# MapVisualizator

Архитектурный документ

**Команда:** 29

**Авторы документа:**
Доржиев Донир Саянович (БПИ203) – раздел №4;
Кондаков Семен Васильевич (БПИ201) – раздел №3, 5;
Насыхова Анастасия Артемовна (БПИ201) – раздел №1, 5;
Неймышева Юлия Петровна (БПИ201) – раздел №2, 4.

**Учебный ассистент:** Щукин Владислав Евгеньевич

## Раздел регистрации изменений

| Версия документа | Дата | Описание изменения | Автор изменения |
|---|---|---|---|
| 1 | 13.03.2023 | Добавление разделов №1 и №2. | Насыхова А. А.<br>Неймышева Ю. П. |
| 2 | 14.03.2023 | Добавление раздела №3. | Кондаков С. В. |
| 3 | 15.03.2023 | Добавление подразделов №4. | Доржиев Д. С. |
| 4 | 17.03.2023 | Добавление подразделов №4. | Доржиев Д. С. |
| 5 | 18.03.2023 | Внесение уточнений в подразделы №4. | Кондаков С. В.<br>Доржиев Д. С.<br>Неймышева Ю. П.<br>Насыхова А. А. |
| 6 | 19.03.2023 | Добавление раздела №5. | Насыхова А. А.<br>Кондаков С. В. |
| 7 | 20.03.2023 | Итоговое оформление документа. | Неймышева Ю. П. |
| 8 | 12.03.2023 | Доработка документа и исправление неточностей | Кондаков С. В.<br>Доржиев Д. С.<br>Неймышева Ю. П.<br>Насыхова А. А. |

# 1. Введение

## 1.1. Название проекта

MapVisualizator

## 1.2. Задействованные архитектурные представления

### Структура архитектурного документа

## 1.3. Контекст задачи и среда функционирования системы

MapVisualizator – инструмент для быстрого отображения карты. Применяется в основном в сфере видеоигр, для отрисовки карты игрового процесса, с отображением различных ресурсов и бонусов на ней.

## 1.4. Рамки и цели проекта

MapVisualizator был разработан в рамках курсовой работы на 2 курсе Кондаковым С. В. Система была одним из компонентов игры, в которой карта отображалась с помощью устройств пользователей и для каждой игры пересоздавалась заново. Важным аспектом разработки было удобство пользователя при использования карты: возможность приближать/отдалять карту, перемещаться по ней.

Цель разработки: создать инструмент, с помощью которого можно быстро и качественно отобразить сгенерированную карту.

В рамках текущего состояния проекта система может отображать только карты, сгенерированные самостоятельно, но в перспективе можно предусмотреть возможность отображать карты пользователей, причем не только игровые.

# 2. Архитектурные факторы (цели и ограничения)

## 2.1. Ключевые заинтересованные лица

При анализе области применения системы были выявлены следующие заинтересованные лица:

| **Действующее лицо** | **Заинтересованность в системе** |
|---|---|
| Разработчик проекта | Простота поддержки и введения новой функциональности. |
| Пользователь | Простота установки инструмента;<br>Простота использования инструмента;<br>Быстрые запросы генерации в инструменте;<br>Генерация карт с указанными параметрами;<br>Просмотр сгенерированной карты. |

## 2.2. Ключевые требования к системе

При анализе ключевых заинтересованных лиц были выделены следующие ключевые требования к системе:

- Наличие пользовательского интерфейса программы;
- Генерация карты с учетом указанных параметров карты;
- Сохранение сгенерированной карты в формате изображения;
- Совместимость с операционными системами Windows, macOS и Linux;
- Открытый исходный код.

## 2.3. Ключевые ограничения

При анализе документации были выделены следующие ключевые ограничения системы:

- Ограничение производительности: (может работать плохо при обработке очень больших карт);
- Программа должны быть активным окном для корректной работы.

# 3. Общее архитектурное решение

Данная система (MapVisualizator) была спроектирована по принципам одноуровневой системы, согласно которой приложение работает, как единая система. Это обеспечивает простоту развёртывания, и отличную скорость связи, а также устраняет необходимость ISC (межсистемного взаимодействия) Тем не менее, такая система подходит только для небольших однопользовательских приложений.

Ядром архитектуры является процесс main.py, обрабатывающий все действия пользователя, и вызывающий, при возникновении соответствующих запросов, генератор – второй структурный элемент этого проекта.

Визуализация работы генератора происходит при помощи подгрузки изображений (далее - текстур) из той же директории. Корневую директорию в дальнейшем, мы будем называть RootDirectory. Сохранение изображений происходит в ту же директорию.

Работа пользовательского интерфейса осуществляется с помощью Tkinter GUI. Сбор всей информации от пользователя происходит с помощью соответствующих объектов интерфейса.

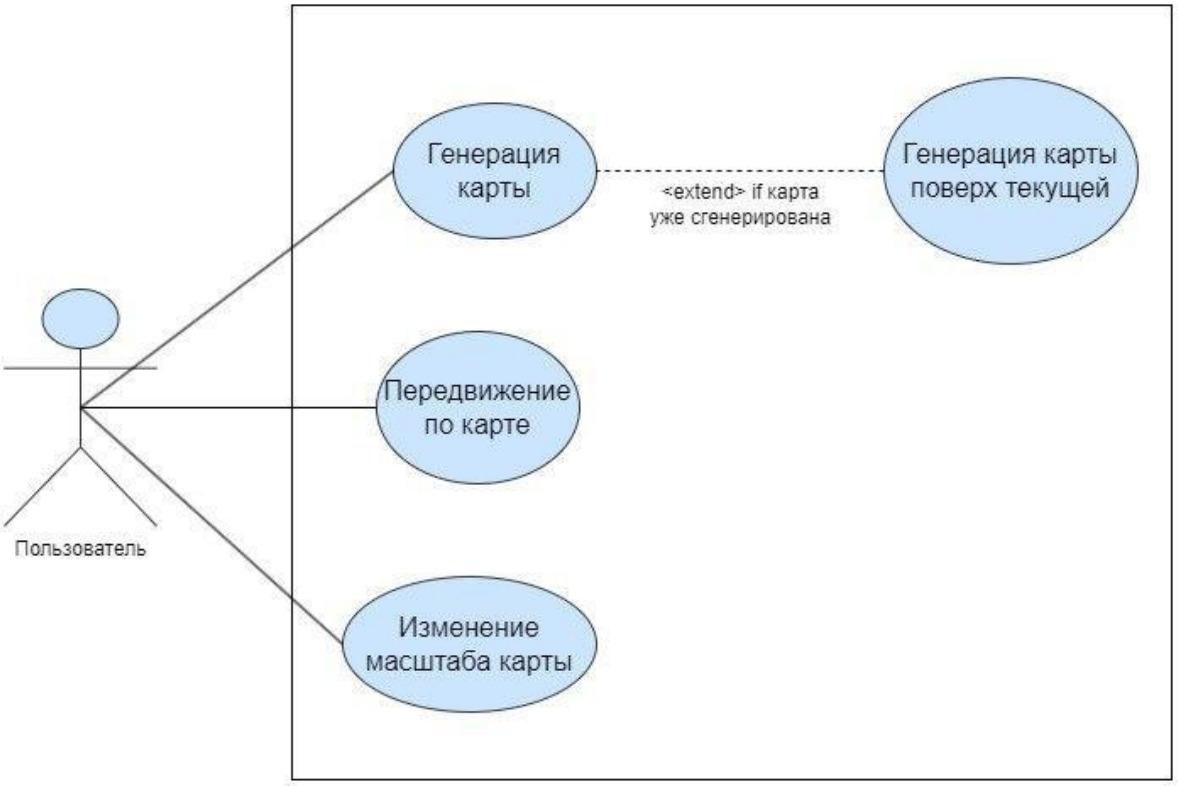
## 3.1. Принципы проектирования

MapVisualizator разработан в соответствии с одноуровневым стилем, когда за обработку всего функционала приложения отвечает один процесс, «жонглирующий» предварительно написанными функциями. Например, функция map_generate – отрисовывает изображение в окне, метод – map_generator – генерирует рельеф, методы draw_image – отрабатывают нажатие пользователя на клавиатуру.

# 4. Архитектурные представления

## 4.1. Представление прецедентов

Модель прецедентов



## 4.2. Логическое представление

Диаграмма пакетов

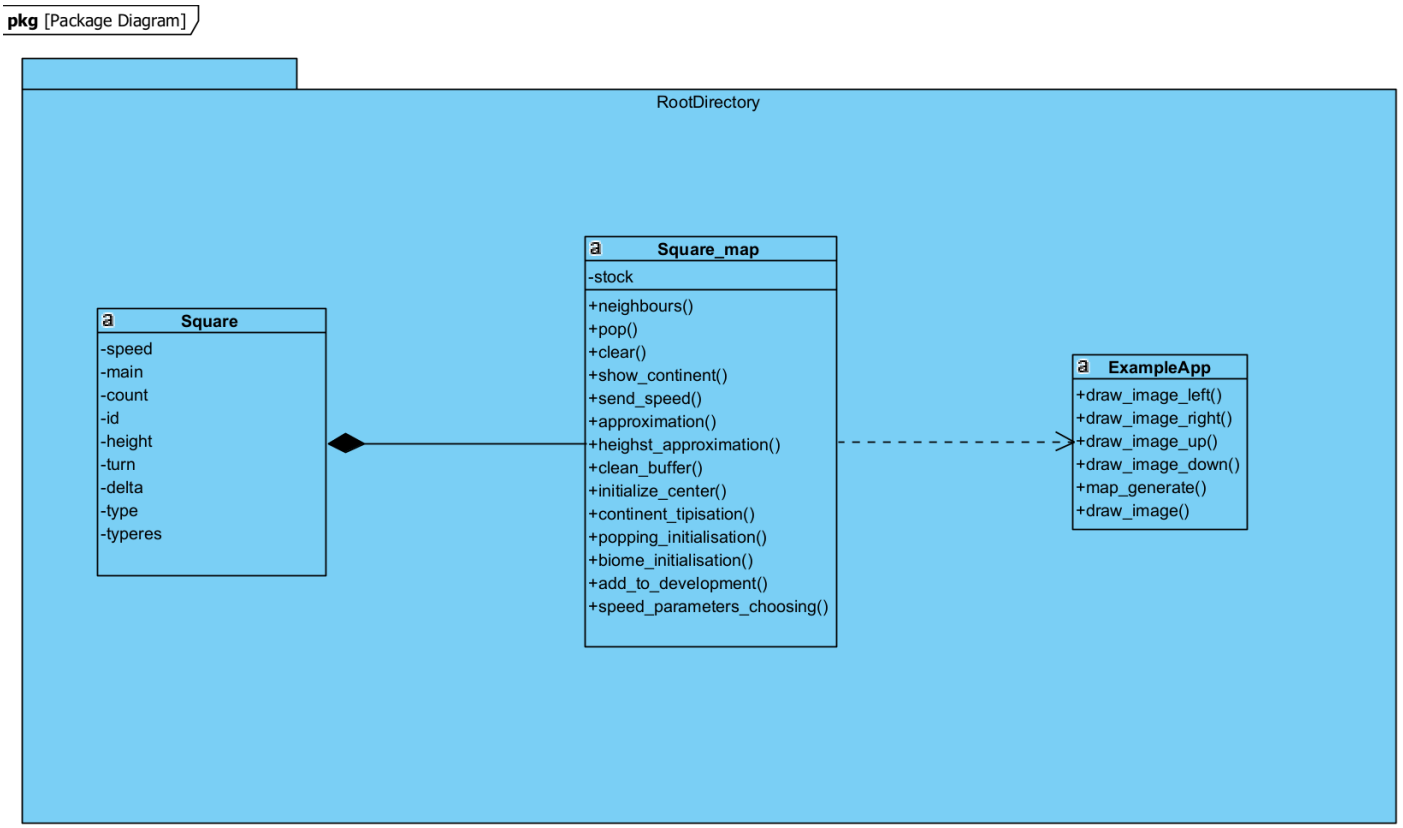
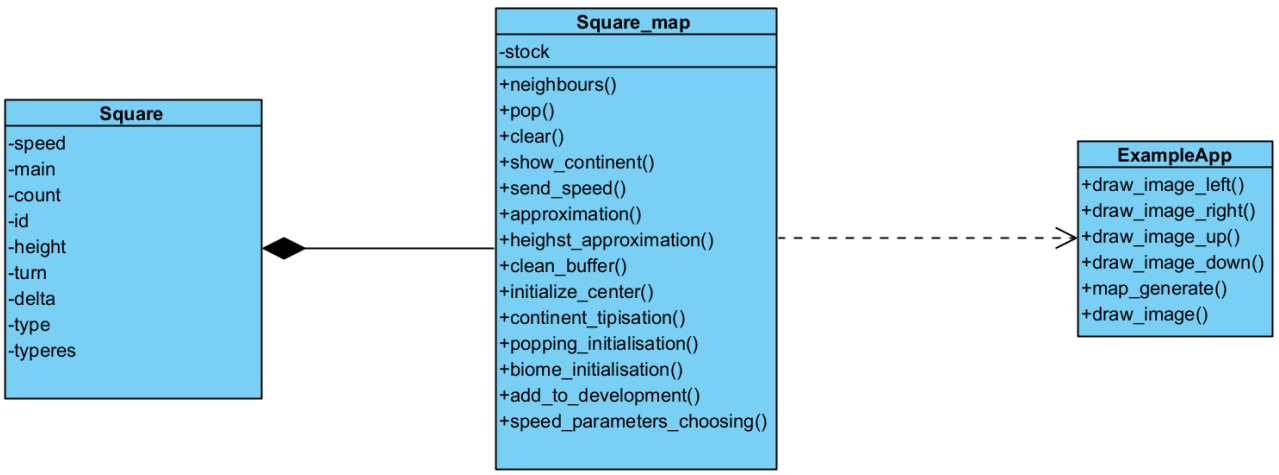
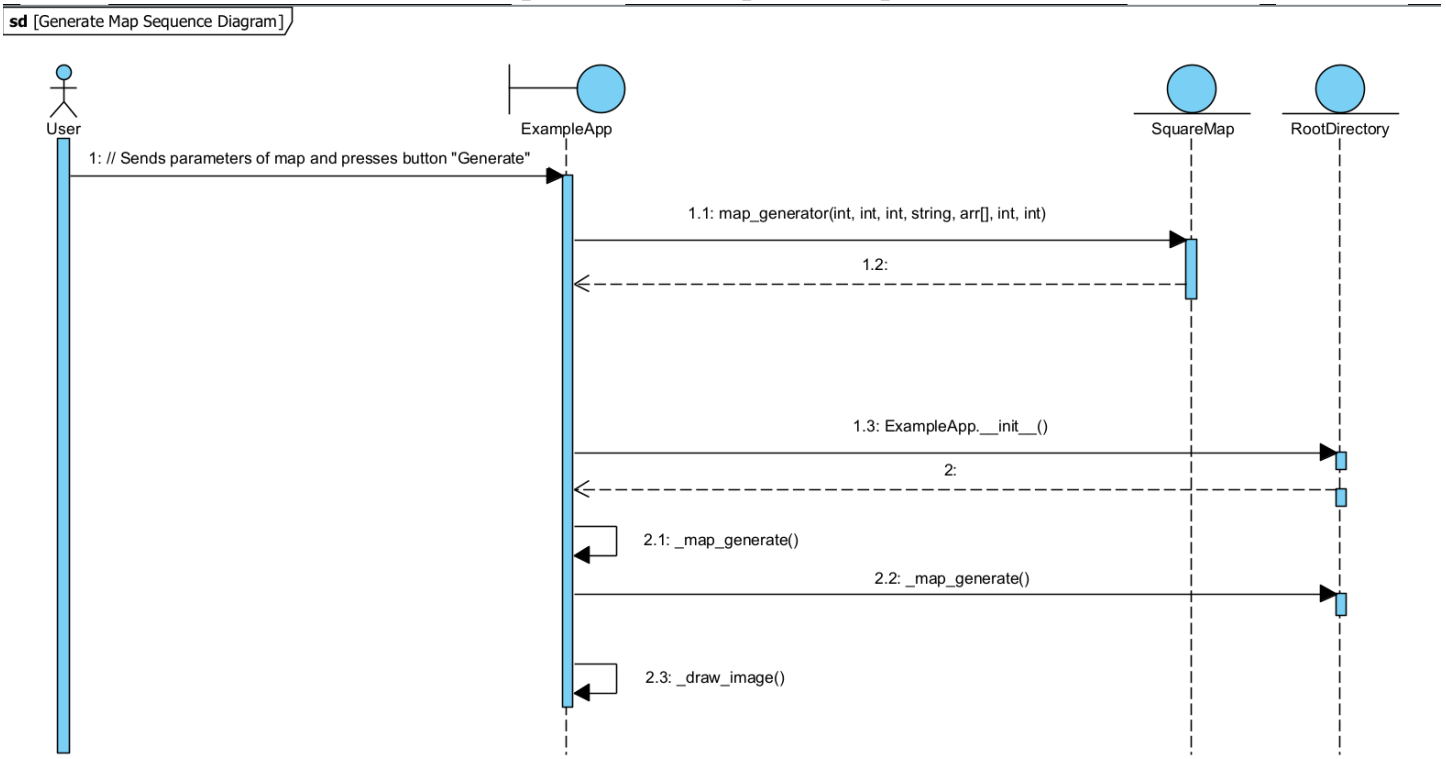



Диаграмма классов

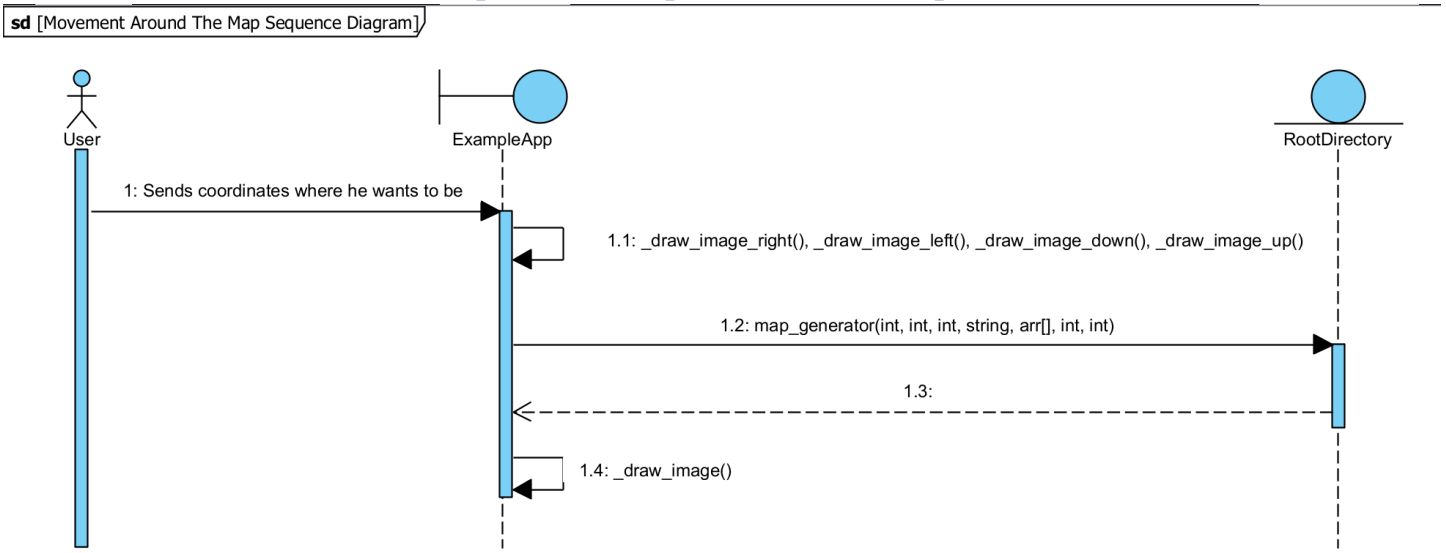


## 4.3. Представление архитектуры процессов

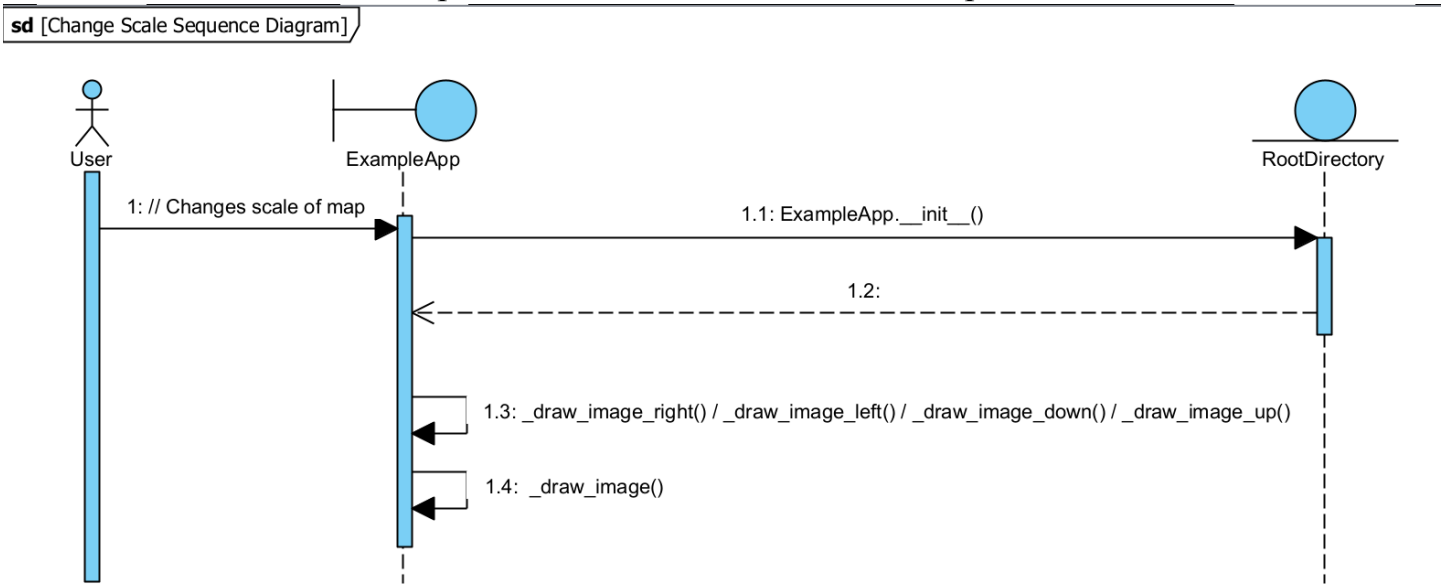
Прецедент генерации карты




Прецедент перемещения по карте



Прецедент изменения масштаба карты




## 4.4. Физическое представление архитектуры

- Windows 7, 8, 8.1, 10
- macOS 10.10.5 or later
- Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+

## 4.5. Представление развертывания

Существуют различные способы установки MapVisualizator.

- Предварительно скомпилированные бинарные файлы;
- Сборка из исходного кода.

## 4.6. Представление архитектуры данных

У приложения нет потребности в базе данных, данные в системе будут сохранены локально. Будет использована локальная файловая система для работы с файлами, а именно, работа будет вестись в корневой директории.

## 4.7. Представление архитектуры безопасности

В данной системе не предусмотрена авторизация.

## 4.8. Представление реализации и разработки

MapVisualizator написан на языке Python. Работа велась в IDE PyCharm. Код хранится в репозитории на GitHub.

На данный момент, история изменений такова:

1) 06.03.2023, 22:15:34 – Первичное копирование директории с локального носителя.
2) 06.03.2023, 23:24:25 - Копирование соответствующей версии Python в архив с проектом.

## 4.9. Представление производительности

Проект не является объемным, однако операции генерации выполняются не быстро. Кроме того, на данный момент, при отрисовке участка карты, обрабаатывается всё изображение, что также сказывается на производительности как генератора, так и визуализатора. Данные проблемы, часто вызванные неправильными архитектурными решениями, создают почву для дальнейшей доработки проекта.

## 4.10. Атрибуты качества системы

Ключевыми атрибутами качества системы являются:

- Производительность: Визуализатор карты должен быть быстрым, чтобы игрокам было комфортно перемещаться по карте и выполнять действия в рамках игрового процесса.
- Реалистичность: Визуализатор карты должен создавать картину мира, которая выглядит красиво, а также соответствует ожиданиям игроков.
- Адаптивность: визуализатор должен настраиваться под разные разрешения экранов устройств пользователей.
- Надежность: визуализатор должен работать стабильно и без сбоев.
- Разрешение: визуализатор должен поддерживать высокое разрешение для отображения детализированных карт.

### 4.10.1 Объем данных и производительность системы

MapVisualizator работает с пользовательским вводом. (несколько параметров карты). Генерирует же довольно объёмные png-файлы. Время генерации файла

варьируется в зависимости от размера карты, но может достигать нескольких минут, при достаточно больших размерах карты.

### 4.10.2 Гарантии качества работы системы

Тестирование является самой главной гарантией качества MapVisualizator. Тесты должны покрывать все возможные сценарии работы с системой. Если у пользователя возникает ошибка, он может написать на github в раздел issues или на почту разработчиков.

# 5. Технические описания отдельных ключевых архитектурных решений

## 5.1. Техническое описание решения №1: Хранение данных карты

### 5.1.1 Проблема

Как должно быть обеспечено хранение данных карты?

### 5.1.2 Идея решения

Следует хранить карту высот и расположение объектов в массиве. Остальные метаданные (текстуры, png сгенерированных карт и т.д.) будут храниться в файловой системе пользователя.

### 5.1.3 Факторы

Быстрая отрисовка является ключевым требованием к визуализатору карт.

### 5.1.4 Решение

Данные, к которым нужен быстрый доступ, будут храниться в массивах, остальные в файловой системе.

### 5.1.5 Мотивировка

Такая система хранения удобна с точки зрения быстрого доступа к значениям. Карта высот – структура с большим количеством значений (100x100 – минимальный размер карты). Ко всем значениям нужен постоянный доступ для корректного перемещения персонажей. Расположение объектов (ресурсов, бонусов, персонажей и т.д.) также будет храниться в массиве в связи с высокой динамикой изменения координат объектов.

### 5.1.6 Неразрешенные вопросы

Нет.

### 5.1.7 Альтернативы

Карту высот и расположение объектов можно было бы также хранить в базе данных, но этот вариант не подходит, поскольку в данном случае снизится скорость отрисовки.

## 5.2. Техническое описание решения №2: Визуализация сгенерированных карт

### 5.2.1 Проблема

Стоит ли вынести части кода, отвечающие за визуализацию, в отдельный модуль?

### 5.2.2 Идея решения

Следует вынести в отдельный модуль.

### 5.2.3 Факторы

Требования технического задания предусматривают возможность дальнейшего масштабирования инструмента.

### 5.2.4 Решение

Компоненты визуализации будут вынесены в отдельный модуль.

### 5.2.5 Мотивировка

Отделение модуля генерации карты от модуля визуализации имеет несколько преимуществ:

1) Расширяемость. При такой архитектуре легче добавлять новые функции.
2) Повышение качества тестируемости. Легче проводить юнит-тестирование.

### 5.2.6 Неразрешенные вопросы

Нет.

### 5.2.7 Альтернативы

Альтернативные решения не рассматривались.

# 6. Приложения

## 6.1. Словарь терминов

GitHub – это онлайн-платформа, которая позволяет разработчикам и командам разработчиков хранить, управлять и совместно работать над своими проектами. Она специализируется на системе контроля версий Git, что позволяет упростить процесс разработки программного обеспечения и обеспечить более эффективную командную работу.

PyCharm – это интегрированная среда разработки (IDE) для языка программирования Python. Она предназначена для упрощения процесса создания, отладки и тестирования программ на языке Python.

База данных — совокупность всех объектов БД (таблиц, процедур, триггеров и т.д.), статических данных (неизменяемых данных, хранящихся в lookup-таблицах) и пользовательских данных (которые изменяются в процессе работы с приложением).

Визуализатор карты – программа, которая выводит на экран изображение игрового мира.

Высотная карта – это тип карты, на которой показаны элементы, которые влияют на рельеф земной поверхности, такие как горы, долины, холмы и др. Эта карта используется в визуализаторе карт для создания трехмерных моделей ландшафта и других форм земной поверхности.

Генерация карты – это процесс создания игрового уровня или мира с помощью алгоритмов и случайных генераторов. Она может создавать различные типы карт, такие как поля, леса, пустыни, горы и т.д., а также управлять размещением объектов игрового мира.

Игровой объект – персонаж, предмет, структура, которые можно разместить на карте.

Инструмент - это программа, приложение или утилита, которая используется для ускорения и упрощения разработки, тестирования, отладки или улучшения качества программного обеспечения.

Карта – игровое пространство, на котором происходят события игры.

Координаты – числовое значение, которое определяет местоположение объекта на карте.

Модуль – это самодостаточная часть программы, которая содержит определенную функциональность и может быть использована другими частями программы.

Текстура – графическое изображение, которое накладывается на поверхности мира.

Юнит-тестирование – это методология тестирования программного обеспечения, при которой отдельные единицы программного кода (юниты) тестируются на предмет соответствия спецификации. Целью юнит-тестирования является проверка корректности работы кода, установка ожидаемых результатов и обнаружение возможных ошибок на ранних этапах разработки.

## 6.2. Ссылки на используемые документы

[1] GitHub репозиторий - https://github.com/seemur1/MapVisualizator