2014

RainBall High Level Design Document

Spatial Cognition

RainBall is an information visualization technology evolved basing on Halo, aimed at optimizing spatial cognition for map navigation.

Jinxiang GOU & Gavin YAO
University Paris-Sud
12/19/2014

RainBall High Level Design Document

Contents

1 General

This report elaborates the concept of off-screen visualization technology RainBall, and the procedure of designing a mobile application (prototype) basing on it.

1.2 Revision History

Table 1 Revision History

Revision	Modifications	Date	Author
RevA	The draft version	19-Dec-14	Jinxiang GOU & Jin YAO

2 Scope

The Scope of this document including:

- Reference of RainBall, i.e. Halo
- Concept of RainBall
- Improvement of RainBall compared with Halo
- Issues and solutions during APP implementation

Others, e.g. specifics in programming, are out of scope.

3 Brief Introduction

RainBall is a technology developed basing on Halo for optimizing spatial cognition, and stands for “Ball in the Rainbow”.

3.1 Motivation

Having gone through all 23 topics posted in website, we decided to choose the paper *Halo: a Technique for Visualizing Off-Screen Locations* belonging to topic *Visualizing Off-screen Locations*, through which, we can not only cultivate the competence in Information Visualization, but also enrich the experience in mobile application development.

3.2 Paper Introduction

This design is based on paper *Halo: a Technique for Visualizing Off-Screen Locations*. Halo accomplishes the task of off-screen location exploring by surrounding off-screen objects with rings that are just large enough to reach into the border region of the display window.

Figure 1 Halo Technology for Spatial Cognition

In this paper, it elaborates below contents:

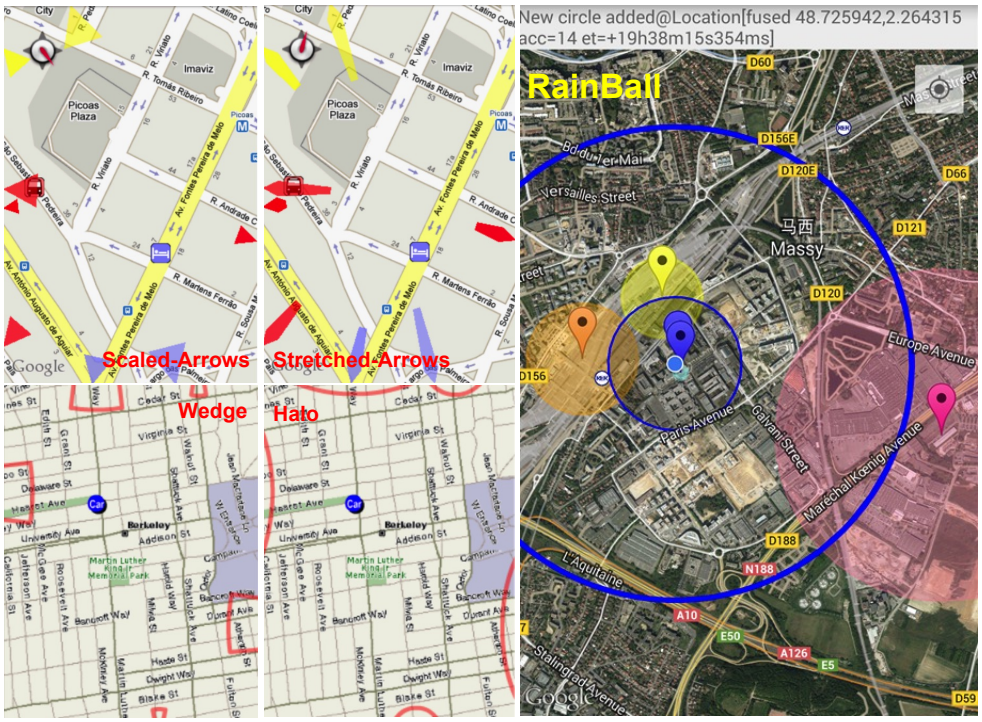
- Motivation and introduction of Halo
- Problem of Halo (overlap) and relevant solutions
- Comparison between Halo and other approaches, e.g. arrows

Among above all, there are some of them inspiring us for latter design. One is the method of spatial cognition. In Halo, it subtly uses the affordance of arc/curvature to adapt Halo's geometric model to the mental model of users. Bigger curvature implies bigger radius and further distance between user's current location and target marker in the map. Furthermore, basing on the route of arc, users can speculate the position of the center, i.e. location of target marker.

After investigation, we have summarized the other popular visualization technologies for optimizing map exploring experience:

- Scaled-arrows
- Stretched-arrows
- Wedge
- Halo

Figure 2 Existing Off-Screen Visualization Technologies & RainBall



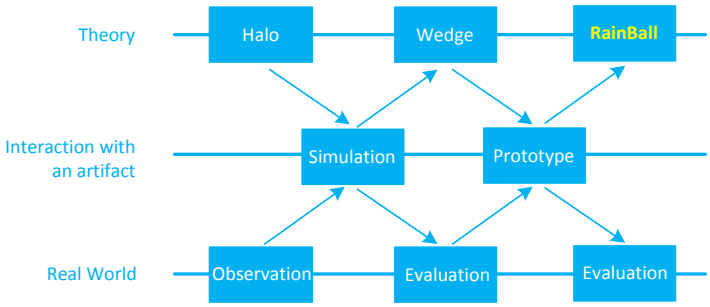
4 Design Evolution

The theoretical underpinning of both Halo and Wedge is the theory of amodal perception or amodal completion. One key weakness for these 2 technologies is using 2 levels connection between the geometric and mental modal, i.e.:

- 1st Level: users speculate the radius according to curvature of arc
- 2nd Level: users speculate the distance between marker and current location according to radius, as radius doesn't equal to the distance

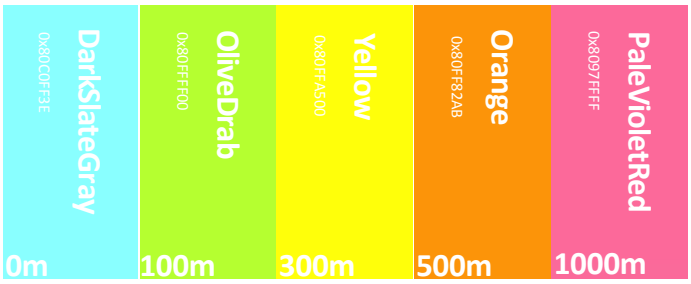
To solve this problem, we introduced a new information visualization technology, RainBall, for spatial cognition, which is improved basing on Halo, and stands for "Ball in the Rainbow".

Figure 3 Peripheral Awareness Research Strategy



In RainBall, we used different color to stand for corresponding distance range as shown in figure 3 below, and the Halo circles of off-screen marked will be filled with corresponding semitransparent color according to the distance between Marker and users current location. Hence, they can obtain the distance directly and intuitively by reading the color like a heat map indication, for instance, if a user sees the Yellow Ball, then the distance between he/she and the target marker is 100m to 300m.

Figure 4 RainBall Color Definition



5 Development Baseline

This application is developed basing on Google Map APIV2, with smart phone, HTC ONE M7, Android 4.4.3, under Android Studio 1.0.

Figure 5 RainBall Development Environment

6 Development Concept

6.1 Key Functions

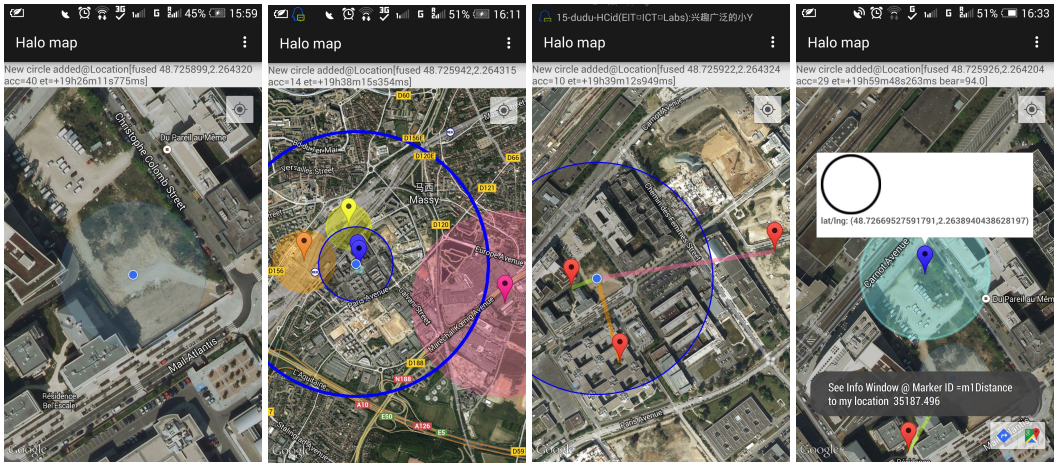
RainBall has below key functions

- Rainbow Ball

- Polyline

- Distance Displaying

Figure 6 RainBall Key Functions



6.2 Geofencing

This API lets our app to setup geographic boundaries around specific locations and then receive notifications when the user enters or leaves those areas. It allows batch addition and removal of geofences, with ability to manage multiple geofences at the same time and the ability to filter alerts for both entry and exit or entry only or exit only.

That notification procedure was thought in our design process firstly, which could sense our user's modality (such as still, walking, driving, and so on [5]). But later we identified the idea of RainBall was to provide with locations indication rather than give a fixed range for notification, so the function is not kept in the final version.

The idea of Geofencing is a standard that implemented by google[5] in mobile app for sensing geolocations, which combines awareness of the user's current location with awareness of nearby features within a range, defined as the user's proximity to locations that may be of interest. Still with/without geofencing feature, to mark a location of interest, we set the latitude and longitude. To adjust the proximity for the location, we add a radius by listening on the mylocation change. Also, inspired by this, we implement our app by adding listeners to the location change event, and thus getting real time feedback of the change of distance between current location and targeted markers (locations), both the radius (as in Halo) and color of the circle/marker changes according to the relative distance to current location.

The final idea is to build a location awareness software that could listen to location change and proximity of locations by changing the circle size of locations automatically and code them with color to indicate range changing.

Activity recognition feature:

With mobile apps being more contextual, to understand what the user is doing is critical to representing the right content. Right now, a solution supported by Googel's Activity recognition API made checking the user's current activity really available [5]—mainly: still, walking, cycling, and in-vehicle—with very efficient use of the battery.

Enhances other services with context:

The API is very helpful for adding movement awareness to location awareness. But other methods are also very effective in showing awareness of off-screen information [6]. Apps can adjust the amount of location awareness that they provide, mainly based on the current user movement. For example, a navigation app can request more frequent updates when the user is driving.

6.3 MVC Model

Our implement app is consisted of 3 parts. Given the interactive software model, model-view-controller, the RainBall is built on:

Model: The Android framework and googlemap API model. Including many core libraries of interactions and geolocation data fetched from google service, also map object models, like markers, and Latlng locations, etc.

View: The XML layout of the information show onscreen, inflated by the model contents, such as geolocation info, circles, markers, current point, etc.

Controller: The main function for listening and reacting to types of events to update the model contents.

1. Location event listeners. Attached to build-in android location sensor and controlled by user movement with the phone. Based on hardware &software sensors.

2. Touch and gesture, etc. event listeners.

6.4 Interaction Table

Table 2 RainBall Interaction Table

Object	Representation	Properties	Operations
Marker	Colorful Icon (Color changes according to distance)	Color, Size	Add
InfoWindow	Text Field	LatLng	Click
Polyline	Color Line between current Location and Marker	Color, Length	Add
Pop-up	Text Field	Distance	-

Window			
CurrentLocation	Blue Dot in screen	LatLng	Move

Operations	Commands	Feedback	Responses
Add a Marker	Click on the target screen pixel	A marker and a circle is added	A marker together with the circle is added in the clicked position.
Add a Polyline	Long click on the target screen pixel	A polyline is added	A polyline is generated between current location and clicked position. The map focused view is changed from current location into the clicked position.
Click	Click on the marker	-	The map focused view is changed from current location into the clicked position. A infoWindow is popped-up with LatLng information
Click	Click on the infoWindow	-	A dialog box is popped-up with distance information
Move	Move the currentLocation by moving geographically	The map view is moving accordingly	The map view is focused according to move direction. The LatLng of currentLocation is updated. The color of RainBall is updated accordingly.
Slide	Slide the screen	The map view is moving according to slide direction	The map view is focused according to slide direction

7 Development Procedure

Before we started the work, we learned on google map API how to retrieve the user's current location. And then we try to campare the functions and implementation requirement between iOS and Android, but found out the former needs more restrictions to test on a real phone (which is necessary) and license which is harder to get during the time, so we decided to simply use the android phone at hand, which resembles the PDA version interface and has a good maintained support and well documented part for map API.

Then we studied several location-aware applications. We understood that one unique feature of mobile applications was location awareness [1,5, 6]. As nowadays people take their devices with them everywhere, and location awareness build in the app offers users the contextual information at any location, and time. Hence the location APIs in Google Play services can help adding “location awareness” to the mobile in daily cases with very convenient location tracking, geofencing, and activity recognition, etc., we thought there is no other better options that we could take to suit our current task of building a location based service.

7.1 Implementation Steps

1 Retrieving the Current Location

We learned on google map API how to retrieve the user's current location.

In example applications, the My Location layer and the My Location button can be used to provide our user with their current position on the map. But the “My Location” layer does not return any data. So we managed to access location data in another way buy adding a location attached to google’s current map pointer.

2 Receiving location updates

By adding on Mylocation Change listener, we tried to request and receive periodic location updates, from both location providers and Google location provided info. After experiments, we found somehow Google provided location is more precise than other location providers.

3 Displaying a location

Then we learned how to convert a location's latitude and longitude into an address using reverse geocoding).

4 Monitoring distance by location awareness

We then defined several geographic areas as locations of interest, when marking the places of interest, we also changed the circle color and size w.r.t my location, and detect if the user is close enough to a place.

5 Tested by using mock locations

For this step, we did not do as Google showed in the API example; we injected mock locations into Location Services just by some manual marking on the map. In mock mode, Location Services sends out mock locations that you inject instead of sensor-based locations.

6 Provide camera update

To apply a CameraUpdate to the map, we move the camera instantly after the location markers are selected, so the view will always go back to the user.

7 separate proximity level

The level is set by adding different numerical level numbers on the location, namely, 100, 300, 500 and can be changed according to user needs. The colors indicate the proximity to my location.

8 Issues & Solutions

Table 3 Issues & Solutions in Development

Issue ID	Slogan	Status	Remarks
1 (Important)	Google Map Service Authentication Failed	Solved	Unique Google Map API V2 Development Key is mandatory for EVERY developer https://console.developers.google.com
2	Retrieving the Current Location	Solved	Use the 'Google Play services Location API' instead of 'LastKnownLocation' which may

			provide outdated location information
3	RainBall Colour Update Failed	Solved	Delete RainBall before each location updating, and then add RainBall according to updated radius
4	Fixed padding of RainBall around the screen	In Progress	Use the mathematics to calculate a radius value for each RainBall to ensure the fixed padding in screen

9 References

[1] P. Baudisch; R. Rosenholtz. Halo: a technique for visualizing off-screen objects. ACM Conference on Human Factors in Computing Systems (CHI 2003); 2003 April 5-10; Fort Lauderdale; FL. NY: ACM; 481-488; 2003.

[2] S. Gustafson , P. Baudisch and C. Gutwin , P. Irani, Wedge: clutter-free visualization of off-screen locations, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Pages 787-796, 2008.

[3] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga and J.-D. Fekete, Topology-Aware Navigation in Large Networks, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Pages 2319-2328, 2009.

[5] Google Map API V2, Android. https://developer.android.com/reference/com/google/android/gms/location/Geofence.html

[6] Ghani et al, Dynamic Insets for Context-Aware Graph Navigation, EuroVis'11 Proceedings of the 13th Eurographics / IEEE - VGTC conference on Visualization, 2011, Pages 861-870.