CityLens: A Mobile Application Utilizing Magnifying Lenses to Enhance the Navigation on Maps

Mobile Information Systems
Summer Semester 2018

Sarah Böning
Mat.-Nr. 119281
Computer Science for Digital Media
sarah.boening@uni-weimar.de

Adrian Teschendorf
Mat.-Nr. 60258
Computer Science and Media
adrian.teschendorf@uni-weimar.de

ABSTRACT

This documentation describes the idea and implementation of the mobile application *CityLens*, which was inspired by the original paper 'Sigma Lenses: Focus-Context Transitions Combining Space, Time and Translucence' by Pietriga et al. The application's intention is to enhance the user experience of navigating through digital maps with the help of various lenses. It is implemented as a web application using Javascript and OpenStreetMap as a basis for the displayed map. The code, including the apk-file and installation instructions, can be found in our Github repository [1] as well as a video of the latest prototype version [2].

CCS CONCEPTS

• Software and its engineering ~ Software design engineering

KEYWORDS

Mobile programming, Android, HTML, Javascript, Maps, Focus and context techniques

INTRODUCTION

Navigating digital maps can be done using standard techniques like *panning & zooming* for one display or *overview & detail* for multiple views in separate windows. However, both of these techniques have not only advantages, but also negative aspects that can confuse or frustrate the user, like getting lost on the map when zooming in too much or having the overview windows cover important details. Pietriga et al. discovered in their paper 'Sigma Lenses: Focus-Context Transitions Combining Space, Time and Translucence' that using *focus & context* techniques are a promising method to reduce the mentioned disadvantages of other techniques especially for the navigation on a map [3]. The authors conducted a user study to analyze various kinds of lenses which are based on the concept of magnifying glasses. In addition to standard lenses, like fisheye or a simple magnifying lens, they also created their own lenses which were enhanced with translucency and transition effects for a better usability.
Our own lens, which we named *CityLens*, was inspired the the mentioned paper, especially by the authors' conclusion that lenses are a convenient way to navigate a digital map. Therefore, we wanted to focus on creating a lens that enhances the user experience when a person has to access a map on a mobile device.

CREATING AN OWN IDEA

In everyday life, people are always in need of a map when visiting foreign places. However, these maps mostly show one specific kind of information and a person would need multiple maps to get all the information on neighboring streets, buildings, bus lines, trains, cycleways etc. With a mobile device, the user has access to all these maps but has to switch between different views back and forth. To eliminate this disadvantage, we wanted to create a map-application that uses lenses which show all these different kinds of information.
As a basis, we chose *OpenStreetMap* and its standard layer as the default map in our application. We implemented lenses, which are also based on magnifying glasses, that show OpenStreetMap's cycle and transport layers which contain information on cycleways, rest stops and bicycle repair shops in one lens, and bus and train lines in an other lens. Additionally, if the user wants to explore the standard map, there is also a simple magnifying lens that zooms in on the default view. Figure 1 shows an overview on the functions of the application.

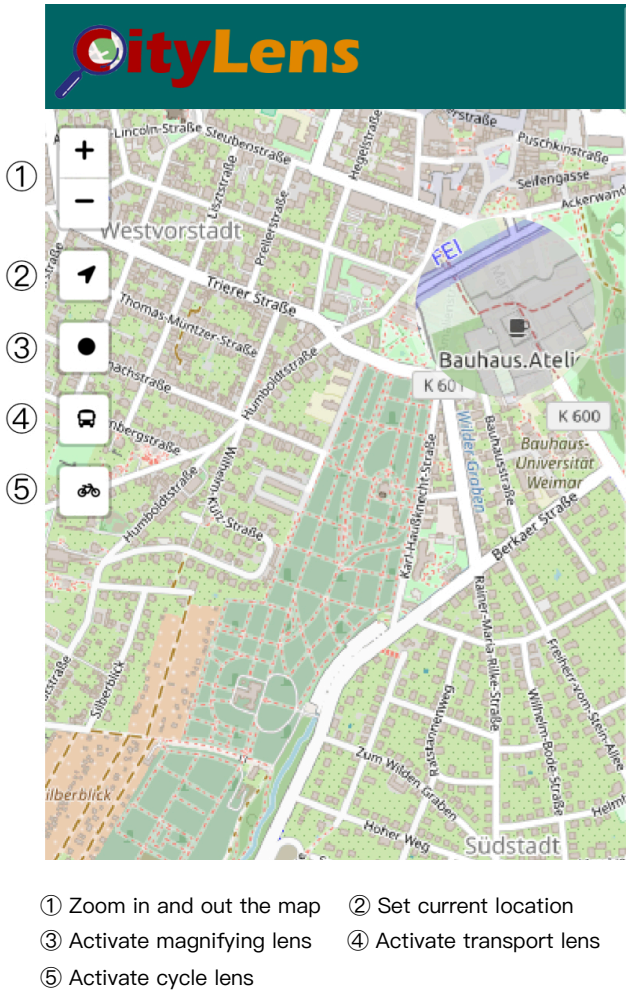

Figure 1: Screenshot of the application with description of the functions.

DEVIATION FROM THE ORIGINAL PAPER

The Sigma lenses presented in the paper were exclusively developed for usage on a desktop PC and use Carpendale's 'Framework for Unifying Presentation Space' as a basis [4]. To create the visual effects, the authors utilized the PC's graphic card for their advanced transition algorithms and used a large-screen monitor and a mouse to navigate on a given map. Since these tools are not available on a mobile device and the visualization techniques require advanced hardware, we chose to focus only on the lenses themselves and implement our own visualization techniques. We kept the idea of translucency of the original paper, but used the different map layers of OpenStreetMap to create three lenses with different kinds of context.

DESIGN CHOICES

For our basis map, we chose to use OpenStreetMap because it is an open source map service and can freely be implemented in external applications. In addition, it also includes other map layers than just the standard one. Therefore, the cycle map and transport map layers could be easily implemented. Since the lenses are supposed to be dragged over the map, we had to decide on how to implement that feature, which is usually utilized with the help of a mouse. Touching the map with one finger will set the lens to the pressed location and moving the finger will drag the lens around the map. To move the map itself, the user has to use two fingers to use this function. Because this function isn't that obvious, a notification is displayed at the bottom of the screen which tells the user how to move the map. Another problem was that a user's finger is mostly covering the whole lens while dragging it over the map. To eliminate this factor, we implemented an offset that makes the lens appear slightly above the pressed location, which can be seen in figure 2. Moreover, we chose to implement our program as a web-application which can also be used with a standard web browser and on iOS devices. However, our program is optimized for Android devices running at least Android 4.3 Jelly Bean. We chose this version, so it is compatible with the majority of people's phones [5]. Our program is a standard Android application that uses a WebView client to display OpenStreetMap and the lenses, which are implemented in Javascript with the help of the *Leaflet* library optimized for mobile devices [6]. Because we could easily customize the leaflet to fit our ideas, using Javascript appeared to be a good solution to develop our program this way. Also, a web application would need less storage space, its appearance would be similar to the desktop version of OpenStreetMap and it's easier to be kept up to date. Additionally, we created an own logo for the application to give it a unique look.

IMPLEMENTATION

The application was developed with Android Studio and uses a WebView client created in an own class, so it doesn't open the website in an installed web browser on the mobile device. When opening the app for the first time, the user is asked for permission to use the current GPS location, which can be denied or passed on to the Javascript program if accepted. Thereby, a ChromeHandler class was needed to pass the permission. We implemented an own class that works together with the asked permission, which sends an answer based on the user's choice after starting the application.

Figure 2: Implemented lens with an offset.

The website is a standard HTML page (index.html) that sets the title and defines the used libraries. The actual content of the web application is defined in the javascript file *citylens.js*. The script loads the OpenStreetMap layers, lenses, control buttons and handles the passed location permission. If the permission was denied, the user's location will be set to the city center of Weimar. The lenses are created with *Leaflet*, which is a Javascript library especially for interactive maps and optimized for mobile devices. There are various tutorials and examples for the usage of Leaflet, which helped us getting familiar with that library. One example was the creation of a simple magnifying glass, which had useful information for the creation of our lenses [7]. The lenses should not only magnify but also load a different map layer and display blending effects as well as translucency. These effects were implemented in the CSS file of the website. After the lenses were created, they needed to be made interactive. Therefore, we needed to implement gesture handlers, which was done with the help of a Gesture Handling library which could be used with Leaflet [8]. Now the lenses could be dragged around the map with one finger, and the map itself could be moved with two fingers. The used libraries had to be customized for our application. For example, if the pressed position would be the center of the lens, a users' finger would occlude most of the lens. Therefore, we changed the lens' position to an offset of the pressed position, to avoid fat finger problems. For the buttons displayed on the map, we used royalty-free icons from Font Awesome [9]. These were implemented with the help of the *EasyButton* library for Leaflet [10]. We chose icons that could easily be identified by the user, such as a bike for the cycle map and a bus for the transportation map. The circle icon describes the standard magnifying lens and the arrow is the default icon for retrieving a user's current location on most operating systems. The user can select the specific lens by touching the corresponding button. If the same button is pressed again, the lens disappears. This can be useful in case the user wants to see the whole map (context) without the details of the lens (focus).

After the prototype of the application was finished, it was tested on various systems, such as the Android Studio emulator, a Samsung Galaxy S4 Mini, a OnePlus E1003, PC web browser (Firefox 61.0) and iPhone 6 web browsers (Safari and Firefox). The installed application size is about 4.63MB, which includes all the needed Leaflet libraries that are stored offline within the app, so there is no need to download them every time the user starts the program. The data consumption after starting the application and setting the user's location was about 434KB. Also, using all the lenses and moving the map a few times added about 351KB to the data consumption.

ISSUES

During development there were minor issues regarding the lenses' properties. While the lens appeared as it was supposed to be, a circular object, on a desktop web browser and also within the Android Studio emulator, it had an ellipsoid form on real mobile devices. This was solved by editing the CSS file of the web application using trial and error until the lens appeared correct on smartphones. However, it should have been this way in the first place. A bigger issue was sending the location permission from the Android program to the web application. While it was working on the Galaxy S4 Mini with Android 4.4, it didn't work on the OnePlus with a newer Android version. It seemed, the problem lied within loading the website and checking for the location permission at the same time. We changed it, so that the application at first checks for the permission after being started, before loading the content of the website. This solved the problem and setting the user's location worked on all platforms. Some minor problems that couldn't be fixed revolve around the loading of OpenStreetMap's content. Because it is a web app, it had to download all its content from the servers of Open-StreetMap. Therefore, selecting a lens doesn't make the content appear smoothly, but the lens rather appears as a grey circle until the content has loaded and pops up. The same goes for moving the map itself, which leads to waiting a few seconds until the next tiles of the map have loaded. However, loading the complete content of an area instantly, especially in combination with the other two map layers, would consume a huge amount of data, which seemed to be a worse solution for mobile devices for us. Therefore, the compromise of having loading times had to be made.

CONCLUSION

This documentation described the inspiration for the idea and the implementation of the mobile application *CityLens*. Furthermore, issues that occurred during development are explained together with their solution approaches. The prototype of the app is working on native devices as well as emulators and different web browsers. Further steps in the development of the application may include the conduction of a user study to analyze the usability of the implemented lenses and their influence on the user experience of navigating digital maps on a mobile device.

REFERENCES

[1] Github: https://github.com/SarahBoening/CityLens
APK: https://github.com/SarahBoening/CityLens/blob/master/app/release/app-release.apk

[2] Video of prototype:
https://github.com/SarahBoening/CityLens/tree/master/Additional%20Material

[3] Pietriga, E., & Appert, C. (2008). Sigma lenses: focus-context transitions combining space, time and translucence. CHI.

[4] Carpendale, M.S., & Montagnese, C. (2001). A framework for unifying presentation space. UIST.

[5] Android Developers. Platform versions.
https://developer.android.com/about/dashboards/
last visited: Sep 18th 2018

[6] Leaflet JS. Open-source javascript library for mobile-friendly interactive maps.
https://leafletjs.com/index.html
last visited: Jun 25th 2018

[7] Leaflet Magnifying Glass example by bbecquet
https://github.com/bbecquet/Leaflet.MagnifyingGlass
last visited: Jun 28th 2018

[8] Leaflet GestureHandler library by elmarquis
https://github.com/elmarquis/Leaflet.GestureHandling
last visited: Jun 28th 2018

[9] Font Awesome Icons
https://fontawesome.com/icons?d=gallery
last visited: Jun 28th 2018

[10] Leaflet EasyButton by CliffCloud
https://github.com/CliffCloud/Leaflet.EasyButton
last visited: Jun 28th 2018

Additional tutorials:

- Android WebView Tutorial:
  https://developer.android.com/guide/webapps/webview
  last visited: Jun 21st 2018

- Geolocation with WebView:
  https://medium.com/@xabaras/android-webview-handling-geolocation-permission-request-cc482f3de210
  last visited: Jun 24th 2018