ВВЕДЕНИЕ

Проникновение интернета в каждую сферу жизни стало неоспоримым фактом в современном мире. С каждым днем растет число пользователей, а, следовательно, и количество передаваемой информации.

Одной из главных задач при передаче данных является сохранение их целостности. Этого можно достичь, если передавать информацию по надежному каналу связи. В свою очередь надежность канала связи подразумевает непрерывность сигнала в различных географических условиях. Для того чтобы оценить возможности создания такого канала на какой-либо местности производят расчет радиотрассы.

Расчет радиотрассы включает в себя построение графика высот для выбранной местности, построение линии сигнала с учетом высоты антенн. В результате расчета можно увидеть существует ли возможность беспрепятственного прохождения сигнала на данной местности. При необходимости определения границ области с беспрепятсвенным прохождением сигнала строится область покрытия радиотрассы.

Помимо проведения расчетов радиотрассы, пользователю также необходима возможность визуализации рельефа области, для которой выполняется расчет, с целью получения приблизительной оценки правильности расчетов. Использование 3х-мерной модели рельефа поможет представить пользователю информацию в более наглядной форме.

Для решения задачи по определению возможности организации канала связи между объектами возможна разработка приложения, включающего в себя описанные выше функции. Именно разработке такого приложения посвящен данный дипломный проект.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

На первым этапе работы были рассмотрены алгоритмы расчета радиотрассы и области покрытия радиотрассой. На основании алгоритма были выявлены данные, необходимые для их реализации. Также было сформулировано техническое задание.

* 1. Расчет радиотрассы

Расчет радиотрассы включает в себя построение графика высот для выбранной местности, построение линии сигнала с учетом высоты антенн.

Для того чтобы построить график высот, необходимо выполнить следующие действия:

1) Выбрать 2 точки на карте

Эти точки – координаты объектов, между которыми планируется организовать канал связи

2) Вычислить координаты точек, находящихся между объектами

Для вычисления координат точек, находящихся между объектами, можно воспользоваться уравнением прямой. Вычисленная прямая вместе с объектами называется маршрутом радиотрассы. Пример маршрута радиотрассы представлен на рисунке 1.

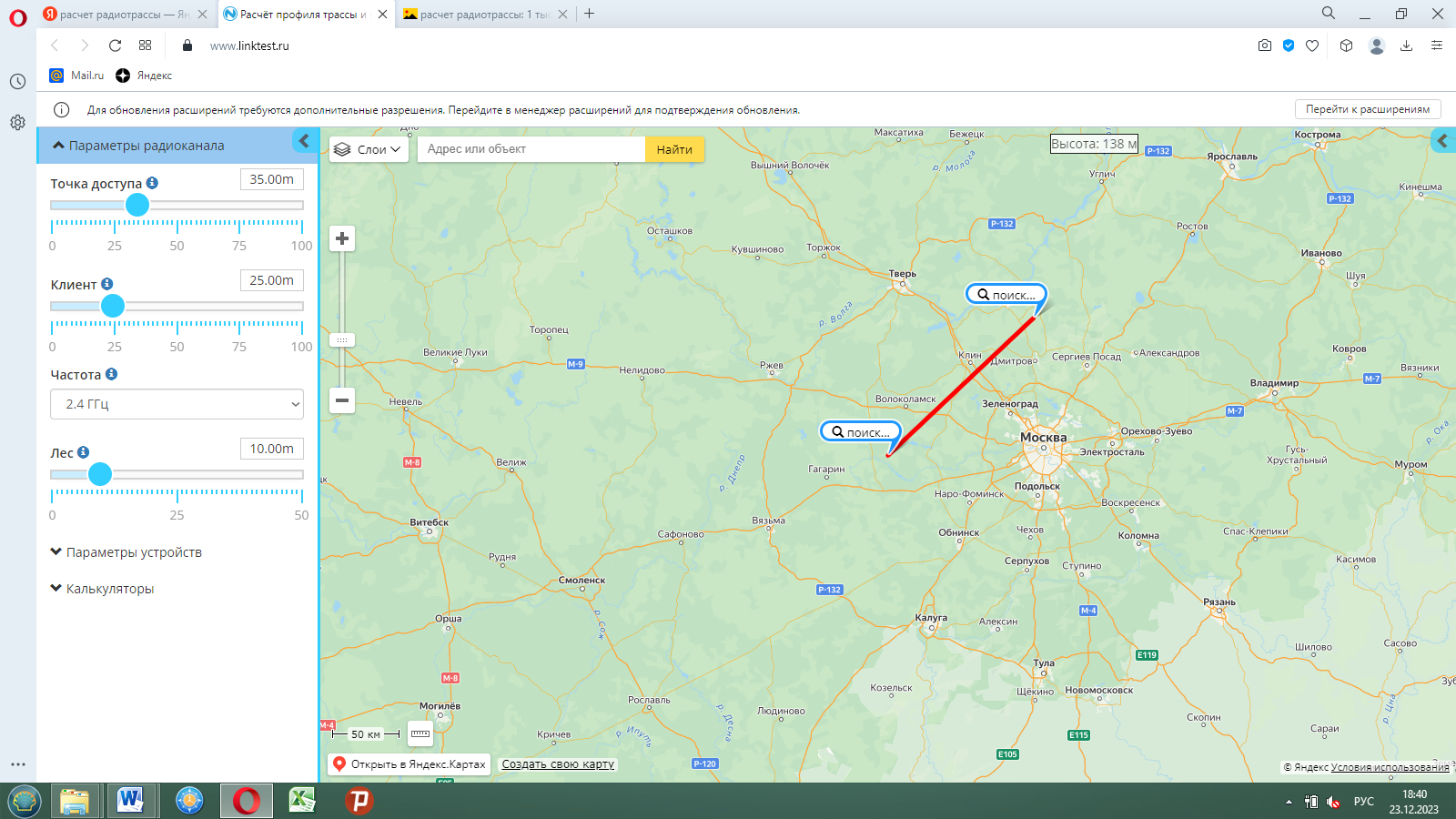


Рисунок 1 – Маршрут радиотрассы

Шаг для вычислений задается из соображений точности, желаемой достичь при расчете радиотрассы.

Если необходимо достичь высокой точности при построении профиля радиотрассы, то берется большое количество точек, находящихся на небольшом расстоянии друг от друга. Следовательно, и шаг должен быть небольшим.

3) Определить высоту для каждой точки маршрута радиотрассы, включая точки объектов

4) Построить профиль радиотрассы

Построение профиля подразумевает построение графика по всем значениям высот, полученным в ходе выполнения предыдущего шага. Стоит отметить, что точки высот наносятся в том порядке, в котором идут координаты точек маршрута трассы.

Дополнительно стоит отметить, что все высоты откладываются относительно нулевой дуги. Нулевая дуга, в свою очередь, рассчитывается по формуле:

Н(Х) = (D²/2R)\*К\*(1-К), км, где

D — расстояние между объектами, км;

R — радиус Земли — 6371 км;

К = Х/D — относительная координата промежуточной точки, 0<K<1;

Х — расстояние от 1-ого объекта до промежуточной точки, км

Для отображения линии связи на графике используются 2 точки: 1 точка – высота 1-ого объекта + высота антенны для 1-ого объекта, 2 точка – высота 2ого объекта + высота антенны для 2-ого объекта. Далее эти 2 точки соединяются линией. Пример расчета радиотрассы представлен на рисунке 2.

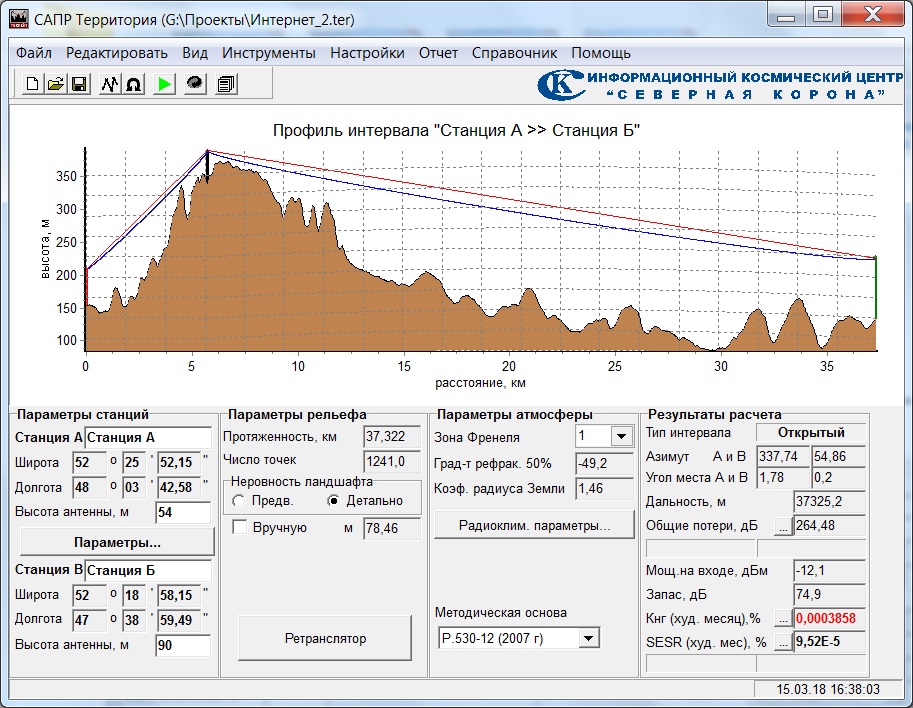


Рисунок 2 – Пример расчета радиотрассы

Пересечение линей связи профиля радиотрассы свидетельствует о невозможности прохождения сигнала. Другими словами, о невозможности установки надежной связи между объектами. Если же линия не пересекает профиль, то связь может быть установлена.

1.2 Расчет области покрытия радиотрассы

Алгоритм расчета области покрытия радиотрассы следующий:

1) выбирается точка – источник сигнала;

2) задается радиус области покрытия;

3) вычисляются точки, лежащие на окружности, образуемой областью покрытия;

4) полученные точки разбивают область видимости на фрагменты. Пример разбиения области видимости на фрагменты представлен на рисунке 1.1.

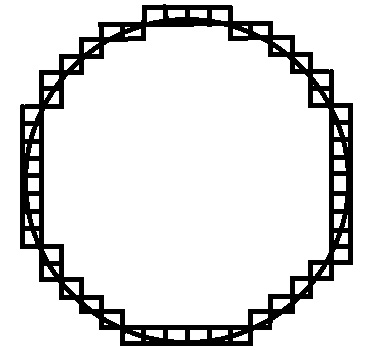


Рисунок 1.1 – Разбиение области видимости на фрагменты

5) для каждой вычисленной точкой и источником сигнала строится радиотрасса;

6) для каждой построенной радиотрассы определяется её тип: открытая, полуоткрытая, полузакрытая, закрытая;

7) каждый фрагмент области видимости закрашивается в цвет, соответствующий типу трассы;

8) уменьшается значение радиуса на заданный шаг;

9) повтор п. 3) – 8) до тех пор, пока значение радиуса не станет равным нулю.

На основании рассмотренного алгоритма для расчета области покрытия радиотрассы был сделан вывод, что данная задача сводится к многократному решению задачи о построении радиотрассы между 2-мя точками. Следовательно, для ее реализации необходимо наличие следующих данных:

1) Координаты 2х точек – источника и приемника;

2) Массив точек, принадлежащих маршруту радиотрассы;

3) Массив тайлов для определения значения высот для каждой точки маршрута;

4) Массив высот, описывающих профиль рельефа;

5) Массив высот для линии сигнала;

6) Массив значений для первой зоны Френеля.

7) Массив тайлов и массив высот также необходим и для выполнения 3D визуализации рельефа местности.

1.3 Техническое задание

В результате выполнения анализа предметной области было принято решение о разработке приложения для расчета радиотрассы, области покрытия радиотрассы, а также 3д визуализации. Рассмотрение алгоритмов поспособствововало тому, чтобы понять какие данные нужны. Для получения данных высот можно использовать тайлы с сервера карт. В этих тайлах в значениях цвета пикселя будет закодировано значение высоты.

(про получение данных высот с тайла сказать еще и в тз)

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

На данном этапе работ мы определяем схему взаимодействия сервера с приложением. Создана карта высот для правильной работы приложения. Спроектировано приложения с учетом предъявляемых к нему требований.

2.1 Взаимодействие с сервером карт

В результате выполнения первичного анализа сервера карт было выявлено, что сервер карт представляет собой сервер, на котором хранятся карты. Для осуществления взаимодействия с сервером карт существует библиотека с открытым исходным кодом, в которой описаны принципы организации работы с ним.

Для упрощения последующих объяснений введем следующую терминологию: под сервером будет подразумеваться непосредственно сам сервер карт, под клиентом или клиентской частью подразумевается библиотека с открытым исходным кодом, а под сервером карт следует понимать сервер и клиент вместе.

Более детальный анализ сервера карт позволил выявить схему взаимодействия с ним. Схема взаимодействия с сервером карт представлена на рисунке 3.

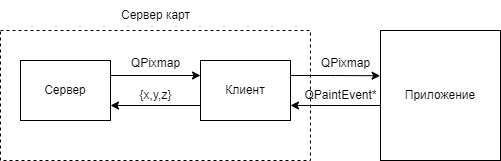


Рисунок –

При определенных взаимодействиях с виджетом: прокрутка колеса мыши, удержание ПКМ и перемещение курсора, возникает событие об изменении содержимого виджета. Параметры для этих событий содержатся внутри класса QPaintEvent, экземпляр которого передается приложением клиенту. То есть в случае возникновения какого-то из событий возникает запрос к клиенту, информирующий о необходимости получения тайлов.

В свою очередь клиент вычисляет координаты тайлов, которые необходимо запросить и делает запрос на сервер.

Формирование запроса от приложения при возникновении таких событий, как прокрутка колеса мыши, удержание ПКМ и перемещение курсора, нам не подходит. Нам нужно, чтобы запрос был сформирован после того, как рассчитан маршрут радиотрассы, то есть после того, как выбраны 2 точки объектов. Следовательно, стоит переписать формирование запроса, но для того, чтобы его переписать надо понимать всю последовательность действий для его формирования.

Данный подход не подходит, поэтому необходимо изменить схему взаимодействия с сервером карт, изменив модуль клиента под себя и вынеся его в отдельный модуль в приложении.

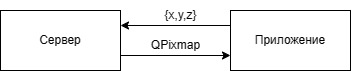


Рисунок – Новая схема взаимодействия

2.2 Создание карты

На данном этапе работы будет выполнен обзор аналогов. На основании этого обзора будет принято решение об используемом формате данных, о необходимости использования сторонних приложений или программной реализации генерации карты.

## 2.2.1 Выбор формата данных

В качестве данных для цифровой модели рельефа можно выделить следующие доступные:

- SRTM

- AsterGDEM

- Etopo2

1) SRTM

Наиболее распространенный формат, используется во многих программах.

В качестве формата для записи высот используются файлы с расширением hgt. Используемое здесь слово «HGT» - это просто сокращение от «высота». Каждый файл представляет собой последовательность 16-разрядных целых чисел, задающих высоту каждой ячейки в метрах, расположенных с запада на восток, а затем с севера на юг. Каждый фрагмент данных продолжительностью 3 угловых секунды содержит 1442401 целое число, представляющее сетку размером 1201 ×1201. Формат доработан для разных систем координат.

В старых версиях данные есть только для области между 60 ° с.ш. и 54 ° ю.ш. С 2014 года данные есть для всей области карты.

Обладает высокой точностью за счет того, что съемка производилась при помощи радиолокации.

2) AsterGDEM

Менее распространенный формат, используется в малом числе программ.

ASTER GDEM охватывает поверхность суши между 83° с.ш. и 83° ю.ш. и насчитывает 22,600 фрагментов размером 1°х1°. Распространяется в формате GeoTIFF в географической системе координат (широта/долгота) и разрешением 1 угловая секунда (примерно 30 метров). Система координат данных WGS84/EGM96.

Точность меньше, чем у srtm, так для использовалась стереофотограмметрическая съемка. И при съемке лесной местности ASTER получает отражение солнечного излучения от кроны дерева, которое затем фотограмметрически обрабатывается для получения ЦМР.

3) Etopo2

Самый нераспространенный формат, потому что является самым неточным из представленных ранее. Его неточность связана с ограничением аппаратных и программных возможностей на момент произведения съемок – 1978г. В последующем доработки не производились.

Результаты сравнения всех рассмотренных данных цифровой модели рельефа сведены в таблицу 1.

Таблица 1 – Результаты сравнения данных цифровой модели рельефа

|  |  |  |  |
| --- | --- | --- | --- |
|  | SRTM | AsterGDEM | Etopo2 |
| Формат данных | Hgt | GeoTIFF | GeoTIFF |
| Система координат | Разные | WGS84/EGM96 | WGS84 |
| Точность | Высокая | Хорошая | Средняя |
| Доступность | 85°с.ш. и 85°ю.ш | 83°с.ш. и 83°ю.ш. | 60°с.ш. и 54°ю.ш |

В результате сравнения было принято решение об использовании SRTM в качестве данных цифровой модели рельефа. После того, как был выбран формат данных, выполняется генерация карты.

## 2.2.2 Генерация карты

В разделе, посвященному анализу работы сервера карт, было сказано, что графический интерфейс предоставляет возможность просмотра позиции тайла. Это возможно благодаря тому, что вся карта разбита на тайлы. Пример разбиения карты на тайлы представлен на рисунке 4.

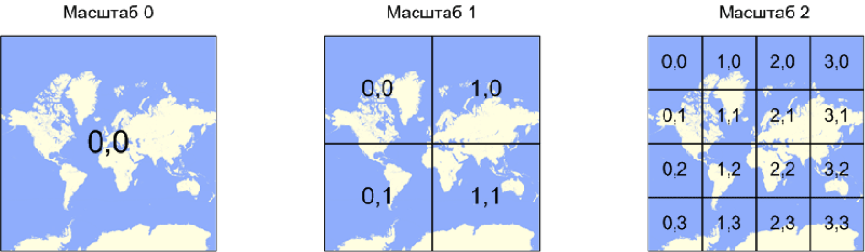


Рисунок 4 – Пример разбиения карты на тайлы

Следовательно, под генерацией карты для использования ее на сервере карт необходимо понимать создание тайлов карты.

### 2.2.2.1 Выбор масштаба

Перед тем, как приступить к генерации тайлов необходимо понимать, для какого или для каких масштабов необходимо генерировать тайлы.

Учитывая, что работа будет происходить с использованием фалов данных SRTM, то можно генерировать тайл для каждого значения высоты, то есть для каждых 3” создавать новый тайл. Это возможно выполнить, если масштаб будет равен 17. Достоинством использования такого уровня увеличения является высокая точность при декодировании. Недостатками такого увеличения является большое число тайлов (30 млрд), что требует большого объема памяти для хранения этих данных, а также большого времени для их генерации.

Кроме того, использование увеличение равного 17, довольно избыточно. Так как в результате будет генерироваться изображение 256х256, закрашенное только в один цвет и покрывающее область в 92 м. Пример сгенерированных тайлов представлен на рисунке 5.



Рисунок 5 – Пример сгенерированных тайлов

Можно уменьшить избыточность тайлов, если генерировать тайлы в которых 92м будут приходится на 1 пиксель. То есть каждое значение высоты будет помещаться в 1 пиксель. Это возможно для увеличения равного 11. При таком масштабе возможен компромисс между количеством сгенерированных тайлов и точностью последующего декодирования.

Последующее уменьшение масштаба приведет к снижению точности декодирования. Так как при уменьшении масштаба происходит сжатие изображения и это приведет к «объединению» нескольких соседних пикселей в один.

Так как для выполнения поставленной задачи выводить карту высот для работы пользователю необязательно, а для определения высоты достаточно только увеличения 11. Следовательно, можно сгенерировать тайлы только для увеличения равного 11.

### 2.2.2.2 Генерация тайлов

Так как разбиение карты на тайлы является довольно стандартным решением для работы с картами, имеющими высокий уровень увеличения, то существуют приложения, в которых эта функция уже реализована. Как правило, такие приложения уже содержат набор готовых карт. Следовательно, среди этого набора может оказаться и карта высот.

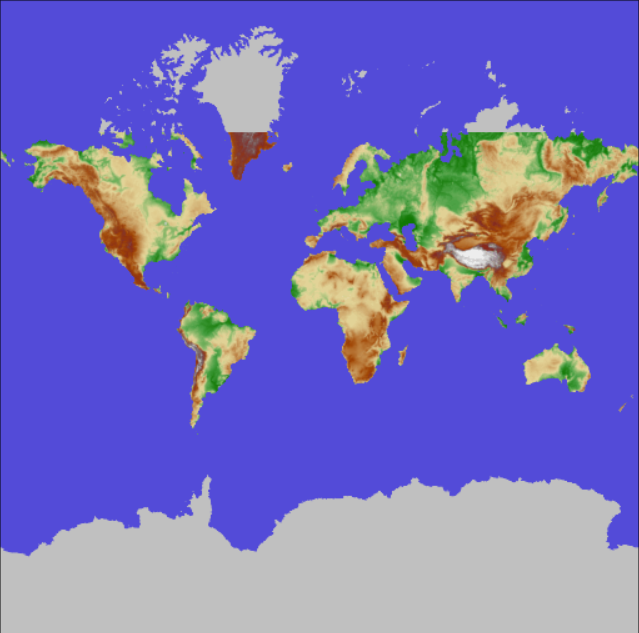
В результате поиска были обнаружены 2 приложения, отвечающие заявленным требованиям (возможность разбиения на тайлы, наличие карты высот): SasPlanet, QGis. Стоит отметить, что оба эти приложения в качестве данных для карты высот используют SRTM.

Далее был выполнен анализ карт высот, которые доступны в приложении. Как итог, карта из приложения SasPlanet не подходит для решения поставленной задачи по 2-м причинам:

1) карта является не полной, так как использует данные от первой версии SRTM.

2) нет сведений о соотношениях между цветом и высотой, а также о методах кодирования высоты цветом.

На рисунке 6 представлена карта, доступная в SasPlanet.

  
Рисунок 6 - Карта, доступная в SasPlanet

Что касается карты высот, доступной в приложении QGis, то она имеет данные для всей области карты, а также позволяет задавать значения цвета для высот и метод кодирования. Окно выбора цвета для высот и метода кодирования представлены на рисунке 7.

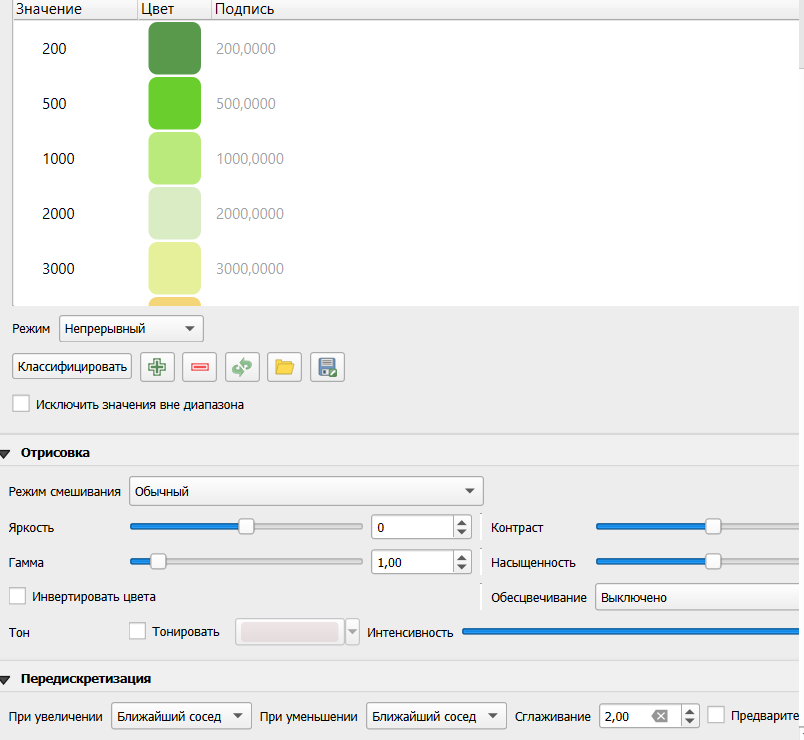


Рисунок 7 – Окно выбора цвета для высот и метода кодирования

В качестве метода кодирования был использован метод ближайшего соседа.

После того, как заданы цвета для высот и выбран метод кодирования, происходит автоматическое раскрашивание карты. Результат приведен на рисунке 8.

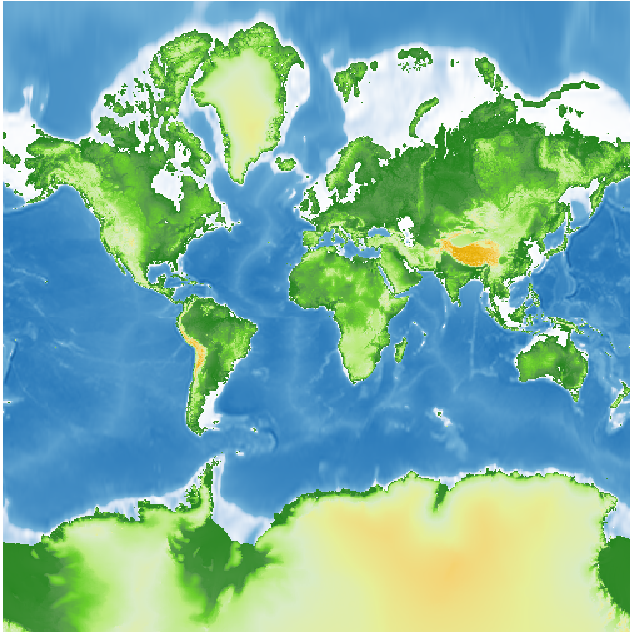


Рисунок 8 – Внешний вид карты после раскраски

После того, как карта готова, можно приступать к разбиению её на тайлы. Для этого, как уже было сказано ранее, в приложении уже есть функция. При выполнении этой функции начинается генерация 4 194 304 тайла. Каждый тайл генерируется, примерно, 1 секунду. Следовательно, для полного выполнения генерации потребуется 48 дней безостановочной работы. Это довольно много.

Можно попробовать уменьшить количество времени для генерации тайлов, если убрать из генерации тайлы, относящиеся к водному пространству, а потом учесть это при разработке приложения.

К сожалению, QGIS не предоставляет возможность генерации тайлов для некоторой области доступной карты, поэтому необходимо искать альтернативные подходы для решения задачи.

Можно попробовать генерировать тайлы программно. То есть написать скрипт, который бы читал данные из hgt файла, кодировал бы их цветом, вычислял по координатам позицию тайла и пикселя и закрашивал бы этот пиксель в полученный цвет.

Предположим, что обработку каждого пикселя потребуется 1 секунда. Так как тайл имеет размер 256х256, то для полной его генерации потребуется 18 часов. Даже если сделать параллельную обработку 6 тайлов на 6 процессорах, то все равно очень долго. Этот вариант не подходит.

Хотя в QGIS нет возможности сгенерировать тайлы для некоторой области, он позволяет выполнять разбиение на тайлы слоя, который можно создать из файла, содержащих данные высот в допустимом формате. Одним из таких допустимых форматов является hgt файл.

Добавить все имеющиеся файлы высот в QGIS в качестве новых слоев для последующего создания тайлов для каждого из них не предоставляется возможным из-за большого объема этих файлов – 70Гб. При попытке загрузить возникает зависание программы QGIS с последующим завершением работы программы.

Если не получается добавлять все разом, можно добавлять последовательно каждый файл, удаляя предыдущий. Но тут возникает ошибка при создании тайлов: файлы содержат информация для области 1°х1°, а тайлы для масштаба 11 захватывает область размером больше, чем 1°х1°. Для решения этой проблемы достаточно добавлять дополнительные файлы, находящие вокруг необходимого файла. Пример представлен на рисунке 9.

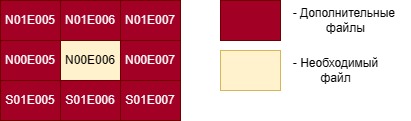


Рисунок 9 – Приме добавления дополнительных файлов

Стоит отметить, что в этом способе генерации карты используется кодирование по методу ближайшего соседа.

Так как добавление, удаление файлов, выбор слоя с последующим разбиением его на тайлы это рутинные задачи, их можно автоматизировать, написав скрипт на python, который будет выполнять все эти функции. Листинг кода этого скрипта приведен в приложении А.

При таком методе генерации тайла, каждый файл обрабатывается 2 мин. Но из-за того, что таких файлов всего 26 тысяч, то для полной генерации карты потребуется 36 суток безостановочной работы.

Это всё-таки меньше, чем при генерации тайлов для всей области карты. Следовательно, это решение нас устраивает.

## 2.2.3 Переименование файлов

После того, как все тайлы сгенерированы их необходимо переименовать и переместить в подкаталоги, которые необходимо создать. Это нужно сделать, потому что сервер карт может создавать карту из тайлов только в том случае, если эти тайлы находятся в нужных директориях, иначе он их не сможет обнаружить.

Для примера рассмотрена директория некоторого файла, созданная в QGIS.

…/656/1114.png

656 – координата тайла по x

1114 – координата тайла по y

Этот же файл, но допустимый для сервера карт будет иметь следующую запись:

…/0/x656/1/y1114.png

0 – координата тайла по х деленная на 1024

x656 - координата тайла по х

1 - координата тайла по y деленная на 1024

y1114 - координата тайла по y

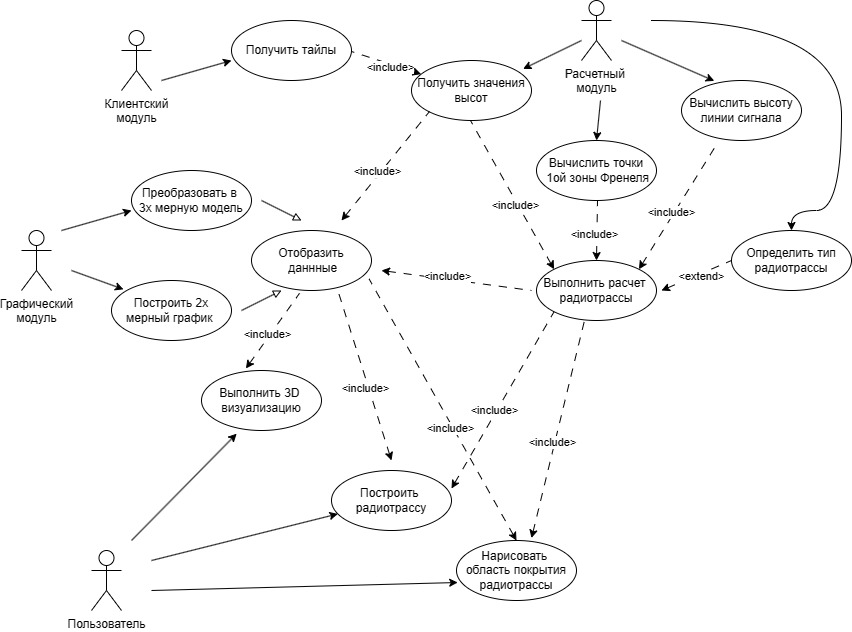
Создание папок, переименование файлов и перенесение их в соответствующие папки – это довольно рутинная задача, которая отчасти реализована в total Commander, поэтому можно прибегнуть к написанию скрипта на python. Листинг кода приведен в приложении Б.

2.3 Проектирование приложения

Проектирование приложение происходило в несколько этапов. Первоначально была разработана структура приложения. Далее был рассмотрен процесс работы приложения с целью… Далее была разработана диаграмма потоков данных для выстраивания правильной последовательности действий. Разработаны алгоритмы функционирования

2.3.1 Разработка структуры приложения

Для демонстрации структуры приложения была разработана диаграмма прецедентов, представленная на рисунке



Рисунок

Как видно из рисунка, пользователь имеет возможность выполнить несколько действий: построить радиотрассу, отобразить область покрытия радиотрассы и выполнить 3д визуализацию рельефа местности, для которой выполнялись расчеты.

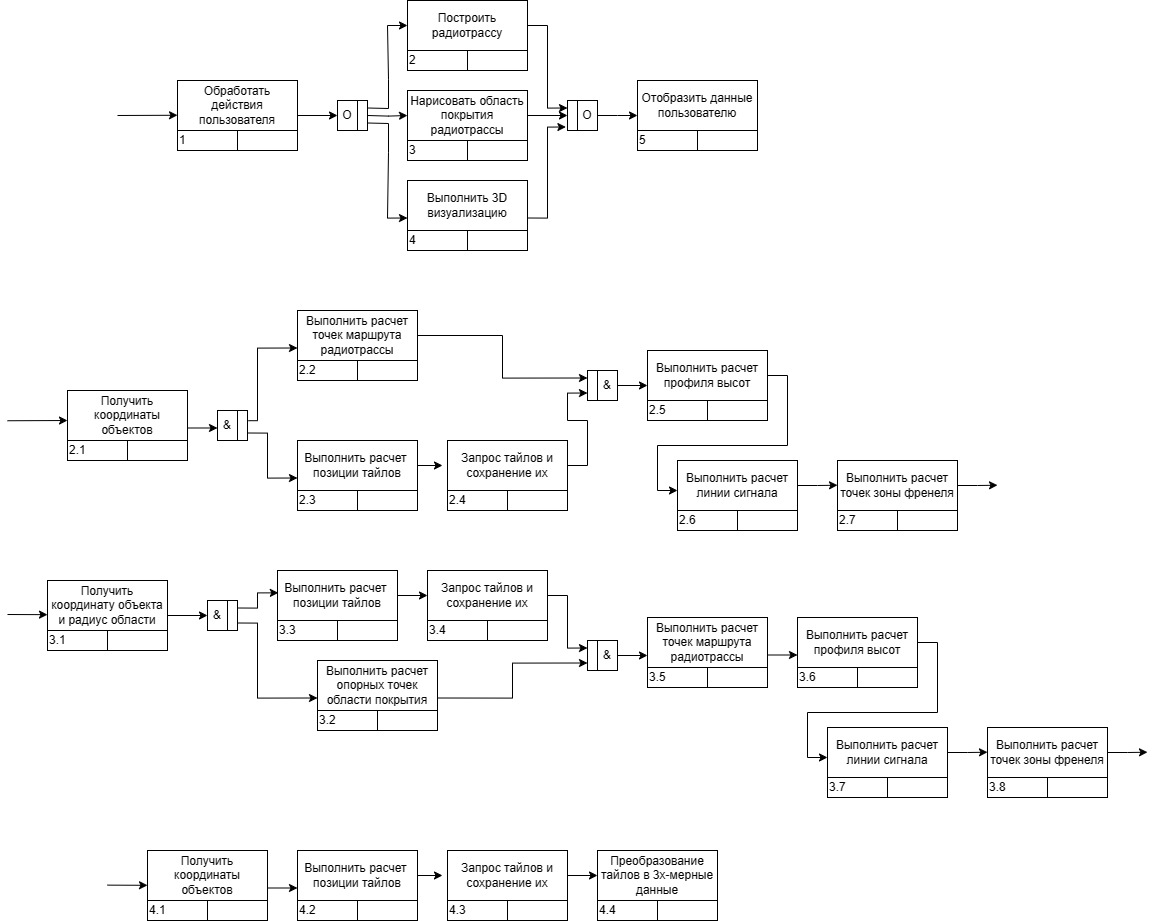
Выполнение 3д визуализации включает в себя отображение 3х мерных данных высот, полученных в результате обработки расчетным модулем тайлов, полученных клиентским модулем.

Построение радиотрассы состоит из расчета радиотрассы и отображения полученных результатов в графическом виде. Отображением результатов занимается графический модуль. Расчет радиотрассы состоит из получения значения высот для профиля радиотрассы, вычисление точек зоны Френеля и определением высоты линии сигнала. Всем этим занимается расчетный модуль. Данные для вычисления профиля высот расчетный модуль получает из тайлов, которые были получены клиентским модулем.

Рисование области покрытия радиотрассы включает в себя многократное построение радиотрассы с дополнительным расчетом в виде определения типа радиотрассы. Но всеми этими расчетами занимается расчетный модуль. А также многократное закрашивание фрагмента области в цвет согласно вычисленному типу. Этим занимается графический модуль.

2.3.2 Описание процесса работы приложения

Для описания процесса работы приложения была построена диаграмма idf3, представленная на рисунке



Рисунок

Как видно из рисунка пользователь может выбрать только одно из действий: выполнить построение радиотрассы или 3д визуализации, нарисовать область покрытия радиотрассы. После того, как все необходимые расчеты будут выполнены, полученные результаты функций будет выведены пользователю на экран в некотором виде.

При построении радиотрассы первоначально происходит получение координат объектов, между которыми необходимо организовать канал связи, от пользователя. Далее при помощи этих координат происходит вычисление позиции тайлов с последующим запросом и координат точек маршрута радиотрасы. Эти события являются независимыми, но оба они должны быть выполнены. Расчет профиля высот возможен только в том случае, если до этого был выполнен расчет точек маршрута и получены тайлы. Следовательно, эта функция возможна к выполнению только тогда, когда предыдущие 2 действия выполнились. Расчет линии сигнала связи возможен только после расчета профиля высот, так как для расчета использует данные о высоте рельефа для начальной и конечной точки маршрута. Расчет точек зоны Френеля использует данные высот линии сигнала, поэтому выполняться должен строго после расчеты линии сигнала связи.

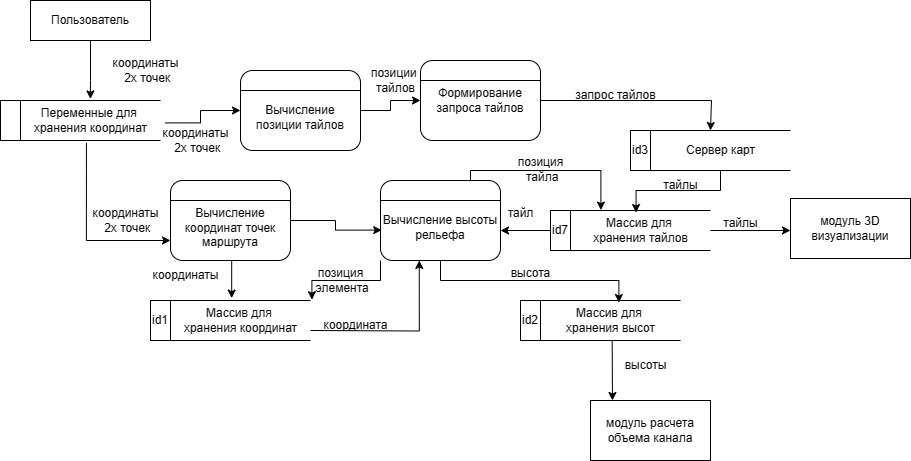
При рисовании области покрытия радиотрассы необходимо задать координаты центральной точки и радиус области, для которой будет строиться зона покрытия. После этого происходит вычисление позиции тайлов и их запрос, координат опорных точек, принадлежащих области покрытия для их последующего использования при расчете радиотрассы. Последовательность расчета радиотрассы не изменяется. Добавляется определение типа радиотрассы

Для 3D визуализациии также первоначально необходимо получить координаты объектов или границы области, выполнить расчет позиции тайлов и их запрос. И после этого выполняется преобразование тайлов в 3х мерные данные при помощи специальных алгоритмов.

Диаграмма прецедентов для того, чтобы просто показать какие модули есть в приложении и что они делают.

Далее диаграмма последовательностей для того, чтобы определить какая последовательность выполняется для каждого алгоритма. Какие действия должны наступать вместе, какие можно запускать параллельно при необходимости. Это диаграмма позволяет спроектировать диаграмму потоков данных. То есть понять, что необходимы хранилища для данных, так как они могут использоваться повторно. Кроме того, выстроить логически алгоритм запроса тайлов. Так как выполняется паралелльно, то нет необходимости для каждой точки. Нам нужно, чтобы они просто вместе были для дальнейшего расчета. А значит можно сначала запросить тайлы, а потом выполнить расчет координат точек. И тем самым мы сможем уйти от высокой степени связности модулей.

2.3.3 Циркуляция потоков данных



На рисунке показано, что вычисление позиции тайлов, используемых при расчетах радиотрассы, происходит сразу после того, как пользователь ввел координаты 2х точек. Далее выполняется обращение к серверу карт для запроса всех тайлов. После их получения, они сохраняются в массив.

Когда все тайлы получены и сохранены, выполняется расчет профиля высот. Первоначально происходит вычисление точек маршрута радиотрассы, при расчете которых используются данные 2х точек, которые изначально ввел пользователь. Полученные значения координат точек маршрута записываются в массив. Далее идет вычисление высоты рельефа для каждой из точек. Для этого для каждой точки определяется позиция тайла, выполняется поиск в массиве тайлов элемента с такими значениями, определяется координаты пикселя внутри тайла и его цвет, полученное значение цвета преобразовывается в значение высоты и записывается в массив высот.

Циркуляция потоков данных будет рассмотрена только для запроса тайлов, потому что для остального она не сильно нужна.

2.3.4 Разработка алгоритмов расчета

2.3.4.1 Расчет позиции тайла

2

2.3.4.2 Расчет опорных точек

Расчет опорных точек должен производится на основании двух параметров: координаты точки источника сигнала и радиуса области покрытия радиотрассы. Дополнительным параметром можно использовать угол поворота радиуса относительно центральной точки. Других параметров нет.

Можно предположить, что расчет опорных точек можно выполнить при помощи следующих формул:

где

– координаты центра области покрытия,

– радиус области покрытия,

– угол поворота радиуса относительно центра

Но данная формула применима только для плоскости, потому что расчет выполняется по длине отрезка, а не по длине дуги, как это необходимо при работе с географическими координатами. В результате неправильного расчета область видимости приобретает форму овала, что является неверным. На рисунке 3.2 изображена граница области видимости при использовании формулы для плоскости.

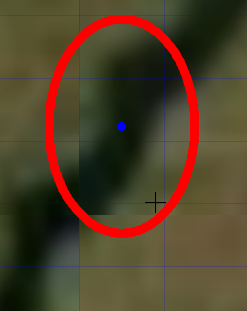


Рисунок 3.2 – Границы области видимости при использовании формулы для плоскости

Как уже было отмечено ранее, правильный расчет возможен в том случае, если он выполняется по длине дуги. Следовательно, необходимо найти формулу для расчета точки через длину дуги.

Эту формулу можно получить из сферической теоремы косинусов. Так как сферическая теорема косинусов позволяет найти длину дуги между точками, находящимися на сфере, то для получения точки через длину дуги, необходимо решить обратную задачу, воспользовавшись следствием из этой теоремы. Формула для расчета координаты точки следующая:

φ₂ = arcsin(sin(φ₁) cos(d/R) + cos(φ₁) sin(d/R) cos(α))

λ₂ = λ₁ + atan2(sin(α) sin(d/R) cos(φ₁), cos(d/R) - sin(φ₁) sin(φ₂)), где

φ₁, λ₁ - широта и долгота источника сигнала,

φ₂, λ₂ - широта и долгота приемника сигнала,

α - направление от точки A до точки B (азимут),

R - радиус сферы,

d - длина отрезка по ортодомии.

Граница области видимости при использовании формулы для сферической поверхности представлена на рисунке 3.3.

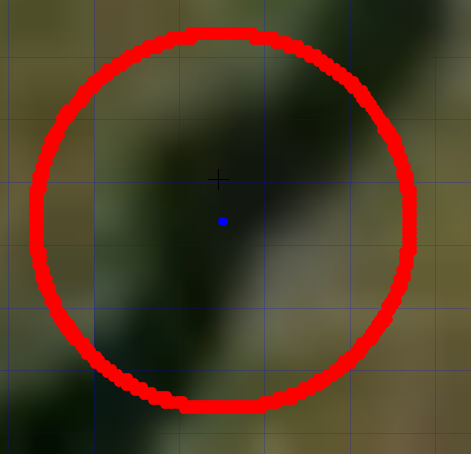


Рисунок 3.3 - Граница области видимости при использовании формулы для сферической поверхности

2.3.4 Определение типа радиотрассы

Cуществует 4 типа трассы: открытая, полуоткрытая, полузакрытая и закрытая. Тип определяется согласно тому, как профиль высот пересекает область сигнала вместе с зоной Френеля:

1) если профиль выше верхней границы зоны Френеля – закрытая;

2) если профиль выше линии сигнала, но не пересекает верхнюю границу зоны Френеля – полузакрытая;

3) если профиль ниже линии сигнала и не пересекает нижнюю границу зоны Френеля – полуоткрытая;

4) если профиль ниже верхней границы зоны Френеля – открытая;

Весь процесс определения типа изображен на схеме алгоритма, представленной на рисунке 3.4

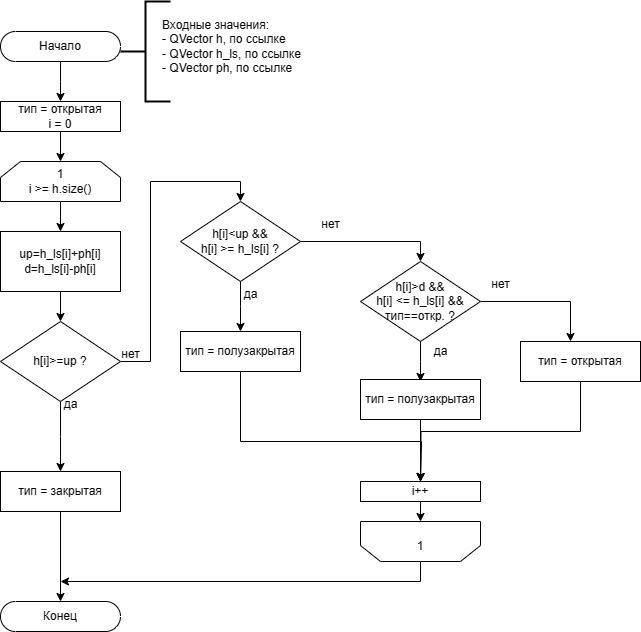


Рисунок 3.4 – Схема алгоритма определения типа трассы

В качестве входных параметров передаются массивы профиля высот, высот линии сигнала и точек зоны Френеля. Далее происходит определение нижней и верхней границы зоны Френеля для каждой точки профиля высот. Для определения верхней границы производится сложение значение высоты линии сигнала со значением расстояния зоны Френеля в этой точке, для нижней – вычитание. Далее происходит сравнение высоты профиля и верхней границы зоны Френеля. Если профиль отказывается выше, то трассе присваивается закрытый тип и алгоритм заканчивает свою работу. Если же нет, не пересекает, то выполняется следующая проверка. И так до тех пор, пока тип трассы не станет закрытым или пока не будут рассмотрен весь профиль высот.

2.3.5 Декодирование цвета в высоту

Что касается декодирование высоты по значению цвета, то тут необходимо учитывать, что для кодирования по методу ближайшего соседа. Этот метод подразумевает, что выбираются несколько значений высот, которые

будут являться границами. Для каждого такого значения задается цвет. Далее считывается значение высоты из файла. Для того, чтобы вычислить значение цвета, соответствующей этой высоте, необходимо выполнить следующие действия:

1) определить между какими заданными границами высотами располагается полученная высота

2) вычислить коэффициент «отдаленности»:

p = (h-h1)/(h2-h1), где

h – полученная высота;

h1–нижняя граница интервала высот;  
 h2 – верхняя граница интервала высот;

3) вычислить значения цвета

r = (1-p)\*r1+p\*r2

g = (1-p)\*g1+p\*g2

b = (1-p)\*b1+p\*b2

r2, g2, b2 – цвет для верхней границы интервала высот

r1, g1, b1 – цвет для нижней границы интервала высот

Следовательно, для декодирования нужно выполнить последовательность действий в обратном порядке.

Программная реализация обоих алгоритмов не будет приведена в ПЗ из-за простоты алгоритмов.

Описать почему другие говно

3. Программная реализация

3.1 Клиентская часть

Под детализацией структуры клиента следует понимать рассмотрение последовательности формирования запроса и получения ответа. Для наглядности была разработана диаграмма последовательностей, представленная на рисунке 10.

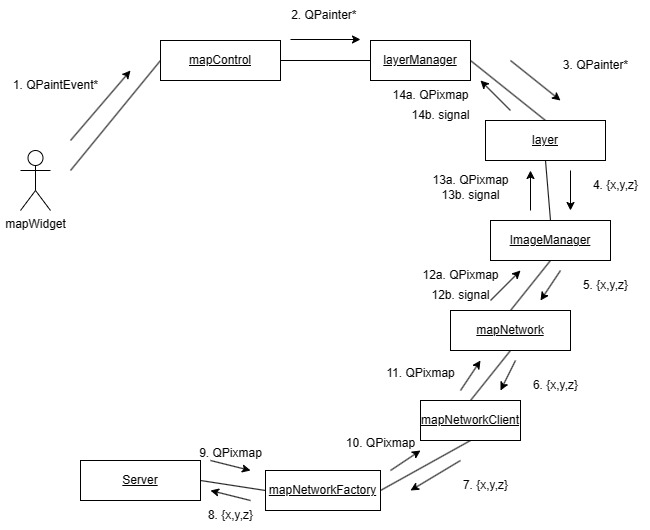


Рисунок 10 – Диаграмма последовательностей

При первом запросе от приложения мы настраиваем взаимодействие с клиентом. Делается это при помощи передачи ссылки на экземпляр класса QPaintEvent, который содержит в себе параметры для событий рисования и принадлежит приложению.

В свою очередь события рисования отправляются виджетам, которым необходимо сделать перерисовку содержимого. Например, при передвижении карты.

mapControl связывает класс QWidget с экземпляром класса QPaintEvent. Это нужно для того, чтобы любые изменения на виджете, сразу обрабатывалась QPaintEvent. Но так как основная задача – отрисовка карты, то mapControl передает управление LayerManager, передав в качестве параметра ссылку на объект класса QPainter, занимающийся отрисовкой объектов. И который необходим для отрисовки области карты

layerManager необходим, потому что карта может представлять собой несколько слоев. Например, гибридная карта создает при помощи наложения osm-карты, где обозначены границы государств и подписаны названия городов, на спутниковую карту. И для правильной отрисовки такой карты просто необходим layerManager.

При принятии управления layerManager передает управление Layer, делая это для каждого слоя. Передавая в качестве параметров ссылку на еще один экземпляр класса QPainter, который необходим для отрисовки одного изображения по верх другого в случае многослойности карты, и координаты центра карты.

Кроме того, layerManager соединяет все тайлы, создавая тем самым область карты и отрисовывает ее поверх виджета при помощи Qpainter, который был передан от mapControl.

При получении управления layer вычисляет значения позиции тайла через смещения.

Далее для всех вычисленных позиций идет проверка наличия тайлов в кэше. Если есть то, тайл извлекается из кэша, отрисовывается и отсылается. Если же нет, то происходит передача управления ImageManager.

В свою очередь imageManager делает запрос на сервер, указывая в качестве параметров позицию тайла и увеличение.

При получении ответа от сервера получаем изображение вместе с его координатами, сохраняем его, а потом перекидываем сигнал, что изображение загружено. Его ловит layermanager, который занимается тем, что запускает метод перерисовки, тем самым запрашивая следующий тайл.

И так делается до тех пор, пока количество тайлов в очереди к серверу не станет равным 0. Когда очередь тайлов пуста посылается сигнал об окончании. Его снова ловит layermanager. Процесс запроса тайлов приостанавливается до следующего события о перерисовке от виджета.

В результате выполнения детализации клиентской части сервера карт стало очевидно, что существующая клиентская часть подходит для задачи отображения области карты для предоставления пользователю возможности выбора точек, где будут располагаться объекты, но не походит для задачи получения высоты для координаты, так как не имеет алгоритма для преобразования координаты в позицию тайла, а также не имеет возможности сохранения позиции тайла для последующей фильтрации изображений, полученных от сервера. Следовательно, необходимо переписать некоторые части клиентской части для того, чтобы они могли решать поставленную задачу.

Для начала в классе ImageManager необходимо добавить метод, который бы реализовывал сохранение позиций запрошенных тайлов для последующего выделения нужного тайла путем сравнения сохраненных значений координат тайла с теми, которые пришли вместе с изображением. Кроме того, необходимо добавить метод, реализующий обработку пустых изображений. О необходимости добавления этого метода было рассказано в главе, посвященной генерации карты. Для того, чтобы сделать добавление этих

методов необходимо создать класс, являющийся наследником от ImageManager, в котором и реализовать эти методы.

Далее необходимо изменить алгоритм определения позиции тайла в классе layer, чтобы он мог работать с географическими координатами.

Алгоритм преобразования координат в позицию тайла заключается в преобразовании географических координат сначала в меркаторские, а затем в пиксели.

Преобразование в меркаторские осуществляется по формуле. Формула зависит от вида проекции. Вид проекции в свою очередь зависит от того, как представляют Землю: в виде сферы, в виде эллипса или в виде геоида. Существует еще одна проекция, в которой Землю представляют в виде сферы, но в которой упрощена формула расчета – web Mercator. Именно ей мы и воспользуемся.

Формула для преобразования географических координат в меркаторские следующая:

y = (1- (log(tan(PI/4+lat)/2))/PI))/2

x = (lon + 180)/360,

где PI = acos(-1.0)

lat – значения широты в градусах

lon – значения долготы в градусах

Получается, что в методе класса layer, который выполнял обращение к ImageManager необходимо изменить функция расчета позиции тайла, добавить вызов метода сохранения позиции тайла у ImageManager. Кроме того, для обращения ко всем методам необходимо создать экземпляр класса наследника ImageManager. Следовательно, для класса Layer также необходимо создавать новый дочерний класс, в котором выполнять переопределение метода.

Так как LayerManager взаимодействует с методом Layer, который был переопределен, необходимо переопределять и метод LayerManager. Раз происходит изменение метода, то из него можно убрать отрисовку изображения при помощи QPainter, так как для задачи этого не требуется.

По той же причине, по которой было выполнено переопределение layerManager, необходимо выполнить наследование класса mapControl, в котором выполнить переопределение метода обращения к классу Layermanager.

Диаграмма классов представлена на рисунке 13.

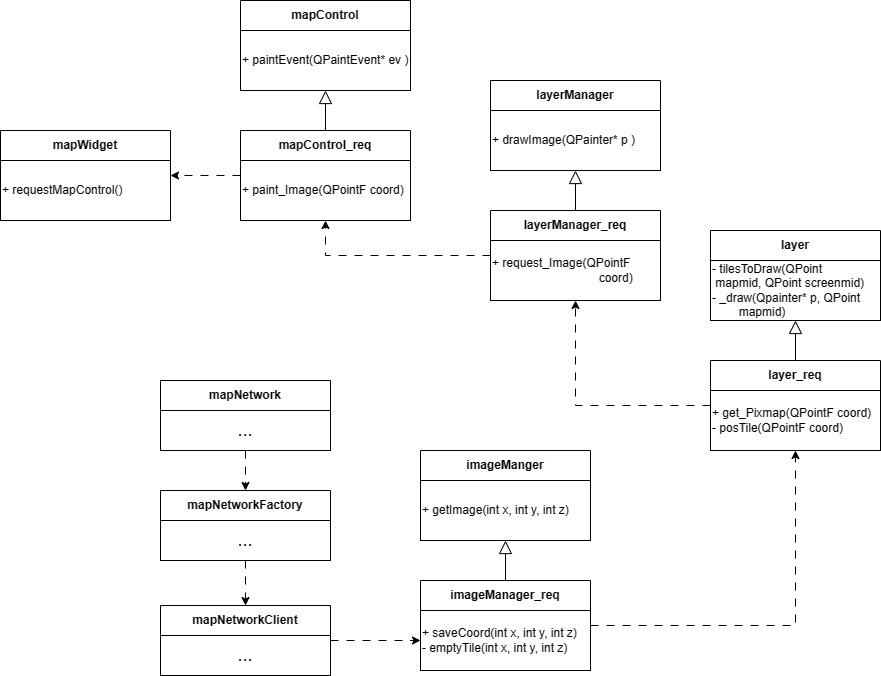
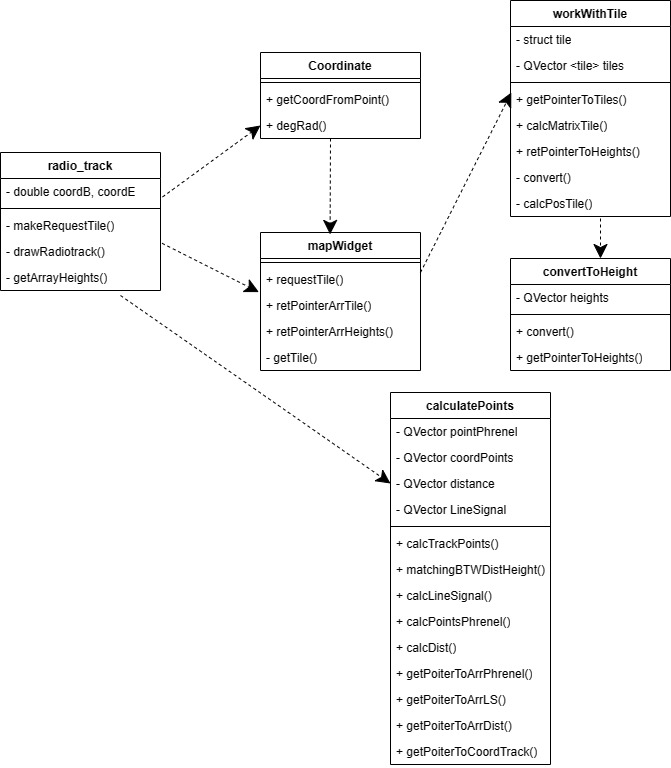


Рисунок 13 – Диаграмма классов

3.2 Расчет радиотрассы



Как видно из рисунка, все методы, а также переменные для хранения данных, относящихся к расчетам различных точек, вынесены в отдельный класс – calculatePoint. Всё что касается работы с координатами: выделение градусов, минут и секунд из дробного значения; перевод градусы в радианы и тд, также вынесены в отдельный класс. Вся работа с тайлами тоже вынесена в отдельный класс. Помимо методов для работы с тайлами в класс была добавлена структура tile, имеющая следующие описание:

struct tile

{

int pos\_x;

int pos\_y;

QPixmap pix\_map:

};

Более подробно стоит рассмотреть функцию calcMatrixTile, выполняющую расчет массива тайлов. Листинг кода, содержащий описание функции приведен ниже:

void workWithTile::calcMatrixTiles(const QPointF& coord\_begin,

const QPointF& coord\_end, int zoom)

{

// перевод географических координат точки в позицию тайла

QPoint coord\_tile\_begin = convertCoordDisplayToPosTile(coord\_begin, zoom);

QPoint coord\_tile\_end = convertCoordDisplayToPosTile(coord\_end, zoom);

//нахождение минимальной позиции тайла

findMinCoord(coord\_tile\_begin, coord\_tile\_end);

//перебор всех позиций тайлов в пределах вычисленной

for (auto i = coord\_tile\_begin.x(); i <= coord\_tile\_end.x(); i++)

{

for (auto j = coord\_tile\_begin.y(); j <= coord\_tile\_end.y(); j++)

{

tile new\_tile;

new\_tile.pos\_x = i;

new\_tile.pos\_y = j;

saveTile(new\_tile); }}}

3.3 Область покрфтия радиотрассы

Расчет области покрытия радиотрассы является ничем иным, как многократным расчетом радиотрассы для разных координат точек приемника. В уже доработанном алгоритме расчета радиотрассы запрос тайлов для последующего построения профиля высот используется вычисление массива тайлов для маршрута радиотрассы. Такой подход приведет к дублированию тайлов при вычислении области покрытия, поэтому стоит сделать перегрузку уже имеющейся функции для расчета массива тайлов, изменив набор параметров.

Код приведен ниже:

void workWithTile::calcMatrixTiles(const QPointF& coord\_right,

const QPointF& coord\_up,

const QPointF& coord\_left,

const QPointF& coord\_down, int zoom)

{

QPoint coord\_tile\_left = convertCoordDisplayToPosTile(coord\_left, zoom);

QPoint coord\_tile\_right = convertCoordDisplayToPosTile(coord\_right, zoom);

QPoint coord\_tile\_up = convertCoordDisplayToPosTile(coord\_up, zoom);

QPoint coord\_tile\_down = convertCoordDisplayToPosTile(coord\_down, zoom);

findMinCoord(coord\_tile\_left, coord\_tile\_up);

findMinCoord(coord\_tile\_right, coord\_tile\_down);

for (int i = coord\_tile\_left.x(); i <= coord\_tile\_down.x(); i++)

{

for (int j = coord\_tile\_left.y(); j <= coord\_tile\_down.y(); j++)

{

tile new\_tile;

new\_tile.pos\_x = i;

new\_tile.pos\_y = j;

saveTile(new\_tile);

}

}}

Диаграмма классов

Был добавлен новые классы, содержащий в себе методы для расчета и отрисовки области покрытия радиотрассы на карте.

3.4 3Д

Все реализованные ранее части приложения разрабатывалась под Qt. И было бы намного удобнее реализовать новый метод также с использованием Qt. Удобство использование этого фреймворка обусловлена удобством и простой дальнейшей поддержки этого проекта.

Задача построения 3D модели для какого-то набора данных является довольно распространенной задачей. Следовательно, во фреймворке Qt должны быть уже реализованы модули, позволяющие выполнять 3D визуализацию.

Фреймворк, действительно, содержит 2 модуля, позволяющие выполнить задачу: Qt Data Visualization, Qt OpenGL

Что касается последнего, то с Qt 5 данный модуль считается устаревшим, а со следующей версии он был убран. Так как разрабатываемое приложение планируется поддерживать и дальше, то использование этого модуля для разработки является нецелесообразным.

Анализ классов для данного модуля производился при помощи рассмотрения диаграммы классов, представленной на рисунке 4.1

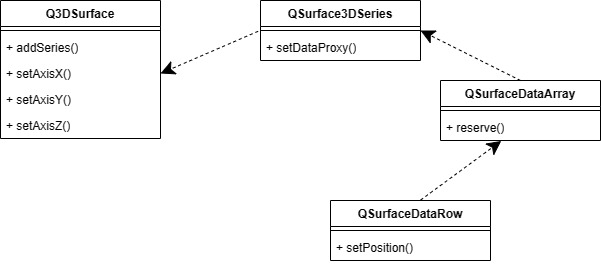


Рисунок 4.1 – Диаграмма классов модуля Qt Data Visualization

В модуле Data Visualization для работы с 3D графикой разработан класс Q3DSurface, его основной задачей является рисование 3х-мерной модели вместе с системой координат. Для задания системы координат используются методы setAxisX, setAxisY, setAxisZ. Для рисования 3D модели используется метод addSeries.

Для отрисовки данных их необходимо сформировать и передать обратно в класс Q3DSurface. Для этой передачи данных был реализован класс QSurface3DSeries, содержащий метод setDataProxy. А для формирования данных используется класс QSurfaceDataArray.

Так как любая 3х-мерная модель состоит из набора 3х-мерных данных, то в модуле был разработан класс QSurfaceDataArray, который сохраняет весь набор данных в 3х-мерный массив для дальнейшего использования. Кроме того, был разработан класс QSurfaceDataRow для создания 3х мерной точки, которая впоследствии записывается в 3х-мерный массив.

Диаграмма разработанных классов с использованием классов модуля представлена на рисунке 4.2

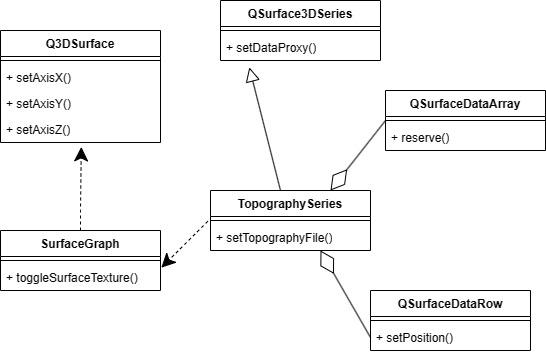


Рисунок 4.2 - Диаграмма разработанных классов с использованием классов модуля

Данные для 3х-мерной модели формируются за счет преобразования каждого тайла, используемого при расчетах, в 3х-мерный массив. То есть координаты пикселя вместе с номером тайла в массиве преобразовываются в координаты X, Z. А значение цвета пикселя, конвертируется в высоту и будет являться координатой Y.

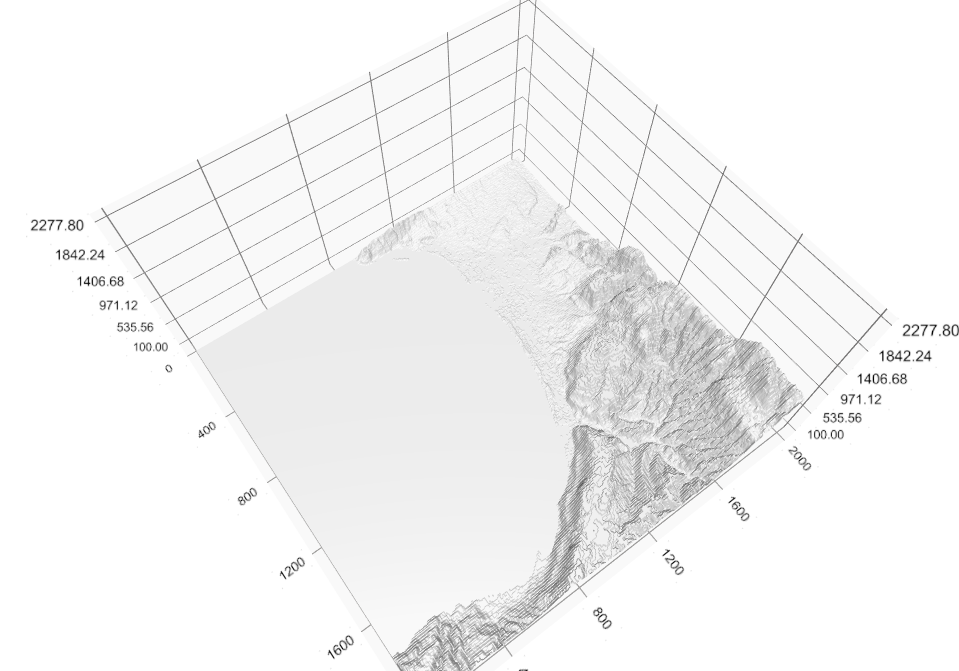
Для формирования данных был разработан класс TopographySeries, который является наследником от класса QSurface3DSeries и в котором добавлен метод setTopographyFile, который и занимается получением данных из тайлов.

Для создания 3х-мерной точки используется класс QSurfaceDataRow, экземпляр которого создается внутри TopographySeries и в который записывается значение полученная в ходе конвертации значений точка. А для сохранения этой точки в массив используется QSurfaceDataArray, экземпляр которого создается внутри TopographySeries.

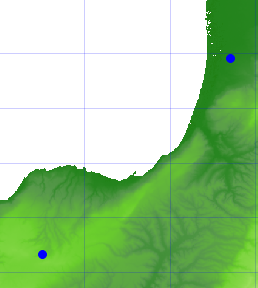
Далее набор данных передается для отрисовки, но так как QSurface3DSeries был изменен через создание наследника - нового класса, то вместо Q3DSurface был разработан новый класс SurfaceGraph, который вызывает методы TopographySeries для получения и формирования данных для 3D модели. Создание координатных осей остается без изменений, поэтому методы для их создания вызываются из экземпляра класса Q3DSurface.

Листинг кода приведен в приложении Б.

Результат построения 3D модели рельефа для области представлен на рисунке 4.3.



а)



Б)

Рисунок 4.3 а) - Результат построения 3D модели рельефа для области, б) – область, для которой выполнялось 3D моделирование