroyTomatoPlant");
                if (destroyTomato==true)
                {
                    readyStorage.addItem(7);
                    De-
stroy(gardenPlow0.transform.GetChild(1).gameObject);
                    destroyTomato = false;
                    gardenCell0.cellIsUsed = false;
                    seedStorageSlot0.item.itemValue -= 1;
                }
                 }
```

Чтобы игрок мог вручную взаимодействовать с грядками был описан код, обрабатывающий нажатие на ячейку грядки с инструментами, семенами.

```
if                                                    (Invento-
ry.transform.GetChild(9).GetComponent<CharacterSlot>().item.Item
Name == "Hoe" && slot.SlotActive9)
            {
                trans-
form.GetChild(0).gameObject.SetActive(true);
                cellIsActive = true;
            }

            if (cellIsActive == true)
            {
                If                                        (Invento-
ry.transform.GetChild(0).GetComponent<CharacterSlot>().item.Item
Name=="Tomato seeds" && slot.SlotActive0)
                {
                    cell0 = true;
                    plant_TomatoSpawn();
                    plantSpawned = true;

               }
```

Создание стартовой машины, которая доставляет игрока на ферму.

```
IEnumerator StartMove()
    {
        float a = 0;
        while (a <500)
        {
            rigidbody.velocity = dirHome;
            a++;
            yield return new WaitForSeconds(0.01f);
            if(a==275)
            {
                StartBack.SetActive(false);
            }
        }
        a = 0;
        rigidbody.velocity = Vector2.zero;
        yield return new WaitForSeconds(5f);
        while (a < 550)
        {
            rigidbody.velocity = dirTarget;
            a++;
            yield return new WaitForSeconds(0.01f);
        }
        a = 0;
        rigidbody.velocity = Vector2.zero;
        destroyBus = true;
        StopCoroutine("StarMove");
    }
```

Описание кода грузовой машины, которая при заполнении отправляется в город на продажу или покупает товары в городе и возвращается на ферму.

```
IEnumerator StartMove()
    {
        float a = 0;
        while (a <= 300)
        {
            rigidbody.velocity = dirTarget;
            a++;
            yield return new WaitForSeconds(0.01f);
        }
        a = 0;
        rigidbody.velocity = Vector2.zero;
        yield return new WaitForSeconds(5f);
        while(a<300)
        {
            rigidbody.velocity = dirHome;
            a++;
            yield return new WaitForSeconds(0.01f);
        }
```

```
        a = 0;
        rigidbody.velocity = Vector2.zero;
        StopCoroutine("StarMove");
    }
```

Описание слотов менеджера грядок, в которые помещаются семена для автоматической засадки на грядки.

```
if (item.itemType != Item.ItemType.None)
        {
            itemAmount.enabled = false;
            transform.GetChild(0).GetComponent<Image>().enabled
= true;
            transform.GetChild(0).GetComponent<Image>().sprite  =
item.ItemIcon;
            if   (item.itemType   ==   Item.ItemType.Seeds   ||
item.itemType  ==  Item.ItemType.Fooder  ||  item.itemType  ==
Item.ItemType.Food || item.itemType == Item.ItemType.Resource)
            {
                itemAmount.enabled = true;
                itemAmount.text = "" + item.itemValue;

            }
        }
        else
        {
            transform.GetChild(0).GetComponent<Image>().enabled
= false;
            itemAmount.enabled = false;
        }
        if (item.itemValue == 0)
        {
            item = new Item();
      }
```

Слоты менеджера грядок в которое будут помещаться готовые продукты после их созревания на грядках.

```
void Start()
    {
        int slotAmount = 0;
        int x = -90;
        int y = 60;
        database                    =                    GameOb-
ject.FindGameObjectWithTag("ItemDatabase").GetComponent<ItemData
base>();
        for (int i = 1; i < 4; i++)
        {
            for (int k = 1; k < 5; k++)
            {
                GameObject        slot        =        (GameOb-
ject)Instantiate(slots);
```

```
slot.GetComponent<ReadyStorageSlotScript>().index = slotAmount;
                Slots.Add(slot);

                Items.Add(new Item());

slot.transform.SetParent(this.gameObject.transform, false);
                slot.name = "Slot" + i + "." + k;
                slot.GetComponent<RectTransform>().localPosition
= new Vector3(x, y, 0);
                x = x + 60;
                if (k == 4)
                {
                    x = -90;
                    y = y - 60;
                }
                slotAmount++;
            }
        }
    }
```

2.3.3 Этап разработки элементов интерфейса проекта

Интерфейс проекта, немаловажная составляющая успешного продукта. Именно то, насколько пользователям будет удобно работать с тем или иным приложением, игрой, определяет дальнейшую судьбу, приложения, игры и финансового состояния разработчика [12].

Удобно классифицировать пользовательский интерфейс по конечному его применению:

- интерфейс программного обеспечения;
- интерфейс игровой;
- интерфейс сайтов.

С первым наименованием списка всё ясно. При создании интерфейса наибольший упор делается на функционал. Поэтому, обычно, в интерфейс самой программы не вкладывают слишком много усилий, придерживаясь правил минимализма и лаконичности. Для примера хочется рассмотреть наш «любимый» Windows. Только в последние версии разработчики начали вкладывать какой-то рационализм. В целом же, интерфейс для пользователя не

только не продуманный, но и, во многом, противоречит основным аспектам психологии восприятия человека.

Во втором случае всё с точностью до наоборот. Интерфейс игровой иногда бывает сложно назвать интерфейсом, так как он внедрён в суть игры и максимально адаптирован под пользователя. Для того, чтобы найти то или иное решение, обычно, не приходится слишком задумываться. Всё интуитивно. Идеальный вариант для пользователя.

С сайтами всё гораздо сложнее. Если обратить внимание на интерфейсы сайтов, то можно наткнуться на невероятное разнообразие решений. И, обычно, кто в лес, кто по дрова. Какие-то разработчики придерживаются строгости и минимализма, какие-то наоборот перегружают интерфейсом.

Довольно много оптимальных решений. Но, так же часто приходится натыкаться на совершенно непроработанные сайты, где, вроде бы, старались, но, видимо, перестарались.

Видимо, те или иные дизайнерские решения необходимо черпать из функциональной составляющей сайта. Но, в любом случае, главная задача любого сайта- донести информацию. И, для того, чтобы она дошла до адресата (в прямом и переносном смысле), она должна быть доступна, читабельна, адаптирована. Для достижения этой цели необходимо немалое значение уделить разработке дизайна. У многих разработчиков «набит» глаз и виртуозная интуиция. Однако, знание азов существенно упрощает выбор цветового решения и экономит время. Есть много концепций выбора цветовой гармонии.

Есть такое понятие, как цветовой круг Иттена (рисунок 18). Цветовой круг — способ представления цветов видимого спектра в условной форме, обозначающей различные цветовые модели. Секторы круга представляют определяемые цвета, размещённые в порядке условно близком к расположению в спектре видимого света, причем в круг добавлен условный пурпурный цвет, который связывает крайние спектральные цвета.

Цветовые круги (а их несколько и их появление связано с появлением различных теорий цвета начиная с 17-18 веков мистические теории и физиче-

ские теории) мнемоническое правило, которое помогает ориентироваться в пространстве цветов, и отображать схематично различные цветовые модели. Как правило цвета, представленные в круге это условные цвета максимальной цветовой насыщенности, выбранные из поля цветового охвата цветовой модели. В модели RYB цвета в цветовом круге — это своего рода символы цветов - условные обозначения цветов без учёта их измеряемых характеристик. Выглядит он примерно вот так:

Рисунок 18 — Цветовой круг Иттен

В его состав входят 12 цветов. Именно эти цвета считаются наиболее привычными и лёгкими для восприятия человеком.

Первообразующими принято считать три основных цвета: жёлтый (ffff00), синий(0000ff) и красный(ff0000). На основе этих цветов создан весь цветовой круг. Первообразующие они потому, что при наличии этих трёх чистых цветов путём смешивания их, можно создать всю палитру. Любой цвет.

Цвета второго порядка получают путем смешения основных цветов: мандариновый (ff8000), пурпурный (ff00ff), зелёный (00ff00).

Цвета третьего порядка получаются смешиванием основных цветов и цветов второго порядка. Их получается шесть:

- янтарный (ffc000);
- киноварь (ff4000);
- светло-вишнёвый (ff0080);
- фиолетовый (8000ff);

- лазурный (0080ff);
- ядовито-зелёный (80ff00).

Этот круг, можно сказать, панацея от всех бед. Именно с помощью него можно подбирать наиболее удачные сочетания цветов. Для этого существует шесть принципов цветовой гармонии. Так называемые, цветовые модели.

Монохромные цвета (рисунок 19). За основу берётся один цвет и разная его насыщенность и прозрачность. Монохромные сочетания весьма просты в использовании и довольно мягкие на восприятие. Но им часто не хватает выразительности.

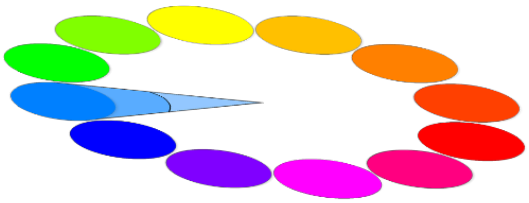

Рисунок 19 — Монохромные цвета

Близкие цвета (рисунок 20). Цвета, которые находятся рядом на цветовом круге. Эти цвета обладают схожими характеристиками световых волн, поэтому сочетаются очень просто.

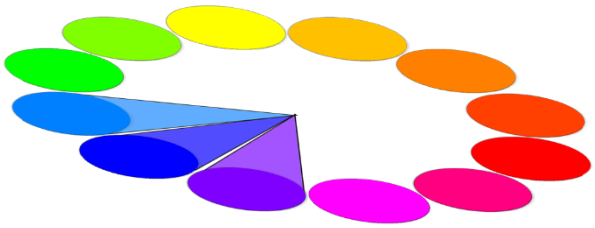

Рисунок 20 — Близкие цвета

Комплементарные (рисунок 20). Это цвета, расположенные строго друг напротив друга. Самый большой контраст дают именно они. Это всё броское, живое, дерзкое.

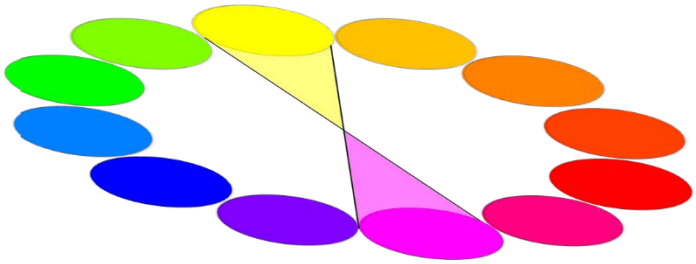

Рисунок 21 — Комплементарные цвета

Близкие и комплементарный (рисунок 22). Тут всё просто! Берутся противоположные цвета, например, А и Б. У Б выделяются соседи Б(л) слева и Б(п) справа. Собственно, сочетание А, Б(п) и Б(л) и есть искомая величина! При игре с яркостью можно получить весьма интересные варианты.

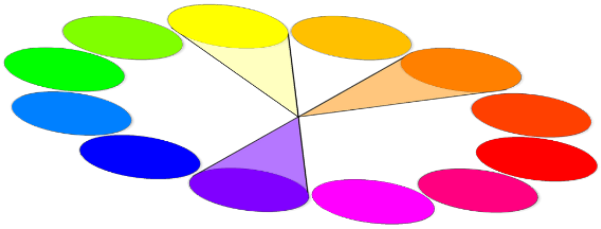

Рисунок 22 — Близкие и комплементарные цвета

Сдвоенные комплементарные (рисунок 23). Тут тоже всё просто. Берём Б(п) и Б(л) и ищем для них комплементарные (противоположные) цвета. Всё то же броское, живое, дерзкое только в квадрате.

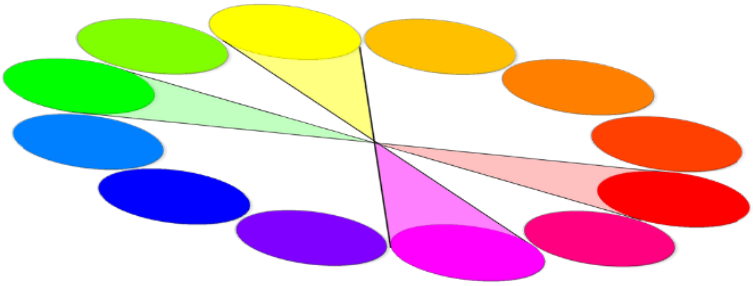

Рисунок 23 — Сдвоенные комплементарные

Триадные цвета (рисунок 24). Расположены через три сектора. Обычно, контрастные цвета сложно сочетаются, хотя, чистом виде, сморятся весьма эффектно в кардинально разных пропорциях. Однако, чем меньше яркость цвета, тем больше возможностей. Триады первичных цветов слишком резкие. Вторичные и третичные триады более мягкие [19].

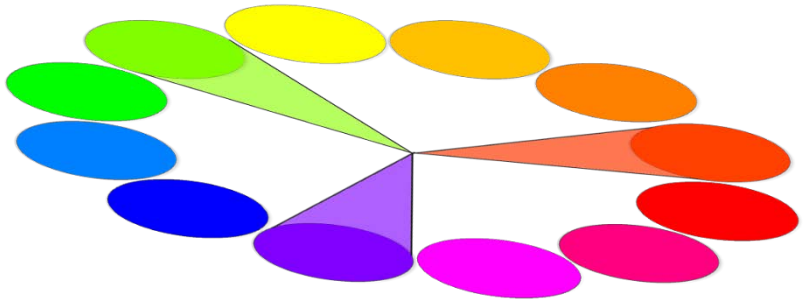

Рисунок 24 — Триадные цвета

Принцип работы с цветом по кругу Иттена прост: подбираете 2-4 цвета через принципы цветовых гармоний и экспериментируете с яркостью-контрастностью, площадью заполнения.

Ну и кроме того, стоит отдельно отметить ахроматические цвета. Никаких красок, никаких оттенков. Только градация от черного к белому. Полный спектр серости. Огромный выбор. Наличие чистого белого или чистого чёрного в цветовом решении основных шести гармоний, усиливает контраст, соответственно, интерфейс будет очень «живым», если не переборщить с площадью покрытия цветов. В противном случае, чрезмерная активность цвета будет раздражать.

Хотите привлечь внимание к цвету и сделать сайт максимально ярким и выразительным? Берите контрастные цвета. Добавляйте немного чистого белого либо бархатного чёрного. Сделайте крупное меню и объемный шрифт. Этого будет достаточно.

Нужен спокойный ненавязчивый интерфейс для серьёзного проекта-монохромные комбинации Вам в помощь.

И, немаловажное правило в цвет подборе интерфейса: минимализм. Пять цветов — это максимальный предел. Не перегружайте свой сайт всеми цветами радуги. Не нужно тут пэчворковского одеяла. Ваша задача- донести информацию, а не пугать людей пестростью.

Проанализировав интерфейсы существующих разработок (рисунок 25) (рисунок 26) и других схожих проектов было решено выбрать монохромный цвет, такая цветовая схема лучше всего подходит для интерфейсов игр так как она не отвлекает своей пестростью от игрового процесса.

Рисунок 25 — Пример интерфейса и цветовой гаммы игры Stardew Valley

Рисунок 26 — Пример интерфейса и цветовой гаммы игры Harvest Moon

Так как мы используем монохромные цвета для проекта были выбраны коричневый цвет и его оттенки.

Данный цвет очень подходит для сеттинга игры (фермерское дело).

Коричневый цвет — цвет земли, плодородия, цвет коры дерева. Этот цвет символизирует стабильность, устойчивость, преданность, здравый смысл, надежность. Он способствует преодолению трудностей, принятию важных решений, концентрации внимания. Но есть и отрицательные характеристики этого цвета – разочарование, депрессия, отказ, разрушение, неуверенность, скованность, скука [7].

Коричневый цвет ассоциируется с сельским домом, бытом, с осенью, с землей. Люди, выбирающие этот цвет, склонны к меланхолии, ностальгии, уединению. Они хотят отдохнуть от суеты, городской жизни, от нерешенных проблем.

В психологии коричневый цвет и все его оттенки считаются цветом чувственности, комфорта, безопасности, уюта и цветом домашнего очага [4].

В древнегреческой мифологии коричневый цвет был цветом богини Геры. Это богиня земледелия, смерти и воскрешения. В христианстве коричневый цвет, особенно его темные оттенки, олицетворял цвет сатаны, страсти и цвет плотских утех. Мирные жители старались обходить стороной все то, что имело коричневый цвет [5].

Основным цветом (рисунок 27) цветовой гаммы проекта был выбран коричневый цвет (RGB: 917746)

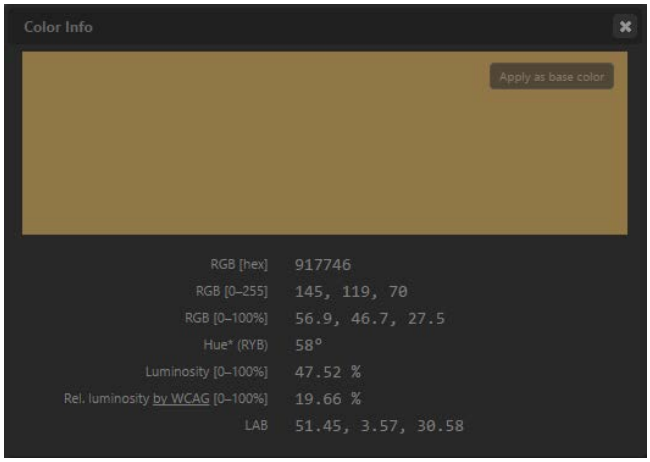

Рисунок 27 — Основной цвет

Дополнительными цветами являются оттенки коричневого цвета:

- яркий телесный цвет (рисунок 28) (RGB DDC69C);

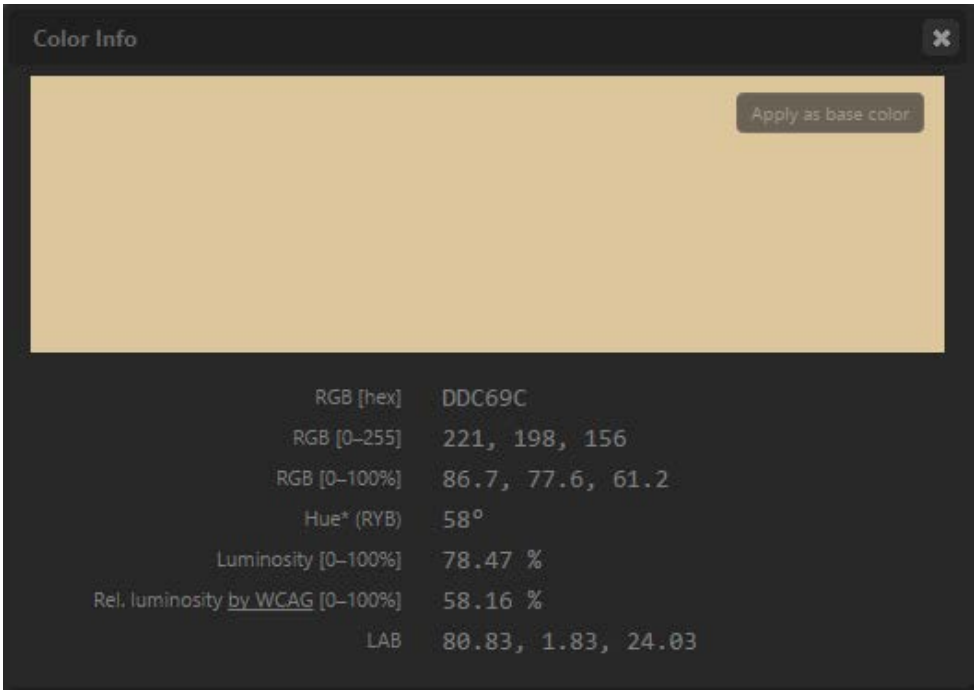

Рисунок 28 — Дополнительный цвет DDC69C

- темный телесный цвет (рисунок 29) (RGB B59A66);

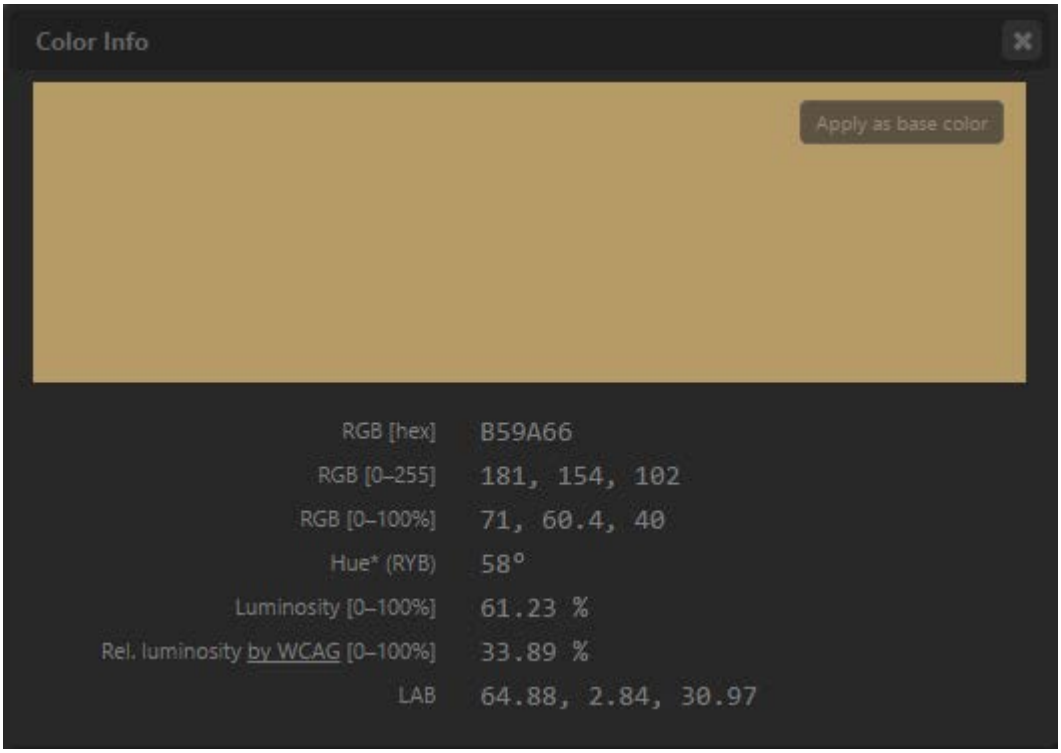

Рисунок 29 — Дополнительный цвет B59A66

- ярко-темно коричневый цвет (рисунок 30) (RGB 7A5D27);

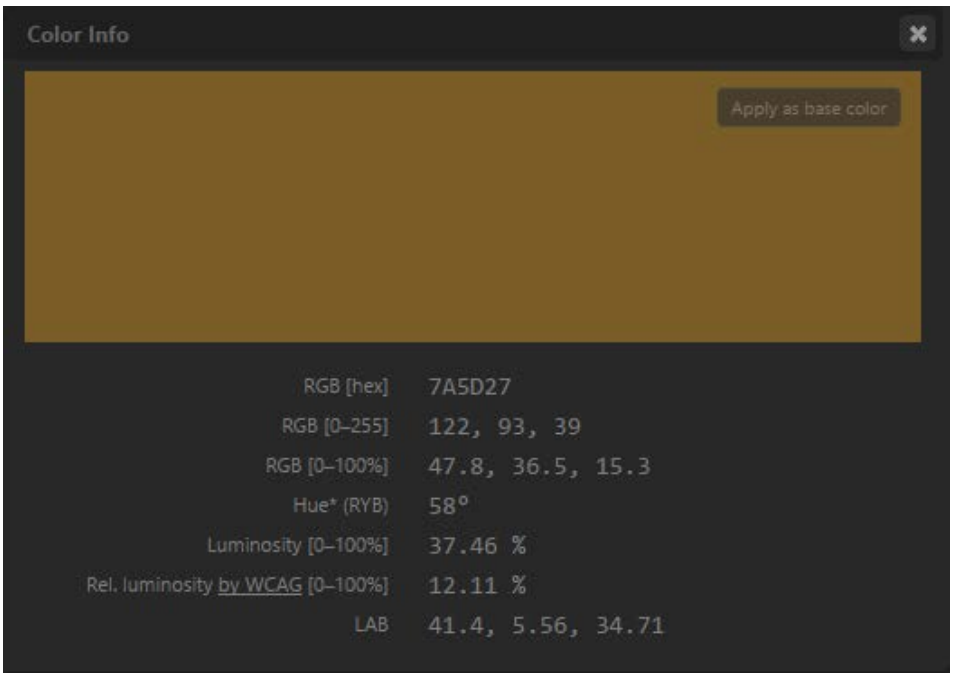

Рисунок 30 — Дополнительный цвет 7A5D27

- темно коричневый цвет (рисунок 31) (RGB 573E0E).

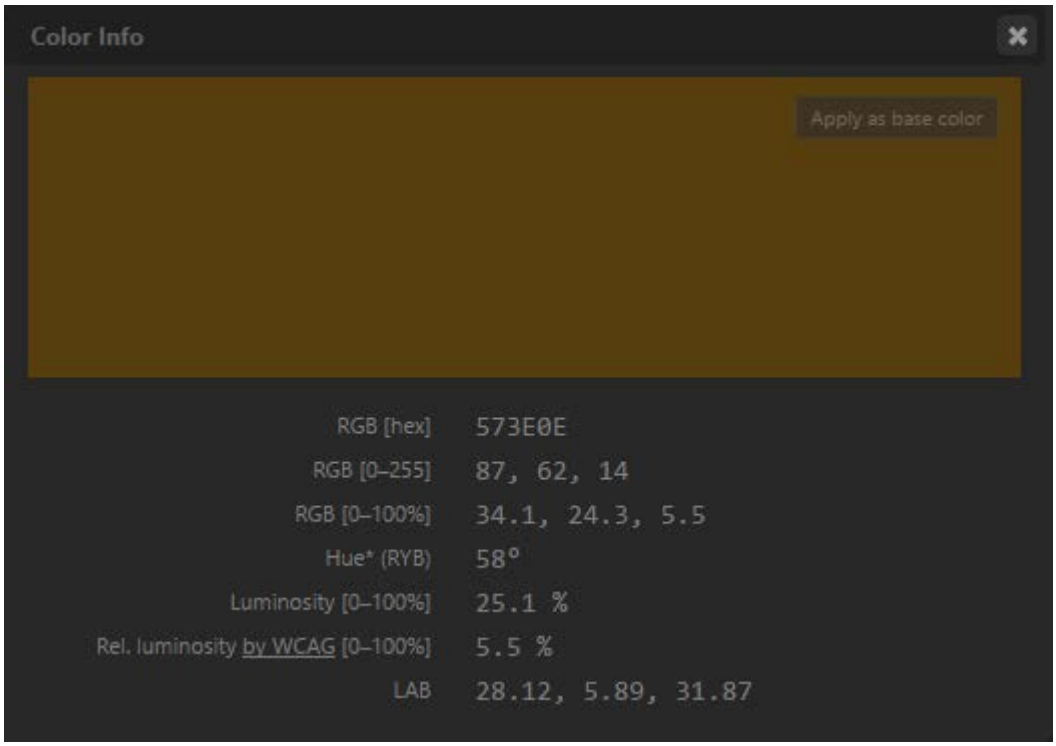

Рисунок 31 — Дополнительный цвет 573E0E

Цветовая схема (рисунок 32) создана по средствам специального сайта, который помогает подобрать цветовую гамму для проектов.

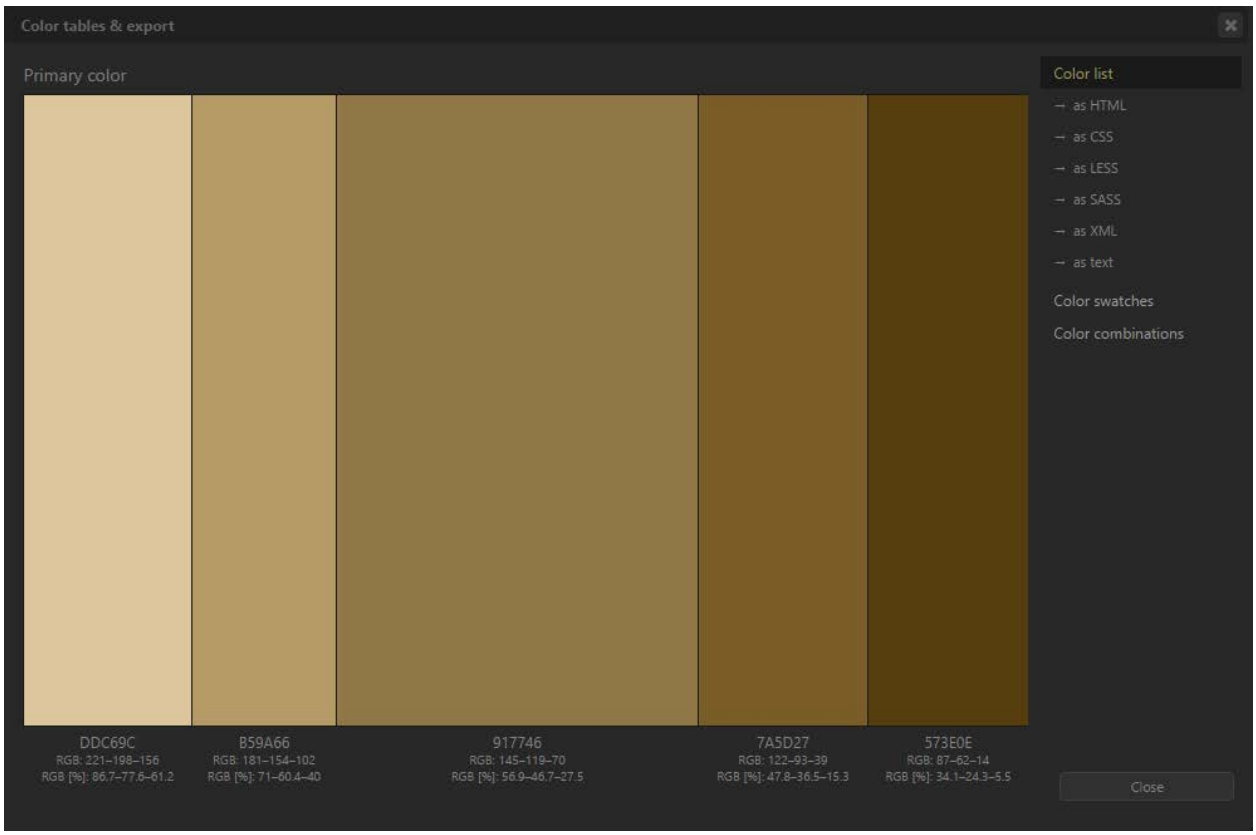

Рисунок 32 — Цветовая гамма проекта

Таким образом были разработан интерфейс игры в цветовой гамме, описанной выше.

Разработан интерфейс инвентаря (рисунок 33), позволяющий взаимодействовать с хранилищем инструментов, семян, товаров.

Рисунок 33 — Интерфейс инвентаря

Разработан интерфейс грузовой машины (рисунок 34), которая предназначена для отправки товаров в город на продажу и доставки инструментов из города.

Рисунок 34 — Интерфейс грузовой машины

Разработан интерфейс для отображения основных характеристик персонажа таких как: энергия, денежные средства (рисунок 35).

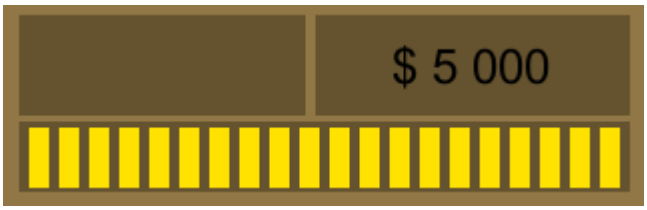

Рисунок 35 — Интерфейс параметров персонажа

Разработан интерфейс дополнительных задач для игрока (рисунок 36), выполняя которые он сможет получить награды.

Рисунок 36 — Интерфейс дополнительных здача

Разработан интерфейс панели инструментов (рисунок 37) в которой игрок может поместить инструменты, семена для быстрого доступа к ним.

Рисунок 37 — Интерфейс панели инструментов

Разработан интерфейс управления грядками (рисунок 38) с помощью которого можно назначать рабочий персонал на грядки, для автоматизации производства.

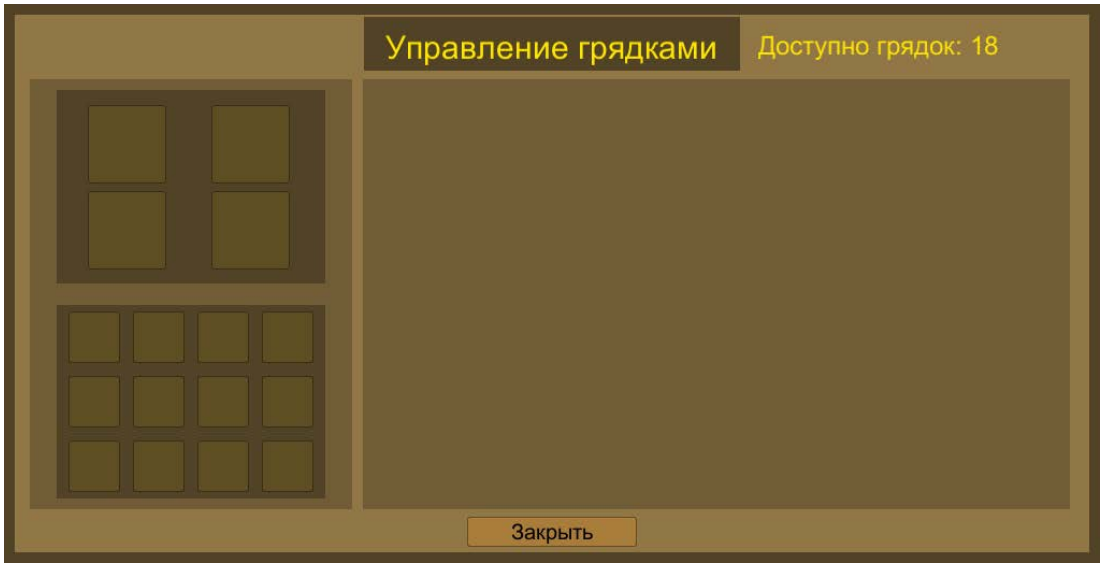

Рисунок 38 — Интерфейс управления грядками

2.3.4 Этап создания игрового уровня (левел дизайн)

Правила игры в виде игровой механики готовы, теперь нам нужно создать площадки, где эти правила начнут работать. Созданные игровые объекты расставляются в отдельных виртуальных пространствах – уровнях (локациях). Игры чаще всего содержат множество отдельных уровней, переход между которыми происходит по ходу сюжета. Но в последнее время, благодаря возросшей производительности компьютеров, выпускаются игры с одним большим цельным миром, лишь условно разделяемом на различные локации (GTA, Skyrim).

На каждом отдельном уровне расставляются игровые объекты, стенки, платформы, декорации, фоны. Уровни создаются в играх всех жанров. Даже в простенькой казуальной игре по перестановке цветных камешков есть уровни – в их роли выступают игровые поля и расстановка камней. В браузерных играх в роли локации выступают отдельные html – страницы.

Если, опять же, представить игру в виде дома, то построение игровых уровней — это планировка этажей, а количество уровней — этажность здания.

Построением уровней занимаются левел дизайнеры.

В идеале левел дизайнеры берутся из числа заядлых игроков. Это происходит потому, что любой другой человек со стороны, пусть даже и творческий, но очень далёкий от темы игр, не сможет хорошо справиться с этой задачей. Левел дизайнер должен хорошо представлять себе игровой процесс, и чувствовать, как от перемещения объектов на уровне будет изменяться игровая ситуация.

Именно от дизайна (не оформления, а планировки) уровней зависит важнейшая составляющая игры — геймплей. (Это правило не действует лишь для большинства казуальных игр, файтингов и спортивных игр, где уровни крайне примитивны) [3]. Неинтересная и однообразная планировка уровней загубила множество игр с великолепным оформлением, подкреплённым новейшими технологиями. Если вы хотите создать полноценную увлекательную игру, а не пустышку в красивой обёртке, то на левел дизайн нужно потратить максимум творческих усилий.

Таким образов был выбран простой инструмент для построения уровней для 2D игр, именуемый Tiled2D.

Tiled Map Editor — редактор пиксельарт тайтлсетов написанный на C++ (существует и Java версия программы). Данную программу можно использовать для разработки игр любого 2D-жанра. Редактор имеет большой набор готовых тайлсетов, что значительно увеличивает скорость разработки

игры. Данный инструмент имеет интеграцию с Unity, что позволяет легко экспортировать созданные уровни сразу в сцену игры без каких-либо потерь.

Игровой уровень разбит на несколько частей:

Основной участок фермы (рисунок 39). На данном участке происходит самое важное для игрока. На данном участке расположен дом игрока, грядки на которых игрок может садить растения, забор, который ограничивает перемещение игрока, курятник, в котором игрок может выращивать куриц, для убийства их на мясо или производства яиц, коровник, в котором игрок может выращивать коров для мяса или производства молока.

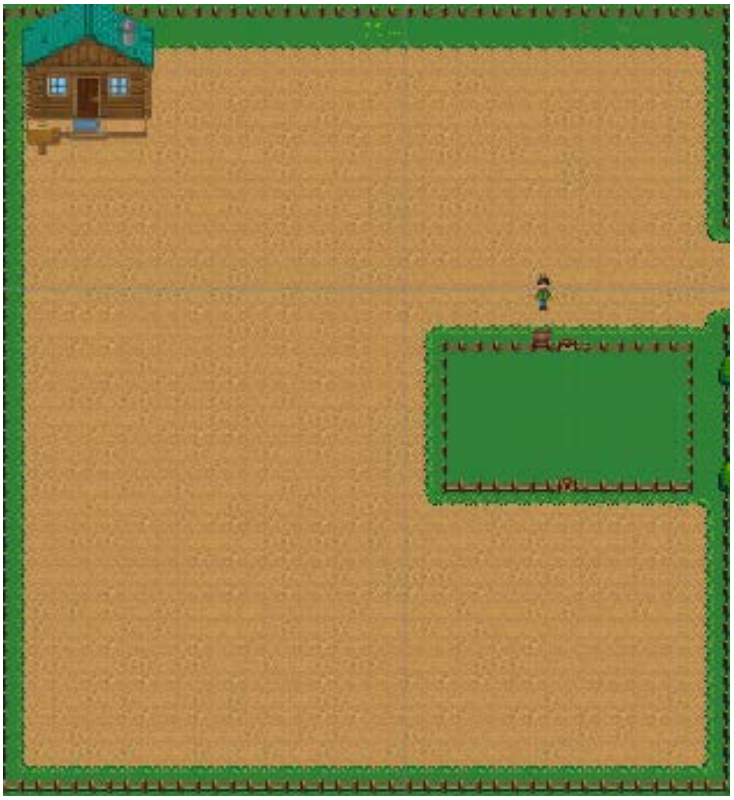

Рисунок 39 — Основной участок

Дорога (рисунок 40), по которой приезжает стартовый автобус с игроком и для грузового автомобиля, который будет ездить в город и из него. Так же рядом с дорогой есть зона разгрузки автомобиля.

Рисунок 40 — Дорога с разгрузочной зоной

Так же был создан помещение дома (рисунок 41), в котором игрок может спать, чтобы восстановить свою энергию и получать свежие новости по ноутбуку.

Рисунок 41 — Внутренние помещение дома

2.3.5 Этап создание спрайтов для игры

Спрайт (англ. Sprite — фея; эльф) —графический объект в компьютерной графике.

Чаще всего — растровое изображение, которое можно отобразить на экране. Наблюдение спрайта в трёхмерном пространстве под несоответствующим углом приводит к разрушению иллюзии. То есть легче всего воспринимать спрайт как перемещающуюся в пространстве проекцию какого-то объёмного тела так, чтобы разница была незаметна.

Слово «спрайт» было придумано в 1970-е годы одним сотрудником компании Texas Instruments: их новая микросхема TMS9918 могла аппаратно отображать небольшие картинки поверх неподвижного фона.

Изначально под спрайтами понимали небольшие рисунки, которые выводились на экран с применением аппаратного ускорения. На некоторых машинах (MSX 1, NES) программная прорисовка приводила к определённым ограничениям, а аппаратные спрайты этого ограничения не имели. Впоследствии с увеличением мощности центрального процессора от аппаратных спрайтов отказались и понятие «спрайт» распространилось на всех двумерных персонажей. В частности, в видеоиграх Super Mario и Heroes of Might and Magic вся графика спрайтовая, анимированная с помощью перебора изображений из атласа спрайтов.

К аппаратно ускоренным спрайтам вернулись в середине 1990-х годов, когда развитие мультимедиа и взрывное повышение разрешения и глубины цвета потребовало специализированного процессора в видеоплате. Именно тогда, как обёртка над аппаратным 2D-ускорителем, вышел DirectDraw. DirectX 8 ввёл общий API для двух- и трёхмерной графики, и в современных спрайтовых играх двухмерные спрайты выводятся точно так же, как и трёхмерные — как текстурированный прямоугольник.

Атлас спрайтов является растровым изображением, объединяющим спрайты в набор раскадровок анимаций, или иным комплектом спрайтов.

Целесообразность применения заключается в уменьшении количества файлов и применение sprite-batch в процессе рендеринга: видеокарте передаётся одно изображение, набор координат, на которых его необходимо «отрендерить» и «трафаретов», вырезающих из атласа отдельные спрайты [6]. В результате, двухмерная графика «рендерится» значительно быстрее, чем одиночными передачами видеокарте одиночных спрайтов с указанием координат отрисовки. Для дополнительного сжатия и уменьшения количества неиспользуемого пространства, атлас может комбинировать изображения не сеткой, а в максимально плотно сжатом виде, таким образом, что, если

описать вокруг каждого спрайта прямоугольник - часть прямоугольников будет пересекаться. Данный формат требует дополнительной информации о вершинах, которыми можно описать каждый спрайт без пересечения с другими, и увеличивает время загрузки приложения, но позволяет сократить память [2].

Таким образом были созданы спрайты (рисунок 42):

- персонажа;
- инструментов;
- семян;
- дома;
- автобуса;
- грузового автомобиля.

Рисунок 42 — Спрайты игровых объектов

2.4 Технические требования к проекту

Для запуска игры нужен компьютер или ноутбук с минимальными требованиями:

- операционная система: Windows 7 SP1 (x64);
- процессор: Intel core i3-4510U;
- оперативная память: 4 GB ОЗУ;
- видео карта: AMD Radeon R5 M230;
- место на диске: 100Мб.

2.5 Калькуляция проекта

В ходе разработке проекта было создано:

- Было написано 52 скрипта общим количеством строк равному 6172;
- Было разработано 12 игровых модулей, отвечающих за различные этапы игры;
- Было разработано 10 интерфейсов для различных модулей;
- Было нарисовано 15 спрайтов для проекта;
- Было создано 2 локации для проекта.

ЗАКЛЮЧЕНИЕ

Компьютерные игры уже давным-давно вошли в жизнь пользователей ПК и являются одним из способов проведения свободного времени. Виртуальная реальность с каждым месяцев выпускает различные новинки этой области, от которых современные геймеры просто не могут отказаться. Несмотря на безусловный вред от компьютерных игр, вызванный привыканием к ним, имеют они и некоторые положительные стороны, которые дают им полное право на существовании в современности.

Во-первых, не стоит забывать, что компьютерные игры в большинстве своем способствуют развитию логики, памяти и, безусловно, внимания. К примеру, достаточно известная игра бойцовский клуб представляет собой организованные бои между двумя или несколькими персонажами, в результате чего происходит безусловная тренировка внимания. Также не стоит забывать про такой вид игры, как стратегии, которые не требуют излишнего напряжения глаз и постоянного внимания, что является достаточным контрастом динамичным играм. Более того, несомненный плюс стратегий состоит в том, что их можно прервать в любой момент, не боясь потерять игру, таким образом, стратегии предназначены для достаточно длительного времяпровождения.

Не стоит забывать и про существенную пользу компьютерных игр для детей-дошкольников, благодаря наличию различных обучающих игр. К примеру, интересными вариантами являются ненавязчивые обучающиеся игры, в ходе которых происходит изучение иностранного языка. Конечно, компьютер вовсе не должен стать единственным источником для знаний вашего ребенка, но запрет на пользование компьютером может стать лишним поводом для агрессии. Таким образом, желательно, чтобы вы самостоятельно подобрали для ребенка игры, которые будут ему полезны.

Однако для взрослого человека польза компьютерных игр несомненна, поскольку та же игра поможет снять стрессовое напряжение, накопившееся в течение дня, а также отвлечься от обыденности и даже поработать мозгами, просчитывая лучшую тактику. Здесь, точно также, как и в ситуации с детьми, важна разновидность игры, к примеру, для выплеска агрессии больше всего подойдут игры из разряда сражений и боев, в то время как стратегии рассчитаны на более спокойную атмосферу, без капли агрессии.

Обобщая все вышесказанное, хочется отметить, что популярность компьютерных игр возникает благодаря возможности уйти из суровой реальности в реальность виртуальную.

Используя навыки программирования на C#, разработки интерфейсов, рисование спрайтов, приобретенные во время обучения в институте, была создана компьютерная игра с использованием игрового движка Unity, среды программирования Visual Studio, редактора уровней Tiled2D, редактора растровых изображений Adobe Photoshop.

В данной работе были раскрыты теоретические аспекты проблемы создания игр, разработки интерфейсов, создания уровней, спрайтов. В ходе создания проекта были получены и закреплены навыки работы в Unity, Adobe Photoshop, Visual Studio, Tiled2D.

Цель выпускной квалификационной работы — разработать компьютерную игру.

Задачи выпускной квалификационной работы:

- изучить игровой движок Unity;
- изучить стадии разработки игр;
- проанализировать аналоги;
- разработка игровой механики;
- тестирование.

Игра реализована в полной мере, таким образом задачи и цели выпускной квалификационной работы достигнуты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Албахари Д. C# 6.0. Справочник. Полное описание языка [Текст]/ Д. Албахари, Б. Албахари. — под общ. ред. Албахари Д. – Москва: Вильямс, 2017. — 1040 с.

2. Дикинсон К. Оптимизация игр в Unity 5 [Текст] / К. Дикинсон. — Москва: ДМК, 2017. — 306 с.

3. Иоханнес И. Искусство формы. Мой форкурс в Баухаузе [Текст]/ И. Иоханнес. — Москва: Дмитрий Аронов, 2016. — 136 с.

4. Иоханнес И. Искусство цвета [Текст]/ И. Иоханнес. — Москва: Дмитрий Аронов, 2015. — 96 с.

5. Купер А. Интерфейс. Основы проектирования взаимодействия [Текст]/ А. Купер, Р. М. Рейманн, Д. Кронин. — Под общ. ред. А. Купера. – Санкт-Питербург: Питер, 2017. — 720 с.

6. Ламмерс К. Шейдеры и эффекты в Unity. Книга рецептов [Текст] / К. Ламмерс. — Москва: ДМК, 2014. — 274 с.

7. Савахата Л. Гармония цвета. Справочник. Сборник упражнений по созданию цветовых комбинаций [Текст]/ Л. Савахата. — Москва: АСТ, 2011. —192 с.

8. Терри Н. Learning C# by Developing Games with Unity 3D [Текст]/ Н. Терри. — Книга по требованию, 2013. — 292 с.

9. Торн А. Искусство создания сценариев в Unity [Текст] / А. Торн. — Москва: ДМК, 2016. — 360 с.

10. Торн А. Основы анимации в Unity [Текст] / А. Торн. — Москва: ДМК, 2016. — 176 с.

11. Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6 [Текст] / Э. Троелсен, Ф. Джепикс. — Под общ. ред. Троелсена Э.– Москва: Вильямс, 2016. — 1440 с.

12. Уэйншенк С. 100 новых главных принципов дизайна. [Текст]/ С. Уэйншенк. — Санкт-Питербург: Питер, 2016. — 288 с.

13. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C# [Текст] / Д. Хокинг. — Санкт-Питербург: Питер, 2016. — 336 с.

14. Язык программирования C# [Электронный ресурс] — Режим доступа: https://metanit.com/sharp/ (дата обращения: 07.05.2017).

15. C# 5.0 и платформа .Net 4.5 [Электронный ресурс] — Режим доступа: https://professorweb.ru (дата обращения: 20.05.2017).

16. Gamesis Art [Электронный ресурс] — Режим па: http://gamesisart.ru/game_dev_create.html (дата обращения: 06.05.2017).

17. Habrahabr [Электронный ресурс] — Режим па: https://habrahabr.ru (дата обращения: 07.05.2017).

18. Unity-руководство[Электронный ресурс] — Режим па: https://docs.unity3d.com (дата обращения: 06.05.2017).

19. Vc.ru [Электронный ресурс] — Режим доступа: https://vc.ru/p/colour-system (дата обращения 20.05.2017).

20. Wikipedia [Электронный ресурс] — Режим па: https://ru.wikipedia.org/wiki (дата обращения: 05.05.2017).

ПРИЛОЖЕНИЕ

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

направление подготовки 09.03.02 Информационные системы и технологии
профилю подготовки «Информационные технологии в медиаиндустрии»

УТВЕРЖДАЮ

Заведующий кафедрой

_____Н. С. Толстова

«_____»____2017 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

студента 4 курса, группы ИТм-401 Съедина Сергея Эдуардовича
1. Тема К
утверждена распоряжением по институту от 07.02.2017 г. № 73.
2. Руководитель старший преподаватель, Садчиков Илья Александрович
3. Место преддипломной практики РГППУ
4. Исходные данные к ВКР Технические документы по Unity. Дизайн документ. Анализ разработок-аналогов[К1]
5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов[К2])
Разработка игровой экономической модели и ее балансировка
Разработка и балансировка основных игровых механик
Изучение Unity 5 Engine
Создание спрайтов и графических элементов игры
Программирование игры и сборка рабочего билда
6. Перечень демонстрационных материалов[К3]
Презентация
Игровой билд
Демонстрационное видео проекта

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа ВКР	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной работе и сдача зачета по преддипломной практике	17.04.2017	15	
2	Выполнение работ по разрабатываемым вопросам их изложение в выпускной работе:			
	Разработка экономической модели, баланс	01.10.2016	100%	
	Создание спрайтов и элементов UI	01.12.2016	100%	
	Программирование основных механик	01.02.2017	100%	
	Создание уровней	01.03.2017	100%	
	Разработка демонстрационной версии	01.05.2017	100%	
3	Оформление текстовой части ВКР	30.05.2017	5	
4	Выполнение демонстрационных материалов к ВКР	01.06.2017	5	
5	Нормоконтроль	03.06.2017	5	
6	Подготовка доклада к защите в ГЭК	09.06.2017	5	

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата

Руководитель _____ *Задание получил* _____ 24.10.2017
подпись дата подпись студента дата

9. Выпускная квалификационная работа и все материалы проанализированы.
Считаю возможным допустить Съедина Сергея Эдуардовича к защите выпускной квалификационной работы в государственной экзаменационной комиссии.

Руководитель _____ _____
подпись дата

10. Допустить Съедина Сергея Эдуардовича к защите выпускной квалификационной работы в государственной экзаменационной комиссии (протокол заседания кафедры от 14.06.2017 №12)

Заведующий кафедрой _____ _____
подпись дата