Министерство образования и науки Российской Федерации

Поволжский государственный технологический университет

Факультет информатики и вычислительной техники

Кафедра информатики и системного программирования

ОТЧЕТ

по педагогической практике

Тема: «Применение картографического API в военной online-игре реального времени»

Выполнил:

студент группы ПСм-21

Кузовков А.В.

Проверил: профессор

Морозов М.Н.

Йошкар-Ола

2015

**Содержание**

ВВЕДЕНИЕ ….................................................................................................... 3

ПОСТАНОВКА ЗАДАЧИ……………………………………………………. 5

1. Обзор существующих игр на реальных картах и анализ популярных картографических сервисов.………………...…..….........................................10

1.1. Обзор существующих игр, использующих реальные географические карты......................................................................... 10

1.2. Анализ сервисов маршрутов и сервисов высотных данных

Google …………………………………………………………………11

1.3. Анализ сервисов маршрутов и сервисов высотных данных

Яндекс……………………………………………………….……...…30

2. Выбор альтернативы сервисам Google и Яндекс …………………….32

2.1. Open Source Routing Machine (OSRM) ………………………..33

2.2. SpatiaLite — расширение SQLite для пространственных

данных………………………………………………………………….34

2.3. GTOPO30 - глобальная цифровая модель рельефа ……………35

2.4. Leaflet ……………………………….…………………………….36

3. Реализация прототипа игры ……………………………………………...37

3.1. Серверная часть …………………………………………………..38

3.2. Клиентская часть …………………………………………………39

3.3. Решение задачи определения окружения ……………………….40

3.4. Использование данных рельефа местности ……………………41

4. Оценка результатов ……………………………………………………….42

4.1. Производительность ……………………………………………...43

4.2. Масштабируемость ……………………………………………….44

ЗАКЛЮЧЕНИЕ ................................................................................................ 47

ЛИТЕРАТУРА……………… ….................................................................... 48

ВВЕДЕНИЕ

В настоящее время существует огромное количество компьютерных игр самых разных жанров. Среди игр можно выделить такой тип игр как браузерные игры, которые требуют от пользователей лишь наличия браузера на компьютере. Некоторые из них являются многопользовательскими, что подразумевает совместную игру многих интернет-пользователей, которые могут взаимодействовать между собой в игровом пространстве, объединятся в группы для достижения определенных игровых задач. Если посмотреть статистику по существующим многопользовательским браузерным играм [1] то можно увидеть что большинство игр это MMORPG (Massively multiplayer online role-playing game) - Массовая многопользовательская ролевая онлайн-игра, либо стратегии ( пошаговые или реального времени ). Суть данных игр сводится к управлению доступными игроку ресурсами для достижения превосходства над своими противниками. Например в военных стратегиях нужно управлять боевыми единицами(юнитами) с целью занять более выгодную по отношению к войскам противника позицию. При столкновении с боевыми единицами противника, как правило, происходит бой, юниты наносят урон друг другу пока один из них не будет уничтожен. Действия происходят на определенной местности, поэтому при разработке игр подобного рода возникает потребность в создании карт игрового пространства, в котором разворачивается действие игры. Карты важны, так как местоположение, рельеф, дороги и наличие ресурсов влияет на действия игрока. Поэтому карты должны быть разнообразны и интересны для игроков [2]. Разнообразие, сложность и непредсказуемость используемых карт очень существенно влияют на сохранение интереса к игре.

На сегодняшний момент существует очень много картографических сервисов, которые предоставляют возможность использовать карты местности для навигации, для поиска местности по адресу, или же по географическим координатам. Практически нет игр, использующих реальные карты для моделирования событий в рамках военных стратегий, где привязка передвижения происходит по реальным дорогам. [3]

Возможность создания военно- стратегической онлайн-игры для моделирования военных конфликтов на реальных картах, включающей поиск маршрутов по реальным дорогам была показана в работе Сидоркиной Т.В. «Создание прототипа многопользовательской игры с использованием API Google Maps». Для создания интересной пользователям игры, целесообразно кроме построения маршрутов по дорогам, включить возможность использовать данные рельефа и определять факт окружения юнитов игрока.

Вместе с тем, возможности применении картографических сервисов для построения игры имеют ограничения. К ним относятся ограничения на количество бесплатных запросов[4], лицензионные ограничения. Службы построения маршрутов позволяют строить маршруты проходящие через заданные путевые точки, но не позволяют строить маршруты не проходящие через ряд заданных точек (что требуется для установления факта окружения, например).

В данной работе исследуется возможность найти альтернативу сервисам Google и Яндекса, использовать в игре данные о рельефе местности и построения сервиса маршрутов для определения факта окружения.

ПОСТАНОВКА ЗАДАЧИ

Целью магистерской диссертационной работы является исследование возможности использования сервисов высотных данных и маршрутов для учета рельефа местности и определения факта окружения игрового юнита в военной многопользовательской онлайн стратегии реального времени и использованием реальных карт. Особенностью является то, что производится поиск альтернатив сервисам маршрутов и высотных данных Google и Яндекс.

В соответствии с поставленной целью в магистерской диссертационной работе решаются следующие задачи:

1. Анализ существующих игр использующих реальные географические карты.
2. Анализ возможностей и ограничений картографических сервисов для получения данных рельефа и определения окружения.
3. Выбор платформы для создания сервиса высотных данных и сервиса маршрутов, а также для создания прототипа игры.
4. Создание прототипа игры на реальных географических картах, использующей данные рельефа и сервис маршрутов для определения окружения.

**1. Обзор существующих игр на реальных картах и анализ популярных картографических сервисов.**

**1.1. Обзор существующих игр, использующих реальные географические карты.**

На сегодняшний день существует ряд игр, использующих географические карты в качестве игрового пространства. Не все из них относятся к многопользовательским онлайн стратегиям.

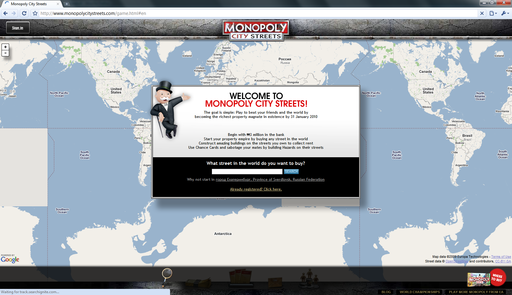
**Monopoly City Streets**

Monopoly City Streets — глобальная онлайн-игра на основе Google Maps.

Разработчик: [Tribal DDB](http://en.wikipedia.org/wiki/Tribal_DDB).

Жанр: многопользовательская онлайн стратегия реального времени.

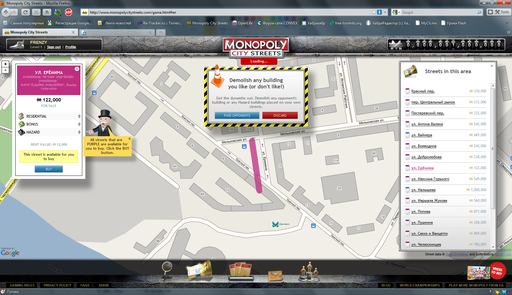
При первом посещении сервис предлагает нам начать с того города, который он определил по нашему IP адресу.



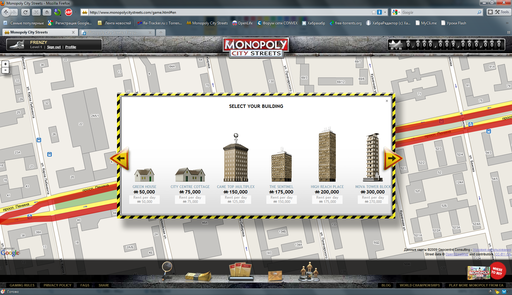
После перехода по ссылке попадаем в свой город и получаем возможность купить любую улицу. С этого и начинается игра. Именно в момент покупки улицы вам предлагается зарегистрироваться.



Фиолетовым цветом отмечены улицы, доступные для покупки, а синим — уже принадлежащие другим игрокам. На данном скриншоте представлена свободная для покупки улица (так же вылезло предложение найти ближайшее здание оппонента и взорвать его:)



Кликнув на свою улицу мы можем ее либо продать, либо построить на ней новое здание:



Доход приносят как здания, так и улицы — ежедневно.

Так же в игре есть так называемые Chance Crads и они делятся на несколько видов:

Hazard Chance Card — позволяет строить такие здания как Power Plants, Prisons, и Sewage Works. Эти здания нужно строить на улицах соперников, т.к. они убивают приносимый улицей доход.

Bulldozer Chance Card — позволяет разрушать здания, построенные благодаря Hazard, или бонусные здания противников.

Bonus Building Chance Card — защищает ваши самые лучшие улицы от попыток построить на них различные здания типа Hazard, а также отменяют эффект от уже имеющихся Hazard-ов.

Другие — карточки, которые случайным образом могут принести прибыль или убыток.

Chance Cards выпадают случайным образом во время строительства или сноса зданий.

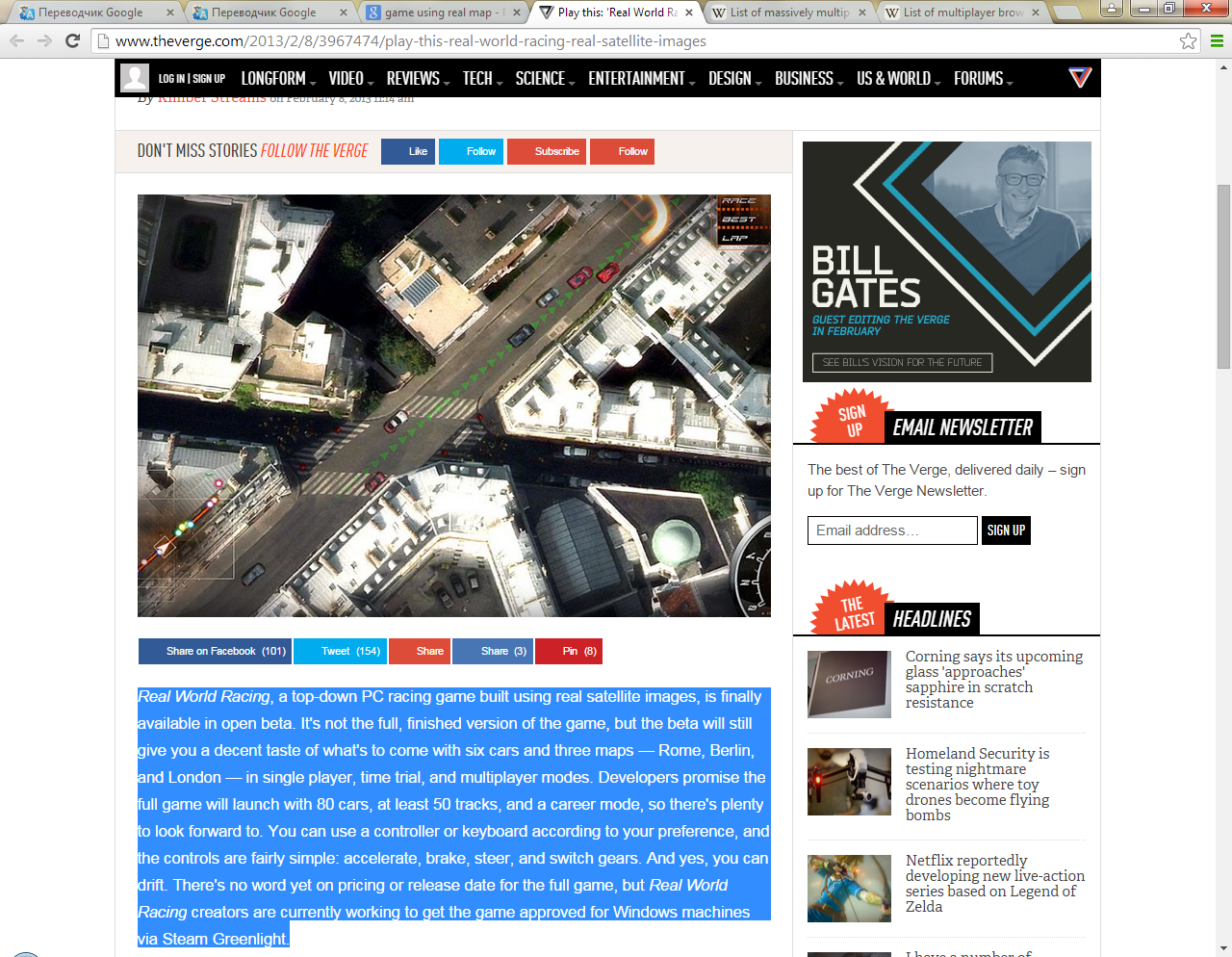
В игре есть три типа рейтинга: локальный (городской), региональный(рейтинг по стране), мировой. Цель игры - стать первым в рейтинге :)

**Real World Racing**

Разработчик: Playstos Entertainment.

Жанр: гонки.

Real World Racing гоночная игра построена для ПК с использованием реальных спутниковых изображений. Игроку доступно 6 типов автомобилей автомобилей и три карты - Рим, Берлин и Лондон - в однопользовательском и многопользовательском режимах. Управление автомобилем производиться с помощью клавиатуры. Скриншот представлен ниже.

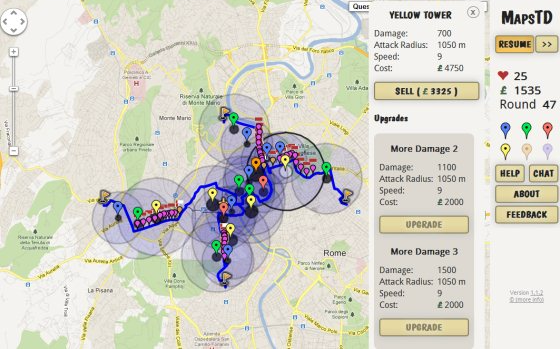


**MapsTD**

Разработчик: Duncan Barclay.

Жанр: Tower defense

Игра «MapTD» использует карты сервиса Google Maps API. Это браузерная онлайн-игра. Цель игры защитить свой дом от вражеских «ползунов», строя башни, которые их атакуют.



Местоположение игрок выбирает из предложенного списка или введя название населенного пункта. Маршрут движения «ползунов» задается в начале игры и далее не изменяется во время игры, могут только добавляться новые.

Игроку доступно масштабирование карты, то есть масштаб карты можно увеличить или уменьшить. Анимация также присутствует: в виде движения «ползунов» вдоль маршрута, проходящий от вражеской базы к дому игрока. Игра однопользовательская, где в качестве противника выступает компьютер.

**World of the Living Dead**

Разработчик: Ballardia.

Жанр: массовая многопользовательская онлайн ролевая игра, стратегия

Действие игры разворачивается постапокалиптическом Лос-Анджелесе, захваченном зомби. Игроку предстоит командовать несколькими отрядами оставшихся в живых людей, бродящими по пустынным улицам и борющимися за выживание. Для отображения игрового мира ранее использовался движок Google Maps, позднее разработчики перешли на Open Street Map.



Игроки могут видеть на карте мира свое местоположение, местоположение других игроков, плотность зомби, ресурсы. Игрокам нужно собирать ресурсы, которые необходимы для выживания, прятать их и т.д.

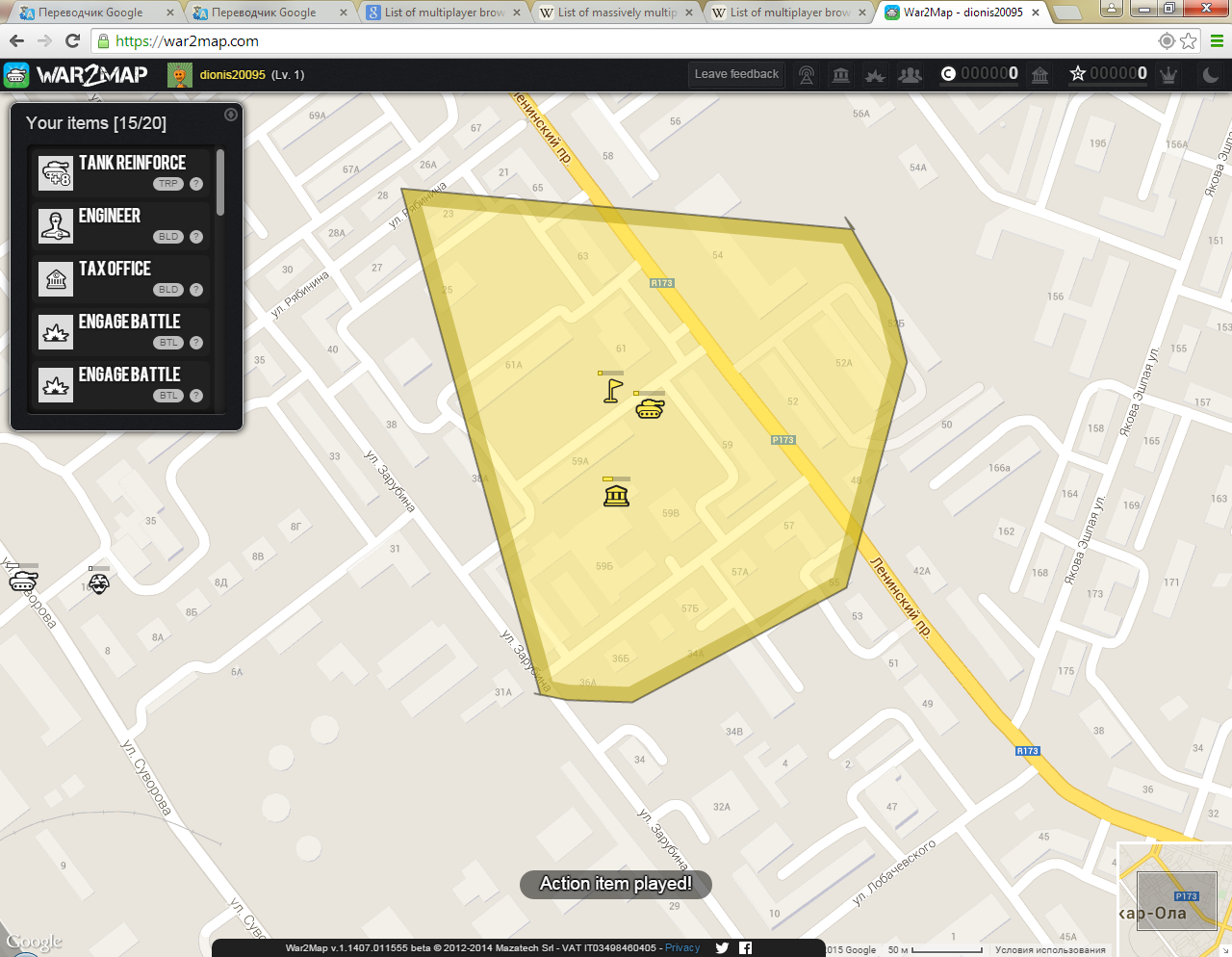


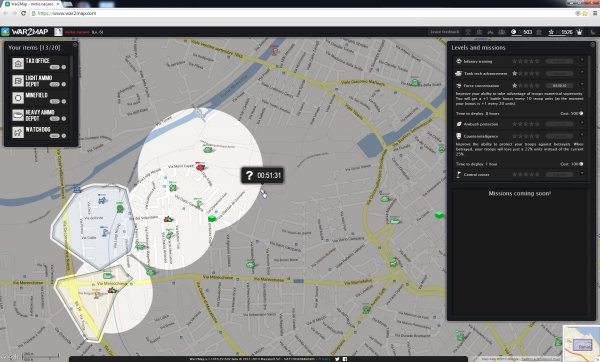
**War2map**

Разработчик: Маттео Муратори

Жанр: смесь с казуальной игрой, военная многопользовательская онлайн стратегия реального времени.

В игре, пользователи могут развертывать войска, строить инфраструктуру, расширять и защищать территории, собирать налоги, торговать игровыми объектами с другими пользователями по всему миру. Сражаясь с другими игроками, игрок набирает очки опыта. Внутриигровые деньги производятся налоговыми органами, расположенными на территории подконтрольной игроку. Цель игры развитие инфраструктуры, войск, экономики расширение территории.

****

****

**1.2. Анализ сервисов маршрутов и сервисов высотных данных Google**

API Google Карт предоставляет веб-службы в качестве интерфейса для запроса данных API Карт из внешних служб и использования этих сведений в приложениях на базе Карт. Согласно условиям [Ограничения лицензии на использование API Google Карт](https://developers.google.com/maps/terms?hl=ru#section_10_12) эти службы не могут использоваться отдельно от карт Google.

Данные веб-службы используют HTTP-запросы к определенным URL, передавая параметры URL в качестве аргументов. Обычно эти службы возвращают приложению HTTP-запрос, содержащий данные в формате JSON или XML для их синтаксического анализа и обработки.

Запрос к веб-службе обычно имеет следующий вид:

http://maps.googleapis.com/maps/api/*service*/*output*?*parameters*

где service указывает запрашиваемую службу, а output – формат ответа (обычно json или xml).

Доступ к веб-службам API Google Карт можно также осуществлять по протоколу **HTTPS**. Для этого следует изменить протокол в URL запроса на https, как показано ниже:

**https**://maps.googleapis.com/maps/api/service/output?parameters

Запросы от приложений, работающих с конфиденциальными данными пользователей, например, с данными об их местоположении, рекомендуется отправлять по протоколу HTTPS.

Для использования Google JS API нужно подключить его код в браузере:

*<script src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&sensor=SET\_TO\_TRUE\_OR\_FALSE" type="text/javascript"></script>*

С помощью объекта *DirectionsService* можно рассчитывать маршруты для различных способов передвижения. Этот объект взаимодействует со службой маршрутов интерфейса API Google Карт, которая получает запрос маршрута и возвращает вычисленные результаты. Вы можете сами обработать эти результаты или использовать объект*DirectionsRenderer* для их визуализации.

В службе маршрутов пункты отправления и назначения могут указываться в виде текстовых запросов (например, "Чикаго, Иллинойс, США" или "Дарвин, Новый Южный Уэльс, Австралия") либо в виде координат LatLng. Результаты возвращаются в виде последовательности отрезков, проходящих через путевые точки. Маршруты отображаются в виде полилинии, показывающей маршрут на карте, или дополнительно в виде последовательности текстовых описаний в элементе <div> (например, "Поверните направо для въезда на Троицкий мост").

Доступ к службе маршрутов осуществляется асинхронно, поскольку API Google Карт требуется отправить вызов на внешний сервер. Именно поэтому необходимо передать метод обратного вызова, который будет выполнен по завершении запроса. Полученные результаты будут обработаны с помощью этого метода обратного вызова. Имейте в виду, что служба маршрутов может возвратить несколько вариантов в виде массива отдельных элементов *routes[].*

Чтобы использовать маршруты в версии 3, создайте объект типа *DirectionsService* и вызовите метод *DirectionsService.route()*для отправки запроса в службу маршрутов, передавая ей литерал объекта*DirectionsRequest*, содержащий условия ввода и метод обратного вызова для выполнения после получения ответа.

Литерал объекта *DirectionsRequest* содержит следующие поля:

*{  
  origin: LatLng | String,  
  destination: LatLng | String,  
  travelMode: TravelMode,  
  transitOptions: TransitOptions,  
  unitSystem: UnitSystem,  
  waypoints[]: DirectionsWaypoint,  
  optimizeWaypoints: Boolean,  
  provideRouteAlternatives: Boolean,  
  avoidHighways: Boolean,  
  avoidTolls: Boolean  
  region: String  
}*

Далее идет описание этих полей:

origin (обязательный параметр) указывает начальное местоположение, от которого следует вычислять маршрут. Этот параметр может иметь значение типа String (например, "Чикаго, Иллинойс, США") или LatLng.

destination (обязательный параметр) указывает конечное местоположение, до которого следует вычислять маршрут. Этот параметр может иметь значение типа String (например, "Чикаго, Иллинойс, США") или LatLng.

travelMode (обязательный параметр) задает способ перемещения, который используется для вычисления маршрута. Допустимые значения указаны далее в разделе [Способы перемещения](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#TravelModes).

transitOptions (необязательный параметр) указывает значения, используемые только в запросах, в которых параметрtravelMode имеет значение google.maps.TravelMode.TRANSIT. Допустимые значения описаны далее в разделе [Параметры маршрутов общественного транспорта](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#TransitOptions).

unitSystem (необязательный параметр) задает единицы измерения, которые следует использовать при отображении результатов. Допустимые значения указаны далее в разделе [Системы измерений](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#UnitSystems).

waypoints[] (необязательный параметр) указывает массив элементов DirectionsWaypoint. Использование путевых точек приводит к изменению маршрута, направляя его через один или несколько указанных пунктов. Путевая точка задается как литерал объекта со следующими полями:

location задает местоположение путевой точки, которая будет геокодирована, как LatLng или String.

stopover содержит логическое значение, указывающее, что путевая точка является остановкой на маршруте, что приводит к его разделению на две части.

Дополнительные сведения о путевых точках см. ниже в разделе [Использование путевых точек в маршрутах](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#Waypoints).

optimizeWaypoints (необязательный параметр) указывает, что с помощью предоставленных элементов waypoints маршрут можно оптимизировать для получения кратчайшего пути. Если в этом поле указано значение true, служба маршрутов возвращает переупорядоченные элементы waypoints в поле waypoint\_order.Дополнительные сведения см. далее в разделе [Использование путевых точек в маршрутах](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#Waypoints).

provideRouteAlternatives (необязательный параметр) – значение true указывает, что служба может предложить несколько альтернативных маршрутов. Обратите внимание, что расчет альтернативных путей может увеличить время отклика сервера.

avoidHighways (необязательный параметр) – значение true указывает, что проложенные маршруты должны по возможности избегать автомагистралей.

avoidTolls (необязательный параметр) – значение true указывает, что проложенные маршруты должны по возможности избегать платных дорог.

region (необязательный параметр) указывает код региона в виде двухсимвольного значения ccTLD (домен верхнего уровня). Дополнительные сведения см. далее в разделе [Предпочитаемый регион](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#DirectionsRegionBiasing).

**Способы перемещения**

При вычислении маршрутов требуется указать используемый способ перемещения. В настоящее время поддерживаются следующие способы перемещения:

google.maps.TravelMode.DRIVING (используется **по умолчанию**) обозначает стандартные автомобильные маршруты по улично-дорожной сети.

google.maps.TravelMode.BICYCLING запрашивает велосипедные маршруты по велосипедным дорожкам и предпочитаемым улицам.

google.maps.TravelMode.TRANSIT запрашивает маршруты общественного транспорта.

google.maps.TravelMode.WALKING запрашивает пешеходные маршруты по прогулочным дорожкам и тротуарам (если они есть).

Сведения о поддержке службы маршрутов в конкретных странах см. в [таблице с информацией о географии охвата Google Карт](http://gmaps-samples.googlecode.com/svn/trunk/mapcoverage_filtered.html). При запросе маршрутов, которые недоступны для указанного региона, в ответе возвращается значение DirectionsStatus="ZERO\_RESULTS".

**Статус запроса маршрута**

*DirectionsStatus*может возвратить следующие значения:

*OK* указывает, что ответ содержит допустимый элемент DirectionsResult.

*NOT\_FOUND* означает, что по крайней мере для одного указанного пункта (исходный, пункт назначения или путевая точка) не удалось выполнить геокодирование.

*ZERO\_RESULTS* означает, что между исходной точкой и пунктом назначения не найдено ни одного маршрута.

*MAX\_WAYPOINTS\_EXCEEDED* указывает, что в *DirectionsRequest* задано слишком много объектов *DirectionsWaypoint*. Максимальное разрешенное количество путевых точек составляет 8 помимо исходной точки и пункта назначения. Клиентам API Google Карт для организаций разрешено использовать 23 путевые точки помимо исходной точки и пункта назначения. Для маршрутов общественного транспорта путевые точки не поддерживаются.

*INVALID\_REQUEST* указывает, что был предоставлен недопустимый объект*DirectionsRequest*. В большинстве случаев этот код ошибки возвращается для запросов без исходной точки и пункта назначения, а также для запросов маршрутов общественного транспорта с путевыми точками.

*OVER\_QUERY\_LIMIT* означает, что в разрешенный период времени веб-страница отправила слишком много запросов.

*REQUEST\_DENIED* означает, что веб-странице не разрешено использовать службу маршрутов.

*UNKNOWN\_ERROR* означает, что обработка запроса маршрута невозможна из-за ошибки сервера. При повторной попытке запрос может быть успешно выполнен.

Прежде чем обрабатывать результат, необходимо убедиться, что запрос маршрута возвратил допустимые данные.

**Объект DirectionsResult**

При отправке запроса маршрута в *DirectionsService* вы получите ответ, состоящий из кода статуса и результата, которым является объект *DirectionsResult*. *DirectionsResult* представляет собой литерал объекта с единственным полем:

*routes[]* содержит массив объектов *DirectionsRoute*. Каждый маршрут указывает путь от исходной точки до пункта назначения, предоставленный в объекте *DirectionsRequest*. Обычно для любого запроса возвращается только один маршрут, за исключением случаев, когда для поля *provideRouteAlternatives*запроса установлено значение true. В таких случаях может быть возвращено несколько маршрутов.

**Маршруты службы маршрутов**

Устаревший объект *DirectionsTrip* был переименован в*DirectionsRoute*. Обратите внимание, что объект route теперь относится ко всему маршруту от начала до конца, а не только к отрезку родительского маршрута.

*DirectionsRoute* содержит единственный результат для заданных исходной точки и пункта назначения. В зависимости от указанных путевых точек маршрут может состоять из одного или нескольких отрезков (типа *DirectionsLeg).* Маршрут содержит также информацию об авторских правах и предупреждения, которые следует показать пользователю в дополнение к сведениям о маршруте.

*DirectionsRoute* представляет собой литерал объекта со следующими полями:

*legs[]* содержит массив объектов *DirectionsLeg*, каждый из которых содержит информацию об отрезке маршрута между двумя точками на заданном маршруте. Для каждой указанной путевой точки или пункта назначения будет представлен отдельный отрезок. (Маршрут без путевых точек будет содержать только один объект *DirectionsLeg*.) Каждый отрезок состоит из последовательности*DirectionStep*.

waypoint\_order содержит массив, указывающий порядок следования всех путевых точек в вычисленном маршруте. Этот массив может содержать измененный по сравнению с исходным порядок, если объекту *DirectionsRequest*было передано значение *optimizeWaypoints: true*.

overview\_path содержит массив объектов LatLng, представляющий приблизительный (сглаженный) путь вычисленного маршрута.

bounds содержит объект *LatLngBounds*, указывающий границы полилинии вдоль заданного маршрута.

copyrights содержит обязательный для отображения текст об авторских правах, относящийся к этому маршруту.Если вы не используете предоставленный объект *DirectionsRenderer*, то вам потребуется самостоятельно обработать и отобразить эту информацию.

warnings[] содержит массив предупреждений, которые следует показать вместе с этим маршрутом. Если вы не используете предоставленный объект *DirectionsRenderer*, то вам потребуется самостоятельно обработать и отобразить эту информацию.

Как отмечено в разделе [Объект DirectionsRequest](https://developers.google.com/maps/documentation/javascript/directions?hl=ru#DirectionsRequests), при вычислении пешеходных, велосипедных и автомобильных маршрутов с помощью службы маршрутов можно указывать также путевые точки типа DirectionsWaypoint. Для маршрутов общественного транспорта путевые точки не поддерживаются. Путевые точки позволяют вычислить путь через дополнительные пункты, и в этом случае результирующий маршрут будет проходить через эти путевые точки.

Максимальное разрешенное количество путевых точек составляет 8 помимо исходной точки и пункта назначения. Клиентам API Google Карт для организаций разрешено использовать 23 путевые точки помимо исходной точки и пункта назначения. Для маршрутов общественного транспорта путевые точки не поддерживаются.

Объект*waypoint* состоит из следующих полей:

*location* (обязательный параметр) задает адрес путевой точки.

*stopover* (необязательный параметр) указывает, является ли путевая точка фактической остановкой на маршруте (true) или только предпочтительной точкой для маршрута, проходящего через указанное местоположение (false). Для остановок в пути используется значение по умолчанию true.

По умолчанию служба маршрутов вычисляет маршрут через указанные путевые точки в порядке их перечисления. Дополнительно можно передать значение *optimizeWaypoints*: true в запросе *DirectionsRequest*, чтобы разрешить службе маршрутов оптимизировать предоставленный маршрут путем выбора более эффективной последовательности путевых точек. Эта оптимизация является приложением [задачи коммивояжера](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0). Все путевые точки должны быть остановками, чтобы служба маршрутов выполнила оптимизацию пути.

Если службе маршрутов дать указание оптимизировать последовательность прохождения путевых точек, она возвращается в полеoptimized\_waypoints\_order внутри объекта DirectionsResult.

**Ограничения на использование**

На использование службы API маршрутов Google накладывается ограничение, составляющее 2500 запросов маршрутов в день. Поиск автомобильного, велосипедного или пешеходного маршрута учитывается при расчете ежедневной квоты как один запрос. Поиск маршрута общественного транспорта учитывается как 4 запроса.

Отдельные запросы автомобильных, велосипедных или пешеходных маршрутов могут содержать до 8 промежуточных путевых точек. Пользователи [API Google Карт для организаций](https://developers.google.com/maps/documentation/business?hl=ru) в течение дня могут запрашивать до 100 000 маршрутов, для каждого из которых допускается до 23 путевых точек. Для маршрутов общественного транспорта путевые точки не поддерживаются.

Кроме того, прежде чем выполнять [кодирование адресов URL](https://developers.google.com/maps/documentation/webservices/index?hl=ru#BuildingURLs), убедитесь в том, что адреса URL для API маршрутов содержат не более 2048 символов. Помните об этом ограничении, формируя адреса URL, поскольку некоторые адреса URL, используемые службой маршрутов, могут привести к включению в маршрут нескольких пунктов.

Примечание. API маршрутов можно применять только для отображения результатов на карте Google. Использование данных службы маршрутов без отображения карты, для которой они запрашивались, запрещено. Кроме того, при вычислении маршрутов формируются данные об авторских правах и предупреждения, которые каким-либо способом должны быть показаны пользователю. Полную информацию о допустимых способах применения см. в разделе [Ограничения лицензии на использование API Карт](https://developers.google.com/maps/terms?hl=ru#section_10_12).

**API высотных данных**

API высотных данных предоставляет данные о высотах всех точек на земной поверхности, включая глубины океана (в этом случае возвращаются отрицательные значения). Если у Google нет точных высотных данных для интересующей вас точки, служба использует интерполяцию и возвращает усредненное значение по четырем ближайшим точкам.

С помощью службы API высотных данных можно разрабатывать приложения для пешего и велосипедного туризма, приложения мобильного позиционирования или геодезические приложения низкого разрешения.

Доступ к API высотных данных осуществляется через интерфейс HTTP. Пользователи JavaScript API Google версии 3 могут также напрямую обращаться к API высотных данных с помощью объекта ElevationService(). Дополнительные сведения см. в разделе[Служба высотных данных](https://developers.google.com/maps/documentation/javascript/services?hl=ru#Elevation).

На использование API высотных данных Google накладываются ограничения в отношении количества запросов: не более 2500 в день (для пользователей API Карт для организаций – не более 100 000 в день). В каждом запросе разрешается указывать до 512 местоположений, но не более 25 000 в день (для пользователей API Карт для организаций – 1 000 000). Это ограничение предотвращает злоупотребление интерфейсом API высотных данных и его использование в иных целях. Ограничение может быть изменено в будущем без предварительного уведомления. Кроме того, для предотвращения злоупотреблений службой существует ограничение на количество запросов. Если превышено суточное (24 часа) ограничение или имеет место злоупотребление другого рода, API высотных данных может временно прекратить работу. При неоднократных нарушениях доступ к нему может быть заблокирован.

Для запросов по одной точке API высотных данных возвращает данные с максимальной точностью. Для пакетных запросов по нескольким точкам могут быть возвращены менее точные данные, особенно если эти точки удалены друг от друга, так как выполняется сглаживание данных.

Кроме того, прежде чем выполнять [кодирование адресов URL](https://developers.google.com/maps/documentation/webservices/index?hl=ru#BuildingURLs) убедитесь в том, что адреса URL для API высотных данных содержат не более 2048 символов. Помните об этом ограничении, формируя адреса URL, поскольку некоторые адреса URL, используемые службой высотных данных, могут привести к обработке большого числа точек.

Примечание. API высотных данных можно использовать только для отображения результатов на карте Google. Использование данных службы высотных данных без отображения карты, для которой они запрашивались, запрещено. Полную информацию о допустимых способах применения см. в разделе [Ограничения лицензии на использование API Карт](https://developers.google.com/maps/terms?hl=ru#section_10_12).

**Запросы к службе высотных данных**

API высотных данных возвращает данные о высоте в разных точках на поверхности земли. Указать требуемые точки можно одним из следующих способов:

В виде набора из одного или нескольких местоположений (элементы locations).

В виде последовательности соединенных точек вдоль пути (элемент path).

Каждый из этих методов использует значения широты и долготы для определения местоположения или вершин пути. В этом документе описаны необходимый для API высотных данных формат адресов URL и существующие параметры.

URL для API высотных данных должен иметь следующий формат:

http://maps.googleapis.com/maps/api/elevation/*outputFormat*?*parameters*

**Совет**. Для доступа к API высотных данных можно также использовать SSL, заменив https на http в URL запроса. Запросы от приложений, работающих с конфиденциальными данными пользователей, например, с данными об их местоположении, рекомендуется отправлять по протоколу HTTPS.

**Форматы вывода**

Форматы вывода указываются в URL запроса с помощью специального флага службы. В настоящее время API высотных данных поддерживает следующие форматы вывода:

/json – результаты возвращаются в формате [JavaScript Object Notation](http://en.wikipedia.org/wiki/JSON) (JSON).

/xml – результаты возвращаются в формате XML, заключенными в узел <ElevationResponse>.

**Ответы на запросы к службе высотных данных**

Для каждого допустимого запроса служба высотных данных возвращает ответ в формате, указанном в URL запроса. Каждый ответ содержит следующие элементы:

**Код состояния высоты (**status**)**, который может иметь одно из следующих значений:

OK, означающее, что запрос к API успешно выполнен.

INVALID\_REQUEST, означающее, что запрос к API сформирован неправильно.

OVER\_QUERY\_LIMIT, означающее, что отправитель запроса превысил квоту.

REQUEST\_DENIED, означающее, что API не выполнил запрос, вероятнее всего, по той причине, что отправитель запроса не указал допустимое значение параметра sensor).

UNKNOWN\_ERROR, обозначающее неизвестную ошибку.

Массив results, содержащий следующие элементы:

Элемент location (содержащий элементы lat и lng) той точки, для которой вычисляются данные о высоте. Следует отметить, что для запросов пути набор элементов location будет содержать выборочные точки вдоль пути.

Элемент elevation, указывающий высоту точки над уровнем моря в метрах.

Значение resolution, указывающее максимальное расстояние (в метрах) между точками данных, на основании которых интерполируется значение высоты. Если разрешение неизвестно, это свойство отсутствует. Обратите внимание, что если передается много точек, точность высотных данных снижается (более высокие значения элемента resolution). Чтобы получить максимально точное значение высоты для точки, следует запрашивать его независимо от других точек.

**Примеры высотных данных для отдельных точек**

В следующем примере запрашивается высота над уровнем моря Денвера, штат Колорадо, в формате JSON (этот город известен как "город высотой в милю", так как находится на отметке 1 мили над уровнем моря):

http://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&sensor=*true\_or\_false*

{  
   "results" : [  
      {  
         "elevation" : 1608.637939453125,  
         "location" : {  
            "lat" : 39.73915360,  
            "lng" : -104.98470340  
         },  
         "resolution" : 4.771975994110107  
      }  
   ],  
   "status" : "OK"  
}

В следующем примере показано несколько ответов в формате JSON (для Денвера, штат Колорадо, и Долины смерти, штат Калифорния):

http://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034|36.455556,-116.866667&sensor=*true\_or\_false*

{  
   "results" : [  
      {  
         "elevation" : 1608.637939453125,  
         "location" : {  
            "lat" : 39.73915360,  
            "lng" : -104.98470340  
         },  
         "resolution" : 4.771975994110107  
      },  
      {  
         "elevation" : -50.78903579711914,  
         "location" : {  
            "lat" : 36.4555560,  
            "lng" : -116.8666670  
         },  
         "resolution" : 19.08790397644043  
      }  
   ],  
   "status" : "OK"  
}

### Примеры высотных данных пути

В следующих примерах запрашиваются высотные данные вдоль прямолинейного пути (path) от горы Уитни, штат Калифорния, до впадины Бэдуотер, штат Калифорния (от самой высокой до самой низкой точки континентальной части США). Для параметра samplesуказывается значение, равное трем, чтобы включить две конечные точки и точку посередине пути.

*http://maps.googleapis.com/maps/api/elevation/json?path=36.578581,-118.291994|36.23998,-116.83171&samples=3&sensor=true\_or\_false*

**1.3. Анализ сервисов маршрутов и сервисов высотных данных Яндекс**

**Условия использования**

* API Яндекс.Карт предназначен для отображения карты реальным пользователям. Нельзя применять его только для получения и обработки данных, например геокодирования точек или прокладки маршрутов без вывода их в интерфейсе.
* Недопустимо скрывать или менять логотипы и копирайты на карте, а также загораживать их другими элементами.
* Бесплатный API Яндекс.Карт можно использовать только на общедоступных сайтах и в бесплатных приложениях. У сервисов с авторизацией должна быть открытая и бесплатная регистрация.
* Бесплатный API Карт нельзя использовать в модулях бухгалтерских программ, интранетах и других закрытых системах, десктоп-приложениях и программах, игровых проектах, а также системах мониторинга и диспетчеризации транспорта.
* Также в бесплатном API есть ограничение — в сутки можно делать не более 25 000 запросов к геокодеру.[7]

## Загрузка модулей при подключении API

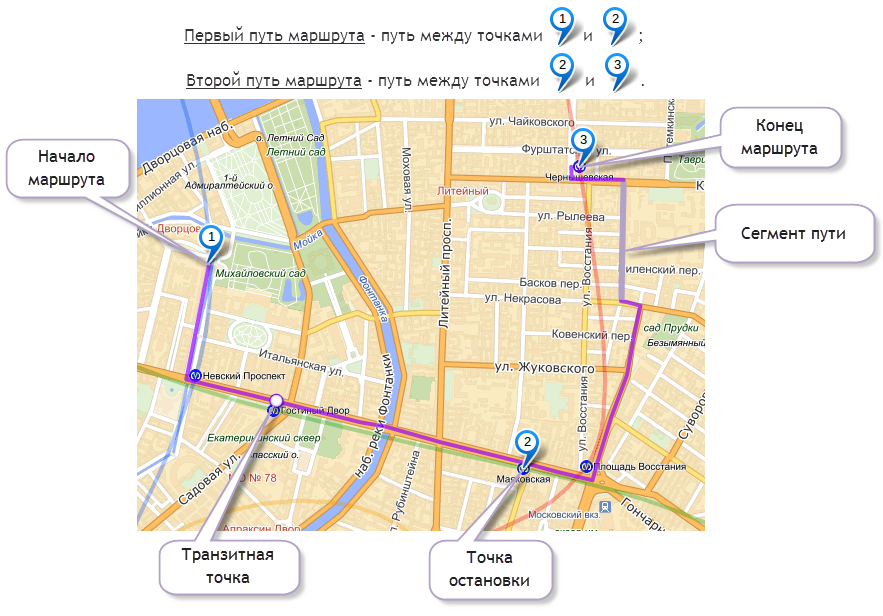
Загружаемые модули задаются HTTP-параметром load в строке подключения API. Можно указать сразу несколько модулей, разделив их запятым. По умолчанию параметр load принимает значениеpackage.full, то есть загружаются все основные модули, необходимые для работы API.

*<script src="//api-maps.yandex.ru/2.1/?lang=ru\_RU&****load****=Map,Placemark" type="text/javascript"></script>*

Загруженные модули будут включены в публичное пространство имен ymaps.

# Маршрутизатор

API предоставляет возможность прокладывания автомобильных маршрутов. Маршрут между начальным и конечным пунктом вычисляется автоматически, при этом можно задать произвольное количество точек остановки и транзитных точек маршрута.



Для построения маршрута предназначена функция [route](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/route-docpage/), в которую передается массив точек, через которые нужно проложить маршрут и, при необходимости, дополнительные опции его построения.

Вычисление маршрута производится асинхронно (аналогично [геокодированию](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geocoding-docpage/#async)). Результат передается в функцию-обработчик в виде коллекции [*GeoObjectCollection*](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/GeoObjectCollection-docpage/). Этот объект реализует интерфейс [*IGeoObject*](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/IGeoObject-docpage/), то есть может быть размещен на карте.

*ymaps.route(['Москва', 'Санкт-Петербург']).then(*

*function (route) {*

*myMap.geoObjects.add(route);*

*},*

*function (error) {*

*alert('Возникла ошибка: ' + error.message);*

*}*

*);*

Существует три способа задания точки маршрута:

Пара координат в виде массива ([59.94, 30.31]).

Строка ('Санкт-Петербург'). Производится автоматическое [геокодирование](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geocoding-docpage/).

Объект с полями *type* и *point*

*({type: 'wayPoint', point: 'Санкт-Петербург'}, {type: 'viaPoint', point: [59.94, 30.31]).*

Этот способ позволяет в явном виде указать тип точки маршрута: *wayPoint* — точка остановки, *viaPoint* — транзитная точка. Если в поле *point* передана строка, производится автоматическое геокодирование.

Если тип точки не указан, она считается точкой остановки.

*ymaps.route([*

*'Кронштадт, Якорная площадь',*

*{*

*type: 'viaPoint',*

*point: [59.93328,30.342791]// или 'Аничков мост'*

*},*

*'Санкт-Петербург, Финляндский вокзал'// или [59.956084, 30.356849]*

*]).then(*

*function (route) {*

*myMap.geoObjects.add(route);*

*},*

*function (error) {*

*alert("Возникла ошибка: " + error.message);*

*}*

*);*

Прокладывать маршрут можно как с учетом так и без учета пробок. Игнорирование или учет пробок регулируется опцией avoidTrafficJams (true/false). По умолчанию пробки не учитываются. Маршрут, проложенный без учета пробок, может отличаться от маршрута, проложенного между теми же точками, но с учетом пробок.

*ymaps.route(['Кронштадт, Якорная площадь', 'Санкт-Петербург, Финляндский вокзал'], {****avoidTrafficJams: true****});*

Построенный маршрут представляется в виде [упорядоченной коллекции](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/GeoObjectCollection-docpage/) путей ([router.Path](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/router.Path-docpage/)), попарно последовательно соединяющих точки остановки. Получить пути маршрута можно с помощью метода[getPaths](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/router.Route-docpage/#getPaths). Для предыдущего примера с промежуточной точкой «Аничков мост» кодroute.getPaths().getLength() возвратит единицу, так как промежуточные точки не влияют на разбиение маршрута на пути.

Каждый путь разбивается на сегменты ([router.Segment](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/router.Segment-docpage/)), получить которые из пути можно с помощью метода [getSegments](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/router.Path-docpage/#getSegments). Концами сегментов являются начальные и конечные точки путей, транзитные точки, точки вынужденной остановки (например, паром) и точки возможного изменения направления движения (развилка, въезд, съезд, поворот, разворот, перекресток).

Сегмент описывается ломаной, координаты точек ломаной возвращаются методом [getCoordinates](https://tech.yandex.ru/maps/doc/jsapi/2.1/ref/reference/router.Segment-docpage/#getCoordinates).

*var routeLength = route.getLength(); // Длина маршрута*

*var firstPath = route.getPaths().get(0); // Первый путь*

*var firstPathLength = firstPath.getLength(); // Длина первого пути*

*var firtstPathTime = firstPath.getTime(); // Время без учета пробок*

*var firstPathFirstSegment = firstPath.getSegments()[0];// Первый сегмент первого пути*

*var firstPathFirstSegmentJamsTime = firstPathFirstSegment.getJamsTime();//* Время с пробками

Для сегментов также доступны данные, позволяющие построить маршрутный лист: направление и угол поворота в конце сегмента, а также название улицы, на которой он начинается.

Сервиса высотных данных у Яндекс нет.

**2. Выбор альтернативы сервисам Google и Яндекс**

Рассмотренные выше сервисы маршрутов и высотных данных имеют серьезные ограничения.

На использование службы API маршрутов Google накладывается ограничение, составляющее 2500 запросов маршрутов в день. Поиск автомобильного, велосипедного или пешеходного маршрута учитывается при расчете ежедневной квоты как один запрос. Поиск маршрута общественного транспорта учитывается как 4 запроса.

Отдельные запросы автомобильных, велосипедных или пешеходных маршрутов могут содержать до 8 промежуточных путевых точек. Пользователи API Google Карт для организаций в течение дня могут запрашивать до 100 000 маршрутов, для каждого из которых допускается до 23 путевых точек. Для маршрутов общественного транспорта путевые точки не поддерживаются.

Кроме того, прежде чем выполнять кодирование адресов URL, убедитесь в том, что адреса URL для API маршрутов содержат не более 2048 символов. Помните об этом ограничении, формируя адреса URL, поскольку некоторые адреса URL, используемые службой маршрутов, могут привести к включению в маршрут нескольких пунктов.

API маршрутов можно применять только для отображения результатов на карте Google. Использование данных службы маршрутов без отображения карты, для которой они запрашивались, запрещено. Кроме того, при вычислении маршрутов формируются данные об авторских правах и предупреждения, которые каким-либо способом должны быть показаны пользователю. Полную информацию о допустимых способах применения см. в разделе Ограничения лицензии на использование API Карт.

На использование API высотных данных Google накладываются ограничения в отношении количества запросов: не более 2500 в день (для пользователей API Карт для организаций – не более 100 000 в день). В каждом запросе разрешается указывать до 512 местоположений, но не более 25 000 в день (для пользователей API Карт для организаций – 1 000 000). Это ограничение предотвращает злоупотребление интерфейсом API высотных данных и его использование в иных целях. Ограничение может быть изменено в будущем без предварительного уведомления. Кроме того, для предотвращения злоупотреблений службой существует ограничение на количество запросов. Если превышено суточное (24 часа) ограничение или имеет место злоупотребление другого рода, API высотных данных может временно прекратить работу. При неоднократных нарушениях доступ к нему может быть заблокирован.

Для запросов по одной точке API высотных данных возвращает данные с максимальной точностью. Для пакетных запросов по нескольким точкам могут быть возвращены менее точные данные, особенно если эти точки удалены друг от друга, так как выполняется сглаживание данных.

Кроме того, прежде чем выполнять [кодирование адресов URL](https://developers.google.com/maps/documentation/webservices/index?hl=ru#BuildingURLs) убедитесь в том, что адреса URL для API высотных данных содержат не более 2048 символов. Помните об этом ограничении, формируя адреса URL, поскольку некоторые адреса URL, используемые службой высотных данных, могут привести к обработке большого числа точек.

API высотных данных можно использовать только для отображения результатов на карте Google. Использование данных службы высотных данных без отображения карты, для которой они запрашивались, запрещено. Полную информацию о допустимых способах применения см. в разделе Ограничения лицензии на использование API Карт.

* API Яндекс.Карт предназначен для отображения карты реальным пользователям. Нельзя применять его только для получения и обработки данных, например геокодирования точек или прокладки маршрутов без вывода их в интерфейсе.
* Недопустимо скрывать или менять логотипы и копирайты на карте, а также загораживать их другими элементами.
* Бесплатный API Яндекс.Карт можно использовать только на общедоступных сайтах и в бесплатных приложениях. У сервисов с авторизацией должна быть открытая и бесплатная регистрация.
* Бесплатный API Карт нельзя использовать в модулях бухгалтерских программ, интранетах и других закрытых системах, десктоп-приложениях и программах, игровых проектах, а также системах мониторинга и диспетчеризации транспорта.
* Также в бесплатном API есть ограничение — в сутки можно делать не более 25 000 запросов к геокодеру.

Кроме того отсутвует возможность искать маршруты исключающие заданный набор мест, что потребуется для определения окружения юнита.

Поэтому возникает необходимость поиска альтернативы рассмотренным выше сервисам.

**2.1. Open Source Routing Machine (OSRM)**

Open Source Routing Machine (OSRM) - проект, с открытым исходным кодом, который позволяет развернуть у себя на сервере, свой собственный сервис построения маршрутов.

OSRM является реализацией на C ++ модуля маршрутизации с высокой производительность для поиска кратчайших путей в дорожных сетях. Она сочетает в себе сложные алгоритмы маршрутизации и использует открытые и свободные данных дорожных сетей проекта OpenStreetMap (OSM). Вычисление кратчайшего пути на сети континентального размера может занять до нескольких секунд, если это если делается классическим способом. OSRM способен вычислять и выводить кратчайший путь между любыми точками в течение нескольких миллисекунд. Так как, он разработан из расчета совместимости с OpenStreetMap OSM файлы данных могут быть легко импортированы. [9]

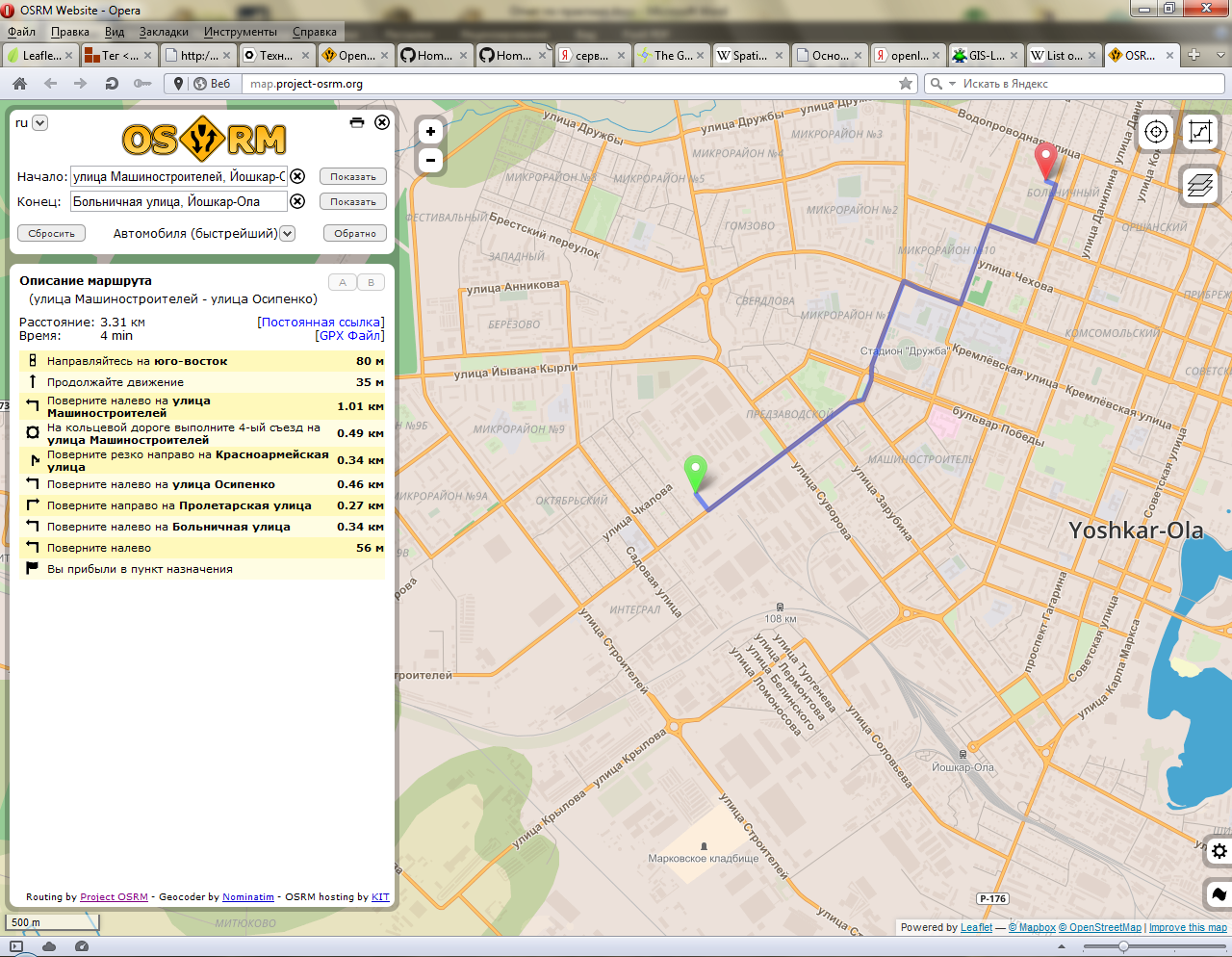
Ключевые особенности OSRM :

* Высокая производительность алгоритма маршрутизации:
* Гибкий импорт файлов данных OSM
* Написана на C ++ и доступно в соответствии BSD лицензией для любого использования.
* Возможность обработки сетей континентальных размеров.

Поддерживаемые операционные системы:

* Linux (kernel >= 2.6.25)
* FreeBSD
* Windows
* Mac OS X

Доступна тестовая страница: <http://map.project-osrm.org/>.



Инструкция по сборке и установке есть здесь: https://github.com/Project-OSRM/osrm-backend/wiki/Building%20OSRM

Чтобы запустить сервер, ему необходимо подготовить данные, для построения графа. Чтобы загрузить их, необходимо их извлечь из карт OpenStreetMap. Делается это, входящими в комплект утилитами.

Но, сначала нужно скачать карту региона, для которого вы хотите строить маршруты.

Разработчики, предлагают использовать сервис GeoFabrik. На котором можно скачать карты для основных материков земли, отдельных стран и регионов в формате **osm** или pbf.

**OpenStreetMap** (дословно «*открытая карта улиц*»), сокращённо **OSM** — некоммерческий [веб-картографический](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) проект по созданию [силами сообщества](https://ru.wikipedia.org/wiki/User-generated_content) [участников-пользователей Интернета](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE) подробной [свободной](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D1%8B) и бесплатной географической карты мира.

Для создания карт используются данные с персональных [GPS-трекеров](https://ru.wikipedia.org/wiki/GPS-%D1%82%D1%80%D0%B5%D0%BA%D0%B5%D1%80), [аэрофотографии](https://ru.wikipedia.org/wiki/%D0%90%D1%8D%D1%80%D0%BE%D1%84%D0%BE%D1%82%D0%BE%D1%81%D1%8A%D1%91%D0%BC%D0%BA%D0%B0), видеозаписи, спутниковые снимкии [панорамы улиц](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BD%D0%BE%D1%80%D0%B0%D0%BC%D1%8B_%D1%83%D0%BB%D0%B8%D1%86), предоставленные некоторыми компаниями, а также знания человека, рисующего карту. Использование для создания карт сервисов, подобных [Google Maps](https://ru.wikipedia.org/wiki/Google_Maps), без разрешения правообладателя невозможно.

В OpenStreetMap при создании карты используется принцип [вики](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8). Каждый зарегистрированный пользователь может вносить изменения в карту.

Данные проекта распространяются на условиях [свободной лицензии](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F) [Open Database Licence](https://ru.wikipedia.org/wiki/Open_Database_Licence). До [12 сентября](https://ru.wikipedia.org/wiki/12_%D1%81%D0%B5%D0%BD%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [2012 года](https://ru.wikipedia.org/wiki/2012_%D0%B3%D0%BE%D0%B4) использовалось лицензия [Creative Commons Attribution Share-Alike 2.0](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%B8_Creative_Commons).

* Проект охватывает всю поверхность [земного шара](https://ru.wikipedia.org/wiki/%D0%97%D0%B5%D0%BC%D0%BB%D1%8F).
* Главной целью проекта является построение не собственно карты, а базы данных, содержащей сведения о точках на земной поверхности. Таким образом, на основе собранных в рамках проекта данных можно создавать карты различного вида и другие сервисы.
* Карты OpenStreetMap двумерные, без отображения высот над уровнем моря, [изолиний](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BE%D0%BB%D0%B8%D0%BD%D0%B8%D1%8F). Однако существуют проекты, которые отображают рельефные карты, используя данные о высотах из сторонних свободных источников.

OpenStreetMap использует топологическую [структуру данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), состоящую из объектов:

* **node** (точка) — точка с указанными координатами;
* **way** (линия) — упорядоченный список точек, составляющих линию или полигон;
* **relation** (отношение) — группы точек, линий и других отношений, которым назначаются некоторые свойства;
* **tag** (тег) — пары «ключ — значение», могут назначаться точкам, линиям и отношениям.

Вот пример OSM файла[9]:

**<?xml** version="1.0" encoding="UTF-8"**?>**

**<osm** version="0.6" generator="CGImap 0.0.2"**>**

**<bounds** minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"**/>**

**<node** id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"**/>**

**<node** id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744" visible="true" version="1" changeset="323878" timestamp="2008-05-03T13:39:23Z"**/>**

**<node** id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible="true" timestamp="2012-07-20T09:43:19Z"**>**

**<tag** k="name" v="Neu Broderstorf"**/>**

**<tag** k="traffic\_sign" v="city\_limit"**/>**

**</node>**

...

**<node** id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"**/>**

**<way** id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606" timestamp="2010-03-16T11:47:08Z"**>**

**<nd** ref="292403538"**/>**

**<nd** ref="298884289"**/>**

...

**<nd** ref="261728686"**/>**

**<tag** k="highway" v="unclassified"**/>**

**<tag** k="name" v="Pastower Straße"**/>**

**</way>**

**<relation** id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637" timestamp="2011-01-12T14:23:49Z"**>**

**<member** type="node" ref="294942404" role=""**/>**

...

**<member** type="node" ref="364933006" role=""**/>**

**<member** type="way" ref="4579143" role=""**/>**

...

**<member** type="node" ref="249673494" role=""**/>**

**<tag** k="name" v="Küstenbus Linie 123"**/>**

**<tag** k="network" v="VVW"**/>**

**<tag** k="operator" v="Regionalverkehr Küste"**/>**

**<tag** k="ref" v="123"**/>**

**<tag** k="route" v="bus"**/>**

**<tag** k="type" v="route"**/>**

**</relation>**

...

**</osm>**

Формат PBF ("Protocolbuffer Binary Format") предназначен в первую очередь для замены Формата XML. Файл всей планеты в формате PBF примерно в два раза меньше по размеру чем OSM XML, сжатый архиватором gzip и примерно на 30% меньше чем OSM XML, сжатый архиватором bzip2. По сравнению с OSM XML, сжатый gzip, запись файла формата PBF происходит примерно в 5 раз быстрее, а чтение в 6 раз быстрее. Формат PBF был разработан для поддержки лучшей расширяемости и гибкости в будущем.

Формат, лежащий в основе файла, поддерживает доступ к произвольному блоку файлу. Каждый блок файла декодируется независимо и содержит ~8k объектов OSM в стандартной конфигурации. Теги элементов не кодируются; все ключи и значения хранятся в неизменном виде как строки. Для масштабируемости в будущем, допускаются 64-битные идентификаторы точек, линий и отношений. Текущий сериализатор (RU:Osmosis) сохраняет порядок элементов OSM, и их тегов. Чтобы гибко работать с разными разрешениями, разрешение (детализация) используемая для записи местоположения и временных меток регулируется до 1 милисекунды и 1 наноградуса. По умолчанию используется разрешение 1000 милисекунд и 100 наноградусов, соответствующих приблизительно 1 см в районе экватора. Такая детализация сейчас используется в базе OSM.[10]

Скачиваем файл с данными:

wget <http://download.geofabric.de/asia/map.osm.pbf>

С помощью утилиты osrm-extract, данные извлекаются из карты и сохраняются для дальнейшей загрузки. Перед тем как извлекать данные, необходимо указать файл профиля (profile.lua), в котором описано какие именно маршруты необходимы. Все профили лежат в папке profiles в корневой папке с исходниками и подключаются созданием ссылки на нужный файл.

*ln -s ../profiles/bicycle.lua profile.lua*

Как видите, мы указали профиль для велосипедистов. Но, там есть профили и для автомобильного транспорта и для пешеходов.

Так же, может потребоваться указать ссылку на папку с дополнительными библиотеками

*ln -s ../profiles/lib/*

После чего, запускаем саму утилиту.

*./osrm-extract map.osm*

И вот, у нас получился файл с расширением osrm. Но, нужно воспользоваться еще одной утилитой из комплекта — osrm-prepare.

Она создает несколько файлов: hsgr — иерархию маршрутов, для нахождения наиболее подходящих, nodes — граф маршрутов, и файлы с индексами.

*./osrm-prepare map.osrm*

Перед запуском, в директории с сервером, файл инициализации server.ini. В нем мы указываем какие файлы и откуда брать, и несколько параметров запуска.

*Threads = 4*

*IP = 0.0.0.0*

*Port = 5000*

*hsgrData=./map.osrm.hsgr*

*nodesData=./map.osrm.nodes*

*edgesData=./map.osrm.edges*

*ramIndex=./map.osrm.ramIndex*

*fileIndex=./map.osrm.fileIndex*

*namesData=./map.osrm.names*

Запуск сервера:

*./osrm-routed*

Запрос маршрутов

Запрос маршрута между двумя точками, заданными координатами выглядит так:

*http://server:5000/viaroute?loc=lat1,lon1&loc=lat2,lon2*

Для получения маршрута, проходящего через заданные точки (текущее ограничение максимум 25 точек) нужно сделать такой запрос:

[*http://server:5000/viaroute?loc=lat1,lon1&loc=lat2,lon2&loc=lat3,lon3&loc=lat4,lon4&...&latX,lonX*](http://server:5000/viaroute?loc=lat1,lon1&loc=lat2,lon2&loc=lat3,lon3&loc=lat4,lon4&...&latX,lonX)

Ответ сервер возвращает в JSON формате. Координаты точек маршрута закодированы в текстовой строке, в параметре *route\_geometry*.

Для декодирования этой строки разработчики предоставляют модуль на JavaScript - *OSRM.RoutingGeometry.js*.

**2.2. SpatiaLite — расширение SQLite для пространственных**

**данных**

Предоставляет функциональность для работы с геоданными, как PostGIS или Oracle Spatial. Но в отличии от последних не использует клиент-серверную архитектуру, то есть все операции выполняет само приложение. Придерживаясь архитектуры SQLite, база данных представляет собой один файл, в котором хранятся все данные (исключением является не монопольный режим, когда ещё создается файл журнала), который может быть свободно скопирован, или удален.

SpatiaLite имеет свою собственную реализацию индексов (R-деревья) и геометрических типов, которые позволяют делать пространственные запросы.

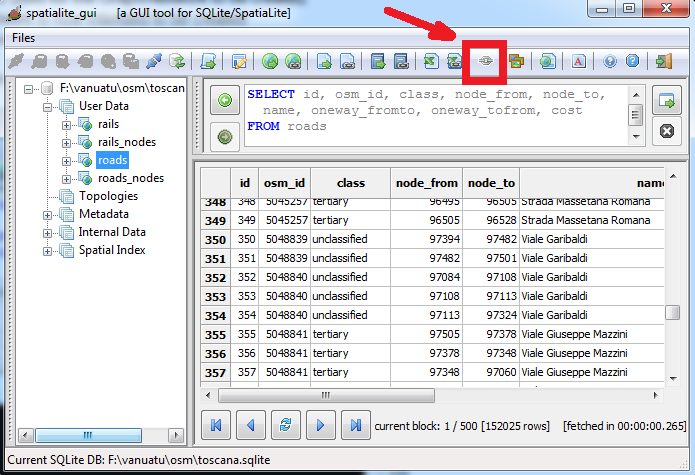
SpatiaLite, доступен для Linux и Windows, в виде библиотеки. В комплекте идут несколько важных утилит, включая GUI для манипулирования SpatiaLite. [11]

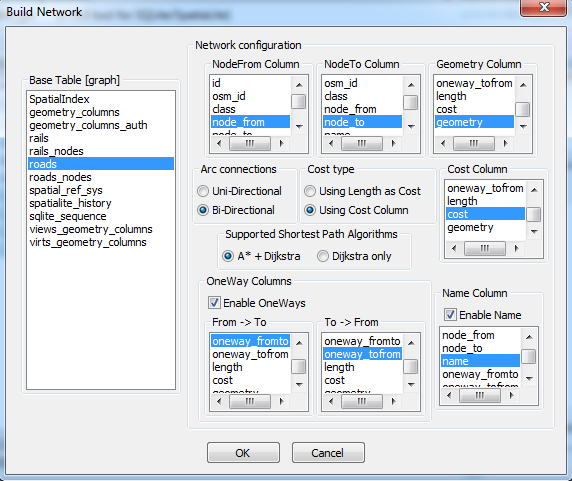
Spatialite включает в себя программу импорта дорожной сети в формате файла OSM, после чего позволяет построить на базе имеющейся информации граф дорожной сети.

После скачивания OSM-файла с данными, необходимо в командной строке указать:

*spatialite\_osm\_net -o map.osm -d map.sqlite -T roads -m*

На выходе мы получим базу данных *map.sqlite* с информацией, размещающейся в таблице roads.

 Теперь эти данные необходимо преобразовать в граф дорожной сети. Для этого, имеется графическая утилита spatialite-gui, с помощью которой необходимо запустить построитель графа:



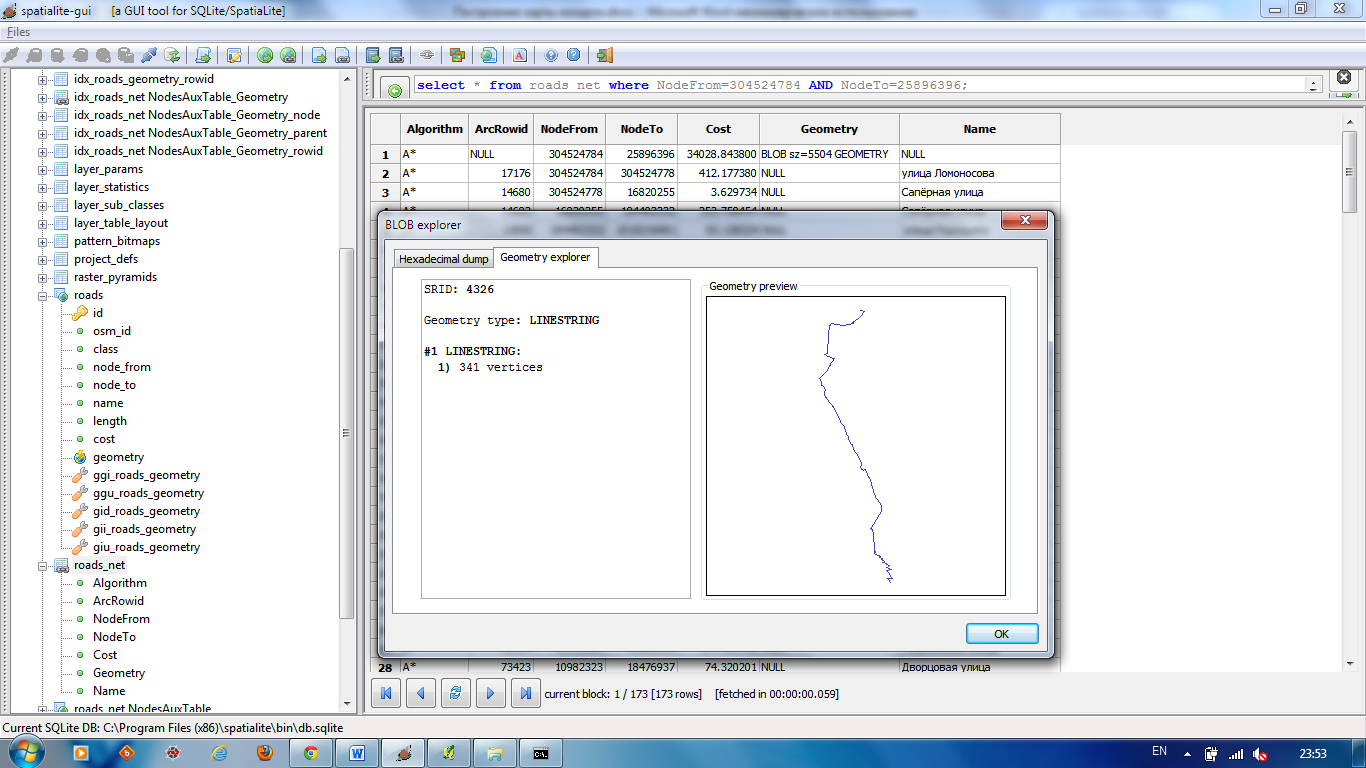
После запуска построителя графа дорожной сети, в базе появится представление roads\_net, к которому можно делать запросы для определения кратчайшего маршрута от узла A к узлу B. Поддерживаются два алгоритма поиска пути в графе: алгоритм Дейкстры и «А-star». Между ними можно переключаться, с помощью запроса:

*UPDATE roads\_net SET algorithm ='Dijkstra или A\*';*

Простейший запрос выглядит следующим образом:

*SELECT \* FROM roads\_net WHERE NodeFrom=304524784 AND NodeTo=25896396;*

Результатом будет выборка, в первой строке которой – сводка по маршруту, содержащая «полилинию» с сегментами маршрута. Все остальные строки – это отдельные участки найденного кратчайшего маршрута, т.е. дуги дорожного графа. Поле cost содержит информацию о длине участков маршрута в метрах.



При создании базы с графом дорожной сети можно задавать дополнительные параметры в файле конфигурации, например:

*spatialite\_osm\_net -o map.osm -d map.sqlite --roads -T roads -tf road\_template.conf*

Для построения графа дорожной сети можно использовать вместо графической утилиты утилиту командной строки *spatialite\_network*

примерно так:

*spatialite\_network -d map.sqlite -T roads -f node\_from -t node\_to -g geometry --oneway-tofrom oneway\_tofrom --oneway-fromto oneway\_fromto -n name -o roads\_net\_data --overwrite-output*

Будет создана таблица с двоичными данными *'roads\_net\_data'*, далее, для создания представления *roads\_net* нужно дать команду:

*spatialite sqlite map.sqlite "CREATE VIRTUAL TABLE roads\_net USING VirtualNetwork('roads\_net\_data')"* .

Получить массив точек маршрута можно следующим запросом:

SELECT AsGeoJSON(geometry) AS geometry FROM roads\_net WHERE

NodeFrom=18283 AND NodeTo=29394

**2.3. GTOPO30 - глобальная цифровая модель рельефа**

GTOPO30 является глобальной цифровой моделью рельефа (ЦМР) с горизонтальным шагом сетки 30 угловых секунд (примерно 1 км). GTOPO30 была получена из нескольких растровых и векторных источников топографической информации.

Данные по рельефу поверхности Земли можно скачать с сайта Национального Геофизического Центра Данных США (National Geophysical Data Center - NGDC) <http://www.ngdc.noaa.gov/mgg/global/global.html>. [13].

Данные доступны в различных форматах, включая формат **xyz**, это формат такого вида: долгота<пробел> широта<пробел> высота<конец строки>

-180 90 -4228

-179.983 90 -4228

-179.967 90 -4228

-179.95 90 -4228

-179.933 90 -4228

-179.917 90 -4228

-179.9 90 -4228

-179.883 90 -4228

-179.867 90 -4228

-179.85 90 -4228

-179.833 90 -4228

-179.817 90 -4228

-179.8 90 -4228

-179.783 90 -4228

-179.767 90 -4228

-179.75 90 -4228

-179.733 90 -4228

Далее эти данные возможно считать в базу данных, например sqlite, и делать к ней запросы для получения высоты точки с заданными координатами. Так как данные дискретны, то потребуется находить точку в базе данных, ближайшую к заданной точке. Значения высоты представлены целыми числами. Для использования в игре этого достаточно. На основе такой базы данных возможно построение локального сервиса высотных данных.

**2.4. Leaflet**

Для построении клиентской части игры работы с картами была выбрана JavaScript библиотека Leaflet. (<http://leafletjs.com/>), разработанная командой разработчиков во главе с Владимиром Агафонкиным.

Библиотека имеет богатый функционал, простой синтаксис, высокую производительность.Она работает эффективно на всех основных настольных и мобильных платформах используя HTML5 и CSS3 на современных браузерах, но в то же время способна работать на старых. Она может быть расширена с помощью огромного количества плагинов, имеет красивый, простой в использовании и хорошо документированный API. [14]

Примеры работы с библиотекой.

Подключение библиотеки:

*<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css"/>*

*<script src="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js"></script>*

Нужно создать блок-контейнер, в котором будет размещаться карта:

*<div id="map"></div>*

Контейнер должен иметь определенную высоту, например, определенную в CSS файле:

*#map { height: 180px; }*

Создание объекта карты с заданным центром и масштабом:

*var map = L.map('map').setView([51.505, -0.09], 13);*

Создание слоя с тайлами (картинками из которых строится карта ):

*L.tileLayer('http://{s}.tiles.mapbox.com/v3/MapID/{z}/{x}/{y}.png', {*

*attribution: 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery © <a href="http://mapbox.com">Mapbox</a>[…]',*

*maxZoom: 18*

*}).addTo(map);*

Создаем маркер и добавляем его на карту:

*var marker = L.marker([51.5, -0.09]).addTo(map);*

Создаем круг и добавляем его на карту:

*var circle = L.circle([51.508, -0.11], 500, {*

*color: 'red',*

*fillColor: '#f03',*

*fillOpacity: 0.5*

*}).addTo(map);*

Создаем полигон и добавляем его на карту:

*var polygon = L.polygon([*

*[51.509, -0.08],*

*[51.503, -0.06],*

*[51.51, -0.047]*

*]).addTo(map);*

Задаем обработчик события клика мышью на карте:

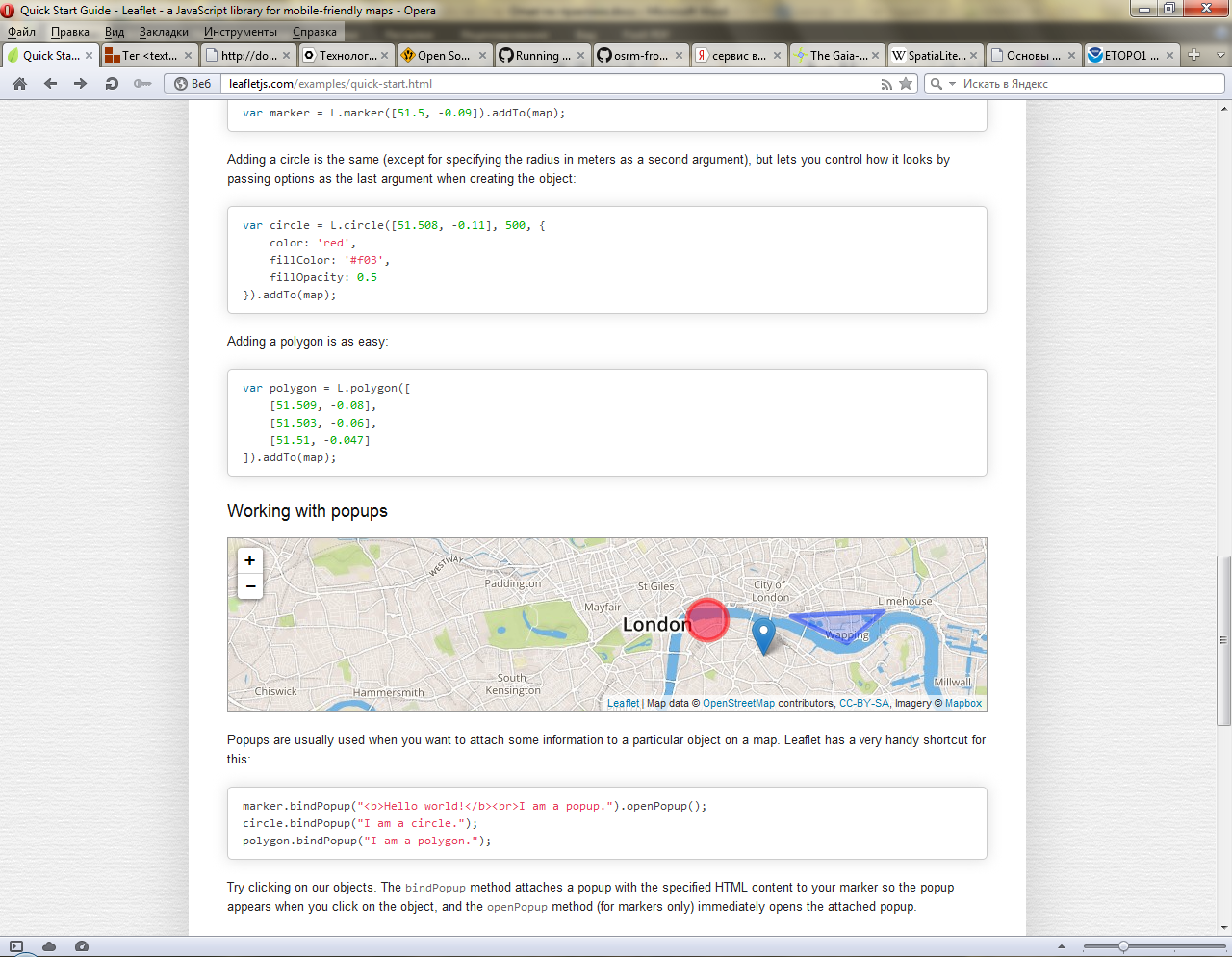
*function onMapClick(e) {*

*alert("You clicked the map at " + e.latlng);*

*}*

*map.on('click', onMapClick);*

Результат показан на рисунке.



Библиотека позволяет динамически добавлять и удалять объекты на карте, менять из свойства, перемещать и т.д. Подробнее с API можно ознакомиться тут: <http://leafletjs.com/reference.html>

ЛИТЕРАТУРА

1. List of multiplayer browser games, Wikipedia (2015), <http://en.wikipedia.org/wiki/List_of_multiplayer_browser_games>.

2. Goncalo Duarte Garcia Pereira (2009), Adaptative Map Generation For Turn-based Strategic Multiplayer Browser Games.

3. Сидоркина Т.В. Создание прототипа многопользовательской игры с использованием API Google Maps.(2014).

4. Ограничения на использование высотных данных Google

<https://developers.google.com/maps/documentation/elevation/?hl=ru#Limits>

5. <http://www.gamer.ru/monopoly-city-streets/monopoly-city-streets-globalnaya-onlayn-igra-na-osnove-google-maps>

6. <https://developers.google.com/maps/documentation/javascript/directions?hl=ru#Directions>

7. <https://tech.yandex.ru/maps/jsapi/>

8. <http://project-osrm.org/>

9. <https://ru.wikipedia.org/wiki/OpenStreetMap>

10. <http://wiki.openstreetmap.org/wiki/RU:PBF_Format>

11. <https://ru.wikipedia.org/wiki/SpatiaLite>

12. <https://www.gaia-gis.it/fossil/spatialite-tools/wiki?name=spatialite_osm_net>

13. <http://www.ngdc.noaa.gov/mgg/global/global.html>

14. <http://leafletjs.com>