MapVisualizator

Предложение по переработке системы

Команда: 29

Авторы предложения:

Доржиев Донир Саянович (БПИ203) — Разделы 1, 2, 3, 4, 5, 6

Кондаков Семен Владимирович (БПИ201) — Разделы 1, 2, 3, 4, 5, 6

Насыхова Анастасия Артемовна (БПИ201) — Разделы 1, 2, 3, 4, 5, 6

Неймышева Юлия Петровна (БПИ201) — Разделы 1, 2, 3, 4, 5, 6

Учебный ассистент: Щукин Владислав Евгеньевич

Дата: 11.04.2023

Версия документа: 1

1. Проблема

1.1. Приложение чересчур атомарное. Часть, отвечающая за генерацию карты, по сути, является отдельным скриптом, однако очень жёстко связана с модулем “Визуализатор”. Кроме того, весь функционал приложения взвален на класс ExampleApp, что также не способствует масштабироемости, и понимаемости приложения.

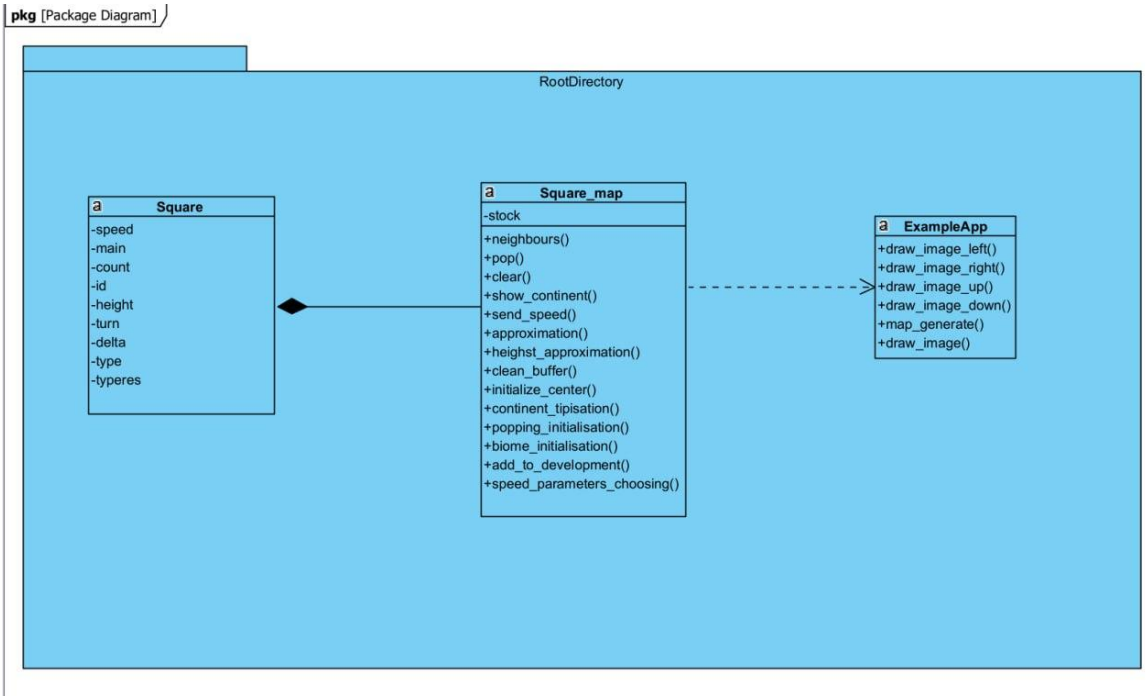
1.2. Нет разделения между интерфейсом и бизнес логикой. Вся логика работы с интерфейсом и работы программы находится в одном классе ExampleApp.

1.3. Приложение разрабатывалось несколько лет назад, из-за чего в нем некоторые части кода являются устаревшими (на данный момент существуют более гибкие и масштабируемые способы реализации пользовательского интерфейса).

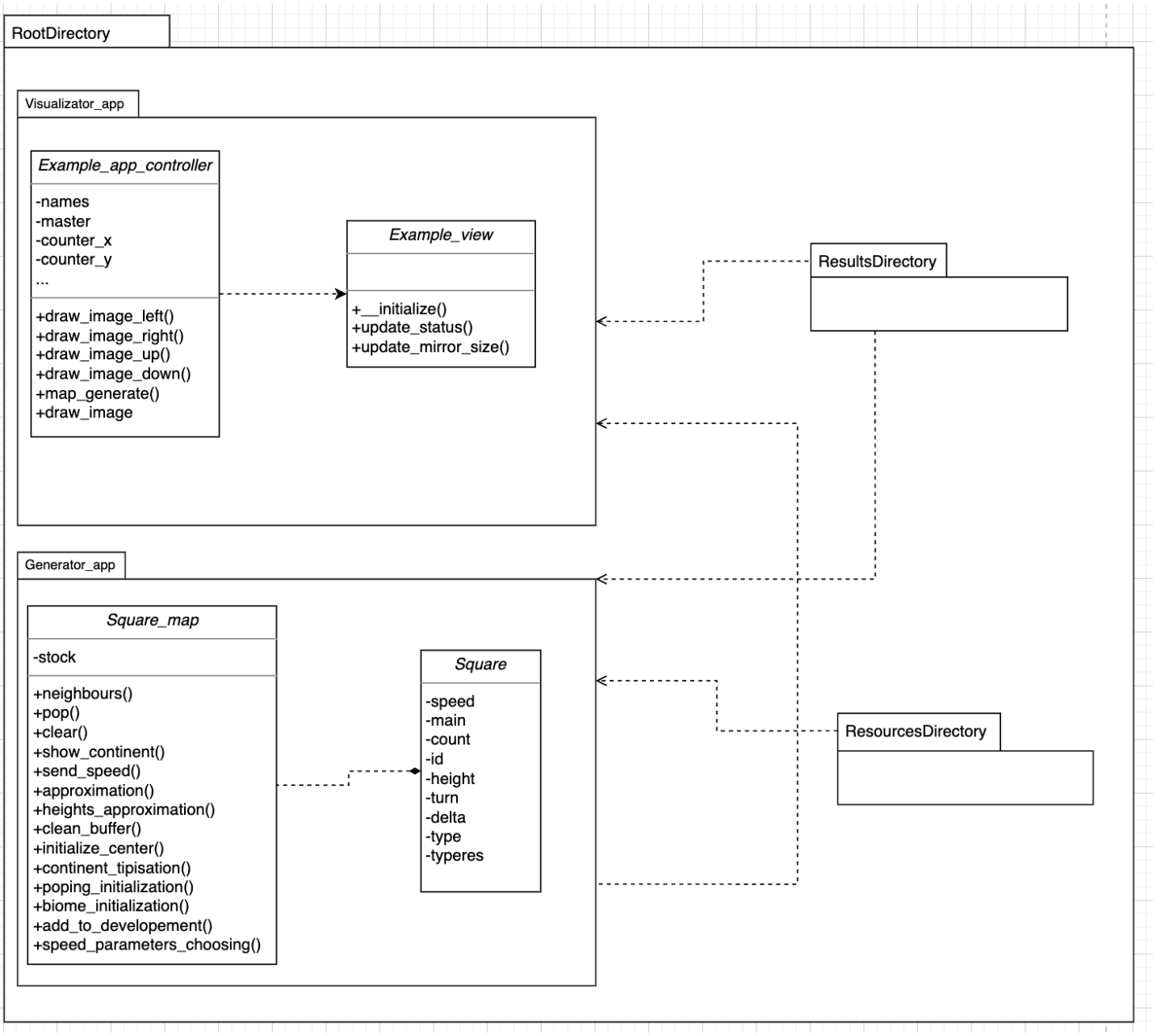
1.4. В текущем состоянии база данных представляет из себя набор элементов в корневой директории, что значит, что базы данных как таковой нет (либо мы называем её супер упрощённой).

2. Общее описание предложения по доработке системы с архитектурной точки зрения

Текущее логическое представление:



Ожидаемое логическое представление:



2.1. Разбиение на два отдельно работающих модуля.

Предлагается разбить main.py на два разных файла: Visualizator.py и Generator.py, каждый из которых будет способен работать независимо. И взаимодействовать друг с другом, также - независимо (Подгрузка карт в Visualizator будет осуществляться через вспомогательные файлы, сохраняемые генератором в results, генерация - через вызов скрипта Generator, с передачей туда требуемых параметров).

2.2. Разделение логики и интерфейса

Предлагается разделить ExampleApp на два класса: Example_View (представление карты), и на Example_App_Controller (контроллер, отвечающий за логику и работу с интерфейсом), тем самым проведя декомпозицию класса.

В данный момент в классе ExampleApp находятся такие методы, как: _draw_image_left, _draw_image_right, _draw_image_up, _draw_image_down, __init__, _map_generate, _draw_image. Планируется все эти методы вынести в Example_App_controller, оставив под Example_View только инициализацию окна, и вызов методов Controller при соответствующей интеракции с пользователем.

2.3. Оптимизация класса ExampleApp

В пользовательском интерфейсе приложения есть хардкод, идея улучшения состоит в том, чтобы избавиться от него. Например, вместо использования ручного задания размеров экрана, будет создаваться отдельный интерфейс, который будет отвечать за получение разрешения экрана и вывод картинок на канвас.

Абстракция размеров экрана позволит также сделать проект более гибким и масштабируемым, что увеличит его функциональные возможности и повысит удобство использования.

2.4. Декомпозиция базы данных

Предлагается разделить базу на две части: база, которая хранит в себе сохраненные карты; и база, которая хранит спрайты (изображения) клеток, для последующей генерации карты.

Таким образом база данных приобретает декомпозицию, что облегчает изучение архитектуры и структурирует данные.

Кроме того, планируется вынесение в отдельные методы логики, отвечающей за взаимодействие с базами данных. Тем самым, код станет более читаемым, а методы станут более декомпозированными.

3. Новые технологии и подходы

Будет использоваться шаблон MVC (Model-View-Controller).

4. План реализации

4.1. (2 дня) Переработка архитектуры модуля generator: написание генератора вспомогательных файлов, в дальнейшем используемых для подгрузки в Visualizator.

4.2. (3 дня) Разбиение main.py на два класса: Visualizator.py и Generator.py. Настройка вызовов Generator.py визуализатором, и подгрузка в Vizualizator.py сгенерированных in-time изображений.

4.3. (1 день) Промежуточное тестирование корректности работы нового приложения Visualizator.py в режиме генерации карт.

4.4. (1 день) Настройка загрузки в Vizualizator.py произвольных изображений из папки Resources.

4.5. (1 день) Промежуточное тестирование корректности работы нового приложения Visualizator.py в режиме загрузки карт.

4.6. (3 дня) Разделение класса ExampleApp на Example_view и Example_app_controller. Выделение методов в данных классах в соответствии с их назначением. Выделение классов в отдельный пакет Visualizator_app.

4.7. (1 день) Замена метода, получающего размеры экрана, на готовой метод из библиотеки tkinter.

4.8. (1 день) Создание папки Sprites и SavedMaps, перемещение в первую спрайты клеток, а во вторую – сохраненные карты. Также необходимо внутри исполняемых файлов зафиксировать изменение директорий. Таким образом реализуется декомпозиция, а исполняемые файлы будут корректно обращаться к базе данных (директориям).

4.9. (1 день) Итоговое тестирование приложение для проверки корректности работы.

5. Исполнители и зоны ответственности

Доржиев Донир Саянович (БПИ203) — 4.7, 4.8.

Кондаков Семен Васильевич (БПИ201) — 4.1, 4.2.

Насыхова Анастасия Артемовна (БПИ201) — 4.3, 4.5, 4.9.

Неймышева Юлия Петровна (БПИ201) — 4.4, 4.6.

Совместное выполнение задач предусматривается.

6. Критерии результативности

В ходе разработки предложений по доработке системы мы выявили следующие критерии результативности:

- Сравнение модульности проекта до и после переработки.
- Успешный результат тестирования программы.

7. Комментарии

Дополнительных комментариев нет.