Главное управление образования Гродненского облисполкома

Учреждение образования

«Гродненский государственный политехнический колледж»

Специальность: 2 – 40 01 01 «Программное обеспечение информационных

технологий»

Специализация: 2 – 40 01 01 35 «Программное обеспечение обработки

экономической и деловой информации»

Дисциплина: «Технология разработки программного обеспечения»

Группа: ПЗТ – 36

**ТЕХНО-РАБОЧИЙ ПРОЕКТ**

Техническое задание на тему: Разработка игрового приложения «Tokyo Ghoul»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Разработал А.Д.Ромпало

Руководитель проекта Е.В. Заяц

2022

**Содержание**

Изм.

Кол

Лист №док

Подпись

Дата

Лист

2

КП 2-40 01 01.35.36.04.22 ПЗ

Разраб.

Ромпало

Пров.

Заяц

Н. контр.

Утв.

Техническое задание на тему:

Разработка игрового приложения ”Tokyo Ghoul”

Стадия

Листов

УО ГГПК

1. Постановка задачи 3
   1. Организационно-экономическая сущность задачи………………3
   2. Функциональные требования……………………………………...4

1.2.1 Регистрация и вход в аккаунт……………………………………4

1.2.2 Игровой процесс………………………………………………….4

1.2.3 Настройки…………………………………………………………5

1. Техническое задание 8
2. Реализация…………………………………………………………...14
3. Проектирование ПИ…………………………………………………18
   * 1. UX – Проектирование……………………………………………18
     2. UI – Проектирование……………………………………………..19
     3. Тестирование и отладка ПО ……………………………………..20
4. Приложение А………………………………………………………..23

5.1 Диаграмма вариантов использования……………………………...24

5.2 Функциональная модель (1)………………………………………...25

5.3 Функциональная модель (2)………………………………………...25

5.4 Функциональная модель (3)………………………………………...26

5.5 Модель данных………………………………………………………27

5.6 Диаграмма объектов………………………………………………....28

5.7Диаграмма последовательности…………………………………….29

5.8Диаграмма деятельности…………………………………………….30

5.9Диаграмма компонентов……………………………………………..31

1. Прототип ПП…………………………………………………………33

**1 Постановка задачи**

**1.1 Организационно-экономическая сущность задачи**

**Наименование задачи:** Игра тамагочи “Токийский гуль”

**Иконка игры: (Рисунок -1)**



**Рисунок - 1**

**Цель разработки:** Дать пользователю возможность почувствовать себя в роли владельца питомца, роль которого будет выполнять известный персонаж из аниме по имени “ Kaneki Ken”.

**Назначение:** Данный игровой продукт имеет развлекательную цель для всех типов игроков. Игра понравится тем, кто не смотрел аниме, а в особенности ее оценят фанаты данного тайтла.

**Периодичность использования:** В любое свободное время.

**Источники и способы получения данных:** Play Марке и по специализированным ссылкам.

**Обзор аналогичных ПП:** На данный момент имеется большое количество игр данного жанра с большим количеством различных персонажей, но именно для фанатов аниме практически ничего нет. Рассмотрим игры подобного жанра:

My anime girl 2: Игра является довольно-таки слабым представителем данного жанра, но при этом у нее достаточно пользователей. В игре не хватает многих функций, с которыми игра могла бы стать более увлекательной. Также нет какого-либо определенного стиля, персонаж отрисован в аниме-рисовке, но остальные объекты игры полностью отличаются.

**1.2 Функциональные требования:**

**Игрок:**

**1.2.1 Регистрация и вход в аккаунт:**

После установки и открытия игры будет выскакивать окно, в котором пользователю будет предлагаться выбрать аккаунт, который привязан к GooglePlay.

* 1. **.2 Игровой процесс**

Игра начнется с того, что пользователю представят персонажа, которого он будет развивать по ходу игры. Появится объяснение каждого показателя его состояния:

**1)Сон**

Показатель сна будет зависеть от того, как долго или как мало персонаж отдыхал. При прохождении мини-игр и при употреблении пищи показатель сна будет падать, в таком случае персонажа надо будет уложить спать. Если показатель сна упадет до минимума – персонаж умрет.

**2)Еда**

Показатель еды будет зависеть от того, как много или как мало персонаж будет есть. По лору манги и аниме, персонаж может употреблять только 2 вещи: Человечина и кофе. Если показатель упадет до минимума – персонаж умрет.

**3)Гигиена**

Показатель гигиены будет падать после употребления пищи и сна. Чтобы повышать показатель надо будет мыть персонажа. Так же гигиена будет резко уменьшаться, если психическое состояние будет минимальным. Если показатель уйдет в минимум – сразу же упадет психическое состояние, персонаж умрет.

**4)Психическое состояние**

Будет зависеть от количества сыгранных мини-игр и от выше перечисленных показателей. Если состояние достигнет минимума – персонаж сойдет с ума и умрет.

**5)Статус “Кагуне”**

Показатель “Кагуне” является чисто косметическим фактором. Разные скины на этот показатель можно будет открыть в мини-играх.

В целом вышеперечисленные факторы и есть геймплей игры. Когда персонаж достигнет 993 уровня – игра будет являться пройденной. Уровень будет расти от пройденных мини-игр. В игре будет 3 мини-игры:

1. Игра по типу классической змейки

“Сколопендра”

-Суть игры будет очень похожа на игру “Змейка”. Игрок на карте должен будет собирать очки, играя за стилизированную змейку. При смерти будет выводится количество очков, заработанных до столкновения со стенкой.

1. Игра по типу игры Doodle Jump

“Сова”

-Суть игры будет похожа на игру Doodle Jump. Игрок должен будет прыгать по островам, зарабатывая очки, при паделнии игрока с острова игра будет окончена.

1. Игра по типу Ping Pong

“Кагуне”

-Классический пинг-понг, но со стилизацией под геймдизайн всей игры.

Уровень будет расти не по 1 единице, а по 7. Данное решение обусловлено лором манги.

**1.2.3 Настройки**

В игре будет лишь одна настройка – регулятор громкости игры.

**Периодичность использования:** Постоянное использование.

**Требования к применению:**

Продукт имеет материал исключительно для лиц не моложе 16 лет. Пользователю требуется войти или зарегистрироваться в Google аккаунт.

**Требования к производительности:**

Программа должна запускаться на слабо конфбюигурированных системах.

**Минимальные системные требования:**

- Оперативная память не менее 1Гб ОЗУ;

- Процессор не ниже модели: Snapdragon 425 MSM8917;

- Версия Android не ниже версии 6.0;

- 200 МБ свободного места на жестком диске;

Максимальное время на запуск программы на запуск программы при минимальных требованиях должно составлять не более 10 секунд.

Требования к реализации:

Для достижения максимальной производительности и оптимизации приложение должна быть разработана на движке Unity3d на языке C# в среде Visual Studio 2022.

**Требование к надежности:**

Приложение должно производить самовосстановление после сбоя работы.

**Требования к интерфейсу:**

Интерфейс должен быть с приятной цветовой гаммой и понятной для пользователя. Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователя было просто и понятно принимать информацию. Пользователь не должен испытывать какого-либо дискомфорта в плане восприятия информация, отображённой на экране. Объекты (рисунки и символы) не должны быть слишком мелкие.

**Геймдизайн:**

В игре будет несколько комнат, нахождение в которых будет зависеть от нажатой иконки показателя. Будет: Кухня, Спальня, Игровая, Ванная, Комната со стулом, кабинет Антейку.

Вот пример главной комнаты: **(Рисунок - 2)**



**Рисунок – 2**

**Требования к программной документации:**

Для удобства использования программного продукта пользователем,

необходимо добавить файл справки, в котором будет находиться вся необходимая информация о работе с программой.

**Технико-экономические показатели:**

Ориентировочная экономическая эффективность не рассчитываются.

Предполагаемое число использования программой в год – 365 сеансов на одном

рабочем месте.

**2 Техническое задание**

**Введение**

**Наименование программы:**

Наименование программного продукта –Тамагочи «Tokyo ghoul».

**Краткая характеристика области применения:**

Данный игровой продукт имеет развлекательную цель для всех типов игроков.

**Основания для разработки:**

Основанием для разработки является задание на практику на тему

«Разработка программного обеспечения, предоставляющего планирование задач пользователя». Тема согласована с преподавателем.

**Назначение разработки:**

Дать пользователю возможность почувствовать себя в роли владельца питомца, роль которого будет выполнять известный персонаж из аниме по имени “ Kaneki Ken”.

**Игровой процесс:**

Игра начнется с того, что пользователю представят персонажа, которого он будет развивать по ходу игры. Появится объяснение каждого показателя его состояния:

**1)Сон**

Показатель сна будет зависеть от того, как долго или как мало персонаж отдыхал. При прохождении мини-игр и при употреблении пищи показатель сна будет падать, в таком случае персонажа надо будет уложить спать. Если показатель сна упадет до минимума – персонаж умрет.

**2)Еда**

Показатель еды будет зависеть от того, как много или как мало персонаж будет есть. По лору манги и аниме, персонаж может употреблять только 2 вещи: Человечина и кофе. Если показатель упадет до минимума – персонаж умрет.

**3)Гигиена**

Показатель гигиены будет падать после употребления пищи и сна. Чтобы повышать показатель надо будет мыть персонажа. Так же гигиена будет резко уменьшаться, если психическое состояние будет минимальным. Если показатель уйдет в минимум – сразу же упадет психическое состояние, персонаж умрет.

**4)Психическое состояние**

Будет зависеть от количества сыгранных мини-игр и от выше перечисленных показателей. Если состояние достигнет минимума – персонаж сойдет с ума и умрет.

**5)Статус “Кагуне”**

Показатель “Кагуне” является чисто косметическим фактором. Разные скины на этот показатель можно будет открыть в мини-играх.

В целом вышеперечисленные факторы и есть геймплей игры. Когда персонаж достигнет 993 уровня – игра будет являться пройденной. Уровень будет расти от пройденных мини-игр. В игре будет 3 мини-игры:

1. Игра по типу классической змейки

“Сколопендра”

1. Игра по типу игры Doodle Jump

“Сова”

1. Игра по типу Ping Pong

“Кагуне”

Уровень будет расти не по 1 единице, а по 7. Данное решение обусловлено лором манги.

**Геймдизайн:**

В игре будет несколько комнат, нахождение в которых будет зависеть от нажатой иконки показателя. Будет: Кухня, Спальня, Игровая, Ванная, Комната со стулом, кабинет Антейку.

Вот пример главной комнаты: **(Рисунок – 3)**



**Рисунок - 3**

**Требования к программе или программному изделию:**

**Требования к применению:**

Продукт имеет материал исключительно для лиц не моложе 16 лет. Пользователю требуется войти или зарегистрироваться в Google аккаунт.

**Требования к производительности:**

Программа должна запускаться на слабо конфигурированных системах.

Минимальные системные требования:

- Оперативная память не менее 1Гб ОЗУ;

- Процессор не ниже модели: Snapdragon 425 MSM8917;

- Версия Android не ниже версии 6.0;

- 200 МБ свободного места на жестком диске;

Максимальное время на запуск программы на запуск программы при

минимальных требованиях должно составлять не более 10 секунд.

**Требования к реализации:**

Для достижения максимальной производительности и оптимизации

приложение должна быть разработана на движке Unity3d на языке C# в

среде Visual Studio 2022.

**Требование к надежности:**

Приложение должно производить самовосстановление после сбоя работы.

**Требования к интерфейсу:**

Интерфейс должен быть с приятной цветовой гаммой и понятной для

пользователя.

Следовательно, каждое окно должно иметь ясную визуальную иерархию

своих элементов. Фрагменты текста должны располагаться на экране так,

чтобы пользователя было просто и понятно принимать информацию.

Пользователь не должен испытывать какого-либо дискомфорта в плане

восприятия информация, отображённой на экране. Объекты (рисунки и

символы) не должны быть слишком мелкие.

**Требования к программной документации:**

Для удобства использования программного продукта пользователем,

необходимо добавить файл справки, в котором будет находиться вся не

обходимая информация о работе с программой.

**Технико-экономические показатели:**

Ориентировочная экономическая эффективность не рассчитываются.

Предполагаемое число использования программой в год – 365 сеансов на

одном рабочем месте.

**Стадии и этапы разработки**

**Стадии разработки:**

Разработка должна быть проведена в три стадии: разработка технического

задания, рабочее проектирование, внедрение.

**Этапы разработки:**

На стадии разработки технического задания должен быть выполнен этап

разработки, согласования и утверждения настоящего технического зада

ния.

На стадии рабочего проектирования должны быть выполнены

перечисленные ниже этапы работ:

разработка программы;

разработка программной документации;

испытания программы.

На стадии внедрения должен быть выполнен этап разработки – подготов

ка и передача программы.

**Содержание работ по этапам:**

На этапе разработки технического задания должны быть выполнены

перечисленные ниже работы:

1. Постановка задачи;

2. Определение и уточнение требований к техническим средствам;

3. Определение требований к программе;

4. Определение стадий, этапов и сроков разработки программы и

документации на неё;

5. Выбор языков программирования;

6. Согласование и утверждение технического задания.

На этапе разработки программной документации должна быть выполнена

работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена

разработка программных документов в соответствии с требованиями

ГОСТ 19.101-77 с требованием п. Предварительный состав программной

документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные

ниже виды работ:

Разработка, согласование и утверждение программы и методики испыта

ний;

Проведение приемо-сдаточных испытаний;

Корректировка программы и программной документации по результатам

испытаний.

На этапе подготовки и передачи программы должна быть выполнена ра

бота

по подготовке и передаче программы и программной документации в

эксплуатацию на объектах Заказчика.

**Порядок контроля и приемки:**

**Виды испытаний:**

Приемо-сдаточные испытания программы должны проводиться согласно

разработанной Исполнителем и согласованной Заказчиком программы и

методик

испытаний.

Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель

документируют в Протоколе проведения испытаний.

**Общие требования к приемке работы:**

На основе Протокола проведения испытаний Исполнитель совместно с

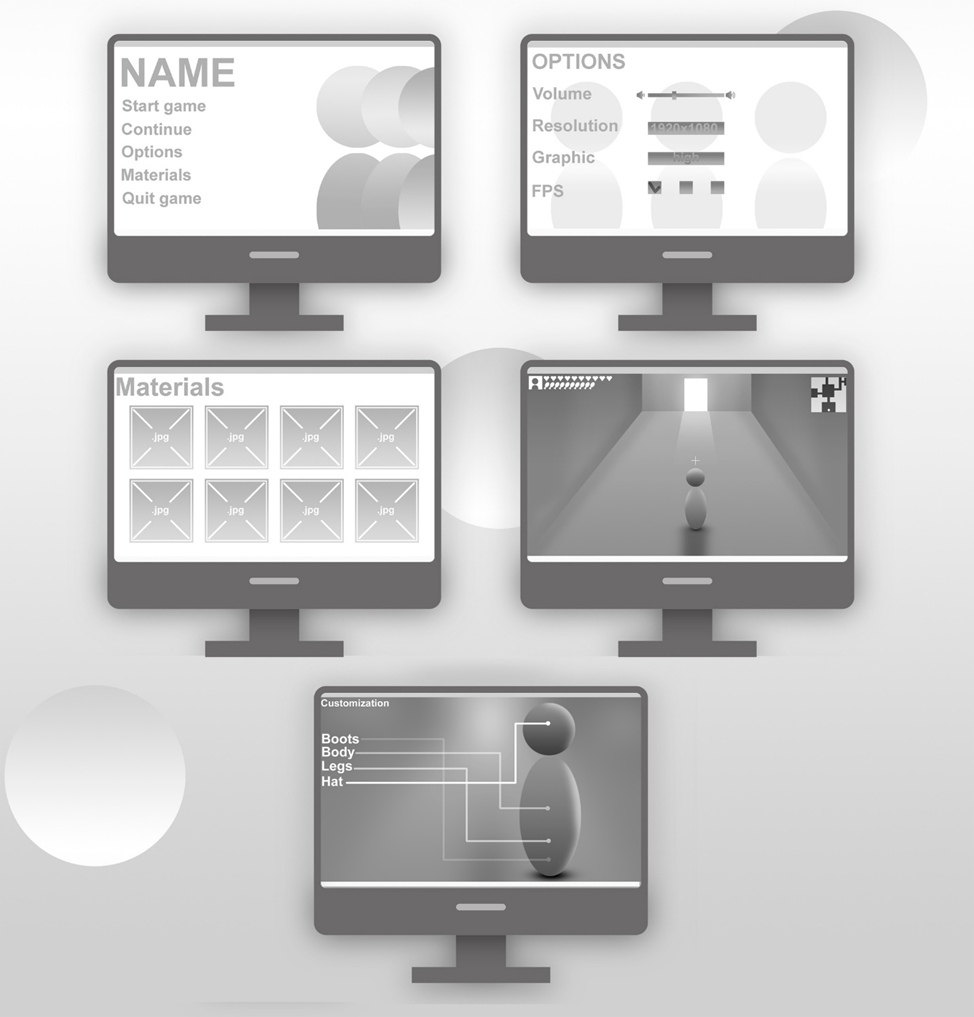
Заказчиком подписывают Акт приемки-сдачи программы в эксплуатацию.

**3 Реализация**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название парадигмы | Сущность  парадигмы,  основные  идеи,  принципы,  объекты. | Языки  поддеживающие данную  парадигму.  (2-3 языка) | Достоинства | Недостатки | **Примеры**  **программ** |
| 1 | Императивное  программирование | Это парадигма, основанная на составлении алгоритма действий (инструкций/команд), которые изменяют состояние (информацию/данные/память) программы. | C, C++, java | Выполняется компьютером сразу, без предварительной компиляции | При императивном подходе к написанию кода (в отличие от [функционального подхода](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), относящегося к [декларативной парадигме](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) широко используется [присваивание](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Наличие операторов присваивания увеличивает сложность [модели вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9) и делает императивные программы подверженными специфическим ошибкам, не встречающимся при функциональном подходе | Fortran, Ada |
| 2 | Декларативное  программирование | Это парадигма, при которой описывается желаемый результат, без составления детального алгоритма его получения. | HTML, SQL | При создании HTML мы с помощью тегов описываем, какую хотим получить страничку в браузере, а не то, как нарисовать на экране заголовок статьи, оглавление и текст. | Декларативные компьютерные языки часто не полны по Тьюрингу, так как теоретически не всегда возможно порождение исполняемого кода по декларативному описанию. | DOM, React |
| 3 | Структурное  программирование | [Парадигма программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в основе которой лежит представление программы в виде иерархической структуры [блоков](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). | Basic, Cg, Pascal | Серьёзно облегчилось понимание программ, появилась возможность разработки программ в нормальном промышленном режиме, когда программу может без особых затруднений понять не только её автор, но и другие программисты. | Главный недостаток структурного подхода заключается в следующем: процессы и данные существуют отдельно друг от друга (как в модели деятельности организации, так и в модели программной системы), причем проектирование ведется от процессов к данным. Таким образом, помимо функциональной декомпозиции, существует также структура данных, находящаяся на втором плане. | [goto](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bgoto%5D), Cgi |
| 4 | Процедурное  программирование | Программирование на [императивном языке](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), при котором последовательно выполняемые [операторы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) можно собрать в [подпрограммы](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), то есть более крупные целостные единицы [кода](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), с помощью механизмов самого языка. | [Tcl](https://ru.wikipedia.org/wiki/Tcl), [Perl](https://ru.wikipedia.org/wiki/Perl" \o "Perl), [Lua](https://ru.wikipedia.org/wiki/Lua" \o "Lua) | Процедурное программирование отлично подходит для программирования общего назначения.Закодированная простота наряду с простотой реализации компиляторов и интерпретаторов.Большое разнообразие книг и онлайн-материалов курсов по проверенным алгоритмам, облегчающих обучение в процессе.Исходный код переносим, ​​поэтому его можно использовать и для другого процессора.Код может быть повторно использован в разных частях программы, без необходимости копировать его. Благодаря методике процедурного программирования требования к памяти также сокращаются .Ход программы можно легко отследить | Код программы труднее писать, когда используется процедурное программирование. Процедурный код часто не может быть использован повторно, что может привести к необходимости воссоздания кода, если это необходимо для использования в другом приложении.Сложно общаться с реальными объектами. Важное значение придается операции, а не данным, что может создавать проблемы в некоторых случаях, связанных с данными. Данные открыты для всей программы, что делает их не очень безопасными | Topogun, Coat |
| 5 | Модульное  программирование | Это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. | Оберон, Компонентный паскаль, Zonnon | Модульные программы легко составлять и отлаживать. Функциональные компоненты такой программы могут быть написаны и отлажены порознь. | Модульность требует большей дополнительной работы. Чтобы писать модульные программы, программист должен быть значительно более аккуратным на этапе проектирования программной разработки. Он должен проектировать свои программы по нисходящей схеме, начиная с верхних уровней всей программы и затем продвигаясь вниз к более детальному проектированию отдельных подпрограмм. | Xerox Parc |
| 6 | Объектно-  ориентированное  программирование | [методология программирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на представлении программы в виде совокупности взаимодействующих [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), каждый из которых является экземпляром определённого [класса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а классы образуют иерархию наследования | Java, C++, Python, C# | Первые программы на языках программирования высокого уровня, по сути, не были структурированы, и это не вызывало проблем, потому что обьёмы кода были, по современным меркам, ничтожны. | В процессе трансляции объектно-ориентированных программ в исполняемый код центрального процесса возникает ряд неоптимальностей по использованию памяти и вычислительного времени процессорных ядер. | Tproger, S.O.L.I.D. |
| 7 | Функциональное  программирование | [парадигма программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в которой процесс [вычисления](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) трактуется как вычисление значений [функций](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) в математическом понимании последних (в отличие от [функций](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) как подпрограмм в [процедурном программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). | Липс, Erlang | Во-первых, для функциональных языков нет эффективного неупорядоченного словаря и множества. Чисто функциональные словари работают медленнее хэш-таблицы, и для некоторых приложений это может быть критично. Во-вторых, не существует чисто функциональных слабых хэш-таблиц, хотя для большинства разработчиков этот недостаток может остаться незамеченным. | высокоуровневые абстракции, которые скрывают большое количество подробностей таких рутинных операций, как, например, итерирование. За счет этого код получается короче, и, как следствие, гарантирует меньшее количество ошибок, которые могут быть допущены. | ST, Goto |
| 8 | Логическое  программирование | [парадигма программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), основанная на [математической логике](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) — программы в ней задаются в форме логических утверждений и правил вывода. Наиболее известный язык логического программирования — [Пролог](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BB%D0%BE%D0%B3_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). | Datalog, Mercury, Oz | Операции, совершаемые в логическом программировании всегда понятны; Результат практически всегда не зависит от выбранного пути реализации;  Может быть использован в качестве невычислительного языка используя только выражения и факты. | 1. Если брать за пример логического языка программирования Prolog, то на лицо невозможность создания комплексных задач. То есть в реальности логический язык может идти дополнением к процедурному, но самостоятельно используется крайне редко; | Prolog, GGB,NVD |

**4 Проектирование ПИ**

**4.0.1 UX – Проектирование (Рисунок - 4)**



**Рисунок - 4**

**4.0.2 UI – Проектирование (Рисунок – 5)**

****

**Рисунок – 5**

* 1. **Тестирование и отладка ПО**

Таблица – 1 Тестирование и отладка программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
|  | Описание | Ожидаемый результат | Фактический результат |
| 1. | Поддаётся ли модель деформированию | Модель поддаётся деформированию | Модель поддаётся деформированию |
| 2. | Поддаётся ли модель ретопологии под анимацию | Топология модели сделана под анимацию | Топология модели сделана под анимацию |
| 3. | Поддается ли топология модели дальнейшему текстурированию | На модель хорошо накладываются текстуры без деформаций | На модель хорошо накладываются текстуры без деформаций |
| 4. | Поддается ли модель качественной анимации | Модель анимируется без дефектов и артефактов | Модель анимируется без дефектов и артефактов |
| 5. | Находится ли модель в нужных форматах для экспорта в игровой движок | Модель находится в формате стл и обж | Модель находится в формате стл и обж |
| 6. | Находятся ли текстуры для модели в режиме ПБР для правильного отражения глобального освещения | Текстуры находятся в режиме пбр | Текстуры находятся в режиме пбр |
| 7. | Имеются ли дефекты на карте нормалей | На карте нормалей дефектов нет | На карте нормалей дефектов нет |
| 8. | Есть ли возможность наложения текстур на модель в игровом движке | Текстуры качественно накладываются на модель находясь в игровом движке | Текстуры качественно накладываются на модель находясь в игровом движке |
| 9. | Работает ли анимация в игровом движке | Анимация модели отлично функционирует внутри игрового движка | Анимация модели отлично функционирует внутри игрового движка |
| 10. | Имеются ли искажения на топологии модели при экспорте в другие программы | На модели не имеется деформаций топологии при экспорте | На модели не имеется деформаций топологии при экспорте |

Таблица 2 – Руководство пользователя

|  |  |
| --- | --- |
| Движение вперед | Нажатие кнопки w |
| Движение назад | Нажатие кнопки s |
| Движение влево | Нажатие кнопки a |
| Движение вправо | Нажатие кнопки d |
| Ускоренный бег | Нажатие кнопки shift |
| Присесть | Нажатие кнопки ctrl |
| Танец | Нажатие кнопки g |
| Прыжок | Нажатие кнопки space |

Таблица 2 – Руководство программиста

|  |  |
| --- | --- |
| ZBrush | В этой программе было реализовано: скульптинг и моделирование |
| RizonUV | В этой программе было выполнено развертка модели |
| Topogun | В этой программе было выполнено ретопология |
| Sabstunce Painter | В этой программе были сделаны текстуры и запекание |
| Cinema 4D | В этой программе была сделана анимация |
| Unity3D | В этой программе была реализован функционал модели (Анимация) |

***Приложение А***

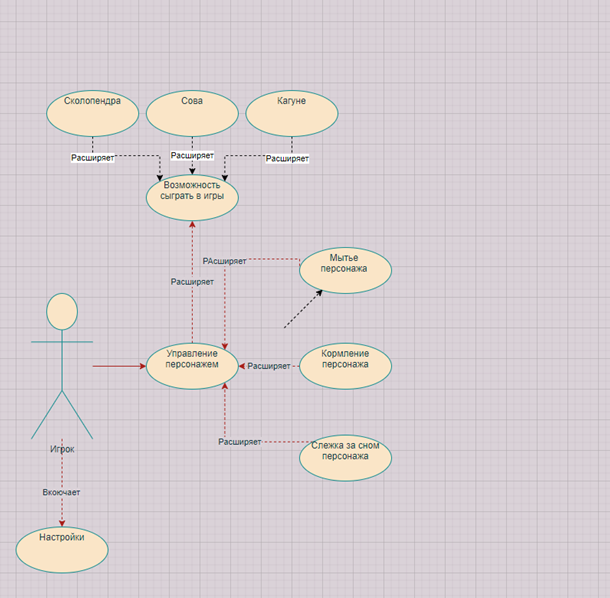


Рисунок 6 — Диаграмма вариантов использования

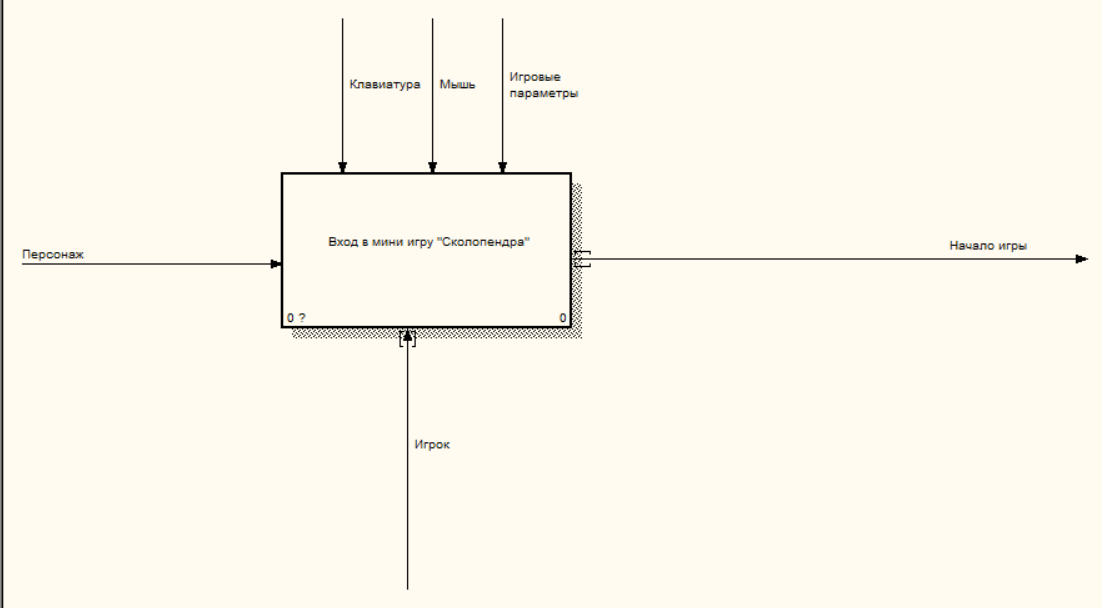
****

Рисунок 7 -Функциональная модель (1)

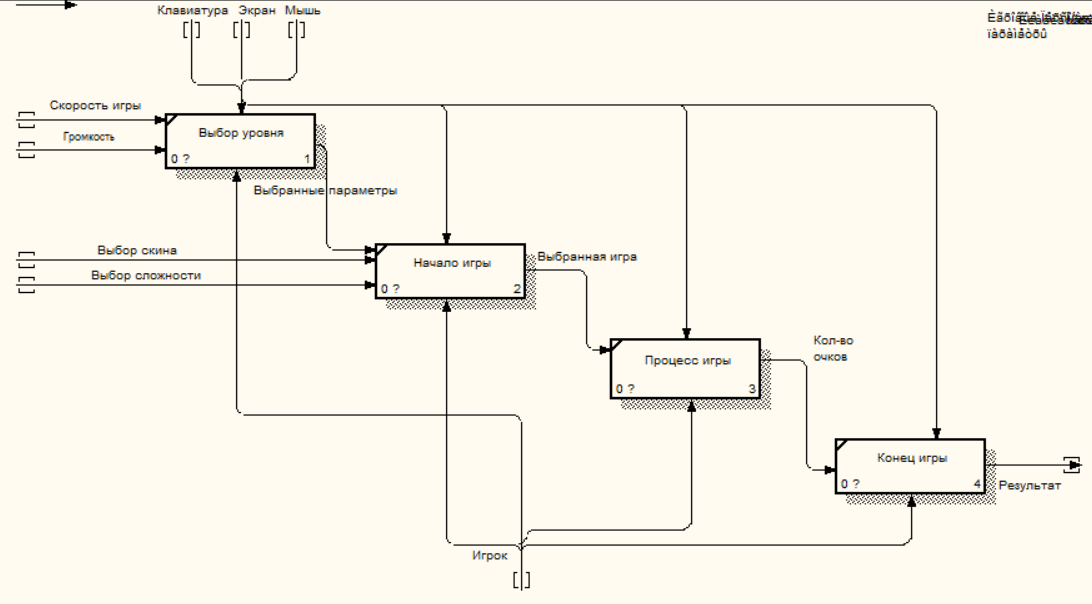
****

Рисунок 8 -Функциональная модель (2)

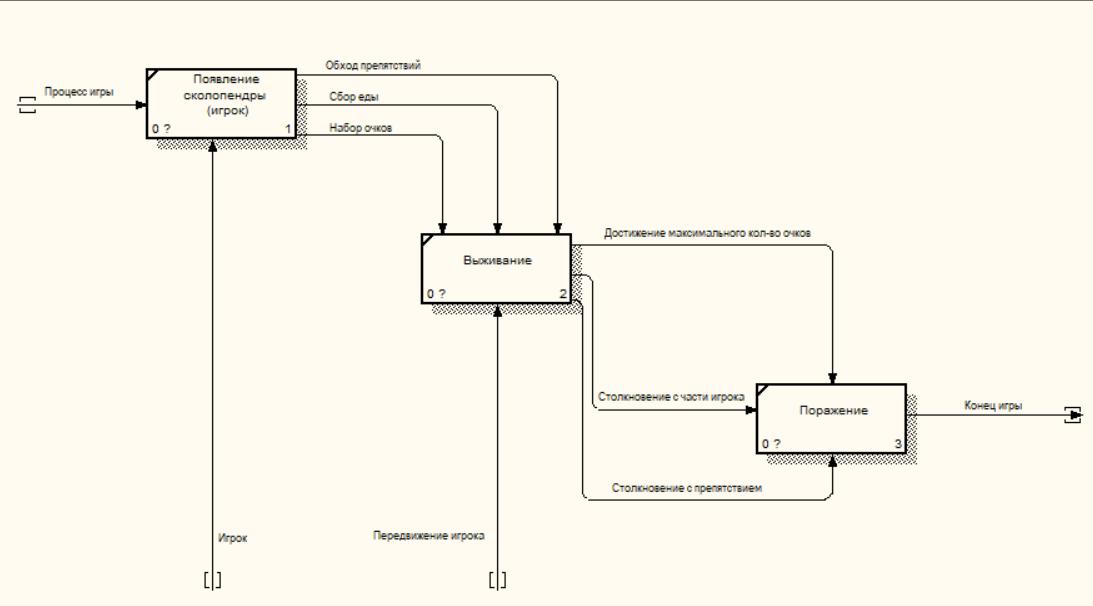
****

Рисунок 9 -Функциональная модель (3)

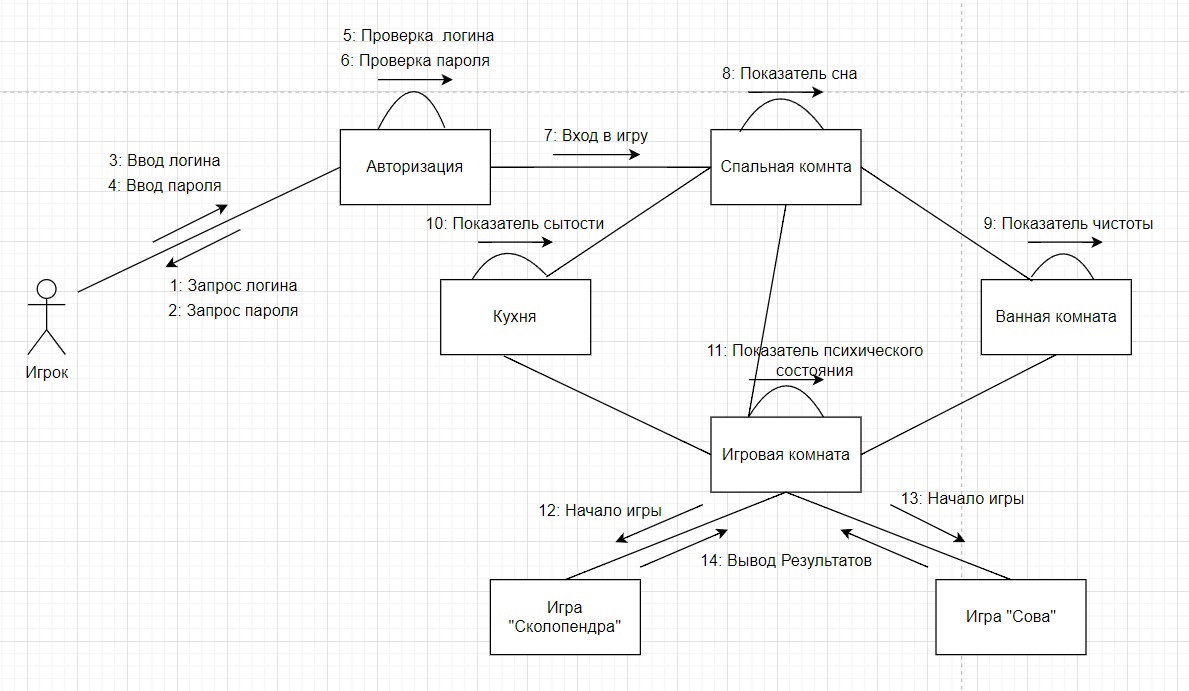


Рисунок 11 — Модель данных

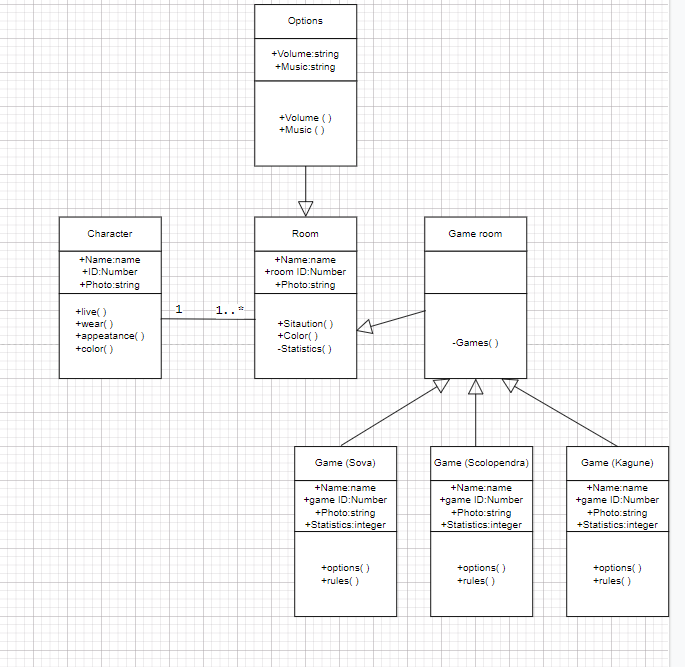


Рисунок 12 — Диаграмма объектов

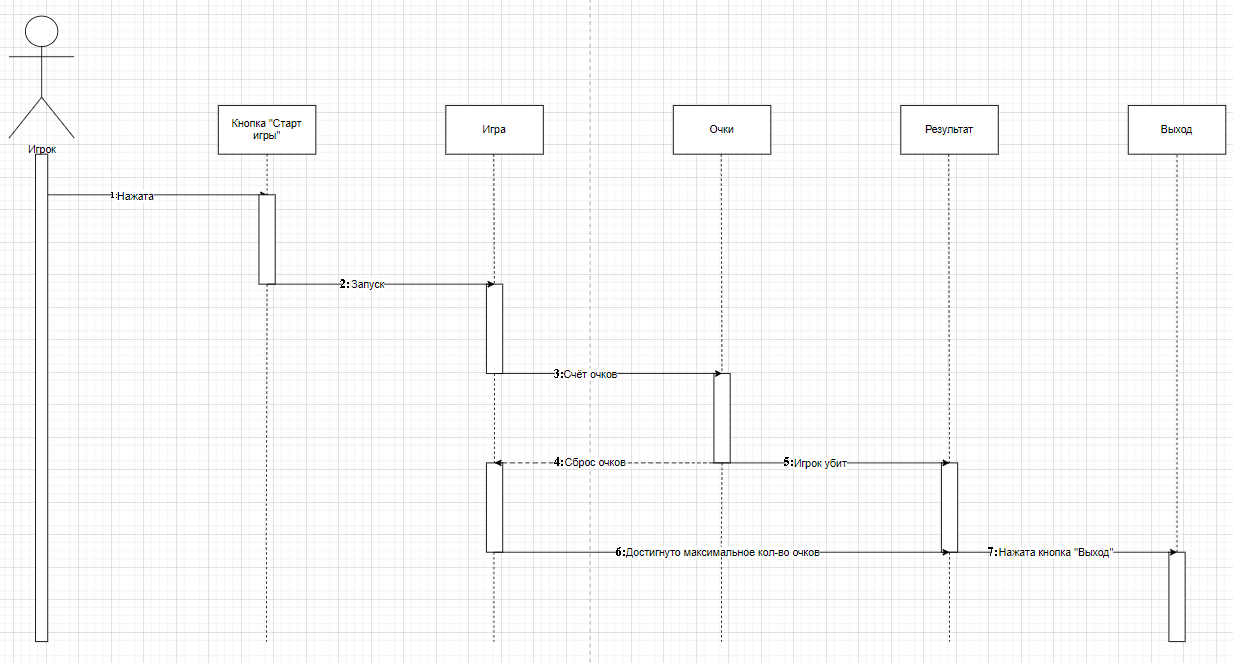


Рисунок 13 — Диаграмма последовательности

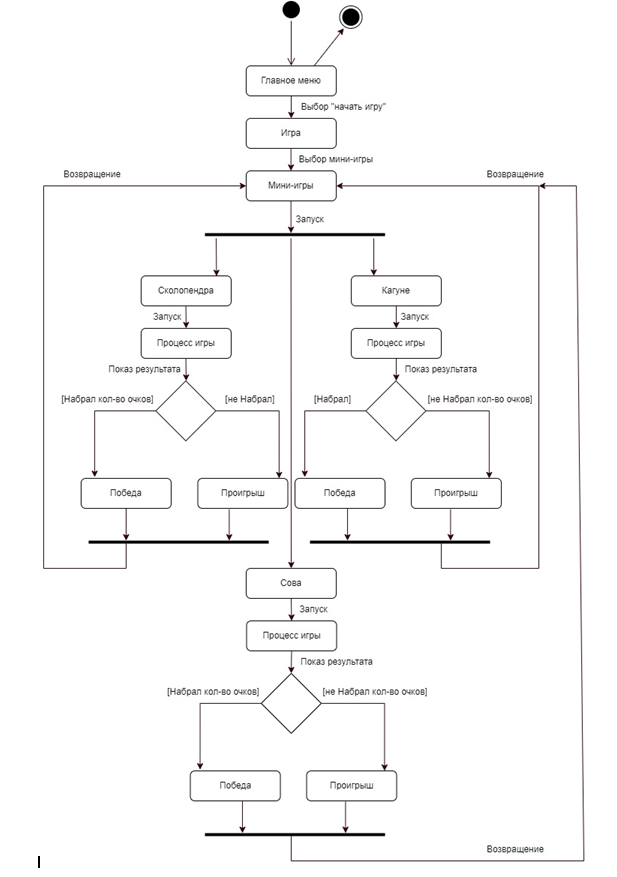


Рисунок 14 — Диаграмма деятельности

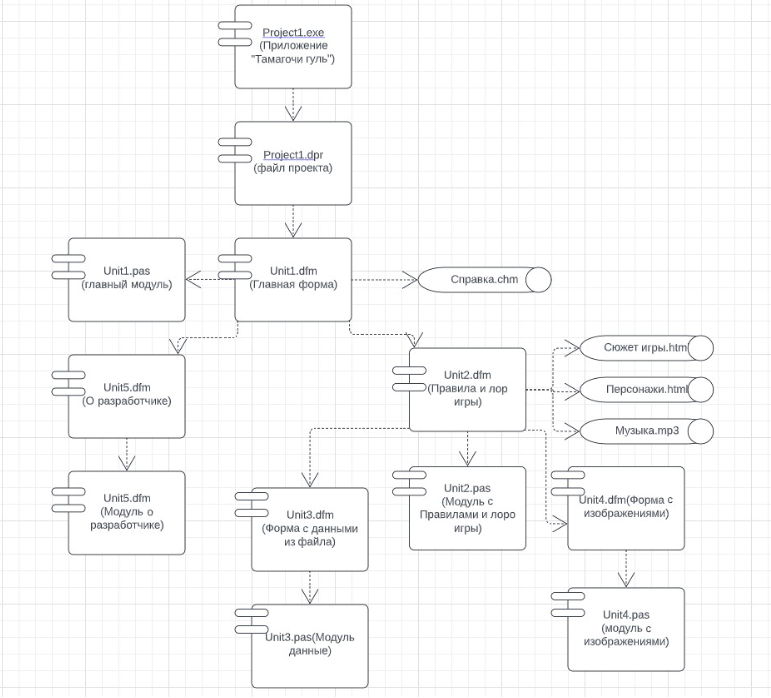


Рисунок 15 – Диаграмма компонентов

1. **Прототип ПП**

Функции модели:

1. Может отражать различные шейдеры



Рисунок – 16

2.Ретопология построена под текстурирование и анимацию

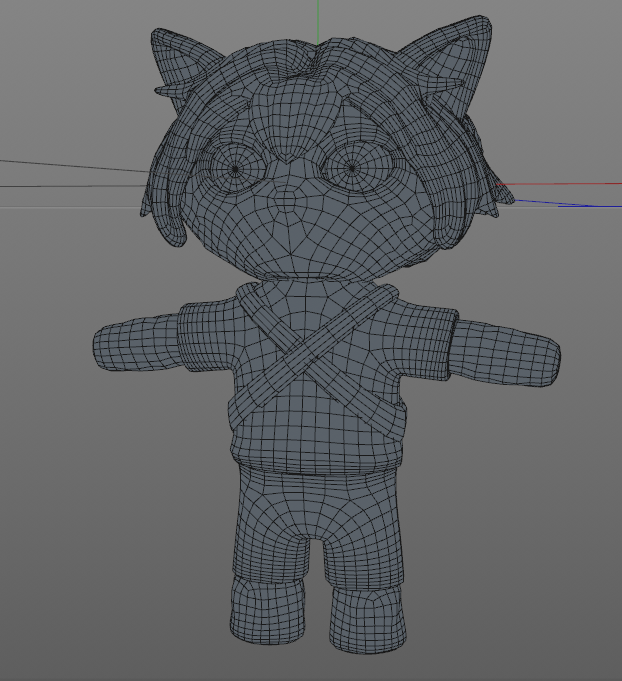


Рисунок - 17

3.Анимация бега

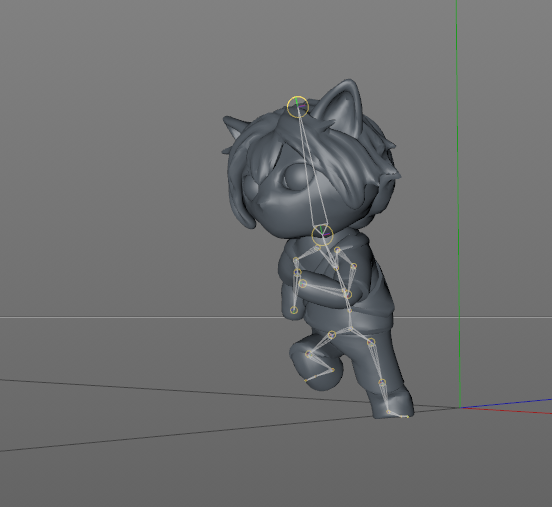


Рисунок - 18

4.Анимация танца

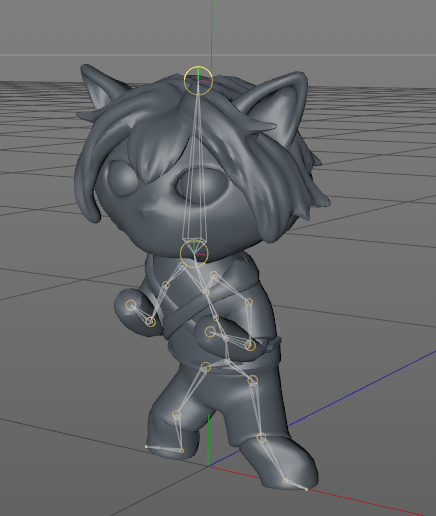


Рисунок - 19

5. Анимация удара

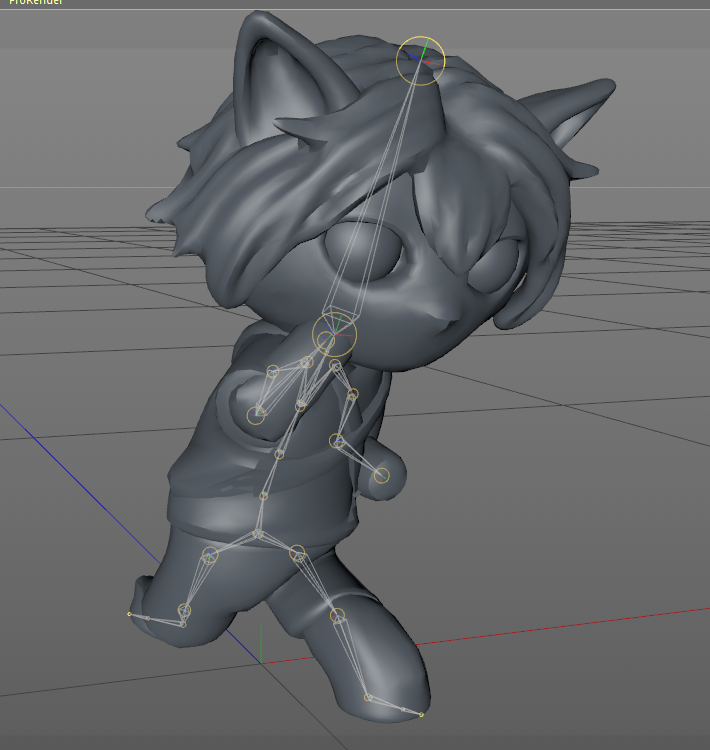


Рисунок - 20

6.Анимация ходьбы



Рисунок – 21

7.Анимация стойки

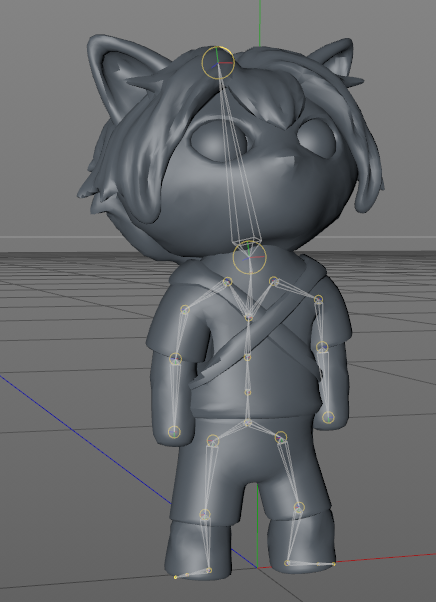


Рисунок - 22

8.Риггинг сделан под анимацию

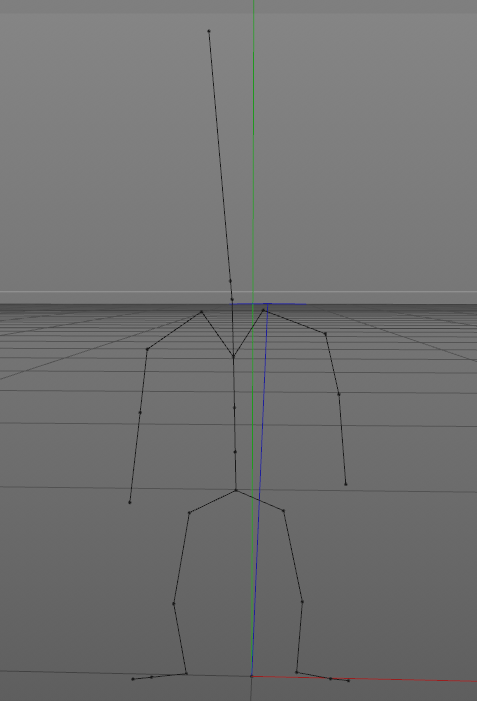


Рисунок - 23

9.Текстуры и шейдинг модели нормально отражает глобальное освещение

Рисунок – 23

10.Модель имеет лоуполи и хайполи полигонаж

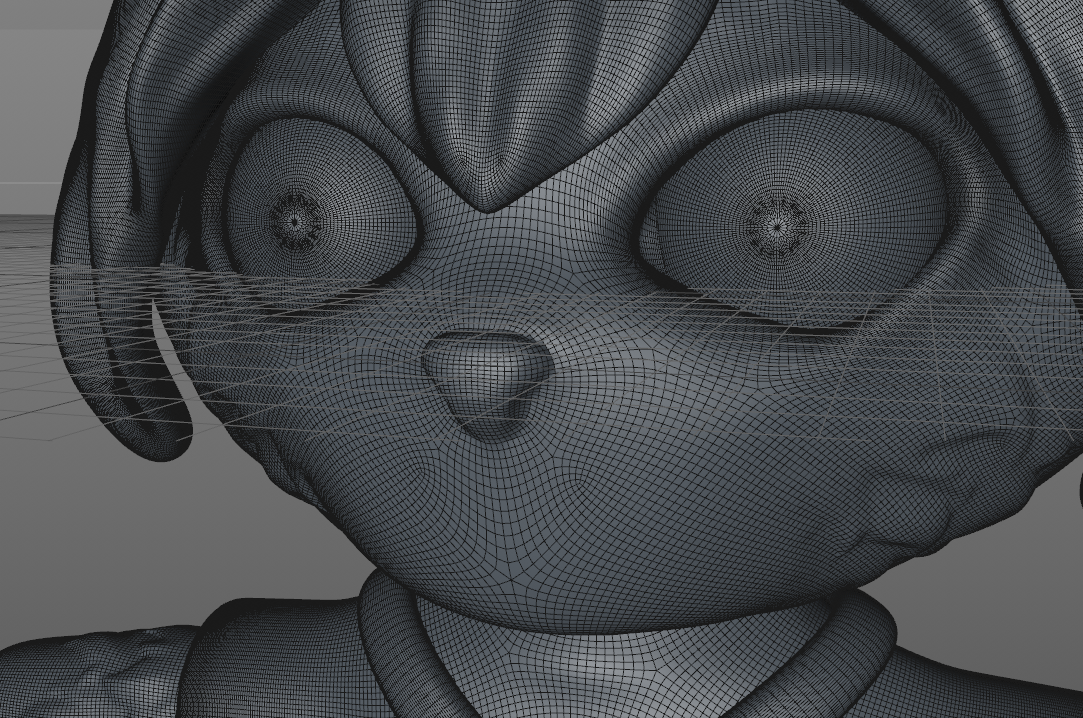


Рисунок - 25