|  |  |
| --- | --- |
|  | **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**  **РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Тюменский индустриальный университет»** |

Строительный институт

Кафедра автомобильного транспорта, строительных и дорожных машин

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Преддипломная практика

Обучающегося 4 курса Мочернюк Артём Николаевича

Наименование практики: производственная

Место прохождения практики: \_\_\_\_\_\_Северречфлот\_\_\_\_\_\_\_\_\_\_\_

Начало практики «\_28\_» \_\_\_апреля\_\_\_\_\_\_\_\_\_\_ 2022 г.

Окончание практики «\_27\_» \_\_мая\_\_\_\_\_\_\_\_ 2022 г.

Руководитель практики от университета к.г-м.н., доцент Антипова А.Н. / \_\_\_\_\_\_

(должность, ФИО) ( подпись)

Руководитель структурного подразделения университета\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_

Тюмень, 2022 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc91174091)

[ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc91174092)

[ГЛАВА 2. МОДЕЛИРОВАНИЕ ДЛЯ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc91174093)

[2.1. Обоснование выбора среды моделирования 12](#_Toc91174094)

[2.2 Функциональное моделирование ИС, стандарт IDEF0 13](#_Toc91174095)

[2.2.1 Контекстная диаграмма 13](#_Toc91174096)

[2.2.2 Диаграмма декомпозиции 15](#_Toc91174097)

[ГЛАВА 3. ПРОЕКТИРОВАНИЕ ИО 17](#_Toc91174098)

[3.1 Концептуальная модель данных 17](#_Toc91174099)

[3.2. Логическая модель данных (ERD) 17](#_Toc91174100)

[3.3. Физическая модель системы 20](#_Toc91174101)

[ГЛАВА 4. ПРОЕКТИРОВАНИЕ ПО 24](#_Toc91174102)

[4.1 Требования к надежности 24](#_Toc91174103)

[4.1.1 Требования к обеспечению надежного (устойчивого) функционирования программы 25](#_Toc91174104)

[4.1.2 Время восстановления после отказа 25](#_Toc91174105)

[4.1.3 Отказы из-за некорректных действий оператора 25](#_Toc91174106)

[4.2 Требования к составу и параметрам технических средств 25](#_Toc91174107)

[4.3 Назначение и описание основных функций 26](#_Toc91174108)

[4.4. Диаграмма классов 26](#_Toc91174109)

[4.5. Диаграмма состояний 29](#_Toc91174110)

[4.6. Диаграмма деятельности 31](#_Toc91174111)

[ГЛАВА 5. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 34](#_Toc91174112)

[5.1. Методы программирования и средства разработки ПО. 34](#_Toc91174113)

[5.2. Диаграмма компонентов 34](#_Toc91174114)

[5.3. Диаграмма развертывания. 35](#_Toc91174115)

[5.4. Индивидуальное задание на разработку 36](#_Toc91174116)

[ЗАКЛЮЧЕНИЕ 38](#_Toc91174117)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 39](#_Toc91174118)

[Приложение 1 41](#_Toc91174119)

# ВВЕДЕНИЕ

Индустрия компьютерных игр (разработка, издание и продвижение игр) – одна из самых быстро развивающихся отраслей компьютерных технологий и одновременно глобального сектора развлечений. Игры становятся культурными феноменами и признаются произведениями искусства. Число геймеров растет (2,5 млрд игроков в 2019 г.), а сам гейминг становится высокооплачиваемой профессией. Формируется мощная экосистема. Вокруг ядра, состоящего из разработчиков игр и издателей, развиваются профильные СМИ, специальные финансовые и рекламные инструменты; появляются специализированные физические площадки (интернет-кафе нового поколения, локации для кибертурниров и др.); разработке игр начинают обучать в высших учебных заведениях и др. Кроме того, занять свою нишу хотят и игроки со смежных рынков – производители электроники (например, Apple) или интернет-корпорации (например, Google), обладающие всеми ресурсами для вывода на рынок перспективных игровых решений. Российская индустрия страдает нехваткой разработчиков игр. Для решения этой проблемы необходимо разработать игру, которая будет выступать на рынке как один из отечественных товаров.

**Объектом исследования** – разработка игры для игровой площадки «Steam».

**Предмет исследования** – проектирование образа и функционала игрового персонажа.

**Цель работы** – заполнить пустой сегмент российской индустрии игр, путём разработки игры жанра roguelike.

Для разработки курсового проекта необходимо будет завершить следующие этапы разработки:

* разработка концепции игры;
* проектирование игры;
* разработка дизайна;
* построение уровней;
* разработка скриптов;
* установка системы управления;
* разработка алгоритмов;
* разработка сюжета.

# ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область – игровая индустрия, мультипользовательские игры.

Игровая индустрия – Индустрия компьютерных игр (также индустрия интерактивных развлечений) — сектор экономики, связанный с разработкой, продвижением и продажей компьютерных игр. В неё входит большое количество специальностей, по которым работают десятки тысяч человек по всему миру [1].

Steam – онлайн-сервис цифрового распространения компьютерных игр и программ, разработанный и поддерживаемый компанией Valve. Steam выполняет роль средства технической защиты авторских прав, платформы для многопользовательских игр и потокового вещания, а также социальной сети для игроков. Программный клиент Steam также обеспечивает установку и регулярное обновление игр, облачные сохранения игр, текстовую и голосовую связь между игроками [2].

Компания Valve была основана 24 августа 1996 года двумя бывшими сотрудниками Microsoft, Гейбом Ньюэллом и Майком Харрингтоном, в Киркланде. Издателем Valve стала Sierra, пережившая в 1996 году тяжёлые времена. Для упрощения работы Valve приобретает лицензию Quake engine, на котором базировалась игра Quake. С этого момента начинается разработка Half-Life.

Рабочим названием игры было "Quiver". Для работы над сценарием был приглашён писатель-фантаст Марк Лэйдлоу. Игра стала научно-фантастическим трёхмерным шутером от первого лица, так как Ньюэлл и Харрингтон считали, что экшн является единственным жанром, где можно придумать что-нибудь новое. По задумке атмосфера Half-Life должна была походить на мрачную атмосферу Doom и, в некотором роде, стать её переосмыслением. Концепция мира игры должна была походить на повесть Стивена Кинга "Туман".

Впервые игру показали на выставке E3 в 1997 году, где она стала настоящим хитом и получила награду лучшая игра шоу. Релиз планировался на конец года, однако, по техническим причинам был перенесён. Ближе к концу 1997 года авторы игры решили, что большую часть сделанного необходимо переделать. Параллельно с Half-Life шла разработка другого не анонсированного проекта, которым, вероятно, должен был стать Prospero. Она была отменена для сосредоточения сил на разработке Half-Life.

Сначала Valve собиралась выпустить игру весной, затем в июне, потом просто летом, затем в сентябре и после этого, наконец, на Рождество. Постоянные переносы даты релиза служили причиной беспокойства в Sierra. На выставке E3 1998 года Half-Life снова получил награду лучшая игра шоу. До выпуска Half-Life компания выпустила демоверсию под названием Day One. Вскоре её стали бесплатно распространять вместе с видеокартами. Уже вскоре демоверсия обрела огромную популярность.

Когда разработка была близка к завершению оставалась лишь одна маленькая программная ошибка. Большая часть сотрудников уже ничего не делала. Когда ошибка была, наконец, найдена, была записана копия игры, которая была отправлена на тестирование. Релиз игры был отмечен в Valve уничтожением хедкраба-пиньяты. Наконец, после окончания разработки, по словам Марка Лэйдлоу, люди вернулись к уже забытой ими обычной жизни. После годового отпуска эти люди снова собрались вместе для работы над Team Fortress, а затем и над [Half-Life 2](https://halflife.fandom.com/ru/wiki/Half-Life_2).

Valve сосредоточилась на разработке SDK, который вышел в апреле 1999 года. Мин Ли, совместно с разработчиком, скрывающимся под ником «Cliffea», использует его для создания модификации для Half-Life под названием Counter-Strike (CS), многопользовательского шутера, основанного на идее противостояния группировки террористов и специального подразделения полиции. Игра стала одной из ведущих киберспортивных дисциплин.

В конце 1999 года Valve начала разработку [Half-Life 2](https://halflife.fandom.com/ru/wiki/Half-Life_2). Для новой игры компания разработала собственный движок Source Engine. Разработка держалась в строжайшем секрете, и впервые игру показали лишь в 2003 году на E3. Игра произвела неменьшее, чем первая часть, впечатление на посетителей выставки.

Новый движок стал отличным подспорьем для мододелов, и Valve предоставила сообществу SDK, документацию и набор утилит. Релиз Half-Life 2 был намечен на сентябрь 2003 года, однако, неожиданно произошла организованная хакером утечка файлов, и игра была отложена на неопределённый срок.

Игру пришлось делать снова. Весь год команда трудилась над игрой, существенно улучшив физику и графику. Half-Life 2 вышел 16 ноября 2004 года, спустя год после кражи. Игра собрала более 35 наград "Игра года". Официальная версия игры требовала регистрации Steam, только что вышедшей системе цифровой дистрибуции от Valve [3].

Примерно четверть. В России в целом хорошее образование, большое количество талантливых и хорошо образованных людей. Конкретно в играх есть одна большая проблема – этому не учат в вузах. Так что игровая индустрия буквально само зародилась много лет назад и сейчас само развивается.

Являясь лидером на рынке в России, мы делаем все, чтобы молодые команды приходили к нам и рассказывали про свои идеи, а мы им помогали. Общий цикл выглядит примерно так: найти очень хороший и интересный молодой коллектив с хорошей идеей, помочь им деньгами и опытом, помочь им с оперированием и маркетингом и в итоге добавить к себе новую студию, которая будет в чем-то уникальна [4].

В наше время сегмент российской индустрии игр в жанре roguelike плохо развит. Это свидетельствует о том, что разработка в данном жанре высоко востребована. Проведя опрос на игровом форуме, выяснилось, что многим игрокам данная задумка показалось очень интересной.

Обоснование необходимости разработки информационная системы:

1. Рынок не занят.
2. Заинтересованность российских игроков в играх жанра roguelike.
3. Разработка подобных игр не занимает много времени и ресурсов.

Вывод: в данной главе была разобрана предметная область. Был проведён опрос, в котором выяснилось, что разработка игры на российском рынке в жанре roguelike очень перспективна.

# ГЛАВА 2. МОДЕЛИРОВАНИЕ ДЛЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Система предназначена для развлечения любителей игр жанра roguelike.

Основным назначением системы является развитие российского рынка видеоигр.

Игровой процесс игр жанра roguelike в целом вдохновляется [Rogue](https://ru.wikipedia.org/wiki/Rogue), однако отдельные игры могут значительно отличаться друг от друга. В 2008 году на конференции International Roguelike Development Conference 2008 в Берлине участники попытались определить ключевые факторы, выделяющие roguelike среди других игр; их определение известно как «Берлинская интерпретация» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Berlin interpretation). Берлинская интерпретация выделила ряд ключевых факторов, позволяющих определить игру как roguelike:

* игра должна быть [пошаговой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0), каждая команда должна соответствовать одному действию и одному ходу;
* и[гровые уровни](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C_(%D0%B8%D0%B3%D1%80%D1%8B)) должны генерироваться случайным образом, будучи уникальными для каждого прохождения;
* игра должна содержать «[перманентную смерть](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D1%81%D0%BC%D0%B5%D1%80%D1%82%D1%8C)», не позволяя игроку продолжить прохождение после гибели персонажа;
* игра должна иметь единый режим и единый набор команд для всех игровых ситуаций, не допуская каких-либо дополнительных меню, головоломок или [мини-игр](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8-%D0%B8%D0%B3%D1%80%D0%B0);
* игра должна предоставлять игроку не какой-то единый линейный путь, а свободу со множеством вариантов прохождения;
* игрок должен самостоятельно исследовать найденные предметы и открывать их свойства.

Система позволяет игроку использовать различные уникальные механики для прохождения игры, такие как:

* cмена гравитации;
* решение головоломок;
* замедление времени;
* телепорт;
* система квестов.

Система состоит из следующих этапов разработки:

* создание игрового сюжета;
* проектирование образа и функционала игрового персонажа;
* проектирование образа и функционала игровых уровней;
* проектирование образа и функционала NPC.

Создание игрового сюжета — это вид сюжета, в котором игрок, попадая в игровую ситуацию имеет несколько вариантов решения игровой задачи, каждый из которых приводит к отличному от остальных результату. Игрок может получать разные бонусы, менять глобальные аспекты игрового мира, или даже зайти в тупик приняв то или иное решение.

Проектирование образа и функционала игрового персонажа —для этого рисуется набросок самого персонажа. Но перед этим требуется знать, что именно персонаж будет делать, как двигаться и множество других мелочей, например: взаимодействие с объектами.

Проектирование образа и функционала игровых уровней—это создание уровней, для которых необходимо спроектировать такую локацию, которая обеспечит достаточное количество интересных игровых ситуаций для применения каждой базовой механики.

Проектирование образа и функционала NPC — рисуется набросок самого персонажа. Но перед этим требуется знать, что именно персонаж будет делать, как двигаться и множество других мелочей, например: взаимодействие с объектами, этот персонаж в играх, который не находится под контролем игрока в компьютерных играх поведение таких персонажей определяется ИИ [5].

Состав документации для всех этапов жизненного цикла проекта представлен в таблице 1.

Таблица 1 – состав документации.

|  |  |
| --- | --- |
| **Этап** | **Документ** |
| Проектирование. Разработка технического задания. Разработка спецификаций. | Ведомость технического проекта |
| Пояснительная записка к техническому проекту |
| Описание автоматизируемых функций |
| Описание комплекса технических средств |
| Разработка. Дизайн. Кодирование. Тестирование. Документирование. | Ведомость рабочей и эксплуатационной документации |
| Формуляр |
| Общее описание системы |
| Руководство пользователя |
| Технологическая инструкция |
| Программа и методика испытаний |
| Ввод в эксплуатацию, техподдержка. Внедрение. Сопровождение | Протокол развертывания системы |
| Протокол первоначального заполнения БД |
| Протокол предварительных испытаний |
| Акт приемки в опытную эксплуатацию |
| Журнал опытной эксплуатации |
| Акт о завершении опытной эксплуатации |
| Протокол приемочных испытаний |
| Акт приемки системы в постоянную эксплуатацию |
| Формуляр |

## 2.1. Обоснование выбора среды моделирования

Для моделирования системы использовались следующие среды – StarUML и Draw.io.

StarUML — это программный инструмент визуального моделирования с открытым исходным кодом, который поддерживает стандартизованный язык графического описания UML (Unified Modeling Language) для моделирования систем и программного обеспечения.

Программный продукт StarUML от разработчика MKLabs предназначен для создания и применения графических моделей в нотации UML. Система основана на UML 2. 0 и предоставляет одинадцать различных видов диаграмм, активно поддерживая таким образом подход построения архитектур на базе моделей (англ. Model Driven Architecture, MDA). Система может эффективно применяться системными аналитиками, проектировщиками и архитекторами систем, инженерами-программистами.

Программное обеспечение StarUML отличается высокой настраиваемостью в соответствии с пользовательской средой и высокой расширяемостью в своей функциональности. Использование StarUML обеспечивает высокую производительность и качество программных проектов [6].

Draw.io — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

Инструмент работает с Google Диск, Google Workspace и Dropbox, глубоко интегрирован и удобен для работы с продуктами Confluence и Jira от Atlassian. Пользователи также могут работать с диаграммами в автономном режиме и сохранять их локально, используя настольное приложение для персональных компьютеров.

Инструмент позволяет создавать: графики, диаграммы, таблицы, презентации, блок-схемы, планы помещений, воронки продаж, ментальный карты, карты сайтов [7].

## 2.2 Функциональное **моделирование ИС, стандарт IDEF0**

Для описания процессов информационной системы использовалось функциональное моделирование в стандарте IDEF0.

Функциональное моделирование — это процесс моделирования функций, выполняемых рассматриваемой информационной системой/объектом, путем создания описательного структурированного графического изображения, показывающего что, как и кем делается в рамках функционирования объекта и объектов, связывающих эти функции, с учетом имеющейся информации [8].

### 2.2.1 Контекстная диаграмма

Контекстная диаграмма — вид IDEF0-диаграммы. Это диаграмма, расположенная на вершине древовидной структуры диаграмм, представляющая собой самое общее описание системы и ее взаимодействие с внешней средой (как правило, здесь описывается основное назначение моделируемого объекта). Контекстная диаграмма состоит из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизмы, вместе с формулировками цели модели и точки зрения, с которой строится модель.

На рисунке 1 изображена контекстная диаграмма, которая представленная была создана в онлайн воркфрейме draw.io.

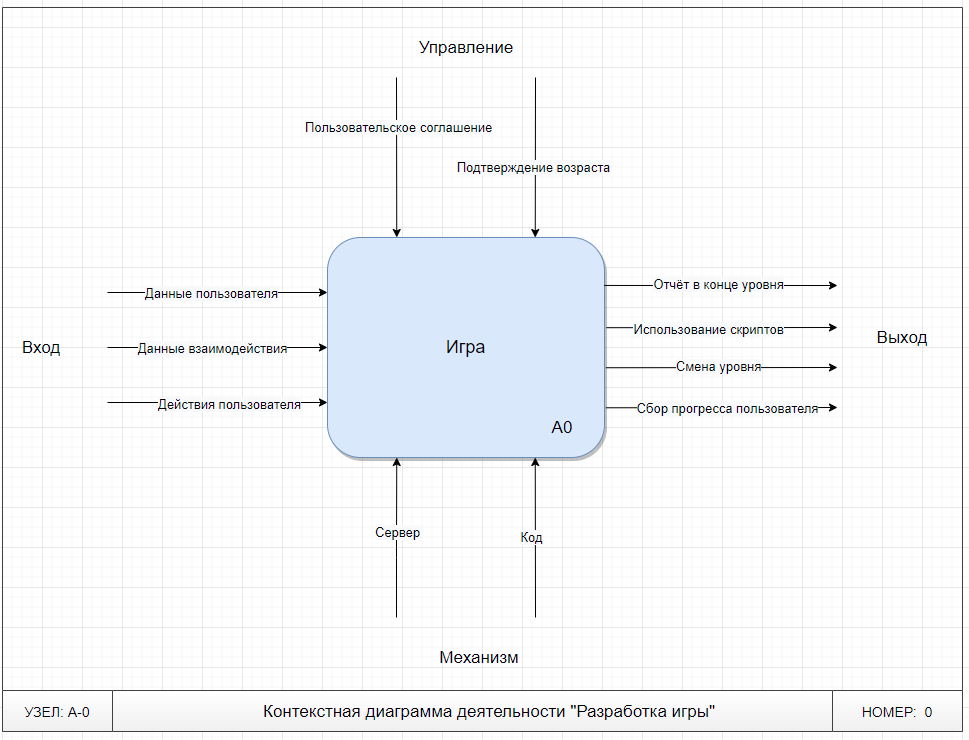


Рис. 1 – контекстная диаграмма игровой сессии.

Входящие потоки:

* данные пользователя;
* данные взаимодействия;
* действия пользователя.

Управляющие потоки:

* пользовательское соглашение;
* подтверждение возраста.

Механизмы:

* сервер;
* код.

Выходящие потоки:

* результат действий пользователя;
* смена локации;
* вывод статистики.

### 2.2.2 Диаграмма декомпозиции

Диаграмма декомпозиции предназначена для детализации функций и получаются при разбиении контекстной диаграммы на крупные подсистемы (функциональная декомпозиция) и описывающие каждый подсистему и их взаимодействие.

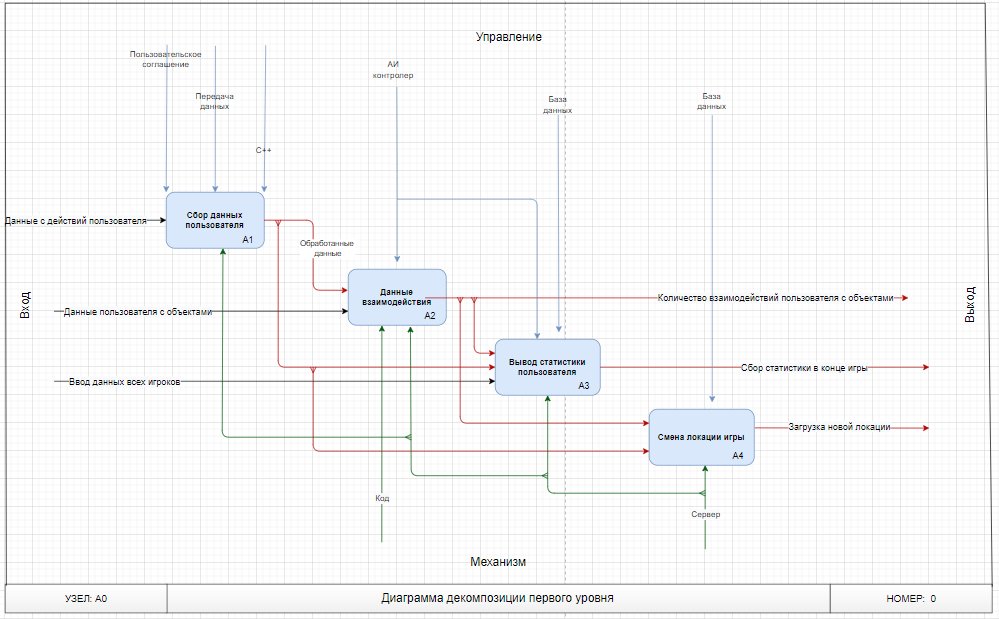
Единственная функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции посредством создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части посредством создания дочерней диаграммы следующего, более низкого уровня, на которой некоторые или все функции также могут быть разложены на составные части. Каждая дочерняя диаграмма содержит дочерние блоки и стрелки, обеспечивающие дополнительную детализацию родительского блока, на рисунке 2 изображена диаграмма декомпозиции нашего проекта.

Рис. 2 – диаграмма декомпозиции игровой сессии.

Работа «Cбор данных пользователя» на входе получает персональные данные пользователя и данные. В качестве управление выступает пользовательское соглашение.

Работа «Данные взаимодействия» на входе получает данные действиях пользователя.

Работа «Вывод статистики пользователя» на входе получает данные о всех проделанных действия пользователя.

Работа «Смена локации игры» на входе получает данные о прохождении локации и загружает новую.

В диаграмме декомпозиции полный цикл процесса пользователя в игре:

* 1. Пользователь соглашается с предоставлением данных и начинает игру один либо в команде.
  2. После этого передаются данные о действиях пользователя и его прогрессе.
  3. После получения все данных выводится статистика пользователя о всех совершенных действий.
  4. После происходит смена локаций.

Вывод: в данной главе были представлены основные этапы создания проекта его видения и свойств, потоки данных и свойства данных потоков.

# ГЛАВА 3. ПРОЕКТИРОВАНИЕ ИО

Информационное обеспечение – создание информационных условий функционирования системы, обеспечение необходимой информацией, включение в систему средств поиска, получения, хранения, накопления, передачи, обработки информации, организации банков данных.

## 3.1 Концептуальная модель данных

Диаграмма DFD наглядно отображает течение информации в пределах процесса или системы. Для изображения входных и выходных данных, точек хранения информации и путей ее передвижения между источниками и пунктами доставки в таких диаграммах применяются стандартные фигуры, такие как прямоугольники и круги, а также стрелки и краткие текстовые метки на рисунке 3 изображена концептуальная модель данных нашего проекта.

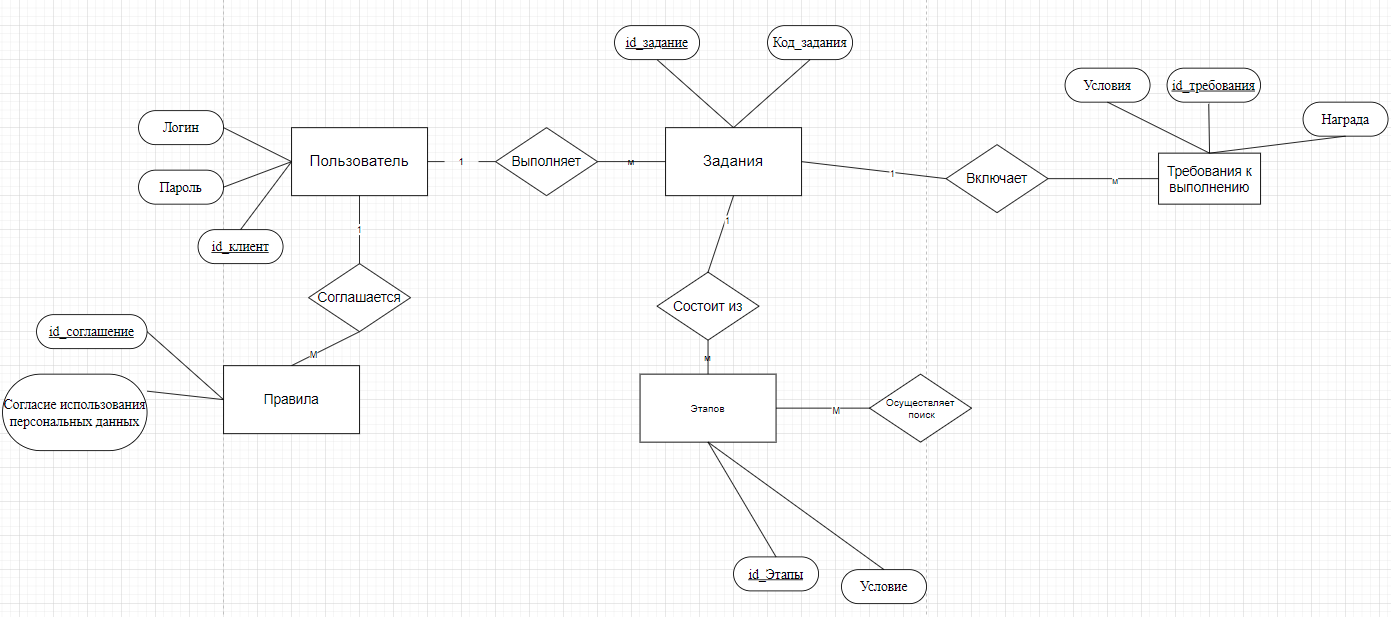


Рис. 3 – концептуальная модель данных.

## 3.2. Логическая модель данных (ERD)

Диаграммы отношений сущностей (ERD) – это визуальные представления баз данных, которые показывают, как элементы в базе данных связаны друг с другом. ERD состоит из двух типов объектов сущностей и отношениями. У отношений есть специальные окончания строк, называемые мощностями, которые описывают, как два элемента базы данных взаимодействуют друг с другом, на рисунке 4 изображена логическая диаграмма нашего проекта.

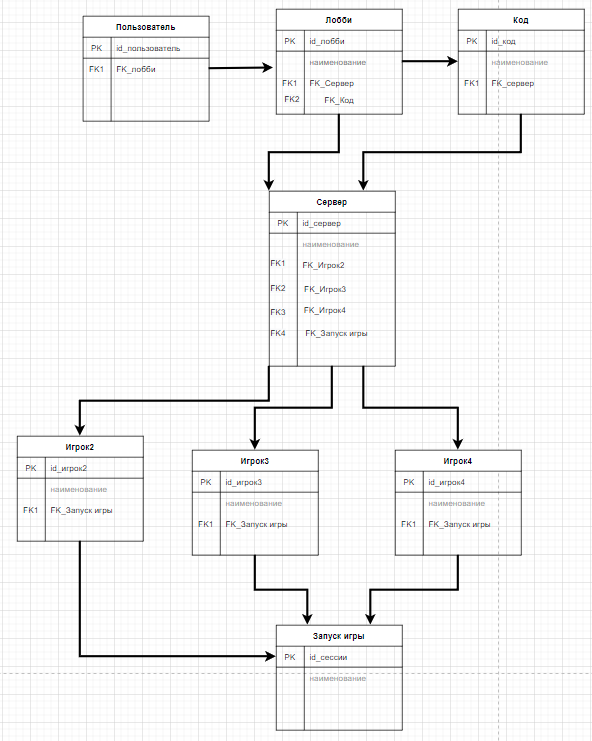


Рис. 4 – логическая диаграмма.

Сервер – сервер, на котором будет происходить подключение.

Лобби – комната, в которую подключаются игроки.

Код – система, которая управляет всеми функциями игры.

Запуск игры – функция запуска игровой сессии.

Пользователь – игрок, который создаёт лобби (общедоступный сервер) для игроков.

Игрок2 – игрок, который подключается к лобби

Игрок3 – игрок, который подключается к лобби

Игрок4 – игрок, который подключается к лобби

Диаграмма IDEF1X – этот метод моделирования используется для моделирования данных стандартным, последовательным и предсказуемым образом, чтобы управлять ими как ресурсом. Он может быть использован в проектах, требующих стандартных средств определения и анализа ресурсов данных внутри организации. Такие проекты включают в себя включение метода моделирования данных в методологию, управление данными как ресурсом, интеграцию информационных систем или проектирование компьютерных баз данных, на рисунке 6 изображена диаграмма IDEF1X нашего проекта.

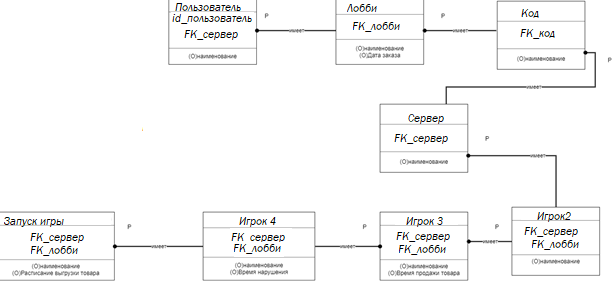


Рис. 6. – диаграмма IDEF1X

## 3.3. Физическая модель системы

Физическая модель базы данных — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных, на рисунке 7 изображена диаграмма физической модели нашего проекта.

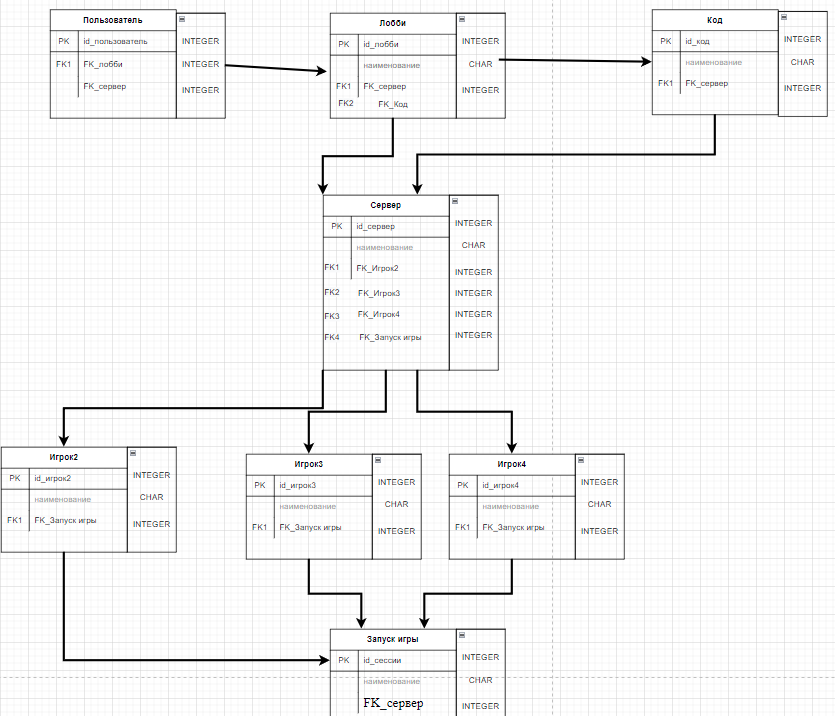


Рис. 7. – диаграмма физической модели.

Спецификация сущностей:

**Пользователь**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ключ** | **Наименование атрибута** | **Тип данных** | **Значение** | **Описание** |
| Первичны  й ключ | id\_пользователь | INTEGER | AUTO INCRIMENT | Идентификатор пользователь |
| Вторичны  й ключ | FK\_сервер | INTEGER | [1..] | Идентификатор сервер |
| Вторичны  й ключ | FK\_лобби | INTEGER | [1..] | Идентификатор лобби |
| Вторичны  й ключ | FK\_код | INTEGER | [1..] | Идентификатор код |
| - | Наименование | CHAR | Строка | Название |

**Лобби**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ключ** | **Наименование атрибута** | **Тип данных** | **Значение** | **Описание** |
| Вторичны  й ключ | FK\_лобби | INTEGER | [1..] | Идентификатор лобби |
| Вторичны  й ключ | FK\_сервер | INTEGER | [1..] | Идентификатор сервер |
| Вторичны  й ключ | FK\_код | INTEGER | [1..] | Подключение кода |
| - | Наименование | CHAR | Строка | Название |

**Сервер**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ключ** | **Наименование атрибута** | **Тип данных** | **Значение** | **Описание** |
| Вторичны  й ключ | FK\_сервер | INTEGER | [1..] | Идентификатор сервера |
| - | Наименование | CHAR | Строка | Наименование |
| Вторичны  й ключ | FK\_игрок | INTEGER | [1..] | Идентификатор игрока |
| Вторичный ключ | FK\_сессия | INTEGER | [1..] | Идентификатор сессии |

**Игрок**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ключ** | **Наименование атрибута** | **Тип данных** | **Значение** | **Описание** |
| Вторичны  й ключ | FK\_игрок | INTEGER | [1..] | Идентификатор игрока |
| - | Наименование | CHAR | Строка | Наименование |
| Вторичны  й ключ | FK\_сессия | INTEGER | [1..] | Идентификатор сессии |

**Запуск игры**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ключ** | **Наименование атрибута** | **Тип данных** | **Значение** | **Описание** |
| Вторичны  й ключ | FK\_сессия | INTEGER | [1..] | Идентификатор сессии |
| Вторичны  й ключ | FK\_лобби | INTEGER | [1..] | Идентификатор лобби |
| Вторичны  й ключ | FK\_сервер | INTEGER | [1..] | Идентификатор сервера |
| **-** | Наименование | CHAR | Строка | Наименование |

Вывод: в данной главе были рассмотрены основные модули будущей системы.

# ГЛАВА 4. ПРОЕКТИРОВАНИЕ ПО

Программное обеспечение для торговой площадки “Steam” представляет собой приложение, на котором будут реализованы подсистемы регистрации, подключения на сервер.

При регистрации на сайте пользователь должен иметь возможность создать свой аккаунт и зайти на сервер, и в случае необходимости изменить его данные.

Требования к составу, структуре, и функциональным характеристикам программных средств.

В состав технических средств входит:

* компьютер оператора, включающий в себя:
* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативную память объемом, не менее 1 Гб;
* видеокарту, монитор, мышь, клавиатура.

Компьютер пользователя, включающий в себя:

* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативную память объемом, не менее 1 Гб;
* видеокарту, монитор, мышь.
* два компьютера для СУБД (основной и резервный), включающий в себя:
* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативную память объемом, не менее 1 Гб;
* видеокарту, монитор, мышь.

## 4.1 Требования к надежности

Вероятность безотказной работы системы должна составлять не менее 99.99% при условии исправности сети (связи приложений оператора и пользователя с базой данных).

### 4.1.1 Требования к обеспечению надежного (устойчивого) функционирования программы

В связи с тем, что в базе данных хранятся данные о совершенных клиентами покупках (финансовая информация) — базу данных стоит резервировать (резервирование замещением).

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* организацией бесперебойного питания технических средств;
* использованием лицензионного программного обеспечения.

### 4.1.2 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 10 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

### 4.1.3 Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

## 4.2 Требования к составу и параметрам технических средств

Состав технических средств:

Компьютер оператора, включающий в себя:

* + процессор x86 с тактовой частотой, не менее 1 ГГц;
  + оперативную память объемом, не менее 1 Гб;
  + видеокарту, монитор, мышь, клавиатура.

Компьютер посетителя, включающий в себя:

* + процессор x86 с тактовой частотой, не менее 1 ГГц;
  + оперативную память объемом, не менее 1 Гб;
  + видеокарту, монитор, мышь.

Два компьютера для СУБД (основной и резервный), включающий в себя:

* + процессор x86 с тактовой частотой, не менее 1 ГГц;
  + оперативную память объемом, не менее 1 Гб;
  + видеокарту, монитор, мышь.

## 4.3 Назначение и описание основных функций

В игре реализован интерфейс, лобби, чат,

## 4.4. Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования, на рисунке 8 изображена диаграмма классов обновления игры нашего проекта.

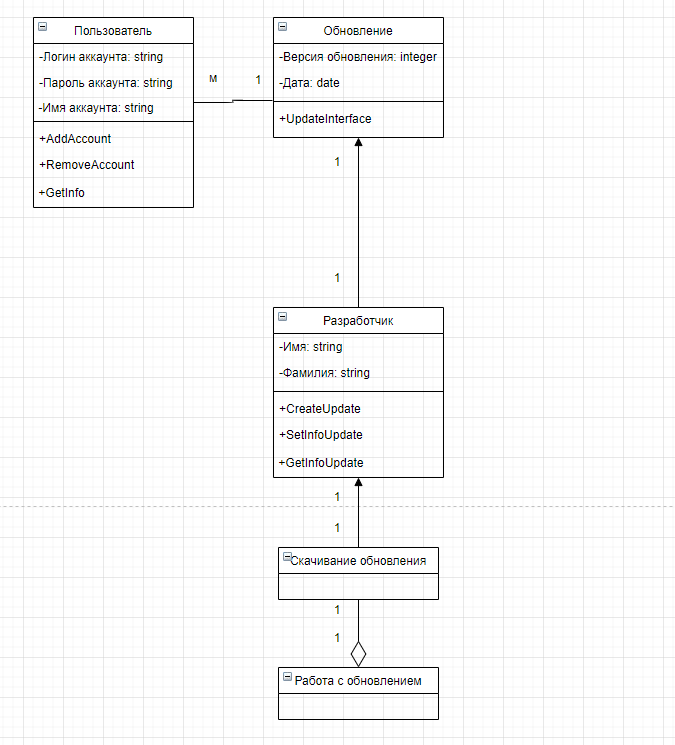


Рис. 8 – диаграмма классов обновления игры

Описание связей:

Таблица 7 представляет собой описание класса.

Таблица – 7 класс Пользователь

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой пользователя игры |
| Атрибуты | Логин аккаунта: string  Пароль аккаунта: string  Имя аккаунта: string |
| Операции | AddAccount – создание аккаунта  RemoveAccount – удаление аккаунта  GetInfo – получить информацию об аккаунте |

Таблица 8 – класс Обновление

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой обновление игры |
| Атрибуты | Версия обновления: integer  Дата: data |
| Операции | UpdateInterface – обновление интерфейса игры |

Таблица 9 – класс Разработчик

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой разработчика игры |
| Атрибуты | Имя: string  Фамилия: string |
| Операции | CreateUpdate – создать обновление  SetInfoUpdate – занести информацию об обновлении  GetInfoUpdate – получить информацию об обновлении |

Таблица 10 – класс Скачивание обновления

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой скачивание обновления игры |
| Атрибуты |  |
| Операции |  |

Таблица 11 – класс Работа с обновлением

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой работу с обновлением |
| Атрибуты |  |
| Операции |  |

## 4.5. Диаграмма состояний

Диаграмма состояний представляет собой состояния, соединенные переходами. Переходы могут сработать в результате заданного в качестве условия перехода события – это может быть истечение заданного таймаута, получение диаграммой состояний сообщения, выполнение заданного логического условия и т.д. Срабатывание перехода приводит к переходу управления диаграммы состояний в то состояние, в которое ведет этот переход. Состояния могут быть иерархическими, т.е. содержать другие состояния и переходы, на рисунке 9 изображена диаграмма состояний нашего проекта.

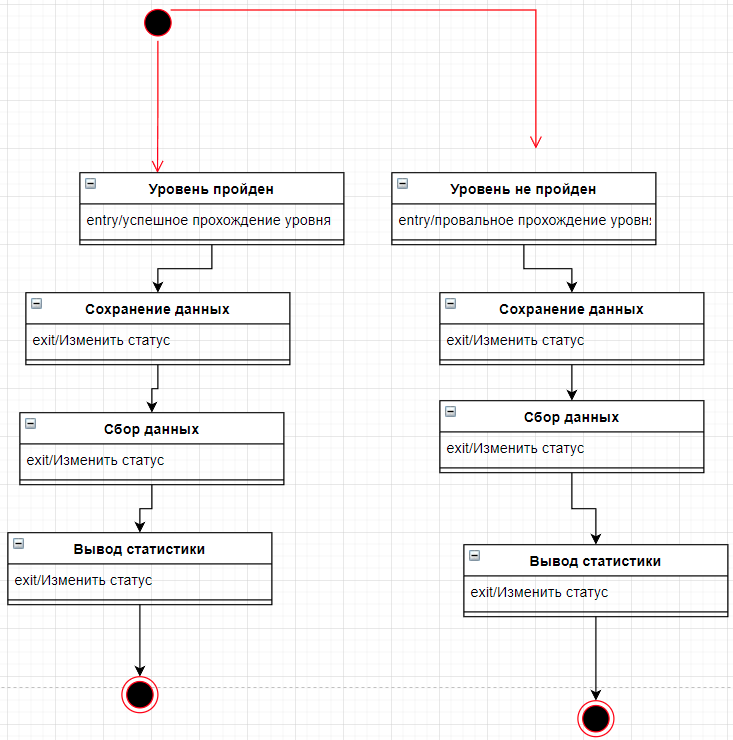


Рис. 9 – диаграмма состояний.

В таблице 12 представлена описание диаграммы состояний

Таблица 12 – описание диаграммы состояний.

|  |  |
| --- | --- |
| **Состояние** | **Описание состояния** |
| Уровень пройден | Присваивается, когда уровень пройден, происходит смена уровня |
| Сохранение данных | Присваивается уровень провален или пройден |
| Сбор данных | Присваивается, когда хранилище данных совершает сбор данных |
| Вывод статистики | Присваивается, когда данные по статистике передаются пользователю |
| Уровень не пройден | Присваивается, когда уровень провален, происходит рестарт |

## 4.6. Диаграмма деятельности

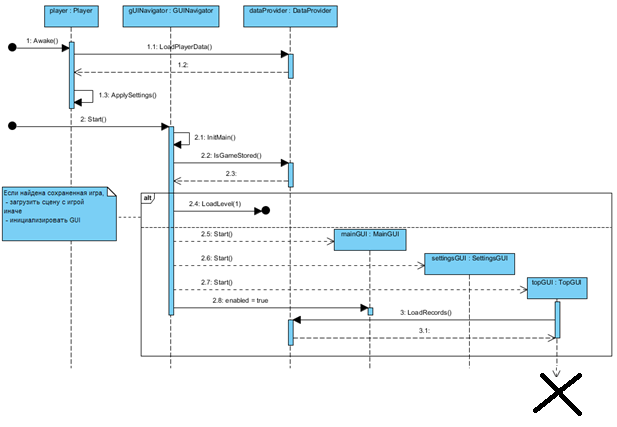
Диаграмма деятельности — UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

На рисунке 10 представлена диаграмма деятельности запуска приложения.

Пользователь выбирает различные функции, которые ему предоставлены в меню.

На рисунке 11 представлена создание объектов в игре в нём описывается пример, как создаётся объект.

На рисунке 12 представлена диаграмма деятельности запуска игры в нём описано как пользователь взаимодействует с меню в игре.

Рис. 10 – диаграмма деятельности запуска приложения.

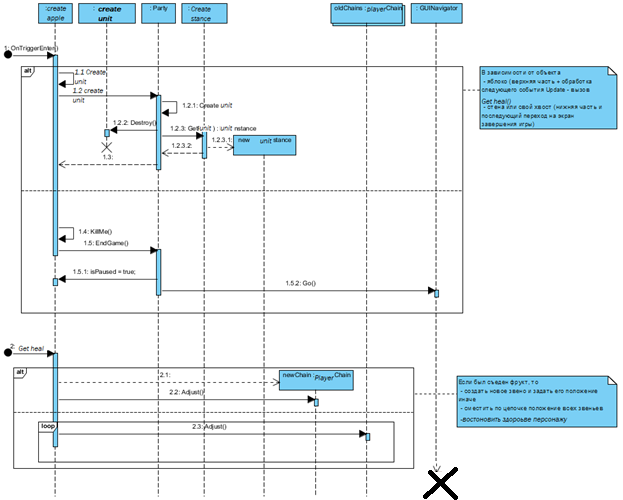


Рис. 11 – создание объектов в игре.

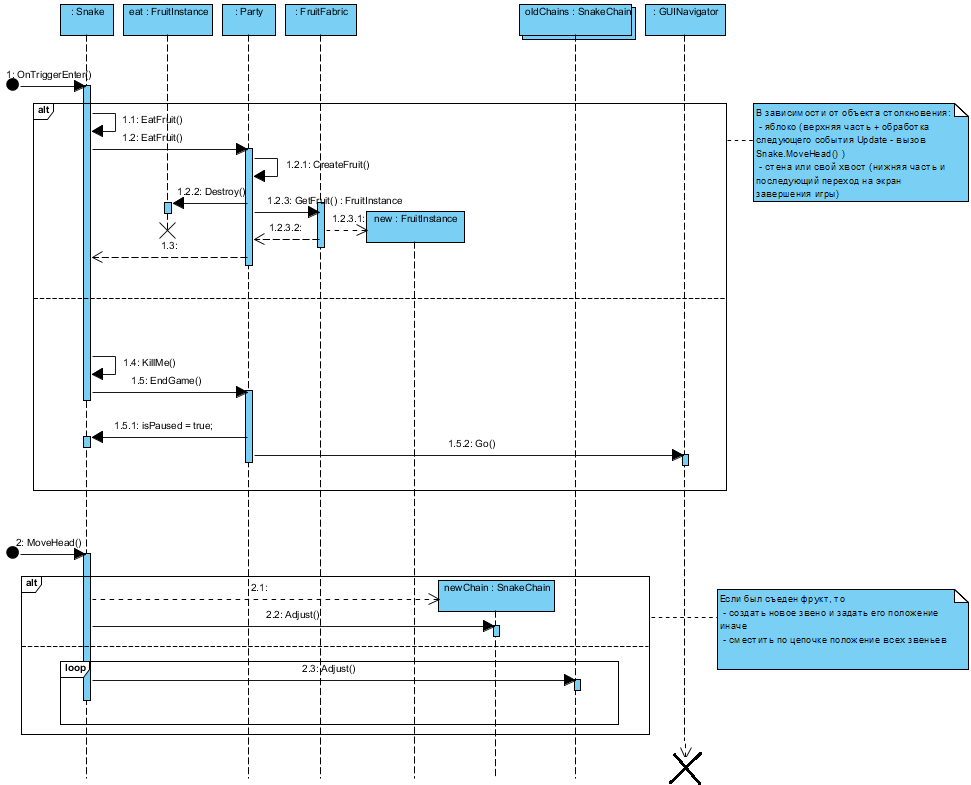


Рис. 12 – диаграмма деятельности запуска игры.

Вывод: в данной главе были рассмотрены состав ИС, режим работы как всей системы, так и отдельных её частей.

# ГЛАВА 5. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В нашем проекте для эффективной работы проектной группы была применена гибкая методология agile, на платформе Trello.

## 5.1. Методы программирования и средства разработки ПО.

Unreal Engine 4 — это набор инструментов для разработки игр, имеющий широкие возможности: от создания двухмерных игр на мобильные до AAA-проектов для консолей. Этот движок использовался при разработке таких игр, как ARK: Survival Evolved, Tekken 7 и Kingdom Hearts III.  
Разработка в Unreal Engine 4 Нашей проектной группой был выбран автоматизированный способ.

Преимущество автоматизированной методики состоит в том, что они позволяют разделять содержание и дизайн ресурса. То есть можно скорректировать дизайн, не затрагивая содержимое.

## 5.2. Диаграмма компонентов

**Диаграмма компонентов** — элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п., на рисунке 13 изображена диаграмма компонентов нашего проекта.

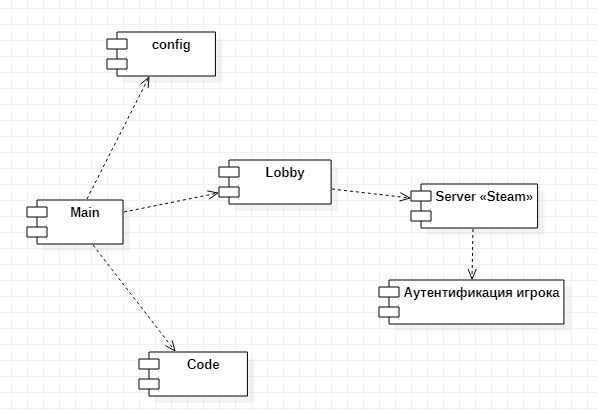


Рис. 13. – диаграмма компонентов.

Code – основные компоненты кода.

Lobby- игровая сессия созданная игроком

Сервер Steam - система управления потоком данных между игроками.

Аутентификация игрока - проверке подлинности данных о пользователе сервером.

Config-конфигурационный файл программы, в котором записаны настройки этой самой программы.

## 5.3. Диаграмма развертывания.

Диаграмма развертыванияпредназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются, на рисунке 14 изображена диаграмма развертывания нашего проекта.

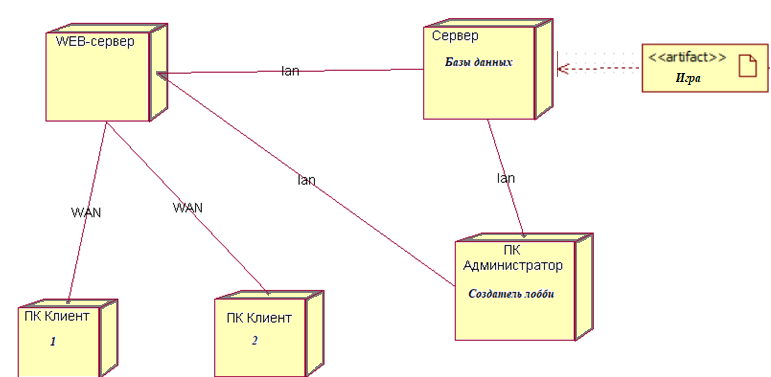


Рис. 14. – диаграмма развертывания.

Узлы:

Web Сервер – сервер на котором будет происходить подключение.

Сервер БД – сервер, на котором храниться БД игры.

Пк Администратор – игрок, который создаёт лобби (общедоступный сервер) для игроков

Пк Клиент – игрок, который подключается к лобби

Артефакты:

Игра – код игры.

5.4. Индивидуальное задание на разработку.

Был разработан макет игры, спроектирован образ и функционал игрового персонажа, и некоторая логика объектов ([Приложение 1](#_Приложение_1)).

Шаблон игры — это проектируемая игра с определённой логикой объектов. Дизайном, сценарием и разработкой кода для шаблона предварительно разрабатываются и реализуются рядом профессионалов — дизайнером, сценаристом, разработчиком.

Разработан макет игры на бесплатном движке unreal engine 4. Ниже приведены макеты вкладок игры.

В Приложении 1 рис.1 представлена демо версия игры, в которой реализован функционал и объекты начальной локации.

В Приложении 1 рис.2, рис.3 представлены начальные механики игры в виде нанесения урона путём атаки и передвижение персонажа.

В Приложении 1 рис.4, рис.5 представлена визуальное программирование объектов.

Визуальное программирование — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста.

# ЗАКЛЮЧЕНИЕ

При выполнении курсовой работы были выполнены следующие работы.

Описание предметной области в данной главе были рассмотрены предмет и объект исследования, компания заказчик, проблема и метод ее решения.

Проектирование ио где были разработаны концептуальная логическая и физическая модель системы. В данной главе была рассмотрена концепция будущей рабочей системы.

Проектирование по в данной главе были рассмотрены основные требования к системе, состав ИС, режим работы как всей системы, так и отдельных её частей.

Разработка программного обеспечения в данной главе была разработана концепция будущей системы сценариев деятельности пользователя, а также сценарии.

По результатам работы был разработана игра, в которой были реализована логика объектов и уровней с реализованной на нем функцией регистрации.

Все запланированные функции были внедрены в игру.

По оценкам рынка предоставления различных аналогов наш жанр игр довольно популярен и будет пользоваться спросом.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Valve [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Valve
2. Игровая площадка Steam [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Steam
3. История Valve [Электронный ресурс]. — Режим доступа: https://www.brandpedia.ru/brand-842.html
4. Российский рынок игр [Электронный ресурс]. — Режим доступа: https://www.vedomosti.ru/media/characters/2020/12/22/852053-rinok-igr
5. Жанр игр roguelike [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Roguelike.
6. Star UML [Электронный ресурс]. — Режим доступа https://freeanalogs.ru/StarUML.
7. Draw IO [Электронный ресурс]. — Режим доступа: https://startpack.ru/application/draw-io
8. IDEF0 [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/IDEF0
9. Истратова Е. Е. Сравнительный анализ свободного программного обеспечения для 3D-моделирования [Текст] / Е. Е. Истратова, Ю. С. Черний // Творчество и современность. — 2017. — №1 (2). — С. 120–125.
10. Обзор приложений для создания 3d моделей — для новичков и профессионалов [Электронный ресурс]. — Режим доступа: https://3dexpo.ru/ru/article/programmy-dlya-sozdaniya-3d-modeley
11. Обзор программы ZBrush: особенности и функционал [Электронный ресурс]. — Режим доступа: https://3ddevice.com.ua/blog/3d-printerobzor/obzor-programmy-zbrush/
12. Обзор самых популярных движков для разработки игр [Электронный ресурс]. — Режим доступа: https://xakep.ru/2014/09/05/game-developmentengines-review/#toc05
13. . Официальный сайт Blender [Электронный ресурс]. — Режим доступа: <http://www.blender.org/>
14. Курсы по созданию игр [Электронный ресурс]. — Режим доступа: <https://itproger.com/courses/games>
15. Курсы по геймдизайну [Электронный ресурс]. — Режим доступа: https://proity.ru/gamedev/gamedesign/

# Приложение 1



Рис. 1 – Демо версия игры

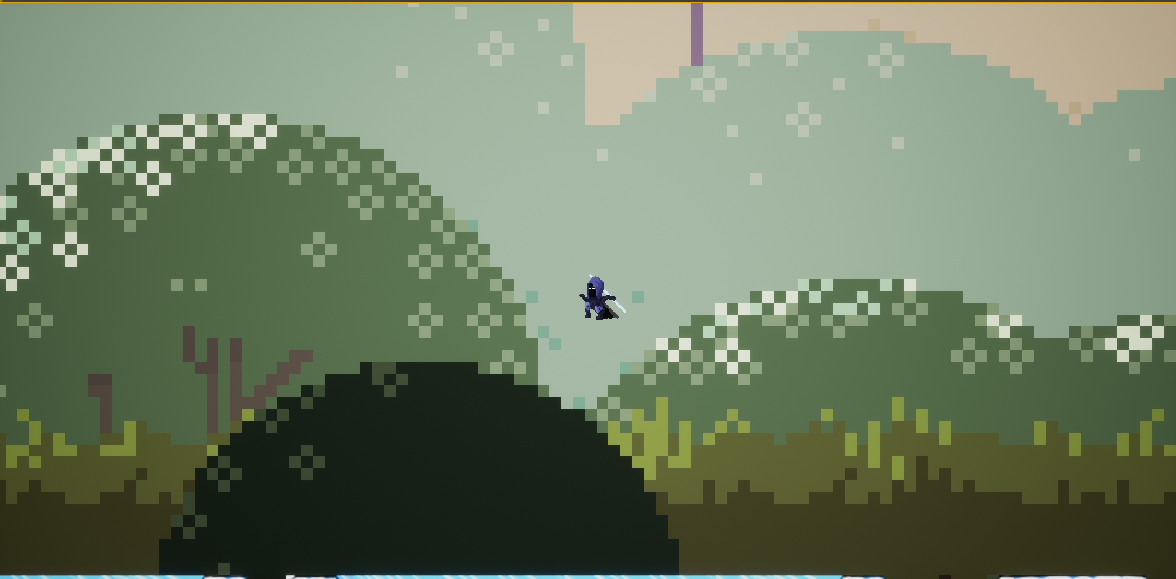


Рис. 2 – Срабатывание анимации при прыжке.

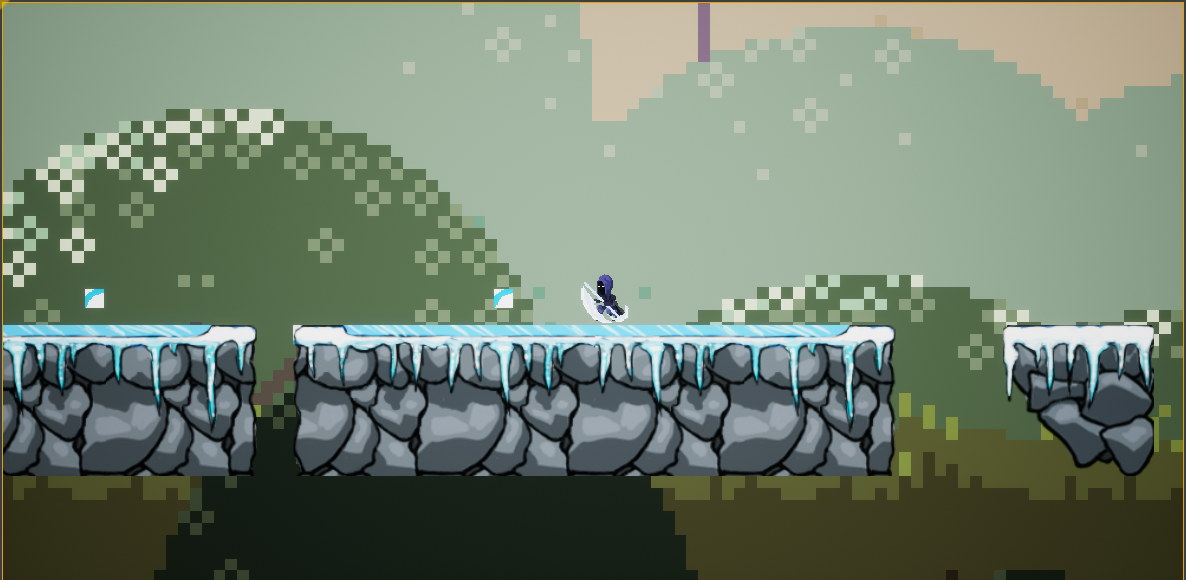


Рис. 3 – Срабатывание анимации атаки.

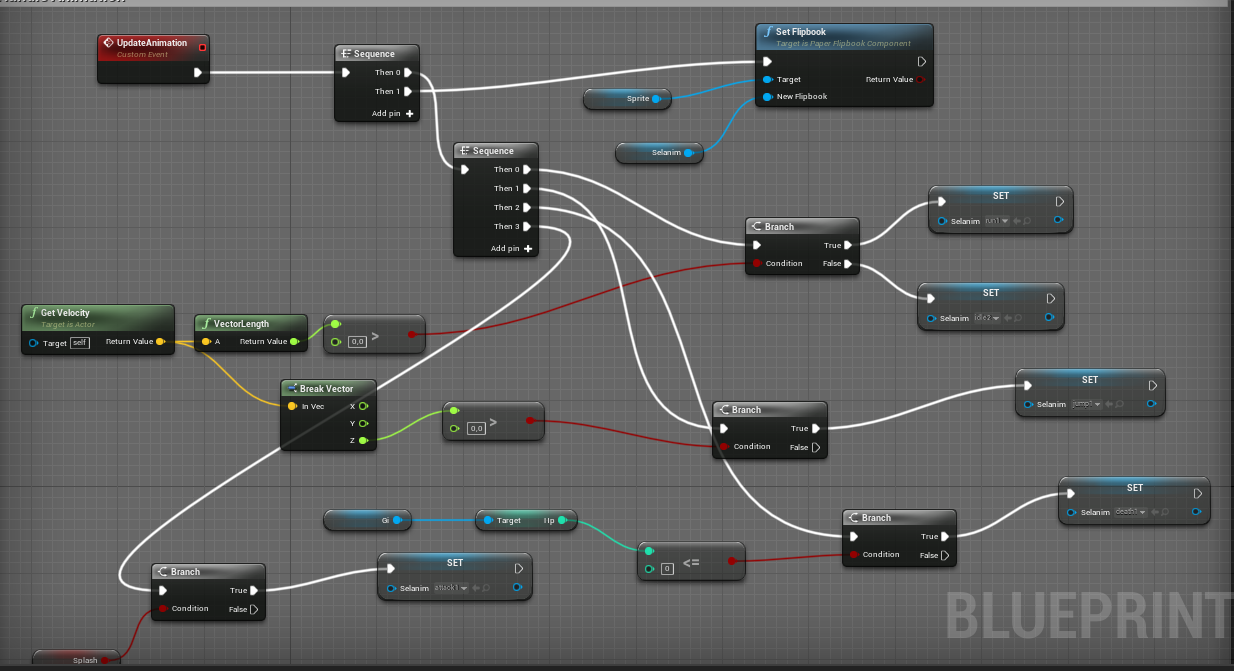


Рис. 4 – Визуальное программирование логика анимации.

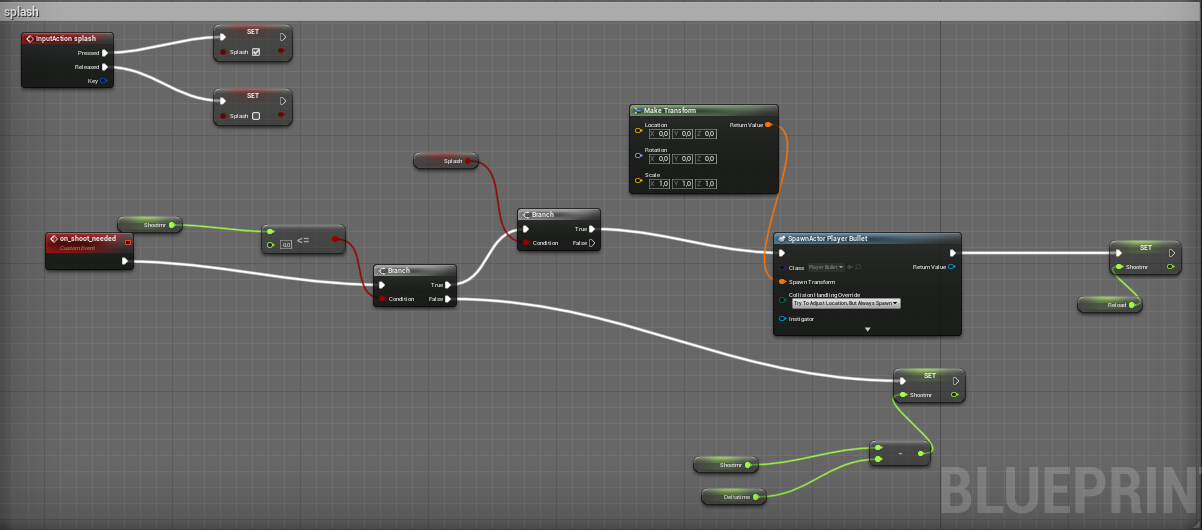


Рис. 5 – Визуальное программирование атаки.