|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент (ФИО) | Группа | Подпись, дата | Оценка |
| Вячеслав Романов | ИУ7-21Б |  |  |
| Зейнал Зейналов | ИУ7-21Б |  |  |
| Анастасия Сушина | ИУ7-21Б |  |  |
| Григорий Будкин | ИУ7-21Б |  |  |
| Азамат Сангинов | ИУ7-21Б |  |  |

Ментор команды **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  *Филатов Сергей*

*подпись, дата фамилия, и.о.*

Ментор команды **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  *Крестов Сергей*

*подпись, дата фамилия, и.о.*

Ментор команды **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  *Кизилов Дмитрий*

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** *Ломовской Игорь*

*подпись, дата фамилия, и.о.*

*2018 г.*

**Оглавление**

1. [**Введение**. 3](#_Toc516036960)
2. [**Аналитическая часть** 4](#_Toc516036961)

[**Обзор существующих решений** 4](#_Toc516036962)

[**Анализ их достоинств и недостатков** 4](#_Toc516036963)

[**Обоснование необходимости нового решения** 4](#_Toc516036964)

[**Постановка задачи** 5](#_Toc516036965)

1. [**Конструкторская часть** 5](#_Toc516036966)

[**Декомпозиция задачи** 5](#_Toc516036967)

[**Разработка структуры создаваемого программного продукта** 8](#_Toc516036968)

[9](#_Toc516036969)

[**Описание назначения, требований к выделенным компонентам и их интерфейса.** 10](#_Toc516036970)

[**Разработка алгоритмов и структур данных для выделенных компонент** 11](#_Toc516036971)

[**Структуры данных** 13](#_Toc516036972)

[**Проектирование пользовательского интерфейса** 14](#_Toc516036973)

[**Описание способов тестирования как выделенных компонент, так и программного продукта в целом** 15](#_Toc516036974)

[**Подготовка тестовых данных** 15](#_Toc516036975)

1. [**Технологическая часть** 17](#_Toc516036976)

[**Выбор и обоснование технических средств** 17](#_Toc516036977)

[**Описание пользовательского интерфейса** 17](#_Toc516036978)

[**Выбор и обоснование модели разработки** 21](#_Toc516036979)

[**Реализация программного продукта** 21](#_Toc516036980)

[**Реализация тестирования** 22](#_Toc516036981)

# **Введение**.

**Цель и задачи практики-2018**

Целью практики является получение навыков командной работы и закрепление знаний и умений, полученных в курсах “Основы программной инженерии” и “Программирование”.

Студенты должны:

* получить практические навыки и умения, связанные с одной из тематик практики;
* научиться работать в команде;
* научиться представлять проделанную работу.

# **Аналитическая часть**

## **Обзор существующих решений**

Перед тем, как создавать новую игру был изучен рынок подобных игр. Более детально были рассмотрены те игры, которые пользуются максимальным спросом и похожи по сюжету и идее исполнения, с тем, что предполагалось реализовать в новом программном обеспечении.

Всего было рассмотрено около 15 игр, в данном же обзоре рассмотрены наиболее популярные:

* **Наездник дракона** - игра представляет собой лабиринт, в котором мы помогаем отважному наезднику на драконах пробираться сквозь подземный лабиринт, в который он залетел. На экране можно видеть двери, которые ведут в другие залы. Чтобы у персонажа оставались силы, ему необходимо собирать свиные ножки.
* **Monster maze** - игра представляет собой также лабиринт с видом сверху, персонаж ходит по лабиринту и должен убегать от монстром-ботов, цель персонажа: дойти до красного прямоугольника, не попав к монстрам.
* **Красная шапочка** - Красная шапочка собирает пироги и должна убежать от волка в домик, собрав все пироги на поле.
* **Случайный лабиринт** - игрок управляет шаром, с помощью которого надо собирать вазы, после чего необходимо найти выход.

## **Анализ их достоинств и недостатков**

Таблица 1. Сравнение с другими играми

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии/название игры | Золотой сундук | Наездник дракона | Monster maze | Красная шапочка | Случайный лабиринт |
| Собирательство предметов | да | да | нет | да | да |
| Обучающие задания | да | нет | нет | нет | нет |
| Подвижность карты | да | да | да | нет | нет |
| ОС | Windows  MacOS  Linux | Windows  MacOS | Windows  MacOS | Windows  MacOS | iOS |
| Цена | бесплатно | бесплатно | бесплатно | бесплатно | условно-бесплатно |

## **Обоснование необходимости нового решения**

Во всех играх отсутствовали обучающие задания, что обуславливает разработку нового продукта, лишенного всех вышеизложенных недостатков.

Кроме этого, планируется оснастить игру следующим дополнительным функционалом:

* Автосохранение прогресса;
* Выбор уровня сложности заданий;
* Различные типы блоков, по которым перемещается персонаж.

## **Постановка задачи**

Цель данной работы - создать игру на платформе python3+cocos2d, в которой персонаж перемещается по лабиринту (вид сверху) и выполняет обучающие задания.

# **Конструкторская часть**

## **Декомпозиция задачи**

Итоговой задачей стала реализация всех действий в игровом менеджере.

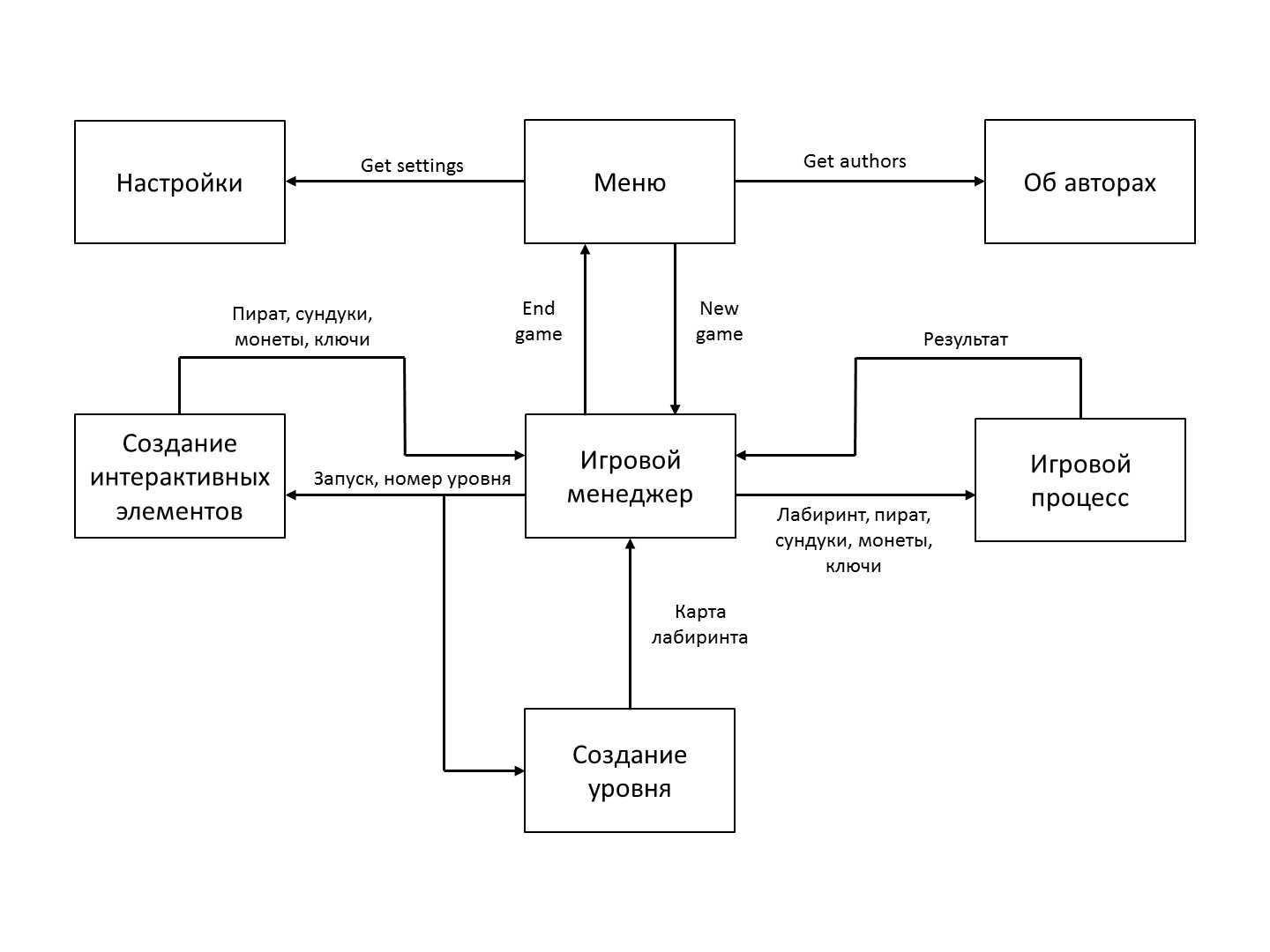


Рисунок 1. Разбиение на модули

В ходе декомпозиции были выделены следующие подзадачи: реализация игрового меню, реализация меню игровых настроек, создание окна с создателями игры, создание карт уровней, создание интерактивных элементов, а также реализация самого игрового процесса. (рисунки 2 и 3).

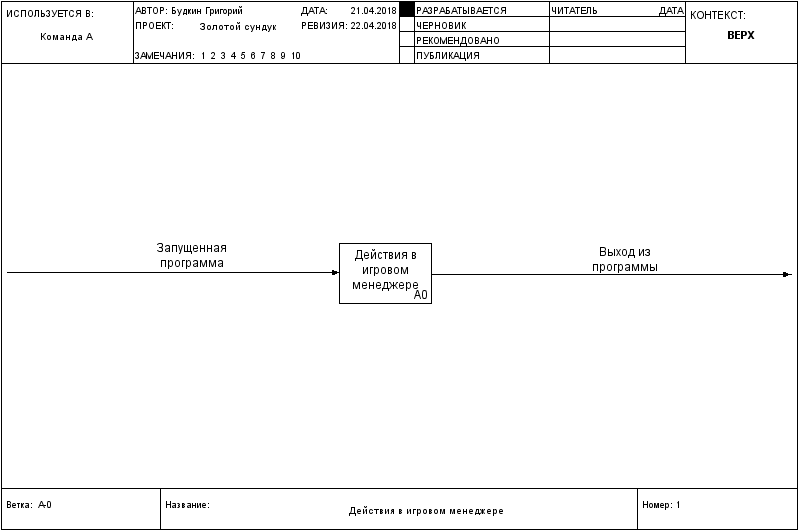


Рисунок 2. Основная декомпозиция работы программы

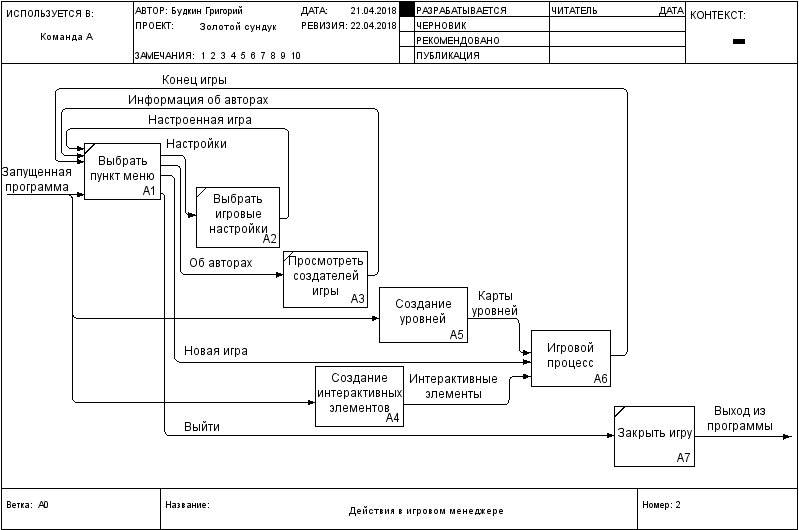
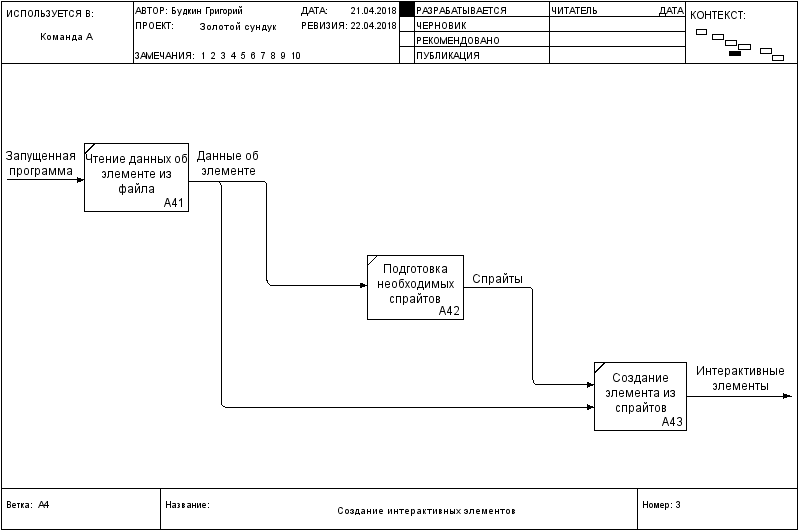


Рисунок 3. Декомпозиция работы в игровом менеджере

Интерактивные элементы и карты уровней для игры создаются путём чтения соответствующих файлов, хранящих данные о спрайтах и их положении, и последующей их постройке. (рисунки 4 и 5).



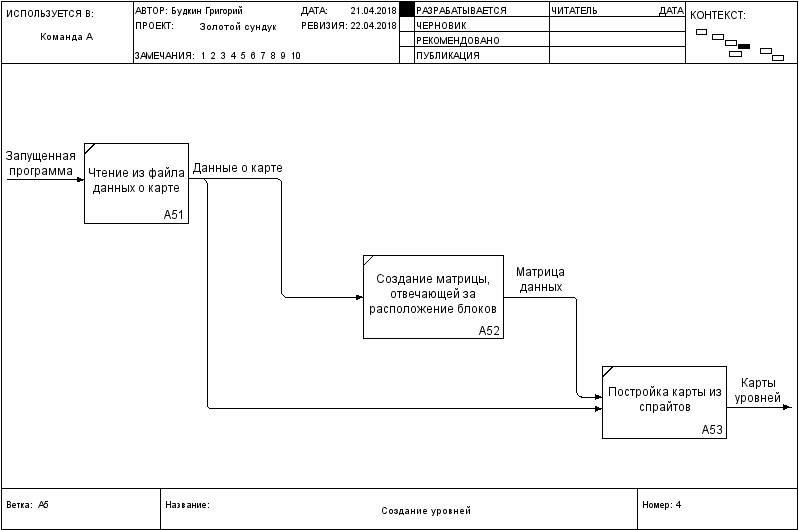
Рисунок 4. Декомпозиция создания интерактивных элементов

Рисунок 5. Декомпозиция создания уровней

Игровой процесс делится на загрузку интерфейсной части: карты уровня, персонажа и необходимых интерактивных элементов, - и действий самого пользователя, в зависимости от которых происходит смещение доступной для пользователя области карты, собирание игровых предметов, а также выдача заданий, которые пользователю предстоит решить для прохождения уровня. (рисунок 6).

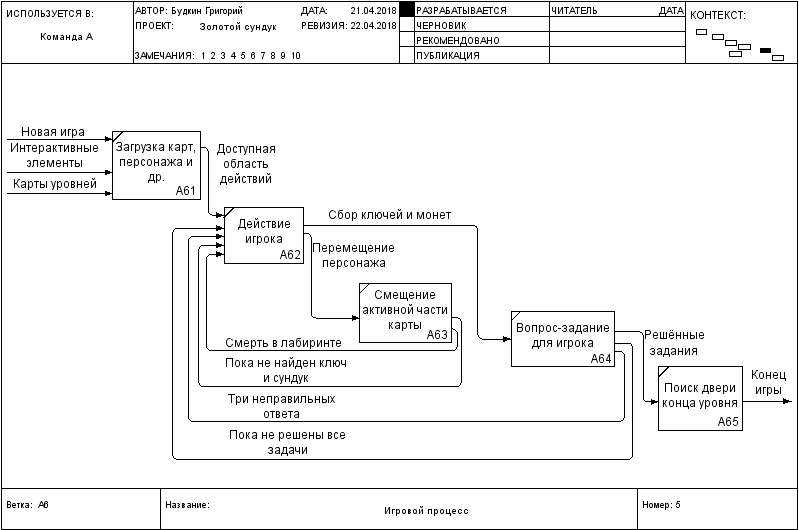


Рисунок 6. Декомпозиция игрового процесса

## **Разработка структуры создаваемого программного продукта**

Инициализация программы начинается с работы с данными, после загрузки информации о состоянии прогресса игрока, запускается уровень, и пользователь получает управление над персонажем, который ограничен правилами игрового мира и его логикой. (рисунок 7).

## arch_212

Рисунок 7. Архитектура проекта

Пунктирными линиями показаны потоки данных.

*Сплошными линиями показаны потоки управления*

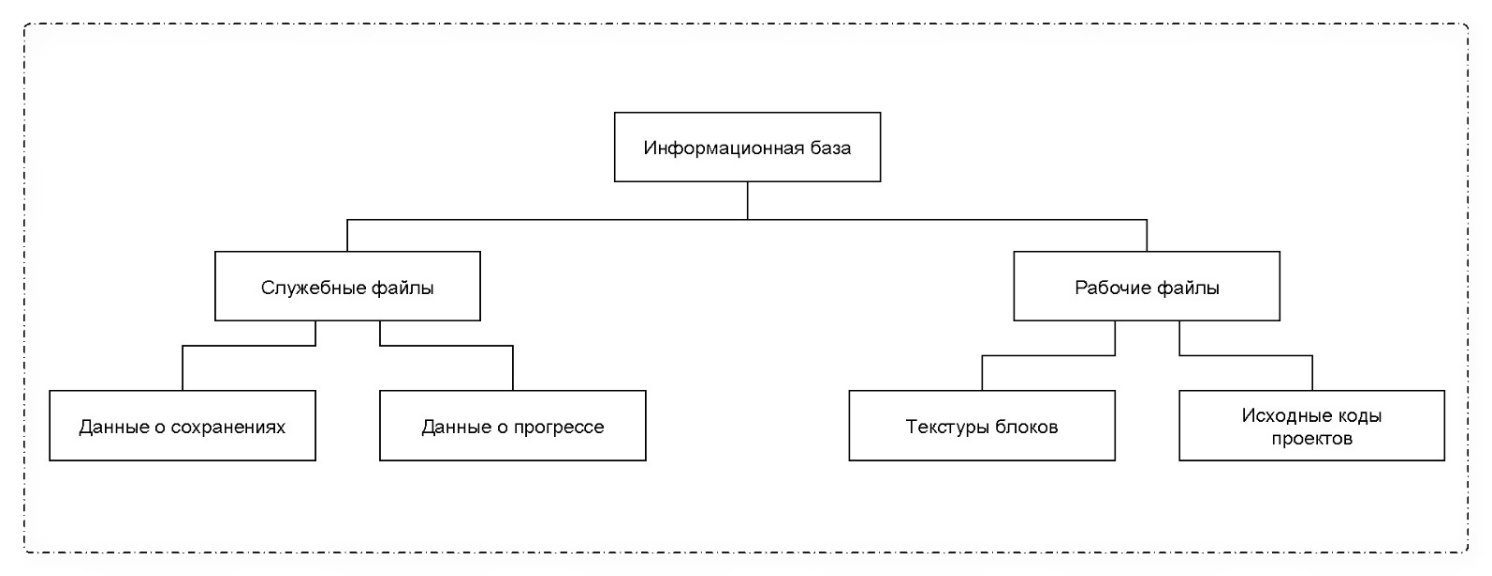


Рисунок 8. Схема информационной базы

Финансовые средства у команды из пяти человек на реализацию задач отсутствуют, поэтому единственными средствами при создании программного продукта являются 5 человек.

Основные задачи при создании программного продукта:

* Придумать основную цель, которую должен выполнять продукт.
* Декомпозировать задачу на подзадачи.
* Каждому разработчику реализовать подзадачу, за которую он несет ответственность.
* Тестировщикам необходимо проверить правильность и корректность результатов работы разработчиков.
* Каждому члену команды написать отчет о проделанной работе.

## **Описание назначения, требований к выделенным компонентам и их интерфейса.**

Игра носит образовательный характер, помогая развивать логику решением разных по уровню сложности задач. В ней есть три уровня сложности для людей разных групп и возрастов. Это обуславливает высокий интерес к игре и позволяет охватить большую аудитории.

Задания, предоставляемые пользователю, создавались таким образом, чтобы они имели однозначное решение и при самом высоком уровне сложности можно было бы дойти до победного конца игры.

Командой было решено, что на карте будет располагаться по три сундука, которые необходимо найти на карте, каждый из которых имеет соответствующий на карте ключ для того, чтобы открыть этот сундук. В каждом сундуке находится по одному заданию, на которое необходимо найти решение. Каждая карта представляет собой запутанный лабиринт.

Требования к компонентам геймплея:

* На каждой карте должны быть по три сундук и по три ключа;
* Каждая карта дает возможность свободно перемещаться на ней;
* Задания должны соответствовать уровню сложности;
* Лабиринт карты должен иметь выход из любой ситуации;
* Прорисовка элементов игры выстроена таким образом, чтобы она имела интуитивно понятный образ.

**Разработка алгоритмов и структур данных для выделенных компонент**

Выстраивание карты происходит после выбора уровня. Сначала считываются данные о карте, затем по считываются элементы по одному. О каждом элементе считывается необходимая информация, далее элемент располагается на карте. (рисунок 9).

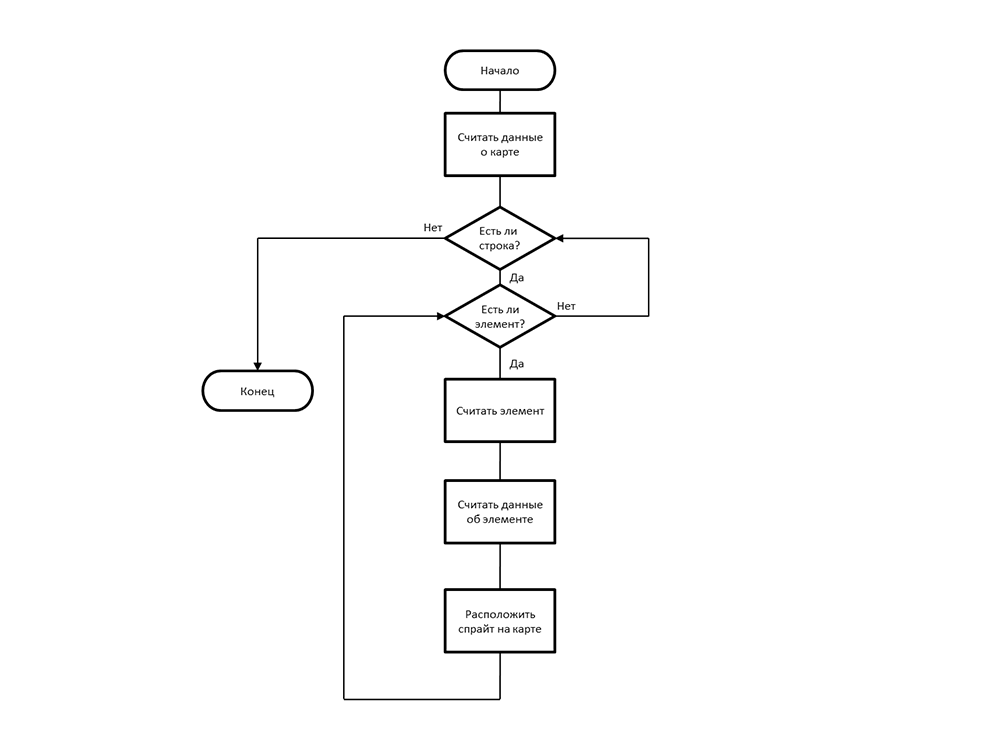


Рисунок 9. Схема алгоритма выстраивания карты

Сначала считываются данные о положении персонажа. Затем пользователь вводит команду. В зависимости от команды и соседних блоков, делается шаг. Если карта содержит еще один ряд блоков, то карта сдвигается в нужном направлении. (рисунок 10).

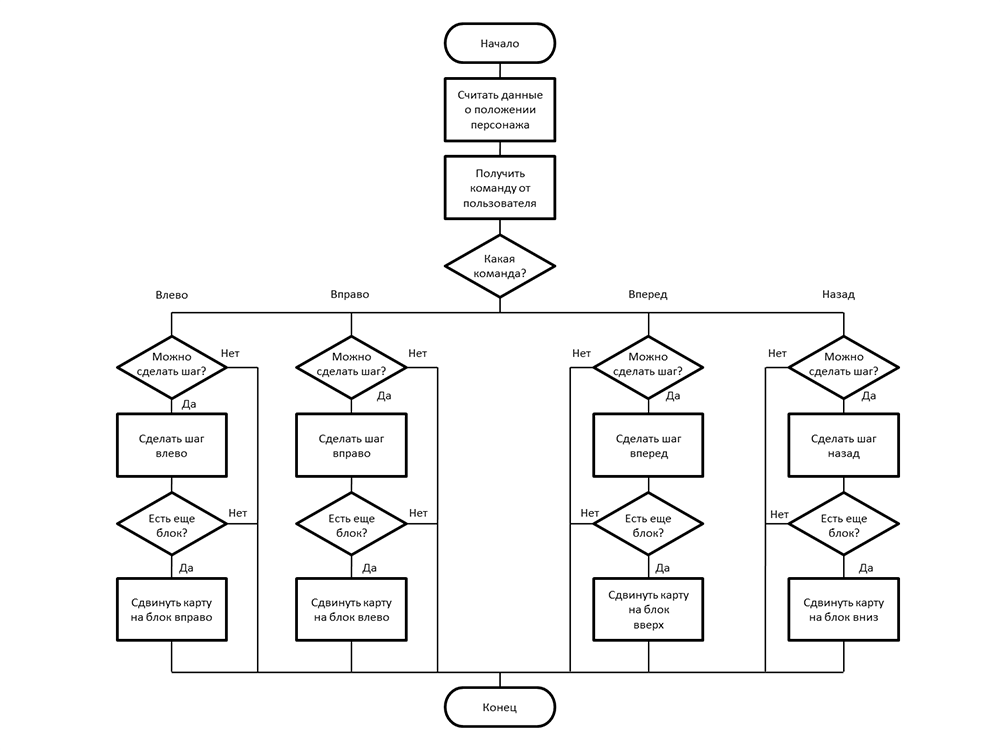


Рисунок 10. Схема алгоритма движения персонажа по блокам

Считываются данные об уровне, о степени прохождения и о положении персонажа. Все эти данные записываются в файл. С этим файлом программа начинает работать при следующем запуске. (рисунок 11).

Считываются данные об уровне и о вопросах этого уровня. Так же считываются данные о степени прохождении уровня и положении сундуков. Если игрок собрал необходимый ключ, то игра выдает нужный вопрос в зависимости от уровня и сундука. Пользователь вводит ответ, и, если он правильный, открытие сундука засчитывается. В ином случае, теряется жизнь. (рисунок 12)

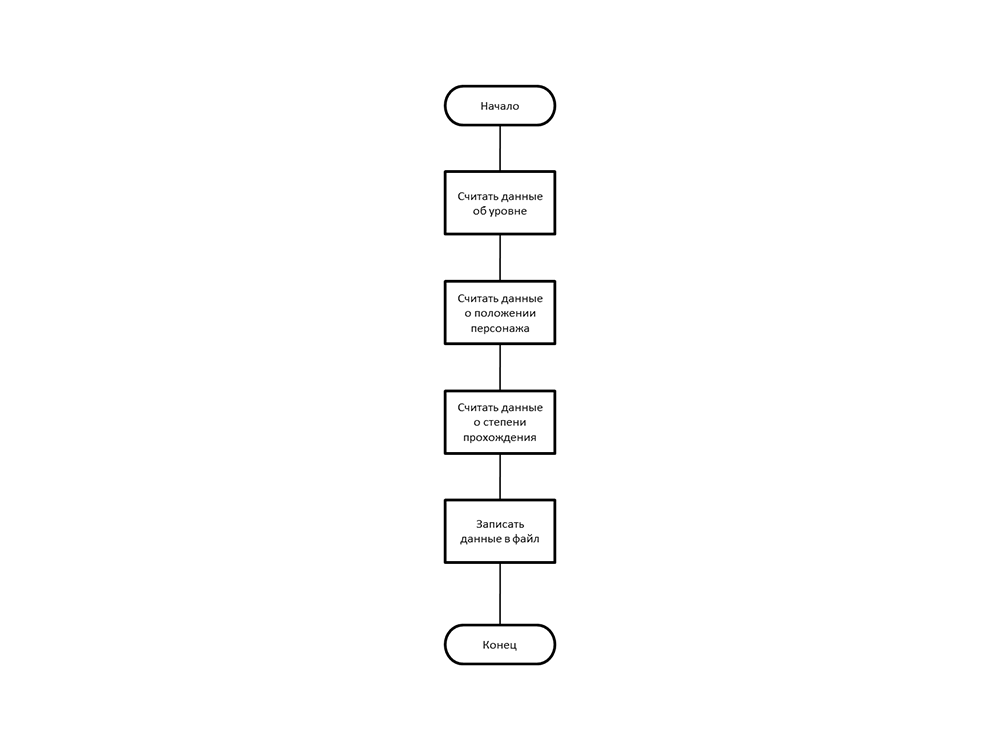
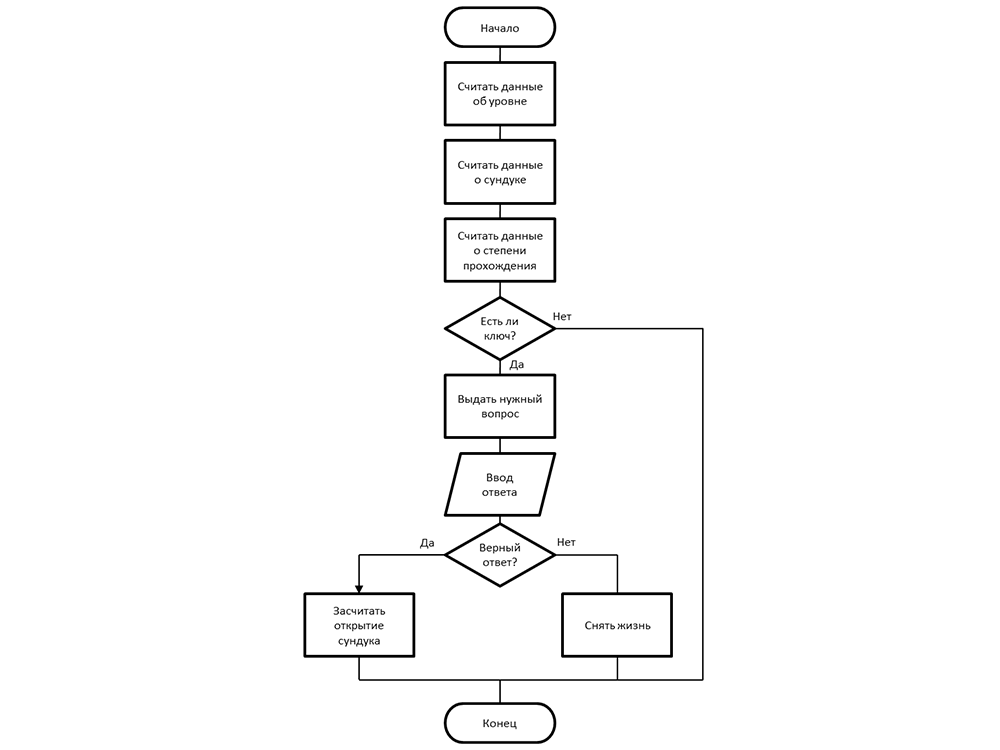


Рисунок 12. Схема алгоритма ответа на вопрос

Рисунок 11. Схема алгоритма сохранения игры

## **Структуры данных**

Уровень строится из 3 компонент, каждая из которых представляет собой файл:

* файл map.tmx содержит информацию, по которой строится карта уровня
* файл tasks.dat содержит информацию о заданиях
* файлы base.dat и levels.dat хранят информацию об уровнях

Карта собирается из нескольких массивов, содержащих информацию, необходимую для построения, а именно:

* массив блоков
* массив ключей
* массив сундуков
* start and end points

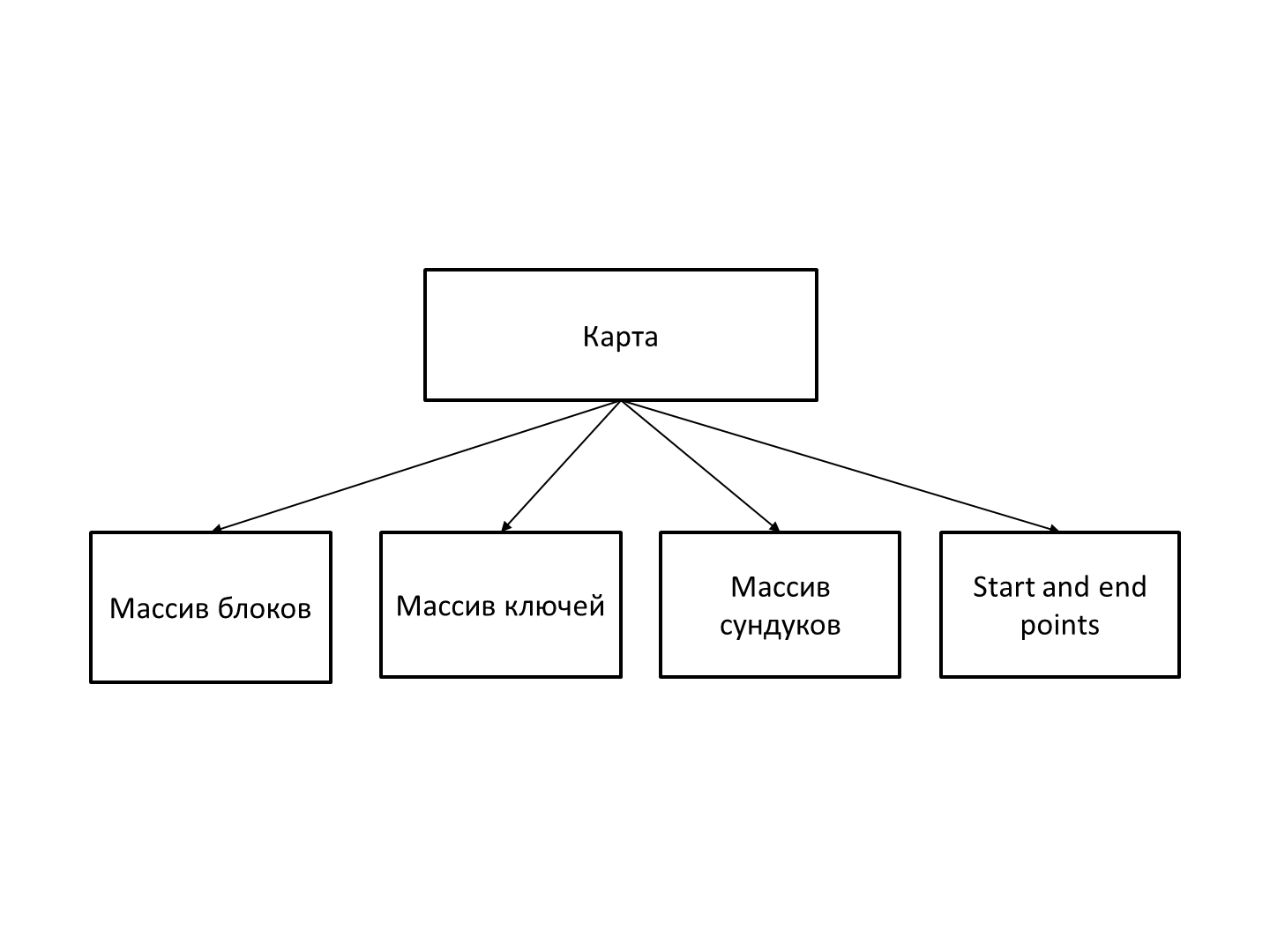


Рисунок 13. Структура данных карты

Информация об уровне хранится в файлах, которую можно использовать на любом этапе игры.

Таблица 2. Структура данных информации об уровне

|  |  |
| --- | --- |
| Файл | Содержимое |
| base.dat | base {'level\_stars': [0], 'level': 1, 'score': 0, 'levels': {'chests': [0, 0, 0], 'money': [0, 0, 0, 0, 0], 'keys': [0, 0, 0]}, 'tasks\_level': 1} |
| levels.dat | {'levels\_name': ['Первое знакомство', 'Невероятные приключения', 'Находка века', 'Золото где-то рядом'…]} |

Структура данных заданий так же хранится в соответствующих файлах

Таблица 3. Структура данных заданий

|  |  |
| --- | --- |
| Файл | Содержимое |
| tasks.dat | {'task\_3\_1': [{'answer': 'Эйлер', 'question': 'В честь кого в комплексном анализе названо тождество, связывающее 5 фундаментальных математических констант?'}…,'task\_2\_1': {..}] } |

## **Проектирование пользовательского интерфейса**

Было определено, что основными разделами игры станут:

* Меню
* Настройки
* Об авторах
* Выбор уровня
* Процесс игры
* Процесс игры - ответ на вопрос
* Окно после прохождения уровня

Все приведенные выше разделы были спроектированы в файле Вики Концепт интерфейса. Основным критерием создания интерфейса является его интуитивная понятность, чтобы пользователь быстро разобрался в нем без дополнительных инструкций.

**Описание способов тестирования как выделенных компонент, так и программного продукта в целом**

Тестирование проекта будет проходить в 3 этапа:

* Базовый тест.
* Геймплей тест
* Детальное тестирование Beta – продукта

1. Базовый тест должен выявить работоспособность движка. Такая проверка проводится на ранних стадиях создания проекта, и она подразумевает изучение работоспособности игры в целом, вне зависимости от действий игрока.
2. Вторая ступень тестирования ориентирована на геймплей. Когда движок более-менее стабилен, можно уделять больше времени балансу и разработке принципов игры.
3. Конкретных приоритетов теперь нет, стоит искать в игре все возможные баги и недочеты. Главная цель - перебор максимального количества тактик и подходов, использовать все предложенные возможности и менять стили игры.

Подробней распишем второй этап тестирования:

В игре будут протестированы сотни различных ситуаций (например, выход за пределы карты, попадание в мертвые зоны карты[[1]](#footnote-1), прорисовка изображения, движения главного персонажа, управление персонажем клавиатурой и т.д.).

## **Подготовка тестовых данных**

Выстраивание карты:

Таблица 4. Тестовые данные для выстраивания карты

|  |  |
| --- | --- |
| Ввод | Вывод |
| Пустая матрица | Пустая карта |
| Матрица из одной строки и одного элемента | 1 спрайт |
| Заполненные элементы матрицы | Карта из спрайтов |

Движение персонажа по блокам:

Таблица 5. Тестовые данные для движения персонажа

| Ввод | Вывод |
| --- | --- |
| Влево; Можно сделать шаг; Есть еще блок | Сделать шаг+сдвинуть карту вправо |
| Влево; Нельзя сделать шаг | Персонаж стоит на месте |
| Влево; Можно сделать шаг; Нет более блоков | Сделать шаг |
| Вправо; Можно сделать шаг; Есть еще блок | Сделать шаг+сдвинуть карту влево |
| Вправо; Нельзя сделать шаг | Персонаж стоит на месте |
| Вправо; Можно сделать шаг; Нет более блоков | Сделать шаг |
| Вперед; Можно сделать шаг; Есть еще блок | Сделать шаг+сдвинуть карту вверх |
| Вперед; Нельзя сделать шаг | Персонаж стоит на месте |
| Вперед; Можно сделать шаг; Нет более блоков | Сделать шаг |
| Назад; Можно сделать шаг; Есть еще блок | Сделать шаг+сдвинуть карту назад |
| Назад; Нельзя сделать шаг | Персонаж стоит на месте |
| Назад; Можно сделать шаг; Нет более блоков | Сделать шаг |

Ответ на вопрос:

Таблица 6. Тестовые данные для ответа на вопрос

|  |  |
| --- | --- |
| Ввод | Вывод |
| Найден ключ и он подходит к сундуку; Верный ответ | Открыть сундук |
| Найден ключ и он подходит к сундуку; Неверный ответ | Снять жизнь |
| Не найден ключ | Вывести сообщение: “Ключ еще не найден” |

# **Технологическая часть**

## **Выбор и обоснование технических средств**

При создании программного продукта будут применяться методологии структурного и объектно-ориентированного программирования на платформе python3+cocos2d. Данная платформа выбрана, так как является доступной и понятной для каждого члена команды из пяти человек и учитывает все условия, поставленные при выполнении учебной практики.

## **Описание пользовательского интерфейса**

На рисунке показано главное меню игры сразу после запуска программы. (рисунок 14).



Рисунок 14. Главное меню игры

На нем есть:

* Логотип игры
* Кнопка начала игры
* Кнопка настроек уровня сложности
* Кнопка просмотра информации об авторах

Для начала игры нужно нажать кнопку «Играть», после чего откроется меню с выбором уровня. Уровни, недоступные для игры, закрыты замком, для их открытия необходимо пройти предыдущие уровни. (рисунок 15).



Рисунок 15. Выбор уровня

Во время самой игры в левом верхнем углу находится информация о количестве собранных на этом уровне монет и сердцах. В правом верхнем углу находится информация о прогрессе в прохождении уровня: количество найденных ключей и количество открытых сундуков. В правом нижнем углу есть раздел подсказки, который поможет игроку понять, что необходимо сделать. (рисунок 16).

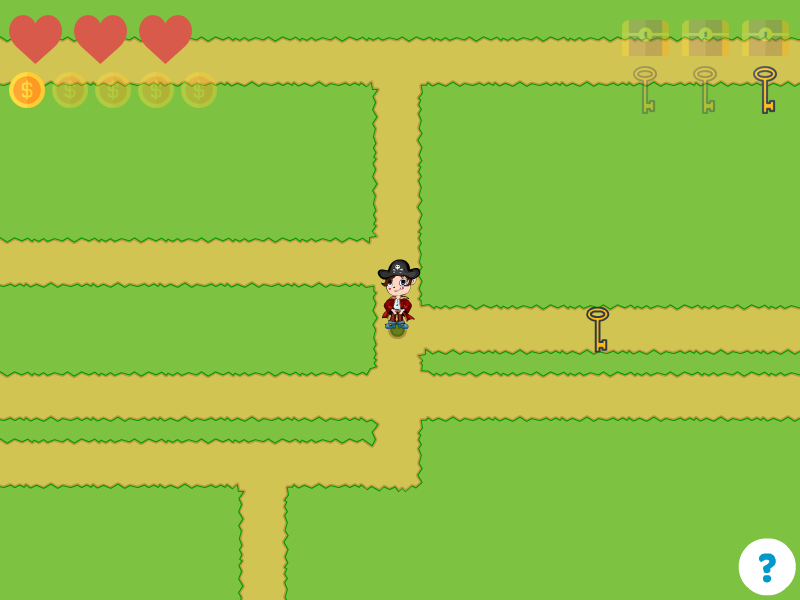


Рисунок 16. Демонстрация игрового интерфейса

При нажатии на кнопку «Настройки» откроется меню с выбором уровня сложности и с кнопкой для сброса прогресса, перед внесением изменений в прогресс или в уровень сложности пользователь должен повторно подтвердить это действие. (рисунок 17).

При нажатии на кнопку «Об авторах» откроет раздел с именами разработчиков ПО. (рисунок 18).



Рисунок 17. Раздел "Настройки"

Рисунок 18. Раздел "Об авторах"

## **Выбор и обоснование модели разработки**

Scrum является одной из наиболее популярных фреймворков разработки ПО. Scrum — это набор принципов, на которых строится процесс разработки, позволяющий в фиксированные и небольшие по времени итерации, называемые спринтами, предоставлять работающее ПО. Также строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость.

Важной причиной выбора именно Scrum для разработки ПО является ее эффективность при работе небольшого числа разработчиков и гибкость самого процесса. Scrum дает шанс каждому члену команды внести свой вклад в реализацию задачи, и, таким образом, ответственность распределяется между всеми членами команды.

## **Реализация программного продукта**

Вячеслав Романов – тимлид, ведущий разработчик, дизайнер интерфейса.

Зейнал Зейналов – тестировщик, разработчик основных этапов игры.

Анастасия Сушина – разработчик, художник элементов игры.

Григорий Будкин – тестировщик, разработчик основных этапов игры.

Сангинов Азамат – технический писатель, разработчик второстепенных этапов игры.

Вячеслав Романов:

* Создание главного меню
* Построение лабиринта
* Рисование анимации
* Создание системы сохранения прогресса
* Автоматическая ходьба
* Рисование график и интерфейса

Зейнал Зейналов:

* Программирование анимации
* Анимация интерактивных элементов игры
* Движение персонажа в лабиринте
* Тестирование

Анастасия Сушина:

* Действия с интерактивными элементами
* Табличка «Старт»
* Табличка «Правила»

Григорий Будкин:

* Раздел меню «Выбор уровня»
* Система ответов на вопросы
* Стрелки
* Движение после стрелок
* Тестирование

Сангинов Азамат:

* Раздел меню «Об авторах»
* Раздел меню «Настройки»
* Создание итоговой версии отчёта

**Реализация тестирования**

Тестирование программного продукта, в дальнейшем ПО, проводилось на протяжении всего времени разработки. Для тестирования методом “белого ящика” были использованы заранее предписанные ситуации.

На ранних стадиях написания проекта проводился базовый тест, благодаря которому удалось выявить и исправить недочеты с “проседанием” частоты кадров в секунду. После налаживания движка игры проводился геймплей тест, направленный на выявление проблем разработки принципов игры.

## Также было обращено внимание на тестирование системы “меню”, которая не дала никаких сбоев.

## После написания и реализации основных принципов игры, было проведено детальное бета-тестирование ПО. Данное тестирование проводилось не только разработчиками, а также “будущими игроками”, не имеющими никакого отношения к написанию игры.

## В ходе данного тестирования были обнаружены ошибки с “мертвыми зонами” карты, связанными с небольшими недочетами в логическом построении карты. Ошибка в анимации движений персонажа также была обнаружена и устранена (Попадая на блок воды, персонаж множество раз проваливался под воду).

## Обнаружены были и ошибки в начальном расположении персонажа, однако устранить данную ошибку удалось достаточно быстро.

В целом, в ходе тестирования ПО было обнаружено небольшое количество ошибок, что свидетельствует о высоком профессионализме и усердном труде наших разработчиков.

1. Мертвые зоны карты - участки карты, попадая на которые персонаж не может выйти, т.е. выйти обратно нельзя, логическая ошибка моделирования карты. [↑](#footnote-ref-1)