Spatial Memories Technical Report

July 2014

Contents

[Summary 2](#_Toc394329047)

[Evaluation methods 2](#_Toc394329048)

[The App and Authoring Tool 2](#_Toc394329049)

[Technical Design 3](#_Toc394329050)

[Architecture: 3](#_Toc394329051)

[Cordova (Geofences): 4](#_Toc394329052)

[Fieldtrip Open Framework: 4](#_Toc394329053)

[Spatial Memories Project: 4](#_Toc394329054)

[Fieldtrip Plugins (e,g, geofence plugin): 5](#_Toc394329055)

[General User Interface Technical Issues 6](#_Toc394329056)

[Long Click / Long Tap Gesture 6](#_Toc394329057)

[Sunlight glare 6](#_Toc394329058)

[Voice Over 7](#_Toc394329059)

[Energy Consumption: 12](#_Toc394329060)

[Geo-fence Events 13](#_Toc394329061)

[Geofence Flow 14](#_Toc394329062)

[Data Capture: 17](#_Toc394329063)

[GPS tracks 18](#_Toc394329064)

[Photo: 19](#_Toc394329065)

[Audio Recorder and Playback: 19](#_Toc394329066)

[Map View 21](#_Toc394329067)

[Symbols 23](#_Toc394329068)

[Grid and List View 23](#_Toc394329069)

[Upload (Sync) 23](#_Toc394329070)

[Authoring Tool 24](#_Toc394329071)

[Interactive Map and Table 24](#_Toc394329072)

[Animation 25](#_Toc394329073)

[Concluding remarks 26](#_Toc394329074)

## Summary

This report details the technical evaluation of the Spatial Memories Proof of Concept App and Authoring Tool that was developed during period from February to July 2014. We document the many issues we came up against and how we attempted to resolve them. Overall we believe we have much to share with developers building geo aware Apps, especially those using the Cordova cross platform development environment and JQuery Mobile and ensuring as much as possible the Apps are accessible and work with assistive technology such as Voice Over.

### Evaluation methods

In April we held a 3 day participatory design event with a group of 10 participants from Artlink Central (<http://artlinkcentral.org/>) , having a range of disabilities from autism, partial sight, focus and memory issues, Downs, epilepsy and motor coordination problems. We were able to evaluate an early baseline version of the App and generate lots of information on usability issues and ideas for subsequent development. The workshop also included materials to inform design of a suitable map style to potentially incorporate into the Spatial Memories app. A workflow was also developed to help us evaluate the potential to use photos that participants had captured as bespoke map symbols. Some changes to our evaluation and development plan followed and we organised follow up sessions with 3 people from the group in June. One obstacle was that none of these users owned or used smartphones themselves, so in the follow up sessions we gave the users devices for 10 days. We then accompanied them on a track they had thought of themselves to get an idea how they might use the App in everyday settings and locations.

From May 2014, we started working with two visually impaired students from Glasgow, both very familiar with using iPhones. One user is blind and uses Voice Over all the time, while a second user who is visually impaired, uses Zoom, Inverted Colours and also uses Voice Over for reading long passages of text.

### The App and Authoring Tool

The most recent iteration of the Spatial Memories App (aka MemeTracks or MindNav ) has the following features:

* Record a GPS track and capture image, audio and text “Memories”.
* Memory pings up (with vibrate and audio cue) when you arrive in proximity (Geofence)
* Grid view of assets associated with each track.
* Bespoke Simple Camera App with audio and haptic feedback.
* Bespoke Simple Audio recorder with Audio and Haptic feedback.
* Can be used with assistive tools such as Voice Over, Siri and Zoom.
* Location “Call Out” in Voice Over so users can stay informed of their current location.
* Can change colour scheme (App theme) to allow greater or less colour contrast and control features such as whether to copy captured images into the camera roll.
* Set assets to trigger iOS notifications, if App running in background.

Authoring Tool

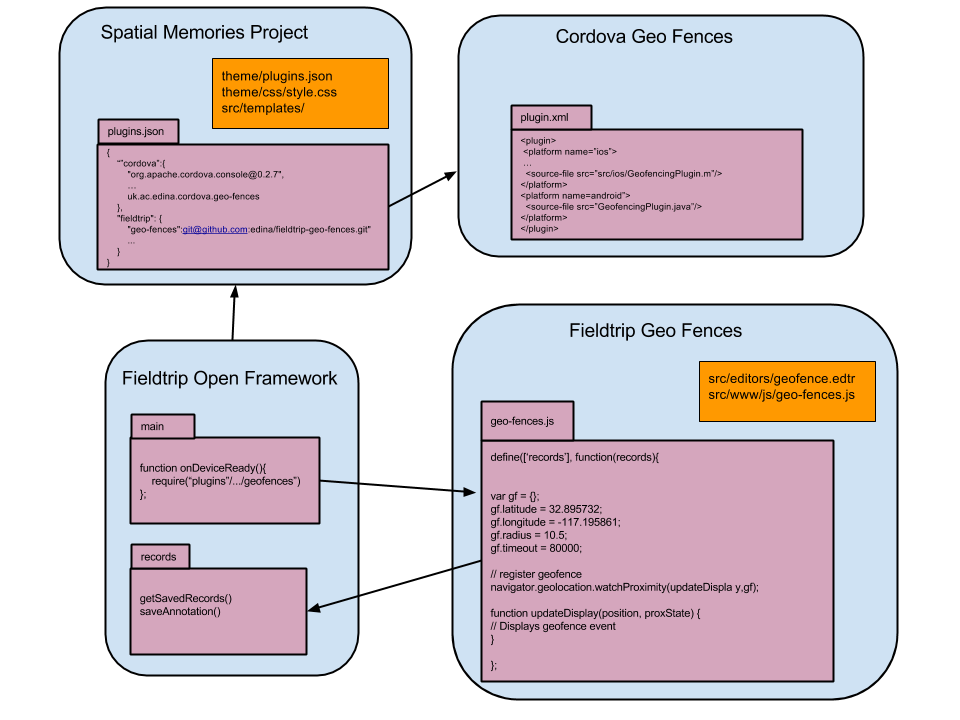
* Supports uploading, sync’ing and sharing of records and tracks from App to Authoring tool using a Dropbox login.
* Can view Memory Tracks and Memory (images, text and audio notes) that user has captured on an interactive map and table.
* Can replay track using an animation that shows Memories popping up as animation progresses.
* Edit Text Memories, and add descriptive text for Image and Audio Memories
* Add new Text Memories to an existing track.

## Technical Design

The framework we have adopted for the Spatial Memories project is based on EDINA’s “Fieldtrip Open” (<https://github.com/edina/fieldtrip-open>) architecture which already incorporates many of the essential components needed and provides a flexible framework for adding new functionality needed for the Spatial Memories project. The Spatial Memories project requires a rapid prototyping environment that will allow us to experiment with various technologies (such as geo-fencing) and design ideas (e.g. customized views and maps) so that we can build iterative versions of the app based on the insights and the creativity of participants involved in participatory design events. The Fieldtrip Open framework provides a modular and flexible architecture for building cross platform Apps that can tap into different device capabilities such as Camera, Audio, Geo-positioning and geo-fencing. The framework extends the popular Cordova (Phone Gap ) App development framework (<https://cordova.apache.org/>). Cordova provides a way for developers to write code for multiple platforms (iOS, Android, Windows mobile), mostly using familiar technologies such as HTML5, Javascript and CSS. These web based technologies are particularly productive for rapid prototyping of user interface components and provide a flexible way for engineers to produce rapid iterations of the Spatial Memories App to try out ideas generated by participants. Cordova is based on a set of plugins that bind native capabilities (e.g. GPS based geo-positioning) to a Javascript API. For the most part the Spatial Memories App will be able to exploit existing plugins, but in some cases engineers need to adapt a Cordova plugin, or author a new one for either Android/iOS or both.

## Architecture:

The diagram below shows the high level architecture for Spatial Memories framework (based on Fieldtrip Open). The blue boxes represent separate code repositories containing different modular components which together can be combined to generate a bespoke App with distinct set of functions. The diagram gives an example of how a geo-fencing component (an existing Cordova plugin we are modifying) could be integrated into the architecture. An explanation of the role of each repository is provided below.

 Architecture diagram show main components of Spatial Memories App

Cordova (Geofences): This repository contains native code implementing platform capabilities (such as geo-fencing in this example) and binding these to a Javascript API using the Cordova framework. (a Cordova plugin). Many plugins already exist, registered in the Cordova Plugin Registry ( <http://plugins.cordova.io/> ). Typically a plugin will bind more than one platform (iOS, Android, Blackberry) to an API.

Fieldtrip Open Framework: This repository provides the logic that brings all the other components together to produce a bespoke App. It queries the configuration (plugins.json) in the Spatial Memories project to download all the components it needs (such as cordova plugins, resources and stylesheets) to create an App with the desired specification. It also handles the storage of assets (photos, notes, GPS tracks.) on the device in a generic manner.

Spatial Memories Project: This repository provides all the code and configuration necessary to make the generic Fieldtrip Open project into a shell for the Spatial Memories App. It defines the set of Cordova plugins and the set of Fieldtrip Open Plugins that the Fieldtrip Open Framework engine will bind together to create a working app with the specified native capabilities and bespoke interfaces.

Fieldtrip Plugins (e,g, geofence plugin): This repository contains the Javascript and HTML5 code that creates a user interface for the App, making use of the Javascript APIs from Cordova plugins to access native functionality.

**Example** **Track Record format:**

{

"1399885416061": {

"record": {

"editor": "track.edtr",

"name": "track (12-05-2014 10h03m29s)",

"fields": [

{

"id": "fieldcontain-textarea-1",

"val": " ",

"label": "Description"

},

{

"id": "fieldcontain-track-1",

"label": "Track",

"style": {

"strokeColor": "red",

"strokeWidth": 5,

"strokeOpacity": 1

}

}

],

"point": {

"lon": -354978.9240334938,

"lat": 7545687.102983582

},

"timestamp": "2014-05-12T09:03:36.061Z",

"geofenceId": "1399885416061"

},

"isSynced": false,

"rate": "5"

}

}

**Example Text Record format:**

{

"1399885427671": {

"record": {

"editor": "text.edtr",

"fields": [],

"name": "Text(12-05-201410h03m43s)",

"point": {

"lon": -3.600254304630154,

"lat": 55.78064324309958,

"alt": null

},

"timestamp": "2014-05-12T09: 03: 47.671Z",

"geofenceId": "1399885427671"

},

"isSynced": false,

"trackId": "1399885416061"

}

}

## General User Interface Technical Issues

### Long Click / Long Tap Gesture

One user interface problem that several participants encountered was generating a response from a long click (a gesture where user presses on a button 1 second or more (note: 500ms registered as long press in Android native interfaces). The current interface listens to Javascript onClick() events which only fire if the press is less than 500ms-1000ms. There are several possible technical solutions to provide a long press listener discussed in the Stack Overflow thread below or as we chose to, use the JQuery Mobile “taphold” method:

<http://stackoverflow.com/questions/2625210/long-press-in-javascript>

Reference JQuery Mobile Taphold:

<http://demos.jquerymobile.com/1.2.1/docs/api/events.html>

### Sunlight glare

Glare from sunlight was another general issue several users encountered. This was especially a problem for using the camera but also affected the users’ ability to use the App UI and see the maps. Our interface uses a colour scheme which we would expected to work well in bright conditions as it uses light text on a dark (blue) background but the interface was still very difficult to see in bright conditions. We implemented a “theme” option in the App settings which allowed users to change the colour scheme but this only helped marginally. We also tried using the “invert colours” accessibility option but again this did not improve visibility.

Reference: <http://ux.stackexchange.com/questions/8153/what-are-the-negative-and-positive-aspects-of-dark-color-scheme>

One might envisage a solution where the ambient light sensor is used to detect brightness and adjust contrast or change a setting (e.g. “invert colours”) automatically. iOS does have an ambient light sensor but it is not made available to developers. Private APIs exist but Apple will not support these so using them would affect ability to market the App in iTunes. The situation is different for Android where programmatic access to the light sensor is possible via the Sensor Manager API.

Reference: <http://developer.android.com/reference/android/hardware/SensorManager.html>

Similarly, it is not possible to programmatically access Accessibility settings in iOS but many Camera filters are available using the CoreImageFilter API.

***Further technical investigation is needed to address this problem, perhaps looking at polarization filters***. However, ultimately sunlight glare is an issue that is best addressed by hardware and operating system manufacturers.

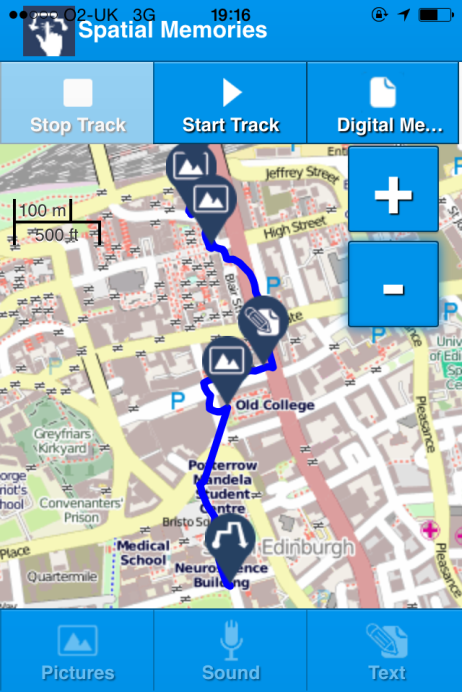
Reference: <https://developer.apple.com/library/ios/documentation/graphicsimaging/reference/CoreImageFilterReference/Reference/reference.html>

Reference: Thread on lack of support to access accessibility settings programmatically in iOS

<http://stackonerflow.com/questions/5591720/how-to-access-iphone-general-accessibility-settings-with-os-4-0>

### Voice Over

The Spatial Memories App uses the web browser to provide all user interface components so the accessibility of the App for users with visual impairment depends on accessibility settings such as Voice Over (<http://en.wikipedia.org/wiki/VoiceOver>) being able to interpret the web pages in the App properly. In May and June 2014 we carried out detailed evaluation of the App with a Voice Over user and learnt a great deal about making the interface work better. The diagram below illustrates some of the techniques we used to make the App interface work better with Voice Over. The main standard we used to mark up our interface was the Accessible Rich Internet Applications (WAI-ARIA) 1.0 (<http://www.w3.org/TR/wai-aria> ).



Map tile images marked as role=”presentation” to stop Voice Over treating them as images to describe. Alternative is to set alt text to a place name, using techniques similar to location “Call Out” but this may cause information overload.

All capture buttons have aria role=”button”, are tab indexed from 1 to 6, and use aria-flow to encourage sequence flow from “Stop” button to “Text” button. Use aria-label to provide alt text and label-for to ensure associated with the button image.

Zoom buttons have aria role=”button”, tab-index=7,8 (default) and aria-label to describe button function.

TODO: Scale bar set as aria-live region to indicate zoom change.

Icons have “alt” and “aria-label” updated whenever the marker layer is loaded with role=”button”. Track icon has tabindex=1 and other makers tabindex starting at 2 for first in sequences and increment 1. Also tried using role=”listitem” and “aria-posinset” attribute but in either case did not succeed in making VoiceOver iterate through sequence in order.

Uses aria-disabled and disabled attribute to indicate to VoiceOver that link disabled?

Diagram showing how Map View Screen is marked up to support Voice Over screen reader.

#### Voice Over: Handling Map Tiles:

In Open Layers (<http://openlayers.org/> ) Tile Images (the 256\*256 images that make up the map) are loaded from a map server as and when needed, so in order to add attributes to prevent Voice Over from describing them as individual images we need to register a function on the OpenLayers “tileloaded” event ( <http://dev.openlayers.org/docs/files/OpenLayers/Tile-js.html#OpenLayers.Tile.events> ) as below.

this.map.layers[0].events.register('tileloaded', this.map.layers[0], function(evt){

evt.tile.imgDiv.setAttribute("role", "presentation");

evt.tile.imgDiv.alt="map tile" ;

});

It would be possible to change the alternative text to something that describes the place represented by the tile (as occurs with “Location Call Out” (see below) and set role=”application” or role=”img”, but users expressed the view that this would create information overload. So we have marked up the map tiles to be ignored by Voice Over using role=”presentation”. This is not the case for other map artefacts. For example, the scale bar is read out by Voice Over if visited. We are experimenting with idea that the scale bar could receive focus after a zoom event to give the user feedback that map is showing a new zoom level. One approach to this is to set the scale div as role=”marquee”.

The pin icons on the map are marked as role=”button” to ensure they are recognised at interactive element (that causes Memory to popup or Track to display / hide). In addition, alt text is added to the elements “alt” and “aria-label” to describe the pin icon. If no description is available for the record, then the record type (“track”, “image”, “audio”) is used instead. We did experiment with aria-poisinset attribute to see if we could get Voice Over to read out the position of each pin in the track, but have not been able to verify this to working reliably as yet.

#### Voice Over: Testing

A general issue we experienced developing for Voice Over is that testing is very time consuming as it is necessary to deploy a new version to the device after each change. With general UI changes we can test quite a lot on the web browser or simulator without needing to deploy to a device. When testing changes for aria mark-up though, the only reliable way of testing if a change has worked is to deploy a new version to a device. The temptation is to stack up a few changes at once, so you can test more on each deploy, but this tactic can backfire as it might be difficult to work out which of the changes is working and which is causing an ongoing problem.

#### Voice Over: JQuery Mobile is not you friend

JQuery Mobile provides a lot of useful tools for developing web applications on mobile. But for some reason it is fond of adding in extra span elements breaking carefully marked up HTML defining a relationship the label text “label-for” and an image button.

For example, Jquery mobile applies it's magic to:   
<input data-theme="a" type="submit" value="Save"/>

transforming it to:

<span class="ui-btn-inner"><span class="ui-btn-text">Save</span>

</span>

<input data-theme="a" type="submit" value="Save" class="ui-btn-hidden" data-disabled="false">

In the visual interface this is not noticed, but screen readers like Voice Over tab to this and the button label causing the button to be announced two times, with no indication which will trigger the button. We couldn’t find a clean fix for this in mark-up so employed a hack to add aria-hidden tag to the spurious span elements dynamically.

$('#text-buttons span').attr("aria-hidden",'true');

#### VoiceOver: Ok we’re British but don’t be too polite

In the Spatial Memories App, we Inform users of system events (e.g. “GPS Not Available” ) using a short message alert, similar to native “Toast” message. The make these messages available to Voice Over we marked the message area as an aria-live region. Initially we set the level to polite (aria-live=”poilite”) which is meant to ensure the user is not interrupted. But we found that Voice Over was not speaking up, even when the User was doing nothing with the interface. We opted then to get aggressive and set “aria-live=”assertive” instead, which did result in the user being better informed.

#### Voice Over: Location Marker

As discussed in section “GPS Tracks” below, we now start continuous location updates running as soon as the Map View is loaded. This means we can update a location marker showing the users current position every five seconds. Recently, Apple Maps App on iOS provided a feature for Voice Over that speaks out a textual description of the geographic feature on the map the user is touching - such as highlighting a street name, name of a park, depending where the user has place their finger. If they touch over the location marker in the centre of the map, a description of the user’s current location is provided. We wanted to at least offer Voice Over users the ability to hear their current location. Rather than users having to find the location marker to hear this, we make the description available to Voice Over by writing it to an aria-live region (hidden from the visual interface) each time the user location is updated. We found that with aria-live=”assertive”, the scheme works quite well. Voice Over seems only to read out the location if it changes, rather than repeating the same location description over.

Some code from the implementation is shown below. We assume from the way it works that Apple is using vector graphics to part-render their maps, so have all the necessary data available already when maps are loaded to the device. As we are not using a vector map renderer our solution relies on downloading a separate file with the location information. We sourced the data from Open Street Map (<http://nominatim.openstreetmap.org> ) which is organised so that every map tile has a single place name description. The level of detail depends on the tile zoom factor – so for example – a tile with a low zoom level might have general area description such as “Westminster, London”, while a tile in same location with a higher zoom (z) level might have more precise description such as “Downing Street, London”. So far we have experimented with just one zoom level.

if(locations){

var tileStr = this.long2tile(this.userLonLat.lon, maxZoomLevel) + "-" + this.lat2tile(this.userLonLat.lat, maxZoomLevel);

In the code above, the description is recovered from a file which has a place name description of each map tile, specified as X, Y, (Z) tile index (<https://developers.google.com/maps/documentation/javascript/maptypes> )

As we get location updates from the operating system as a WG84 longitude / latitude co-ordinates, we first have to call functions (long2tile and lat2tile) to convert this to a tile index.

long2tile: function(lon, zoom){

return (Math.floor((lon:

+ 180) / 360 \* Math.pow(2, zoom)));

}

lat2tile: function(lat, zoom) {

return (Math.floor((1 - Math.log(Math.tan(lat \* Math.PI / 180) + 1 / Math.cos(lat \* Math.PI / 180)) / Math.PI) / 2 \* Math.pow(2, zoom)));

}

We then look up the tile in our locations file, and post the text to the aria-live region as below, (creating the live region div if not already present):

locationText = locations[tileStr];

if($('#map-location-text').length === 0){

$('#gpscapture-page').append('<div ><p aria-live="assertive" id="map-location-text"><p></div>');

}

$('#map-location-text').text(locationText);

}

At the moment we envisage the locations file to be specified for relatively small areas (e.g “Central Glasgow”) as the file is likely to get too large otherwise. The Authoring Tool provides a mechanism to specify the bounding box of a location file, which can then be generated and downloaded to the App. To make this work we will need to do further work to store and serve the data ourselves and perhaps look at more efficient ways to hold the data.



Screenshot showing location marker at map centre

## Energy Consumption:

A detailed technical evaluation of battery usage has not been carried out on the App yet. The main expected demands of the App on battery consumption other than display are geo-tagging, geo-tracking and geo-fencing capability as these all potentially use GPS that can consume a lot of CPU. Geo-tagging is a relatively short lived, discrete and user initiated so will not have to use geo-location services very often. Both geo-tracking and geo-fencing use CPU at regular (approximately 5 second) intervals so can potentially consume a lot of CPU.

For geo-tracking a user might start a track running and then forget they are still recording and within hours they may have run their battery flat. Some options for reducing this risk are:

1. Stop track running when app is paused by OS. For Spatial Memories this is not acceptable as users may well wish to continue recording their track while device is in their pocket.
2. Stop track running after a period of inactivity (e.g. user has not moved for five minutes). This may cause issue when user has stopped to take photo or record audio.
3. Prompt user after period of inactivity. Could annoy user is using another app or making a call.
4. Stop track after period of inactivity but only if battery level is below a threshold.
5. Pause track after period of inactivity and notify user silently. Resume track when user re-enters the App or clicks on notification.
6. Leave track running but increase the interval between calls to geo-locations API and/or reduce accuracy requirement. Depending how tracking implemented and how much battery consumed the OS itself may apply either of these strategies.

Optimising geo-fencing is more complicated as we can expect geo-fence events to fire when we are not expecting them to fire (i.e. we forgot about them). However strategies such as only activating geo-fences when user starts to come into range (detected using low accuracy, low consumption geo-position techniques) can greatly reduce energy consumption demands. Applying a timeout to geo-fences could also be useful (assuming geo-fences timeouts are reset when the app is restarted), although it is possible users have an expectation that the geo-fence will persist permanently.

Our current implementation on iOS activates the geo-fences when the App is started and geo-fences continue to be monitored as long as the App is running (in foreground and background). It is left to the OS to optimise the geo-location detection method and apply power saving strategies.

## Geo-fence Events

The Spatial Memories App uses a technology generally referred to as “geo-fencing” to alert user that they are in the vicinity of a point where they previously made an audio, text or image “memory”. When the device detects a geo-fence is in range the user is alerted by a vibration and/or audio alert and a window pops up in the App Map View showing either a photo, an audio media player or the Text note that the user previously captured at the location.

All Text, Audio and Image memories are registered as active geo-fences. This was designed so that the user interface could be made as simple as possible, allowing users to create a geo-fence as a side effect of creating a memory rather than having to tick an extra box or option button. Similarly the radius of the geo-fence (the circle around point where geo-fence is triggered) and transition types (ENTER. EXIT) are all hard coded so that no additions to the UI are needed. We set the radius to 20 meters during the user testing at the workshop and we found this setting worked well. It seemed to be large enough to enable the event to trigger reliably but small enough to avoid too many overlapping geo-fences.

The diagram below explains how Geo-fences work in Spatial Memories.

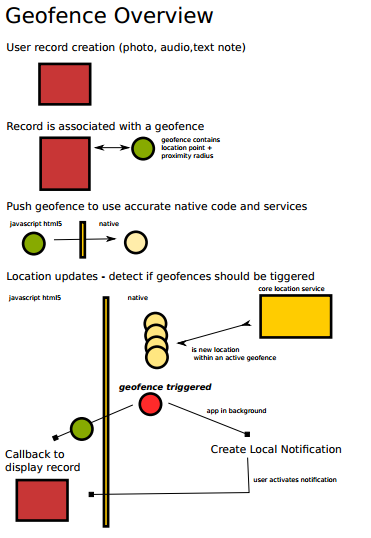


Diagram showing the control flow for geo-fence events

### Geofence Flow

1) User Record Creation (Photo, Audio, Text memory)

2) On record creation, the record is associated with a geo-fence using a unique id. The geofenceId is a guid, the point is the location from device GPS

var geofenceRecord = function(geofenceId, point){

var gfparams = {"fid": geofenceId, "radius": GEOFENCE\_RADIUS\_METERS, "latitude": point.lat , "longitude": point.lon };

if(typeof(geofencing) !== 'undefined'){

geofencing.addRegion(

function() {

console.debug("region added");

},

function(e) {

console.debug("error occurred adding geofence region") ;

}, gfparams);

}

};

The geofencing object is our concrete implementation of a geofencing mechanism adhering to cordova plugin interface.

3) The javascript geofenceRecord function calls across boundaries to the native method addRegion.

- (void)addRegion:(CDVInvokedUrlCommand \*)command {

[self checkMonitoringStatus];

NSMutableDictionary \*options = [self parseParameters:command];

NSString \*regionId = [options objectForKey:@"fid"];

NSString \*latitude = [options objectForKey:@"latitude"];

NSString \*longitude = [options objectForKey:@"longitude"];

double radius = [[options objectForKey:@"radius"] doubleValue];

if (radius > locationManager.maximumRegionMonitoringDistance) {

radius = locationManager.maximumRegionMonitoringDistance;

}

CLLocationCoordinate2D coordinate2D = CLLocationCoordinate2DMake([latitude doubleValue], [longitude doubleValue]);

CLRegion \* region = [[CLRegion alloc] initCircularRegionWithCenter:coordinate2D radius:radius identifier:[NSString stringWithFormat:@"cordovaGeofencing:%@", regionId]];

@synchronized(self.monitoringRegions){

[self.monitoringRegions addObject:region];

}

[self returnStatusOk:command];

}

4) Still on the Native side The Location Manager working asynchronously and updates the users location.

This method calculates whether a geofence is triggered.

- (void)locationManager:(CLLocationManager \*)manager didUpdateToLocation:(CLLocation \*)newLocation fromLocation:(CLLocation \*)oldLocation{

CLLocation \*aUserLocation = newLocation;

for (CLRegion \*region in self.monitoringRegions) {

if([region containsCoordinate:aUserLocation.coordinate]){

if (![self.insideRegions containsObject:region]) {

@synchronized(self.insideRegions){

[self.insideRegions addObject:region];

}

[self notify:region withStatus:@"entered Region"];

AudioServicesPlayAlertSound(kSystemSoundID\_Vibrate);

[self playSoundAlert];

}

} else {

if ([self.insideRegions containsObject:region]) {

@synchronized(self.insideRegions){

[self.insideRegions removeObject:region];

}

[self notify:region withStatus:@"leaving Region"];

}

}

}

}

5) If the geofence is triggered and the application is the the background, a local notification is created and alerts the user.

UILocalNotification \*notification = [[UILocalNotification alloc] init];

notification.fireDate = [NSDate date];

NSTimeZone\* timezone = [NSTimeZone defaultTimeZone];

notification.timeZone = timezone;

notification.alertBody = statusMessage;

notification.alertAction = @"Show";

notification.soundName = UILocalNotificationDefaultSoundName;

[[UIApplication sharedApplication] scheduleLocalNotification:notification];

6) If the geofence is triggered and the application is in the foreground there is a simple callback to the Javascript layer to display the record.

- (void)notify:(CLRegion \*)region withStatus:(NSString \*)status{

NSString \*regionId = [self getRegionId:region];

NSString \*json = [NSString stringWithFormat:@"{fid:\"%@\",status:\"%@\"}", regionId, status];

NSString \* jsCallBack = [NSString stringWithFormat:@"%@(%@);", self.callbackId, json];

[self.webView stringByEvaluatingJavaScriptFromString:jsCallBack];

## Data Capture:

As a result of evaluation during the Stirling workshop in April, we decided to make the steps involved in capturing images, audio and text as simple as possible.

In our existing “Fieldtrip GB” app, to take an image the user must follow the workflow:

1. Click the image capture button.
2. A form opens up where the user can enter a name for the image. (e.g. “Lecture Theatre 2”)
3. The user can optionally enter a longer description.
4. The user clicks the Camera button – the Camera Viewfinder now opens.
5. The user presses the Camera shutter button to take a photo.
6. The Camera app shows the user a preview of the photo.
7. The user selects “Use” to confirm they want to use the photo.
8. The user is shown a preview of the image and a map preview showing where GPS has located the user position with a “balloon” icon indicating the captured location.
9. The user can move the balloon icon on the preview map to correct any GPS inaccuracy.
10. The user confirms the positioning of the record by clicking a “Save” button and the image capture is completed.

In Spatial Memories, the process is greatly simplified.

1. Click the image capture button.
2. Press the “Take” button
3. The photo is taken and popup appears
4. Press “Close” button on the pop up Screen

While this simplifies the process and makes the data capture as quick as possible there are two drawbacks.

1. There is no opportunity to add metadata such as a description to the image. This can be added later using the Authoring Tool, but immediate effect is that there is no alternative description for image and audio Memes. This is particularly important for Voice Over users.
2. If GPS signal is not accurate at the time of data capture there is no way to correct it.

Discussing the first issue with Voice Over users, one user came up with a clever solution where the description could be included as a field in the pop up that appears immediately after the image or audio is captured, or when it is selected from the Grid View. The user then has an option of adding a description. The second issue is harder, but we think we have implemented a strategy to reduce the likelihood of incorrect GPS fix being associated with the captured data. In recent iterations we start a continuous location update (every 5 seconds) as soon as the App starts up and display a location marker as soon as a reliable fix is obtained. As the GPS track is an ongoing process repeatedly obtaining location updates from the operating system, a good location fix should already be obtainable by the time a user seeks to create an image, audio or text memory. This workflow should also align better with the operating systems location strategies where typically less accurate methods are used to obtain an initial location, allowing time to obtain fixes on satellites needed for more accurate location updates. Generally it appears this new strategy is working well as we have not had problems with location fixes since this implementation.

### GPS tracks

GPS tracks are initiated in the App by pressing Start button (on Home Page or in Map View) and ended by pressing Stop button in Map View. If the user navigates away from Map View, tracking continues in background. The GPS tracks are persisted as a GPX file (<http://en.wikipedia.org/wiki/GPS_Exchange_Format>) and this file is updated periodically while track is running so will be recoverable if App is stopped or paused. In the Map View the track is plotted in real time as a line so that the user can see their progress.

Overall GPS tracks are working well. In the April workshop we found that some tracks were recording had large inaccuracies. It was difficult to replicate this but we think the following improvements have rectified the issue.

1. We start continuous location updates as soon as the user enters the Map View in the App. This give the operating system a chance to obtain a clean high accuracy location fix by the time the user clicks the start button.
2. The user has visual indication that the location fix has been obtained as the map automatically pans to the users current location.
3. The system will not start a new track unless it has obtained a fresh location fix.
4. The system makes more attempts to obtain an initial location fix.
5. The system now continues location update when the App is not in foreground or screen locked.

### Photo:

The initial baseline App (evaluated during the April workshop in Stirling) revealed a number of usability issue with the default Camera plugin for Cordova. The default App had several options and buttons such as switching front and back camera, a preview step where a user can choose whether or not to use the photo they just took and a video/still toggle. We implemented a new “Simple” Camera plugin that just has a “Take” and “Cancel” button. The Simple Camera also provides an audio and vibrate feedback to confirm button pressed. There are still a few details to sort out such as the buttons not responding to long tap gestures, poor colour contrast in the buttons and a delay between pressing the button and the feedback but we think de-cluttering the Camera has improved the accessibility of the App. Some features that would be good to evaluate in future, would be the ability to use filters such as polarisation and colour inversion.



Screenshot of Simple Camera Cordova plugin

### Audio Recorder and Playback:

During evaluation with users we noted a number of usability issues with the default Cordova Audio Recorder plugin.

1. It was difficult for users to know they had started the recording as the feedback ( a timer and thin red bar) was too subtle.
2. Several users pressed the toggle button twice by mistake (it seemed very sensitive to me too) which resulted in the previous recording being lost.
3. Users did not realise they had to navigate back to the “Done” button to save the recording.

As a result we modified the default plugin to simplify the workflow and provide a beep (and vibrate) as soon as recording starts. When the user presses the record toggle button again, the recording is immediately saved and Audio recorder exits, so there is no chance of accidently wiping the recording. The audio feedback (beep) is particularly helpful for Voice Over users.

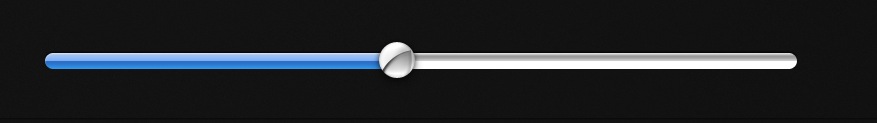
The biggest issue users reported with Audio was the volume of playback. There are two variables to consider. The current volume setting of the device (System output volume) (v1), and the volume level set by the media player (v2), which will have a maximum value equal to v1. (i.e. max(v2) = v1)

This interaction is important for safety of users (hearing) but can also be confusing. The latest version of iOS (iOS7) has deprecated a previous system volume method on MPMusicPlayerController and now predicates the use the [MPVolumeView](https://developer.apple.com/library/ios/documentation/MediaPlayer/Reference/MPVolumeView_Class/Reference/Reference.html" \l "//apple_ref/occ/cl/MPVolumeView" \t "_self) class, which “provides media playback controls that iOS users expect and whose appearance you can customize”.

Reference Deprecated method:

<https://developer.apple.com/library/ios/documentation/MediaPlayer/Reference/MPMusicPlayerController_ClassReference/DeprecationAppendix/AppendixADeprecatedAPI.html#//apple_ref/occ/instp/MPMusicPlayerController/volume>

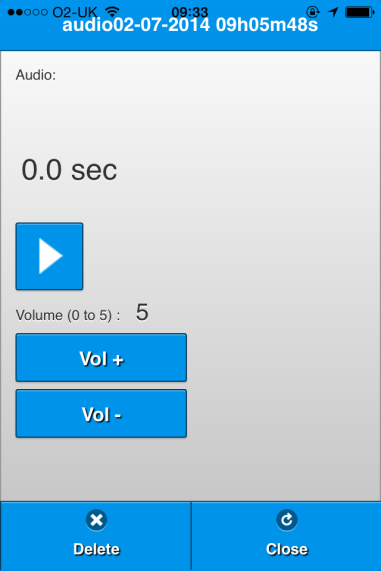
Reference: <https://developer.apple.com/library/ios/documentation/MediaPlayer/Reference/MPVolumeView_Class/Reference/Reference.html#//apple_ref/occ/cl/MPVolumeView>



Reference: <http://ios-blog.co.uk/tutorials/controlling-system-output-volume-with-the-mpvolumeview-class-part-one/>

We attempted to use the above in a new version of the Audio Playback but ran into some problems as our design relies on popups using HTML rather than native interface components. We did implement a system volume button on the playback screen, which should help improve playback. One important factor in obtaining a good recording volume is to have the user speak with their mouth close to the device microphone. We might investigate ways to encourage this in the Audio Recorder plugin – for example, using proximity sensors or detecting amplitude of input and vibrating device if too quiet.

When we tested the first Audio Playback implementation using Voice Over we noticed that the timer which updates the number of seconds into the playback was being read aloud, drowning the actual audio track itself. To prevent this we simply set elements role=”presentation” in accordance with the aria specification. (we also experimented with role=”timer” and aria-live=”polite”, but Voice Over was still interrupting media playback with these settings).

Screenshot of first Audo recoder interface (left) and latest version with button reorganised (middle) and the audio playback screen with volume control (right).

## Map View

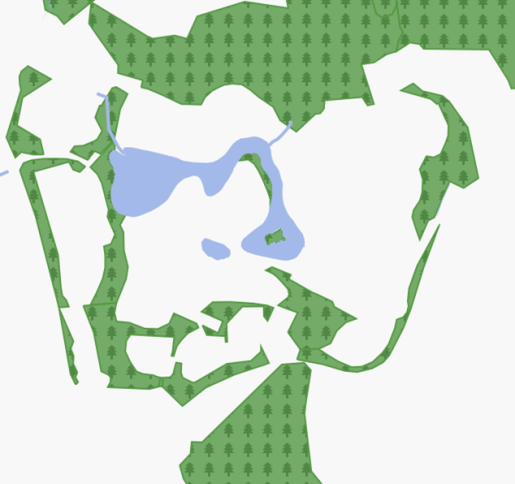
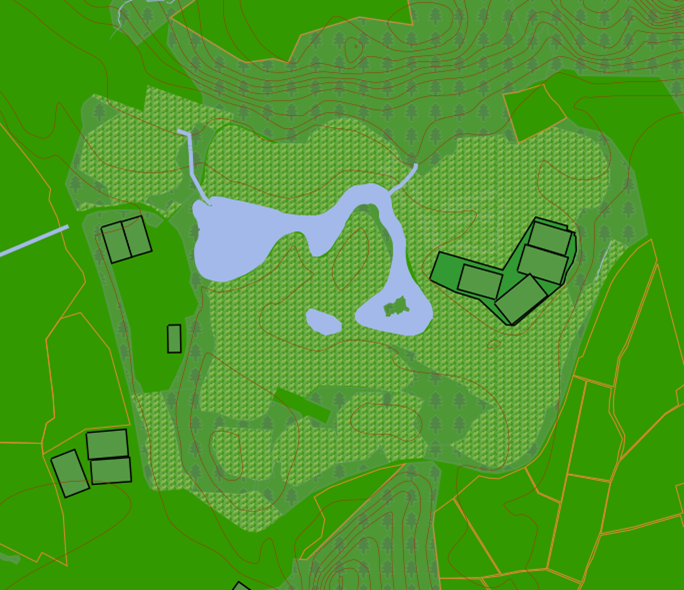
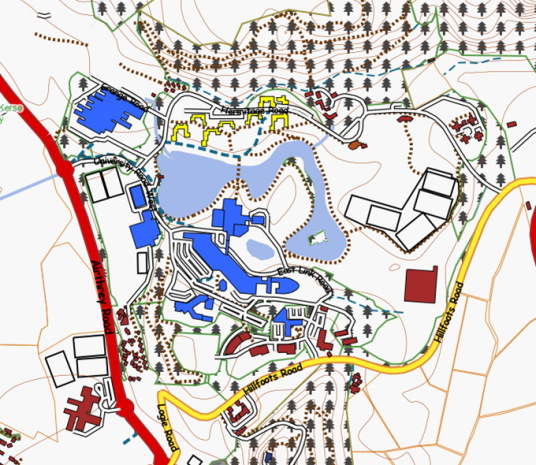
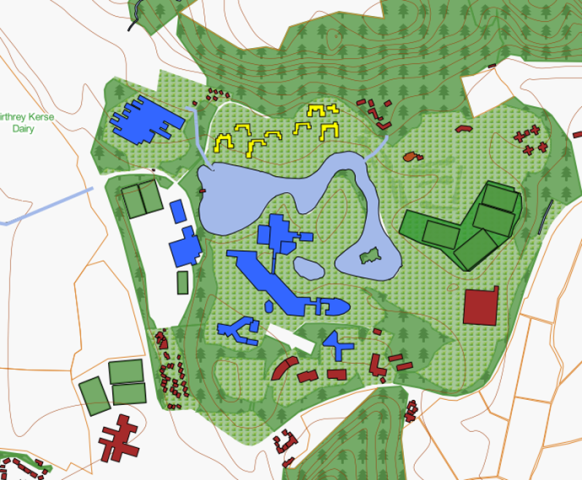
Spatial Memories uses the Open Layers Javascript library (<http://openlayers.org/> ) to display maps, map controls and gestures (panning and zooming). We are mostly using the public Open Street Map server for evaluation of the App. In preparation for the Stirling workshop in April, we also prepared some custom styled maps of the University of Stirling campus location using raw vector data from Open Street Map and Ordnance Survey (Open data). Maps were styled using the Open Source TileMill application (using the Carto CSS styling language) and served via the Mapbox Cloud service. Our intention was create several versions of the same map using different layers, features and styles, and use these maps in both the workshop design sessions and the App to help us evaluate what cartographic styles might suit some users more than others.

TileMill applies Carto CSS style rules to the raw vector data to create map (raster image) tiles that are packaged into a dynamic archive format (MB Tiles). A feature of MBTiles is that it can be used to serve map tiles both on an online map server instance (such as Mapbox or Mapcache) or as an offline package in the App itself. The ability to use MBTiles offline makes the format ideal for mobile applications where network connectivity may be an issue in some locations. Only relatively small areas are possible to cache on the device as images take up too much space. However a workflow can be envisaged where a user selects an area of interest, applies some styling preferences and then downloads the generated map tiles to cache locally on the device. EDINA has previously worked on a solution to integrate MBTiles with the Open Layers Javascript API.

Reference: Mbtiles with OpenLayers (<http://mobilegeo.wordpress.com/2013/06/07/mbtiles-and-openlayers/> )

We initially intended to use TileMill in the workshop map design session to show different maps being produced in real time. But we realised that it would take too long to render each map from raw data and this would make it hard to keep discussion going - so instead different layer combinations and styles were rendered in advance of the workshop and screenshots collected so we could quickly show participants how maps would differ.



Some example maps showing different layering of features

## Symbols

Another idea we experimented with during the Stirling workshop in April was using photos or “sketchified” versions of photos as point symbols on maps. Concept of using photos or sketches as point symbols on maps had some technical difficulties. The CSS scaling technique we intended to use did not produce high enough quality images as too much information was lost. We required full blown image processing software (GIMP) to produce acceptable results. Participants in the workshop enjoyed using the “Sketch Guru” App to turn photos into sketches from an aesthetic point of view but this was actually worse for scaling down as the original image had lost information before scaling occurred . Another problem is handling overlapping points. Mapping software such as Map Server and Mapcache can decide to omit overlapping images but this is likely to confuse users.

For this idea to work we would need to vectorize the original image and apply completely different algorithms for abstracting the original image – “symbolising” rather than “sketching” the image. This is an interesting technical challenge but represents a new research question.

## Grid and List View

The Grid and List View did not raise any particular technical issues. One usability issue that needed to be addressed was the Grid View becoming too crowded and requiring extensive scrolling. This was improved by filtering Memories by track id, so that only the Memores associated with the current open track are shown. We evaluated the Grid View with Voice Over and there is an issue with swiping to tiles that are not visible. This is an issue we are still working on.

## Upload (Sync)

The data users collect (GPS tracks and image, audio and text Memories) need to be uploaded to a server so that they can be accessed and edited by the web-based Authoring Tool (<http://fieldtripgb.edina.ac.uk/sm-authoring/> ). Similarly, changes and new Memories added in the Authoring tool need to sync back to the App. The current implementation uploads and syncs via Dropbox (<https://www.dropbox.com/home> ). However it should be noted that our design is not closely coupled to Dropbox or any other Cloud Storage Provider. All the code that handles sync operations uses an abstraction layer called the PC-API ( Personal Cloud API <https://github.com/edina/pcapi/tree/master/docs> ). EDINA has already developed an implementation of the PC-API on its own servers that is close to being released publicly. Once this is available we anticipate that Spatial Memories will use this API instead of the Dropbox implementation. Generally sync seems to work well but we have noticed some occasional problems where images belonging to one Memory get mixed up with another one. We are still investigating this issue but suspect it has something to do with the intricacies of the Dropbox implementation of PC-API. Another issue is the accessibility of the user login to Dropbox. This is something that is not in our control as we pass over to the Dropbox authorisation page when the user enters their credentials. We also have noticed a significant delay before the page loads up which is particularly problematic for Voice Over users. We are not sure at the moment if this delay is due to the PC-API or Dropbox itself.

## Authoring Tool

The Authoring Tool is still being actively evaluated so we have limited information on technical issues at the moment. There are 3 main features in the current interface.

1. An interactive Map and Table which shows Tracks and Memes.
2. An Animation which allows users to retrace their steps (literally!).
3. An interface which allows users to create a location file (partially implemented) by specifying the bounding box of the area for which locations should be generated.

Below we describe briefly some implementation details for each of these components

### Interactive Map and Table

As with the App, the Interactive map uses the Open Layers Javascript API (<http://dev.openlayers.org/releases/OpenLayers-2.12/doc/apidocs/files/OpenLayers-js.html> ). The data (Tracks and Memories) is obtained by querying the uploaded records in Dropbox using the PC-API implementation (described in Upload / Sync section above). The table is implemented using AJAX and standard HTML. The main technical consideration for this part of the interface was how to make the table work well with Voice Over. There were two concerns: One was handling the delay between the user clicking the “Get Tracks” button and the table being populated; the second was making sure that the user could use a hierarchical tree structure where expanding a track displays the Memes associated with that track. As with the App itself we relied heavily on W3C ARIA constructs such as aria-hide, aria-expanded to achieve this:

MapViewer.prototype.onRowExpanded = function(evt){

var ariaMsg = '';

var $track = this.\_findClosestTrack(evt.currentTarget);

var trackName = $track.attr('record-name');

var trackid = $track.attr('trackid');

// Collapse any other track expanded

$track.siblings('.track')

.removeClass('expanded')

.addClass('collapsed')

.attr('aria-expanded', 'false');

$track.siblings('.poi:not([trackid="' + trackid + '"])')

.addClass('hidden');

// Expand this track

$track.removeClass('collapsed')

.addClass('expanded')

.attr('aria-expanded', 'true');

$pois = $track.siblings('[trackid="' + trackid + '"]');

$pois.removeClass('hidden');

switch(n = $pois.length){

case 0:

ariaMsg = "No memories associated to " + trackName + " track.";

break;

case 1:

ariaMsg = "Showing " + n + " memory for " + trackName + " track.";

break;

default:

ariaMsg = "Showing " + n + " memories for " + trackName + " track.";

break;

}

aria.notify(ariaMsg);

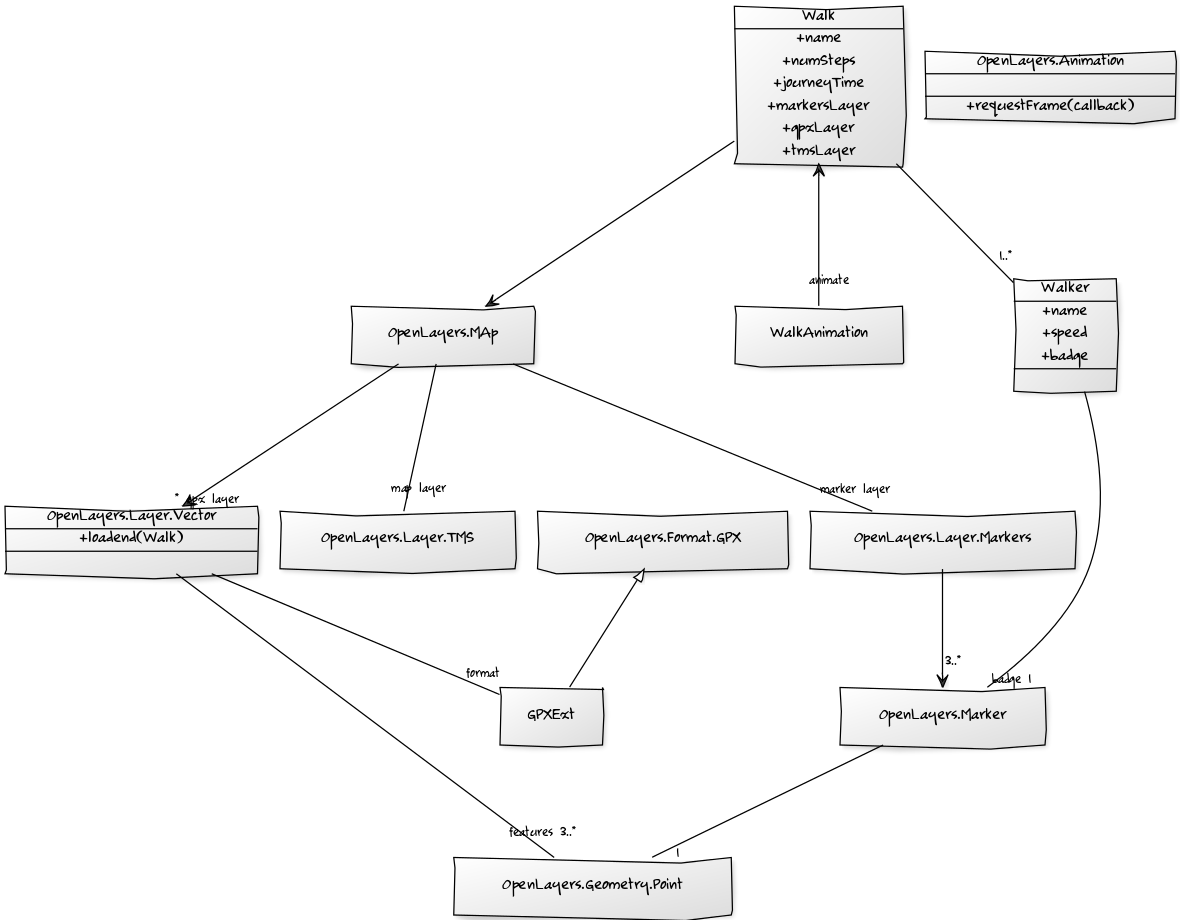
}

For both notifying Voice Over users about the table results being available and when track expanded in the table, we used ARIA live region=”assertive”.

### Animation

The animation uses HTML5 Canvas to draw footsteps on the map following the tracks users have uploaded, similar to Harry Potter Marauders’ Maps. As the footsteps pass by Memories, a popup opens briefly to show the image or audio or text the user captured. One technical issue we encountered was that it took too long in some cases to load the image or audio file from Dropbox, so the animation moved on and closed the popup before it could be shown. To resolve this we added controls to speed up or slow down the pace of the animation. Also we implemented an option to pause the animation at each popup, so the user has to press resume after the popup has loaded. An alternative might be to load all the necessary images and audios and cache them locally before the animation starts.

The design of the Marauders Map animation is shown below:



The diagram illustrates how the components involved in the Marauder animation, interact with the OpenLayers Map API (including the GPX format) and the web browser. Note that the animation uses requestAnimationFrame to enable the Web Browser to optimize the animation using native routines (<http://www.paulirish.com/2011/requestanimationframe-for-smart-animating/> )

## Concluding remarks

We have gained a great deal of technical knowledge and developed several techniques for making Apps based on the Cordova and jQuery mobile framework and Open Layers mapping API more accessible. Our approach of using W3C ARIA standard attributes to support Voice Over produced good results, but was time consuming and difficult to debug. In outdoor locations the geo-fence technology worked well, but we may have to optimize our location update strategy to improve battery efficiency as the App develops further. We will investigate also Implementing geo-fences to trigger in indoor locations using a technology such as iBeacons. The “Location Call Out” worked better than expected but requires compression and / or caching functionality to become a practical solution. We need to look at alternative to Dropbox as a storage provider so that it is easier to sync and share data. Another area to investigate is how we could link up to social networks.