Accessibility of maps for the blind on mobile platforms

Daniel Dalton

June 2015

Contents

Abstract

Maps or mobile applications that present maps on screen are not usable by the blind without modification. Although many applications exist to aide with navigation and mobility, few options exist for understanding spatial and relative layout of a map. The current strategies that are used to present map data to the blind will be explored along with their benefits and limitations. A mobile solution utilising a web service and IOS application will be proposed to provide access to maps for the blind in the hope of overcoming some of the current limitations.

1 Introduction

Traditionally, it has been difficult for a vision impaired person to access and interpret a map of a particular area. There are limited options currently available to access this type of spatial information and those options currently available are not always practical or realistic. Although over the last decade or so the increased availability of navigation apps for mobile platforms has assisted with this problem there are still many limitations in conveying spatial data efficiently and effectively to a blind user.

The second complication of this problem is that a standard map aimed for a visual user contains an overwhelming amount of data, which is difficult to convey to a sight impaired user relying exclusively on their finger tips or audio for such data. In addition sight impaired users may require greater detail about their surroundings to successfully navigate the region [1].

This project seeks to provide a mechanism for a blind or vision impaired person to access a map of an area in a practical and timely manner gaining the essential information they need to safely and confidently navigate. This project relates to the retrieval of spatial information and human computer interaction. Ultimately, the project will investigate the most appropriate way to provide this spatial information for a given area to a blind or vision impaired person utilising a mobile platform. This is a conceptual and practical work.

1.1 Goals and Objectives

The aim of this project is to build a web service which has the ability to produce a SVG graphic of a particular area. The image should render streets, buildings and other places of interest of a neighbourhood. The image can then be parsed by the GraVVITAS software for ipad developed by Dr Cagatay Goncu. This will facilitate a blind or vision impaired user to explore the graphical layout of the neighbourhood rendered by the proposed web service. The produced SVG will contain the data required to meet the requirements of this solution.

This research will investigate the following two areas:

1. Determine the most effective way to filter information from a map to a form which can be comprehended by someone without the ability to view it visually.

2. Investigating navigation techniques to navigate the presented information.

Visual maps contain a considerable amount of information. Not all of this is relevant, and it will need to be decided what is crucial information and what is not. In addition useful information should be presented on the produced map such as buildings within a university campus for example.

Conveying the layout of the setting such as where buildings in a university campus are relative to each other is the second aspect in this research. Devising practical and efficient strategies for a blind person to explore the map is critical to its success. The blind user can not be expected to scan the map every time they are looking for something in its entirety as this is time consuming and tedious.

Ultimately, it is hoped that the final product of the research will allow a caller to pass a geographical location to the web service with defined bounds and to have a SVG result returned. The SVG result will be suitable for access by a blind person on ipad in an effective and efficient manner. It is hoped that this can help vision impaired users reliably understand their surroundings around them before they set off on a trip. Secondly, it is hoped that this can be adapted to a smart phone device such as Iphone so that the project can be taken advantage of by a blind person while moving around in society although this is likely a future addition to the project.

2 Literature Review

2.1 Current Solutions

2.1.1 Tactile Graphics

Historically, the most popular form of conveying diagrams to the vision impaired including maps was by means of a tactile graphic, utilising "swell paper" and "thirmoform diagrams" [2]. Roads, buildings and other points of interest are displayed tactically with supporting Braille labels in some cases so that readers can identify what each raised point or line represents.

2.1.2 Virtual Acoustic Map

Another alternative is the "Virtual Acoustic Map", which associates objects on the map with different audible sounds and uses these sounds to convey the map data to the blind user. Another implementation is speech output of the layout of streets including their direction and name [3].

2.1.3 Virtual Tactile Maps

Virtual tactile maps are another approach to communicate map data to a site impaired user. This type of presentation utilises tactile displays, or other haptic devices such as "joysticks" and "haptic mice" [4]. These devices are used to communicate the input and output of the map with the user. In addition acoustic information is sometimes used for greater detail.

2.1.4 Mobile solutions

With the innovation of mobile devices came innovative mechanisms to present map data to a sight impaired user. Several approaches of this nature were explored and presented in [5]. Edge projection is the first technique. This technique relies on projecting map objects to two sides of the touch screen. The user can explore the map by moving both hands simultaneously to locate the x and y projections. Having discovered these the user can then move their hands inwards towards the point of interest on the map. Neighbourhood browsing is another technique used by mobile applications. The application calculates area that is not occupied around each object and reclaims that area. The result is when touching the screen the closest object is spoken to the user. A gesture can exist to provide directions. Touch and speak is the final approach utilising touch input and voice commands to overcome the inefficiency of exploring the map.

Various mobile app solutions were explored first hand on the IOS and Android platforms. They will be summarised here.

Apple's maps application facilitates exploring a map and gaining spatial insight among other features. When Apple maps is presenting a map of some location the user may explore with one finger to hear names of streets, towns or larger areas such as cities and states (depending on the zoom factor). If the map is showing streets the user can hold their finger on a street to trace it. The user may also use standard voice over flicks to flick between objects on the map eg. cities for a map of Australia. However, such a technique will give the user very little spatial layout information. The user can control the map easily. Scrolling the map left/right or up/down is straight forward. Zooming is also very easy, but it seems hard to make the map zoom in on a particular area such as Sydney for a map of Australia. Double tapping the city to zoom in on and then zooming in seems to work sometimes. As the zoom changes or the map is shifted voice over will announce a quick summary of the map such as what is on screen. Works OK, but sometimes overwhelming. Apple maps was tested with a map of Australia. Exploring with one finger was painful and time consuming as you must touch a specific area of the map to hear the name of a place spoken. In other words there is a lot of empty space on the map where the user is given no feedback as they move their finger. Moving around the map is not particularly efficient.

Blind square is a very popular app among the vision impaired. The description here is based off the apps description in the app store as ‘the app costs $37 and could not be tested first hand [6]. Blind square claims to be the most popular gps app for the blind. “When Blind Square has determined your location using your iOS-devices GPS capabilities, it will look up information about your surroundings on FourSquare and Open Street Map. Employing unique algorithms, it will then ascertain the information most useful to you and speak it in a clear synthetic voice” [6]. Blind square will announce places of interest and your location as you move. If you set a destination it will also indicate the clock direction of the place from you and the distance it is from you has you move towards it. Blind square facilitates marking your current location so that you can easily find this later on. It appears that blind square is very good at speaking important information back to the user, but does not provide a way for the blind user to explore the area in a tactile fashion and gain a spatial layout understanding.

Ariadne gps is an app available for ipad and Iphone. It has a variety of navigational and mapping features. For the scope of this project the mapping functionality will be described. The app has two main features of interest, explore an area and look around. Both these modes present a map to the user. Look around will position the map to the location of the device in the direction that the user is facing. When the map is shifted the user will hear which direction it is shifted relative to the way they are facing eg. “forwards”, “backwards”, “left” or “right”. The explore area allows the map to be positioned at some fixed address. Moving the map around says east/west/north/south rather than left/right etc. Both modes have exactly the same functionality. Moving the map around in the four compass directions. After each motion the app will speak what area the map is centred around and the radius size. Zooming in and zooming out works with a pinch. The map will update and speak to the user eg. showing localities and cities or streets and numbers. Otherwise, the map allows the user to explore the neighbourhood by tracing their finger around the screen and street names and intersections get announced along with places of interest eg. Monash university Clayton campus and street numbers. The user can locate a point on the screen and have the map re-calibrated and centred around that point.

Intersection explorer is an android application by Google designed for the vision impaired. It allows a user to navigate a neighbourhood with their finger to explore streets and intersections to gain a spatial layout of streets. The user moves their finger around on the screen in a direction until they hear that their is a street in the given direction that they would like to follow. Releasing the finger will move down the street to the next intersection. This pattern is repeated so that the user can understand how streets are set out and intersect each other. Unfortunately the app is only good for streets and intersections. It does not announce any street numbers or places. It does have some problems as many streets are spoken as unnamed road.

2.2 Limitations of existing work

Although the discussed solutions do a reasonable job at making maps accessible to the sight impaired they do have some substantial limitations. The manual techniques such as tactile graphics are very resource intensive to produce. They require work on the part of a sighted transcriber for every map that a blind user may wish to explore. In addition this is an expensive process. The tactile maps are not overly portable and can not be used while walking for instance.

The scope of this project is to develop a mobile solution to this problem. The rest of the limitations will discuss those faced by current apps on the market. As indicated apps such as apple maps are inefficient to navigate and for instance do not speak when the user touches empty space. The problem is that the map may potentially contain a vast amount of empty space. Other apps such as intersection explorer do not show places or points of interest on the map which is valuable information for a blind user.

Ariadne GPS is perhaps the best solution currently available. It has a variety of features, but also some important limitations. Navigating the map is still done by exploring with the finger tip which is relatively slow and it only shows some places. It does not necessarily provide all of the details that the user may require.

Although there are many other navigational apps designed for the vision impaired these are designed primarily for guided directions or finding places near by. They do not make any effort to present any form of map or relative spatial layout information.

2.3 Opportunities

Presently, there is no solution which depicts buildings and their shapes as well as entrances for example. In addition there is no way to determine the relative spatial layout between two objects on the map aside from tracing with the finger tip which is a fundamental point of this research. However, perhaps most importantly filtering is quite limited on all solutions. If the blind user is only interested in cafes they will have a hard time locating these in among all of the other points of interest on the map. This area will also be addressed by the proposed solution.

3 Requirements

The project is broken into two components the user interface and web service. The user interface relies on the web service to present the map experience to the sight impaired user.

3.1 User interface

3.1.1 Opening the map

Name: Open map.
Identifier: UC1
Description:
Load a map of a given area of the user's choosing.
Preconditions:
The application has access to the web service.
Postconditions:
A map of the chosen area is presented provided the chosen location was valid.

Basic course of action

User	System
1. User selects "Use current location" or "Enter address manually" [Alt A]	
	2. System presents the map centred around the desired region with default zoom factors and radius.

Alternate course A

User	**System**
A1. The user selects “enter address manually”.	
	A2. System displays text field for address input.
A3. User types address to centre map around.	A4. System populates auto-completion area of potential addresses.
A5. User makes the selection of the address. The use case now continues at step 2 in the basic course.	

3.1.2 Filtering points of interest

Name: Points of interest
Identifier: UC2
Description:
Allow the user to pick points of interest they would like to see on the map in addition to the basic features such as streets. This filtering is done by displaying possible categories to filter. There shall be a wide range of categories to select from for example “cafe”.
Preconditions:
A map of a given area is loaded and being presented.
Postconditions:
The presented map now reflects the changes in the category filters.

Basic course of action:

User	**System**
1. User performs a two finger twist (roter gesture).	
	2. System announces “Points of interest”.
3. User flicks up or down until desired category is heard.	
4. Double tap anywhere on screen to activate selected category.	
	5. System updates map to show points matching the newly selected category. [Alternate a].

Alternate course A

User	System
	A5. This category is already being displayed on the map the result is to deactivate it and map no longer shows places of this type.
	A6. Use case ends.

3.1.3 Showing a path between two points on the map

Name: Depict a path between two distinct points
Identifier: UC3
Description:
A user has two distinct points on the map they wish to find a path between following roads and other pathways. This is not necessarily trivial for a blind user to achieve.
Preconditions:
There is a map of an area being displayed on screen.
The map has at least two distinct points.
Postconditions:
Map is updated to play a unique tone whenever the newly created pathway is touched or approached.
No change to map if a path could not be found.
Basic course of action:

User	System
1. User Locates and double taps the first point.	
2. The user locates and double taps a second (distinct) point on the map (point B).	
	3. System calculates and presents a path between the points on the map. An overview of this path is spoken. back to the user. [Alternate A]
4. User explores map with a single finger.	5. When the user touches the path the pathway tone is played back to the user.

Alternate course A

User	System
	A3. System could not find a path between the two points, error message is spoken.
	A4. Use case ends

Name: Remove paths

Identifier: UC4
Description:
Clear the map of any paths if any have been created by UC3.
Preconditions:
Map of an area is currently being shown.
At least one pathway is depicted on the map otherwise it has no effect.
Postconditions:
Remove all pathways from the map.
Basic course of action:

User	**System**
1. Two finger twist action (roter gesture).	
	2. System announces “clear paths”
3. User double taps anywhere on screen.	
	4. System clears map of paths and speaks “paths cleared”.

3.1.4 Finding points on the map

Name: Locating points on the map
Identifier: UC5
Description:
It is time consuming for the user to explore the entire screen looking for specific features on the map. Allow the user to hear the features contained within a certain sub-area of the map. **Preconditions:**
There is a map of an area being displayed on screen.
Postconditions:
No change to map.
Basic course of action:

User	**System**
1. User touches and holds their finger on a single point of the screen.	
2. User double taps on a point some distance away from the initial point they are still holding.	
3. The user releases the initial finger.	
	4. The system creates a circle around the initial point with radius equal to the distance between the initial point and the second point and announces all points of interest and streets within the circle.

3.2 Web Service

The web service is invoked by means of HTTP parameters by means of a HTTP request. The following use cases will be necessary for the web service to fulfill its requirements to the UI.

3.2.1 Request an SVG map

Name: Produce SVG map
Identifier: UC6
Description:
Construct an SVG map of an area of certain bounds with data from one or more third party services. Return the resulting data in the form of an SVG image.
Preconditions:
Third party services are available
Request is a correctly formatted HTTP request.
Request contains at a minimum radius, latitude and longitude parameters.
Postconditions:
None

Basic course of action:

Client	**System hosting service**
1. Client provides a HTTP request with parameters radius, latitude, longitude, filter, paths.	
	2. System contacts open maps and additional third party services to acquire data matching filter list. [Alternate A], [Alternate B], [Alternate C], [Alternate D]
	3. System combines all retrieved data into a single accessible SVG.
	4. System returns the svg.
5. Client receives the svg image.	

Alternate course A:

Client	**System hosting service**
	A2. Radius, latitude or longitude of the centre point is missing, return error code to client.
A3. Client receives error code.	
A4. Use case terminates.	

Alternate course B:

Client	**System hosting service**
	A2. One of the third party services is down return an error code indicating this to the client.
A3. Client receives error code that external services are down.	
A4. Use case terminates.	

Alternate course C:

Client	**System hosting service**
	A2. Filter parameter contains a value such as “cafe” or “restaurant”, system constructs a suitable query.
	A3. The query is sent to the appropriate service.
	A4. System receives the place data back and converts it to a standardised form.
	A5. Use case continues at step 3.

Alternate course D:

Client	**System hosting service**
	A2. System creates objects to be placed on the SVG representing a path between two unique points within the SVG.
	A3. Use case continues at 3.

4 Design

4.1 User Interface Design

As specified within the requirements there are four key features of this application: loading of the map around a given area which includes features within a specified radius, the filtering of map features, marking a pathway/route between two points on the map and enhancing the exploration of the map from the point of view of a sight impaired user. The design of each of these four aspects will be discussed.

4.1.1 Opening the map

When the application is first invoked it is logical to display a map to the user of the current location. This is done by using the current location as determined by the device’s location services. The user can then begin exploring the map of where they currently are.

If the user wishes to look around another area they may load the map of this area by supplying the address. To do this the user will make use of the rota or two finger twist method. They will repeat this until they hear “search”. A double tap with a single finger will present the search screen.

The search screen is now active. This screen will consist of an input text field at the top labeled “Address”. A search button will be located to the right of this. When the user is within the text field the system keyboard will be active and they may begin typing a location. As the location is being typed the potential candidate addresses appear below the text field. The user then either presses the search button or selects one of the suggested addresses.

The map may fail to load due to either an invalid address being supplied or other area. If this is the case a standard error dialog should be displayed informing the user of such an event. The user can dismiss this dialog returning them to the search screen again.

If the result is successful the map screen which appeared initially when the application was launched will re-appear, but this time for a map of the selected location.

The two finger twist can be used to cycle to the “radius” option. Flicking up will decrease the radius of the map while flicking down will increase the radius of the map. When the user finds a radius they are happy with they will double tap to refresh the map with the new radius. The radius specifies the points that shall be included within the map. For example a radius of 500 m will include only features that are within 500 metres of the centre point that the map is centred around.

The map screen may be explored with a single finger. The feedback provided is controlled by the graViewer software. As the finger moves objects under the finger are announced to the user. These objects include roads and points of interest. Audio tones will be played for additional audible feedback. In addition the vibration finger sensors may be used for haptic feedback when exploring the map.

4.1.2 Filtering points of interest

A key consideration for the project is how to best filter the map so that it can be navigated in a practical manner. The user will have the liberty to choose what appears on the map so they can choose what fits within their user capabilities.

Assuming the map screen is active showing a map of an area the user may use the conventional two finger twist voice over command until they hear “points of interest”. The user will then flick up and down with one finger to cycle between the potential options such as streets, cafes, restaurants, fast food, buildings, public transport etc. The user can double tap on any of these options. If the option was not previously selected the user will hear "checked" and the map will be updated to include features matching that parameter. eg. if cafes was now checked the map would be populated with all cafes within the mapped region. If the item was already selected eg. cafes a single finger double tap would announce "unchecked", and the cafes on the map would disappear. Note that the state of any of the options in this list (cafes, streets etc.) should have their state announced after the category name either checked or unchecked.

The user will have the freedom to select or unselect whatever items they would prefer. The map will be initialised with streets shown only.

4.1.3 Showing a path between two points on the map

A very important use case is traversing from a starting point to a destination point.

Assuming the map screen is being displayed showing a map of a region. The user will double tap any area within the bounds of the map. That is the point must actually reside on the map such as a street or point of interest. The user will then move their second finger around the map locating the destination point. The user will then double tap on this second point. If an error occurs the system will announce this otherwise a pathway/route between the two points will be plotted on the map.

The pathway will be in a different colour to other features on the map. As a result when the path is touched the street name will be announced, but a unique tone specific to the path will be played. This will reassure the user that they are indeed following the path.

Between stages of the path eg. where the route twists and turns the relevant directional instructions will be spoken. For instance if the user is traversing along the route over street a, as the route is about to turn the system will announce "route turns left onto street b 400 metres". This will be very helpful to not only give the user directional instructions, for real world navigation, but it will make following the route much easier on the screen of the ipad. Less time is wasted trying to constantly locate the route.

The starting points and end points of the route will also be marked accordingly.

The rota action (two finger twist) can be used to navigate to "clear paths". When this is double tapped the pathways will be removed as will the directional points discussed above. The map will behave normally again.

4.1.4 Finding points on the map

A challenging task for a blind user is to find a certain point of interest on a map that is potentially quite small hidden within other features. If the user has some idea where this is they can place their first finger roughly where they think this is. Next they will move their second finger some distance away from the initial point and double tap. The initial finger can then be released. The circle can be as large or as small as the user desires. A circle with a radius equal to the distance between the initial point and the second point touched will be created surrounding the initial point. It does not matter if the circle extends passed the edge of the map on a given side as only points up until the edge of the map on this side will be included. In the other directions which are not restricted by the map dimensions i.e. where traversing from the initial point outwards by the specified distance does not touch a map edge no limitations will exist (points within the radius of the circle will be included). All features that lie within this circle will be announced to the user including their direction from the centre point. The centre point is the point where the initial finger was first placed. For example “building 1 12 o’clock”. The features will be announced in terms of distance. That is those closest to the centre are spoken first followed by those further out. If objects overlap i.e. exist both within and outside of the theoretical circle they will be included in the spoken list of places and roads. The speech can be interrupted at any time by touching another point on the map.

4.2 Web Service Design

4.2.1 Class overview

The following classes will be created to implement the web service with the above features.

- Map: A class which represents a map on a grid with specified dimensions. The map object holds all points that will appear on the map.
  - MapPoint: Represents a single point on the map eg. a circle.
  - MapPolyShape: Represents either a polyline or polygon on the map. The PolyShape must refer to at least two coordinates for the polyline and at least 3 for the polygon.
- Utilities: General utility methods. All of these methods are static.
- Main: Main functions to make the program run.
- OpenStreet: Responsible for retrieving and adding geographical data from the OSM database to the map.
- Route: Responsible for finding a route between two points (using the here maps api) and adding this to the Map object.
- Location: Represents a geographical location on the earth.
- Parameters: Represents the parameters for an open street map request.

4.2.2 Map

The Map class consists of the following public constructors:

- Map(Location centre, int radius): Construct a Map around centre with a specific radius.

The Map class has the following public methods:

- void addPoint(String id, double lat, double lng, String name, String description): Add a location on the earth to this map instance with a given name and unique id.
- void addLine (String id, double[][] coordinates, String name): Add a polyline to the map which is a location with the name specified and a unique id. The polyline passes through the geographical coordinates specified.
- void addPolygon (String id, double[][] coordinates, String name): Add a polygon to the map. It passes through the specified geographical coordinates and has the name and id specified.

The Map has the following members:

- int height: Height of the map in svg pixels.
- int width: Width of the map in SVG pixels
- Location centre: A Location object that represents the geographical point the map is centred around.
- MapPoint[] mapPoints: Array of MapPoints on the map.
- PolyShape[] mapLines: Array of polylines on the Map.
- MapPolyShape[] mapPolygons: Array of polygons (buildings) on the map.
- MapPoint mapCentre: Represents the point centre on the map i.e. as a mapPoint.

MapPoint subclasses Map and has the following public constructor:

- MapPoint (Location point, double distance, double angle, Location MapPoint centre): Construct a point on a map for a given location (point). Calculates the position of the x and y values of this point on the actual map based on the MapPoint of the centre.
- MapPoint (double x, double y): Set a MapPoint with x and y values.

Members of MapPoint:

- double x: x coordinate on the map for this point.
- double y: y coordinate on the map for this point.
- String name: The name of this point.
- String description: The extra description for this point.
- String id: The unique id of this point.

The MapPolyShape class subclasses Map. The MapPolyShape class has the following public constructors:

- MapPolyShape (String id, double[][] coordinates, String name): Create a PolyShape on the map for the given coordinates.

Fields of the MapPolyShape class:

- String id: unique Id of this polyshape.
- String name: The name of this polyshape.
- double[] coordinates: The map coordinates for this shape on the map.

4.2.3 Utilities

The Utilities class has the following public methods they are all static:

- double distance (Location location1, Location location2): Returns the distance between location1 and location2
- double toRadians(double deg): Return deg in radians.
- double initialBearing(Location location1, Location location2): Return the initial bearing from location1 to location2.
- toSVG(Map map): Return a string of the svg image that is created from map.

4.2.4 Main

The Main class has the following methods:

- static bool handleCommandLineArguments(String[] argv): Check to see if command line arguments are valid and set up request parameters.
- static void run(): Runs the application with the given parameters set by handleCommandLineArguments and coordinates the requests to third party apis and the writing of the svg output.

The Main class has the following fields:

- Map map: The map for this session
- Parameters parameters: parameters for this request.
- String outputFile: The path to the outputFile for the SVG image.
- Location[] points: The points involved in any path for this map.
- Location centre: Centre of this map.
- int radius: Radius of area the map is for.

4.2.5 OpenStreet

The OpenStreet class has the following public static method:

- void readFromOSM (Map map, Function(Map map), Parameters parameters): Add data to the map from open street and invoke the callback once finished.

4.2.6 Route

The Route class consists of the following public static methods:

- void readRoute(Location[] points, Map map, function(Map map)): Add a route between two points to the map and invoke the callback once finished.
- void readRouteFromFile(String file, Map map, function (Map map)): Add the route specified in file to the map and then invoke the callback.

4.2.7 Location

The Location class consists of the following public constructors:

- Location (double lat, double lng): Initialise a location with the specified lat and lng coordinates.
- Location (double lat, double lng, String id, String name, String description): Initialise a Location with the specified lat lng, name, id and description.

The Location class has the following fields:

- String name: Name of this location.
- String description: Extra description about this location.

- String id: Unique id for this location.
- double lat: Latitude of this location.
- double lng: Longitude for this location.

4.2.8 Parameters

The Parameters class consists of the following public constructors:

- Parameters(): Initialise empty parameters object.

The Parameters class has getters and setters for each individual parameter.

4.2.9 Class relationships

The following relationships exist between the classes.

- Map has many MapPoint
- Map has many MapPolyShape
- Map has one Location.
- Map depends on Utilities
- Map depends on Location
- Map depends on MapPoint
- Map depends on MapPolyShape
- MapPoint depends on a MapPoint
- MapPoint depends on Location
- MapPoint depends on Utilities
- MapPolyShape depends on MapPoint
- MapPolyShape depends on Location
- MapPolyShape depends on Utilities
- Utilities depends on Map
- Utilities depends on Location
- OpenStreet depends on Parameters

- OpenStreet depends on Map
- Route depends on Location
- Route depends on Map
- Main has many Location.
- Main has 1 Parameters
- Main has 1 Map
- Main depends on OpenStreet
- Main depends on Route
- Main depends on Utilities

5 Implementation

The implementation of the web service is located in git and can be found at: `https://github.com/danieldalton10/FIT3144-accessible-maps.git`.

5.1 Platform and Programming languages

The web service will be built using javascript. It has been designed to run under nodejs. Nodejs is built using Chrome's JavaScript runtime environment and is designed for building fast and scalable distributed network applications.

Nodejs will make the implementation of this web service very straight forward. Many libraries and apis that are required can be installed very easily with npm and used in the code as simple function calls. It also provides the flexibility of making the service cross platform and usable in a wide range of situations.

Javascript has been used so that the system can be portable. In addition it allows the system to be used by a variety of platforms on a variety of devices. The idea is that local mobile applications will call the web service for an svg map given a set of parameters. Not only does this take the load off the mobile device to perform any intensive processing it also ensures that all maps will be using up to date data. However, the most attractive gain here is that the service can be used by many applications on many devices. It can be used for GraViewer under IOS and in future it could be used to implement a mapping app on android. Javascript is relatively high level and will facilitate the solution to be implemented in a timely fashion. Many of the API have a javascript interface available so it makes retrieving data from third party's much more straight forward.

Git will be used to manage the project. This will ensure that code is centrally located and can be viewed and modified by relevant people from anywhere. It also makes sure the code will not be lost in case of a disaster. However, the most appealing reason git was chosen is for its version control capabilities. This allows features to be added concurrently thanks to the power of branching. Code can be added and tested separately to other features if desired. In addition a complete history of changes exists so the code can be reverted back to any of its previous states at any time. Changes between revisions can be compared to see what could possibly have introduced a fault in a particular version. Git will be used to manage the code base for the web service component of this project.

5.2 Code layout

The web service is spread across three javascript files.

The mapconstructor.js is the main file of the project and is used to run the web service currently under nodejs. It currently holds the Map class (and it's inner classes) as well as the Main class.

openstreet.js holds the OpenStreet class and is used for retrieving data and updating the map with the open street data. This file is called upon by mapconstructor.js.

The final file is the route.js file. It is responsible for implementing the Route class. It will manage communication with the third party here maps API and updating of the map with the retrieved data.

6 Results and Evaluation

6.1 Current work

As of June 2015 the web service component of the project has been implemented. Work on the mobile application to interface with the web service and GraViewer has not yet started.

The web service can currently be controlled by command line arguments under nodejs. The README file within the git repository has details and examples of how the web service application is to be used. In addition the –help switch will provide a quick usage summary.

The value of the project can certainly be appreciated. A map can be generated by the web service on a computer and the resulting SVG is uploaded to a Google drive folder. The file can be then opened with GraViewer on the ipad.

The map can be explored as any other image would and points of interest and roads can be found by the user. The spatial layout and details of the map can now be explored by GraViewer. One of the main objectives of the project has also been achieved. A pathway can be marked on the map between two points. At this stage the pathway can be found by the vision impaired user using graViewer and provides functionality to make the path easy to follow.

The web service itself has filtering implemented as per the design of this project. The web service currently allows the creation of a map with a range of filters. Some of these include the type of roads (residential, primary, secondary etc), amenities (cafes, restaurants, fast food etc), railways and buildings.

6.2 Future work

The web service provides all the required basic functionality. Future work will improve the SVG output by using different colours to represent roads, points of interest and routes. This is reasonably straight forward to implement and will provide a better experience with graviewer.

The web service will be extended to mark intersections on the map. This means that when a user travels along a road they will be notified of intersecting streets. The web service will be updated to take advantage of asynchronous processing offered by the nodejs apis. This relates to reading data from a file or a http request. This means that requests such as open street maps and routing (here maps), can be done concurrently. The open street data is currently retrieved from a local file. A way to do this by means of a third party api will be developed in order to use up to date data. Some options are migrating to here maps for data retrieval, but this requires some extra consideration including the licensing from here maps.

The placement of objects on the map is reasonably good currently. It should be investigated whether this can be improved at all and working out ways to scale objects would also be useful. The map can sometimes become cluttered. Error handling has only been implemented at a basic level and this can be improved by passing down more details in the request stage.

7 Conclusion

Ultimately, this project aims to provide an accessible way for sight impaired users to interpret maps. Their are two components the web service and a local application to interface with the web service. The goal of this 14 week project was to create a web service that can generate SVG images that can be used by graViewer. This has been achieved and maps of an area can be made by means of a nodejs command line driven applications. The maps can then be loaded onto graViewer for exploration by the blind user.

The maps provide a large degree of very structured useful information. The future of this project will be to write a local application to communicate with the web service such that maps can be made accessible on the device on the go. The web service can be improved as discussed previously. Hence, the web service addresses the major goals and objectives of the project and presents very usable maps suitable for use by graViewer.

References

[1] M. Laakso, T. Sarjakoski, T. Sarjakoski, L. Harrie, J. Andreasson, A. Vilén, G. Claassen, and A. Stoer, "Accessible map and lbs content guidelines," *HaptiMap Haptic, Audio and Visual Interfaces for Maps and Location Based Services*, 2012.

[2] D. McCallum and S. Ungar, "An introduction to the use of inkjet for tactile diagram production," *British Journal of Visual Impairment*, vol. 21, no. 2, pp. 73–77, 2003.

[3] H. Klaus, D. Marano, J. Neuschmid, M. Schrenk, and W. Wasserburger, "Accessiblemap," in *Computers Helping People with Special Needs.* Springer, 2012, pp. 536–543.

[4] P. Parente and G. Bishop, "Bats: the blind audio tactile mapping system," in *Proceedings of the ACM Southeast Regional Conference.* Citeseer, 2003, pp. 132–137.

[5] S. K. Kane, M. R. Morris, A. Z. Perkins, D. Wigdor, R. E. Ladner, and J. O. Wobbrock, "Access overlays: improving non-visual access to large touch screens for blind users," in *Proceedings of the 24th annual ACM symposium on User interface software and technology.* ACM, 2011, pp. 273–282.

[6] BlindSquare, "What is blindsquare," 2015. [Online]. Available: http://blindsquare.com/about/