

Map Visualizator

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. 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. Доржиев Д. С. Неймышева Ю. П. Насыхова А. А.
6	19.03.2023	Добавление раздела №5.	Насыхова А. А. 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. Доржиев Д. С. Неймышева Ю. П. Насыхова А. А.
9	26.04.2023	Корректировка исходного документа	Неймышева Ю. П.
10	28.04.2023	Обновление информации во всех разделах в соответствии с перестроенной архитектурой	Насыхова А. А.
11	29.04.2023	Добавление нового логического представления и новых диаграмм последовательности	Кондаков С. В. Доржиев Д. С.

1. Введение

1.1. Название проекта

MapVisualizator

1.2. Задействованные архитектурные представления

Структура архитектурного документа

1. Введение.....	3
1.1. Название проекта.....	3
1.2. Задействованные архитектурные представления.....	3
1.3. Контекст задачи и среда функционирования системы.....	4
1.4. Рамки и цели проекта.....	4
2. Архитектурные факторы (цели и ограничения).....	5
2.1. Ключевые заинтересованные лица.....	5
2.2. Ключевые требования к системе.....	5
2.3. Ключевые ограничения.....	5
3. Общее архитектурное решение.....	6
3.1. Принципы проектирования.....	6
4. Архитектурные представления.....	7
4.1. Представление прецедентов.....	7
4.2. Логическое представление.....	8
4.3. Представление архитектуры процессов.....	9
4.4. Физическое представление архитектуры.....	10
4.5. Представление развертывания.....	11
4.6. Представление архитектуры данных.....	11
4.7. Представление архитектуры безопасности.....	11
4.8. Представление реализации и разработки.....	11
4.9. Представление производительности.....	11

4.10. Атрибуты качества системы.....	12
4.10.1 Объем данных и производительность системы.....	12
4.10.2 Гарантии качества работы системы.....	13
5. Технические описания отдельных ключевых архитектурных решений.....	14
5.1. Техническое описание решения №1: Хранение данных карты.....	14
5.2. Техническое описание решения №2: Визуализация сгенерированных карт.....	15
6. Приложения.....	16
6.1. Словарь терминов.....	16
6.2. Ссылки на используемые документы.....	17

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 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 на устройствах пользователей и для каждой игры пересоздавалась заново.

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

Важным аспектом при разработке была необходимость в том, чтобы карта отображала местность, максимально приближенную к реальности (с плавными переходами от суши к воде, без резких перепадов высот и т. д.)

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; Простота использования инструмента; Быстрые запросы генерации в инструменте; Генерация карт с указанными параметрами; 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:

- Должна быть обеспечена интеграция с сервисом, отвечающим за основную механику игрового процесса (перемещение по карте нескольких персонажей, сбор ресурсов и т. д.).

3. **Общее архитектурное решение**

Состояние ДО

Данная система (MapVisualizator) была спроектирована по принципам монолитной системы, согласно которой приложение работает, как цельный модуль. Это обеспечивает простоту развертывания и разработки, а также устраняет необходимость ISC (межсистемного взаимодействия). Тем не менее, такая система подходит только для небольших однопользовательских приложений.

MapVisualizator изначально разработан в соответствии со стилем main и подпрограммы, когда за обработку всего функционала приложения отвечает один процесс, «жонглирующий» предварительно написанными функциями. Так процесс, соответствующий запуску main.py, вызывает класс ExampleApp, обрабатывающий все действия пользователя, а он в свою очередь вызывает методы генерации и визуализации карты.

Визуализация карты происходит при помощи подгрузки изображений (далее – текстур) из корневой директории (далее – 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.

Состояние ПОСЛЕ

В результате работы по улучшению проекта архитектура изменилась следующим образом:

- 1) main.py был разбит на два разных файла: Visualizator.py и Generator.py, каждый из которых способен работать независимо. И взаимодействовать друг с другом, также - независимо. (Генерация – через вызов скрипта Generator, с передачей туда требуемых параметров, подгрузка карт в Visualizator осуществляется через вспомогательные файлы, сохраняемые генератором в results).
- 2) Корневая директория RootDirectory была декомпозирована на две папки: ResourcesDirectory (директория, в которой хранятся текстуры) и ResultsDirectory (директория, в которую выводятся результаты работы генератора и визуализатора).

3) В проекте были удалены хардкоды и использование устаревших библиотек.

Таким образом, теперь архитектура системы состоит из двух независимых модулей и основного процесса, в котором они взаимодействуют.

3.1. Принципы проектирования

В изначальном виде разработка приложения не подразумевала принципов проектирования.

На этапе внесения предложенных изменений в код все разработчики придерживались следующих принципов:

- Соблюдение стандарта PEP8 для языка Python;

4. Архитектурные представления

4.1. Представление прецедентов

В данной системе реализуются use case, изображенные на следующей модели прецедентов (рис. 1).

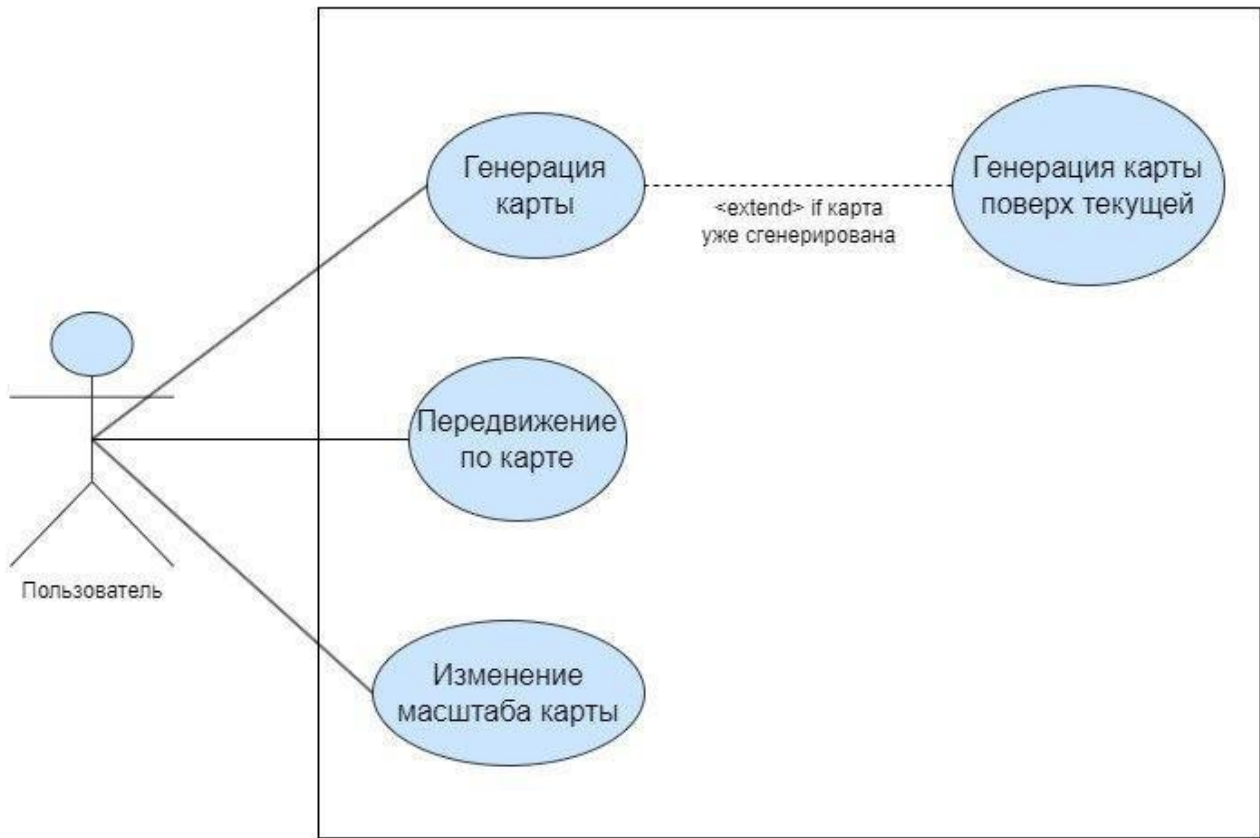


Рис. 1: Модель прецедентов

Внесенные в рамках третьего этапа архитектурные изменения не повлияли на модель прецедентов.

4.2. Логическое представление

Исходное логическое представление архитектуры выглядит следующим образом:

Архитектурный документ: MapVisualizator

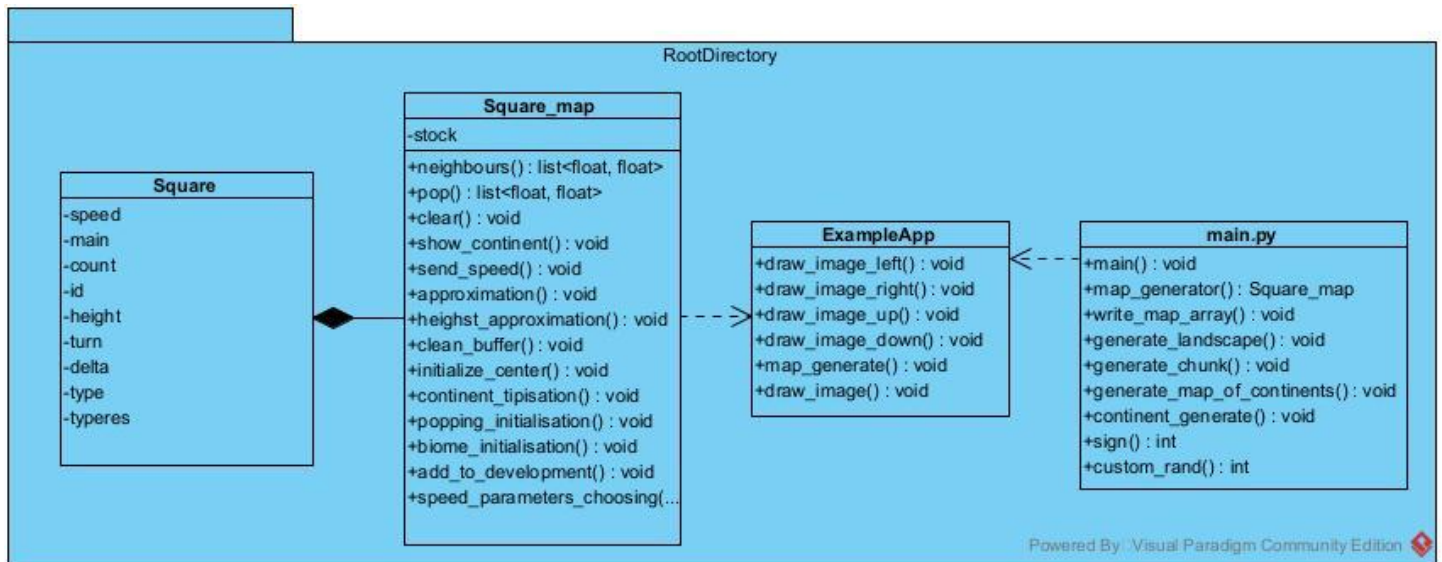


Рис. 2: Диаграмма пакетов до внесения архитектурных изменений

После внесения изменений логическое представление архитектуры MapVisualizator приняло следующий вид:

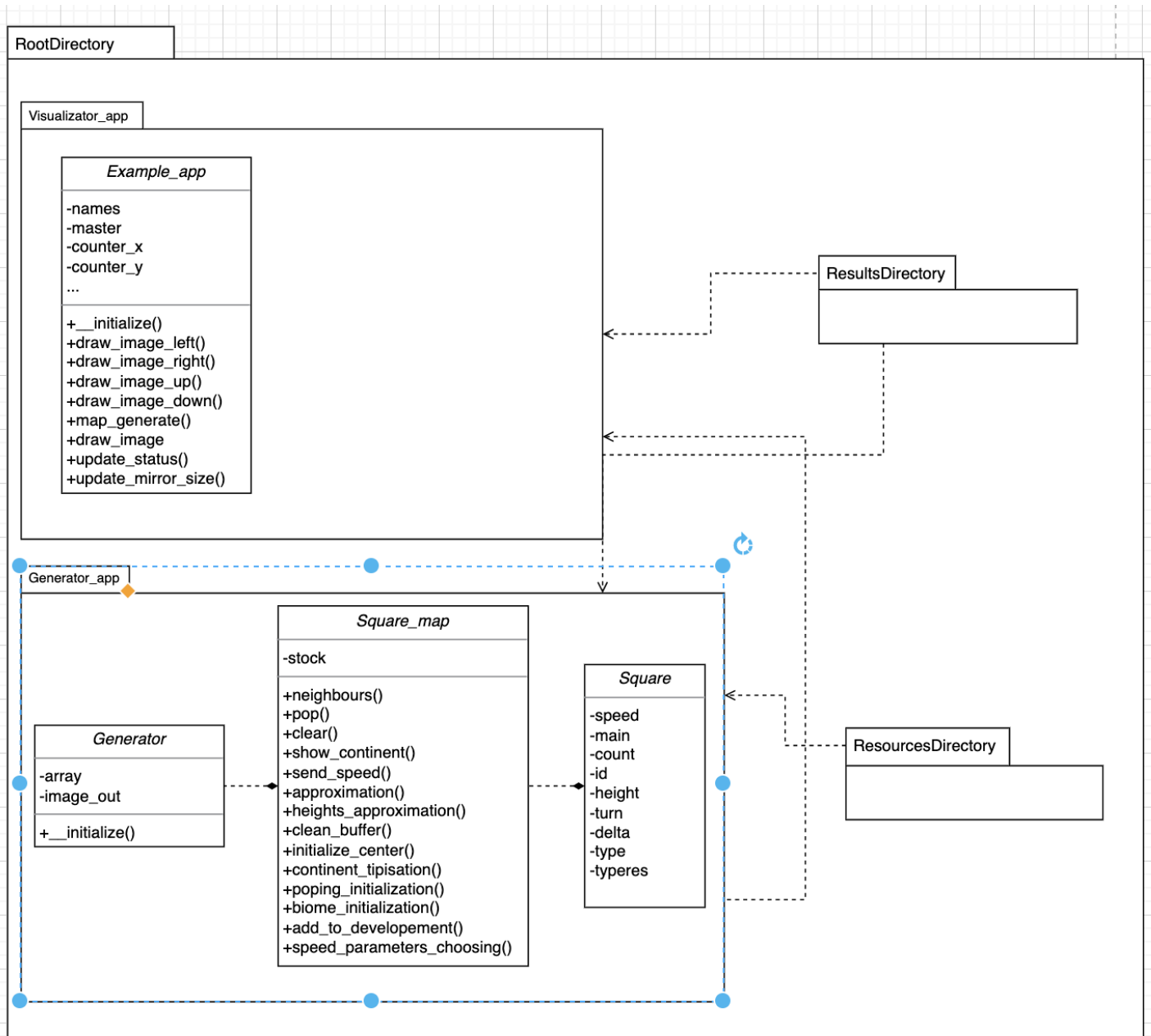


Рис. 3: Диаграмма пакетов после внесения архитектурных изменений

4.3. Представление архитектуры процессов

Изначально был один процесс, который в зависимости от действий пользователя вызывал необходимые собственные методы или методы класса ExampleApp.

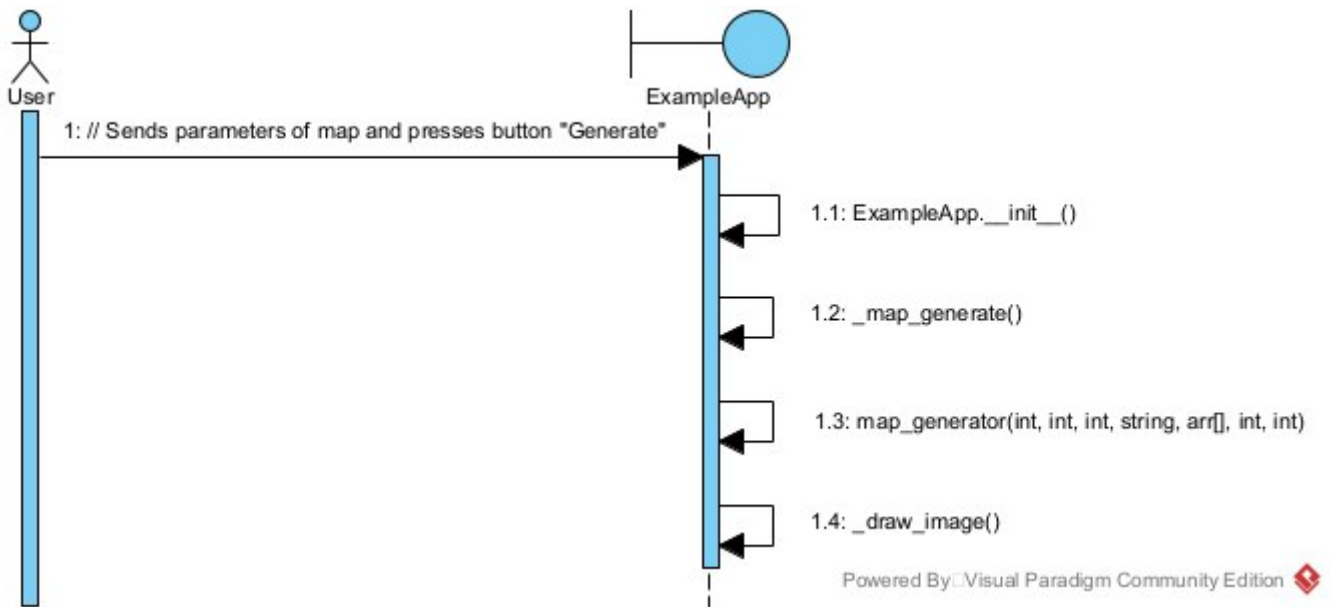


Рис. 4: Диаграмма последовательности для прецедента «Генерация карты»

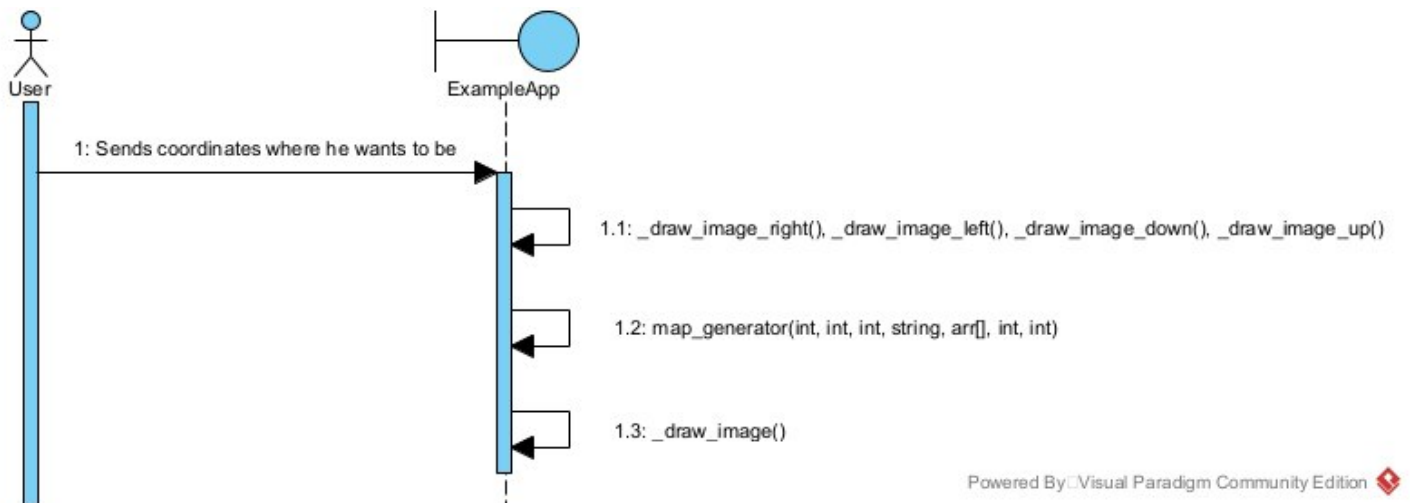


Рис. 5: Диаграмма последовательности для прецедента «Перемещение по карте»

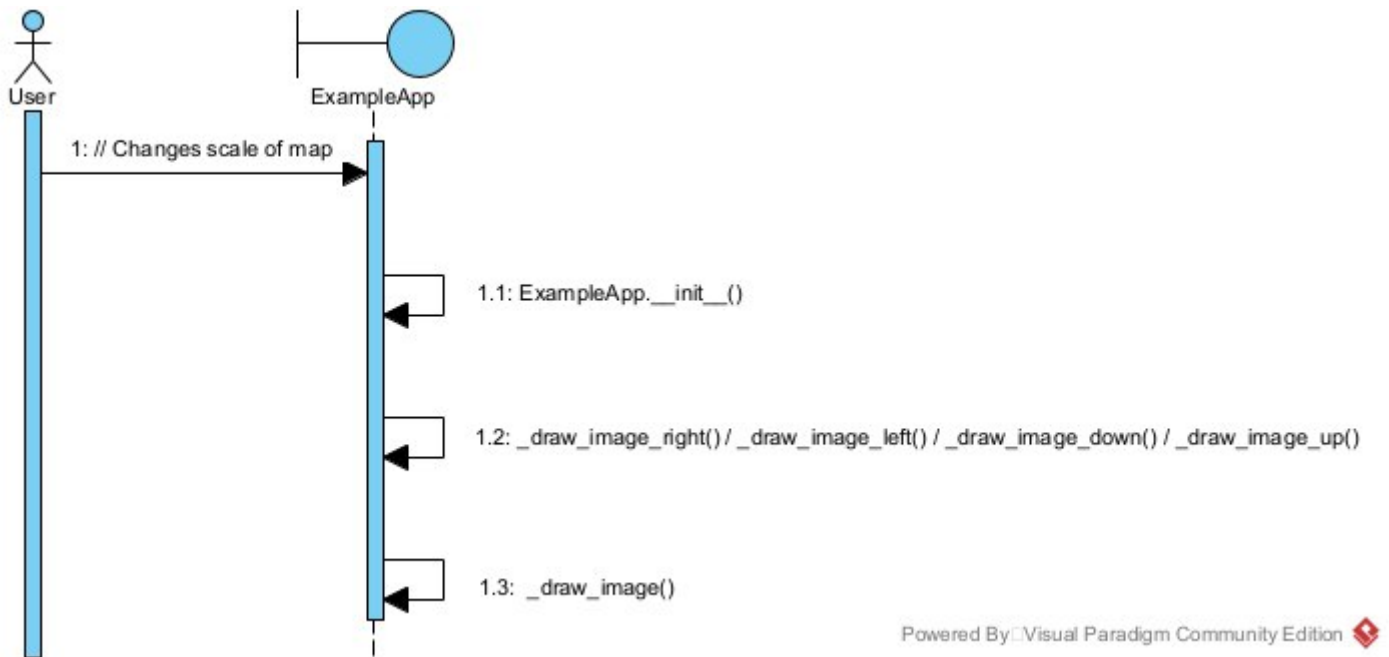


Рис. 6: Диаграмма последовательности для прецедента «Изменение масштаба карты»

После внесения изменений исходный процесс был разбит на три:

1. Исходный процесс, который отслеживает действия пользователя;
2. Процесс, который отвечает за генерацию карты и сохранение ее текстового представления на устройстве пользователя;
3. Процесс, который отвечает за вывод результата (визуализацию в окне пользователя + сохранение png-версии карты на на устройстве пользователя).

4.4. Физическое представление архитектуры

MapVisualizator – кроссплатформенное desktop-приложение. Оно может быть развернуто на устройствах с одной из следующих операционных систем:

- Windows 7, 8, 8.1, 10
- macOS 10.10.5+
- Ubuntu 18.04+, Debian 10+

Минимальные требования для устройства, на которое может быть установлен MapVisualizator:

- 64-разрядный компьютер с оперативной памятью не менее 2 ГБ;
- Не менее 4 ГБ свободного места на жестком диске.

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, была использована локальная файловая система для работы с файлами, вся работа будет вестись в корневой директории.

После внесения изменений корневая директория была декомпозирована на две папки: ResourcesDirectory и ResultsDirectory, описанные в разделе 3.

4.7. Представление архитектуры безопасности

В данной системе механизмы безопасности не требуются.

4.8. Представление реализации и разработки

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

При разработке были использованы следующие фреймворки:

- Библиотека Tkinter – используется для создания графических интерфейсов. Он предоставляет набор виджетов, таких как кнопки и поля ввода. Tkinter также обеспечивает возможности работы с графикой, включая отображение изображений (вывод текстур на экран).
- Модули Image и ImageTk из библиотеки Pillow – используются для работы с изображениями в Python. Image предоставляет удобный способ чтения и записи изображений в различных форматах, а также основные функции обработки изображений, такие как изменение размера и обрезка. ImageTk, с другой стороны, используется для отображения изображений на экране.

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.
- 3) 23.04.2023 11:37 – Декомпозиция базы данных
- 4) 29.04.2023 12:18 – Оптимизация исходного кода программы (избавление от хардкода и устаревших технологий)
- 5) 29.04.2023 xx:xx – Разделение логики и интерфейса
- 6) 29.04.2023 xx:xx – Разбиение исходного монолита на два модуля (генератор и визуализатор).

4.9. Представление производительности

Проект не является объемным, однако операции генерации выполняются не быстро при больших размерах карты (больше 200x200). Кроме того, на данный момент, при отрисовке участка карты, обрабатывается всё изображение, что также сказывается на производительности как генератора, так и визуализатора.

При генерация карты размерами 20x20 время ожидания результата составляет 10 секунд.

При генерация карты размерами 50x50 время ожидания результата составляет 50 секунд.

При генерация карты размерами 100x100 время ожидания результата составляет 200 секунд.

Для сохранения карты размерами 100x100 необходимо 3,5 мб памяти.

Результаты изменения производительности в результате внесения изменений в проект будут представлены на следующем этапе работы.

4.10. Атрибуты качества системы

Ключевыми атрибутами качества системы являются:

- **Производительность:** Визуализатор карты должен быть быстрым (время ожидания результата меньше 5 минут для карты размером до 200x200), чтобы игрокам было комфортно перемещаться по карте и выполнять действия в рамках игрового процесса. Для перемещения от одной стороны карты размерами до 100x100 до другой необходимо не более 20 секунд.

- **Реалистичность:** Визуализатор карты должен создавать картину мира, которая выглядит красиво, а также соответствует ожиданиям игроков. Средняя оценка реалистичности и красоты карт игроками в анонимном независимом опросе должна составлять не менее 7 из 10.
- **Адаптивность:** Визуализатор должен настраиваться под разные разрешения экранов устройств пользователей. При разработке должны использоваться методы автоматического получения размеров экрана.
- **Надежность:** Программа должна работать без сбоев минимум в 8 запусках из 10 с генерацией карт размерами до 200x200.
- **Разрешение:** Визуализатор должен поддерживать высокое разрешение (до 1920x1080) для отображения детализированных карт.
- **Кроссплатформенность:** Визуализатор карт должен работать на операционных системах, описанных в разделе 4.4 Физическое представление архитектуры.

4.10.1 Объем данных и производительность системы

MapVisualizator работает с пользовательским вводом (несколько параметров карты). Генерирует png-файлы размерами до 4 мб для карт размерами до 100x100. Время генерации файла варьируется в зависимости от размера карты, но может достигать нескольких минут (2-3 минуты), при размерах карты от 100x100.

4.10.2 Гарантии качества работы системы

Тестирование является одной из гарантий качества системы. Тесты должны покрывать все возможные сценарии работы.

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

Текстуры и карты высот, 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.

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.

Модуль – это самостоятельная 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>