MASTER'S THESIS 2010:XX

Augmented Reality on iPhone

Full implementation of a Client-Server Application.

JEAN-LOUIS GIORDANO

Department of Signals and Systems
Division of XXXXXXXXXXXXXXX
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010

Augmented Reality on iPhone
Full implementation of a Client-Server Application.
JEAN-LOUIS GIORDANO

Master's Thesis 2010:XX

Signals and Systems
Division of XXXXXXXXXXXXXXXX
Chalmers University of Technology
SE-41296 Göteborg
Sweden

Tel. +46-(0)31 772 1000

Cover:
Artwork of the application used on the AppStore, drawn by the author.

Reproservice / Department of Signals and Systems
Göteborg, Sweden 2010

Augmented Reality on iPhone
Full implementation of a Client-Server Application.
Master's Thesis in Communication Engineering
JEAN-LOUIS GIORDANO
Department of Signals and Systems
Division of XXXXXXXXXXXXXXXX
Chalmers University of Technology

Abstract

Bla bla.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Keywords: iPhone, Augmented Reality, Realtime, AppStore, Ruby, Erlang, Objective-C, Mnesia, Google Map

Contents

Acknowledgements

1. Introduction

Augmented Reality is a dream coming true. Of course, the concept itself is not brand new and has been used and studied for many years, but modern technologies and devices now enable Augmented Reality to be implemented relatively painlessly and be used on a day-to-day basis, in a vast range of applications from advertising to medical assistance.

At the same time, the last generation smartphones are equipped with sensors that locates its users and can interact with them in cleaver and innovative ways. The iPhone is one of those. That is why Augmented Reality on smartphones, and especially on iPhone, has been growing fast for the last couple of years.

This master thesis focuses on the design of an application of Augmented Reality on iPhone 3G-S and its implementation from both a Client and Server point of view.

At first an overview over the basic principles of Augmented Reality (chapter 2) will help understanding Augmented Reality, how it works and how it can be used. Then we will have a look at the Client side of the application (chapter 3) to see what the application is from a user's point of view, and we will go through the implementation on the iPhone to understand the underlying code structure. We will also have a look at the Server implementation (chapter 4) to get an idea of the require architecture to provide data in real-time. Finally, we will describe the results achieved with the current implementation (chapter 5), and a view on further improvements will finalize the thesis.

2. Basic Principles of Augmented Reality

2.1. What is Augmented Reality?

Augmented Reality is a term that is used to describe digital systems that display an artificial visual layer of three-dimensional or two-dimensional models upon our natural perception of reality in real-time. It aims at improving or completing our limited way of seeing the real world by presenting artificial elements that would otherwise be hidden to our senses.

The most commonly known Augmented Reality application is the weather report system. The reporter is in a real environment in front of a blue/green screen, but the end user in front of his TV will see the reporter standing in from of an interactive map: the content of the background virtual data mapped on a real world object in real-time. That is Augmented Reality.

It is during the year 1992 that a Boeing researcher named Tom Caudell first defined "Augmented Reality" in a paper called "Augmented reality: an application of heads-up display technology to manual manufacturing processes" [Cau 92]. The term was used to describe a digital system that displays virtual graphics onto a physical reality.

The concept of Augmented Reality itself is older than its name, since the first Augmented Reality device was build and presented in a 1968 paper by Ivan Sutherland [Sut 68]. The system was designed to track the head position of the user to project before his eyes two-dimensional models in order to create an illusion of three dimensions. It relied on the idea that moving perspective images would appear three-dimensional even without stereo presentation, a principle which is called the "kinetic depth effect". In such a system, the user will perceive the object as three-dimensional only when he moves, otherwise the object will only be perceived as planar.

It has to be noted that Augmented Reality is not Virtual Reality. In Virtual Reality, there are no elements of Reality, as much as in Reality there are no elements of Virtuality by definition. In his paper called "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum" [Mil 94], Milgram defines the Reality-Virtuality Continuum shown in Fig. 2.1.

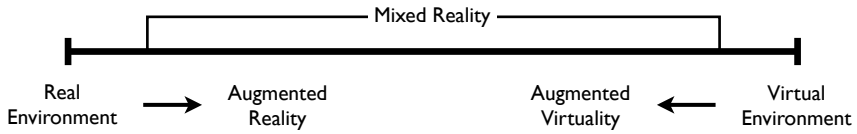


Figure 2.1.: Milgram's Reality-Virtuality Continuum [Mil 94]

As can be seen from this diagram, Augmented Reality is based on reality and tends to Virtuality in an area called "Mixed Reality". It lies near the real world end of the line, which means the predominate perception is the real world. On the symmetric position, Milgram defines Augmented Virtuality. Behind this concept lie systems where real world items added to a Virtual Environment, such as texture mapping on virtual objects in video games.

So the limits of Augmented Reality are defined by the furthest one can go at adding virtual data on a real world representation without depriving the user of its predominate perception of Reality.

2.2. How does it work?

In order to get an Augmented Reality application, one must integrate artificial object into a real scene. Of course, such an object must be rendered to scale and at the correct position and orientation. To do so, the system requires to know the position of the camera or the user from the object. This is the core principle of Augmented Reality.

To solve this problem, there are two main approaches:

- Use sensors to know the position of the object from the camera

By the mean of tags on a real object, one can get a coordinate system on which to project a virtual model. Those tags can be visual or based on any technology that can be use to locate that object, for example a set of Infra-Red diodes.

- Use sensors to know the position of the camera in a known space

If the environment is known, i.e. the application is internally aware of the position of the object, or if the object is fully virtual and does not rely on any physical counterpart,

one can evaluate the position of the camera in this environment and render the object accordingly. The position of the camera can be known by the mean of a large enough set of sensors that will be able to position it in the environment space.

Once the virtual environment is known, each virtual object must me rendered on top of an existing view by the mean of geometric projections that fits the estimated position of this object in the virtual space.

2.3. What is it for?

Applications are numerous and are invading a greater and greater number of domains: entertainment of course, with video games or virtual scavenger hunt, but it also has medical uses and various advantages for the industry.

3. On the Client Side

3.1. The iPhone

3.1.1. The device

The iPhone is a last generation smartphone developed by Apple and was revealed for the first time in 2007. It features advanced functionalities such as multimedia support, internet access, video games and many more. Up to this day, forty-two millions iPhone have been sold across the globe.

Its main characteristic is that its User Interface (UI) is almost only based on two inputs:

- a large multi-touch screen
- a 3-axis accelerometers

As such, the iPhone does not posses any physical keyboard. Instead, a virtual keyboard can be summoned any time it is necessary, leaving a larger screen for applications.

The last generation of iPhone, the iPhone 3G-S, also includes a GPS and a compass, and an upcoming version of its Operating System (OS) will allow the access to the video data of the camera.

In order to install new applications on an iPhone, one must download it from the AppStore, the internet-based retail store from Apple for iPhones, iPods and the iPad. Apple having a highly proprietary policy on its product, this is actually the only way to distribute iPhone applications.

3.1.2. The Development Tools

To develop an application for iPhone, a specific Framework is required.

First of all, any software developed for iPhone must be programmed in Objective-C, although it is possible to call C and C++ functions from the code.

Objective-C is an Object-Oriented (OO) reflexive programming language build upon the C language. It is comparable to C++ from this point, but differs greatly in many ways, especially by its dynamic message passing system, by being weakly typed and by being able to perform dynamic loading. In its latest version, Objective-C also features a Garbage Collector which abstract the programmer of the memory management consideration. Unfortunately, this feature is not available for iPhone development.

In order to compile code for the iPhone, the use of Xcode as Integrated Development Environment is almost unavoidable. Apple has a highly proprietary approach for its products, and programming for an Apple environment is much restrictive to this regard. Fortunately, Xcode and the set of tools provided by Apple offer a great comfort of use in many cases, especially for debugging, or for creating interfaces with the use of an Interface Builder (IB).

Once Xcode is set up, the iPhone Software Development Kit must be installed on Xcode. The iPhone SDK takes advantage of all the features of Xcode, including its set of external tool to the largest extent. After being registered to the iPhone developer's program, Xcode can also be linked directly to an actual iPhone device in order to test an application in real-time with any monitoring tool available.

This SDK contains Application Programming Interfaces (API) to all the accessible libraries of the iPhone, including Cocoa for the user interface. Unfortunately, many functionalities such as the access to raw data from the video camera are not part of a public API, so their libraries are considered as not accessible and therefore are forbidden by Apple.

3.2. The Application

3.2.1. What it does

This application has been designed for travellers in Sweden that are looking for a nearby public transport stop. Augmented Reality makes it very intuitive to find a nearby stop: the user just has to ”look around” with his iPhone, and the closest stops will appear at their positions. To make it even easier for the user, a Google Map is available when the device is held horizontally if directions are required.

This makes the application very interesting for people that are easily confused when reading maps when discovering a new city.

But this application has also been designed for people that are already familiar with

the city they visit. Simply by pointing at a stop they are interested in, they can get the next departures without having to go till the stop to check the timetable.

For now, the application is available for Göteborg and Stockholm, with their respective public transport companies Västtrafik and Storstockholms Lokaltrafik. But additional providers could be added later on.

3.2.2. How it works

The application takes advantage of the GPS capabilities of the iPhone to locate the user on a map. Then thanks to the compass, we are able to estimate the direction he is looking at, and eventually the 3-axis accelerometer allows us to evaluate the angle at which he holds his phone. Note that this application requires a GPS and a compass, and therefore is only compatible with the iPhone 3G-S.

Once the position of the user is known, a request is made over the internet to determine the bus stops that are close to him. The answer to this request will be a list of stops with their names and locations together with a forecast list for each of them.

An item in the forecast list is composed of a line number, a destination and a set of attributes indicating the time of departure in some information on the quality of the travel.

Once the positions of the bus stops are known, we can project them in the virtual space according to the user's coordinates, heading direction and phone holding angle.

To cope with the precision errors of acceleration and heading measurement, a buffer is used to get the average values on an arbitrary period of time before updating the knowledge on the user's position. Animations are also used to make transitions between two views appear smoother.

3.3. Implementation of the Augmented Reality View

3.3.1. View Hierarchy

When developing for iPhone, the Model-View-Controller (MVC) is the best architecture to go with. There are well defined classes for views and controllers in the Cocoa framework, so we took advantage of that when designing our application.

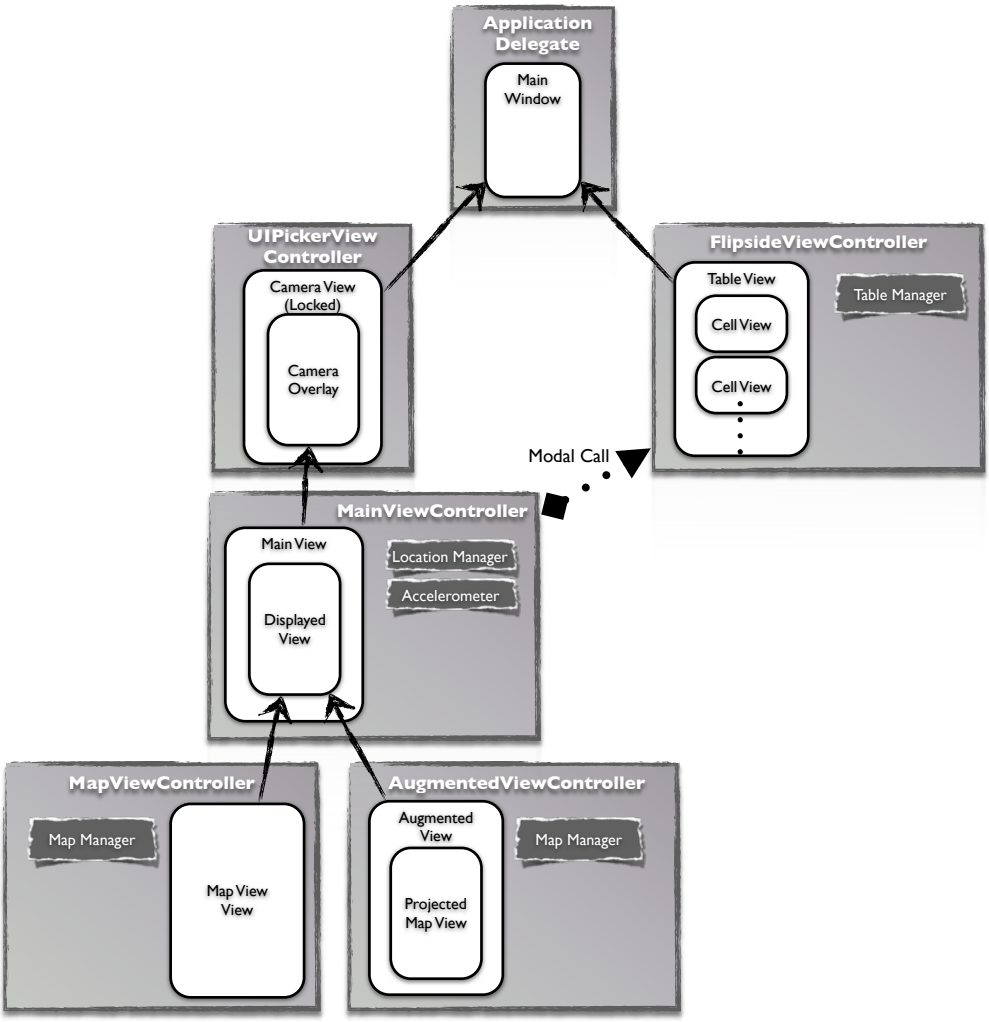


Figure 3.1.: Scheme of the View Hierarchy of the application

When using the Cocoa framework, each view must be a subclass of the UIView class, and each controller of the UIViewController class. A view is automatically linked to a view controller.

A view can contain other views that will be considered as its subviews, and each view has a frame in which it renders itself. Views can have transparent or semi-transparent backgrounds, so it is possible to create a complex view hierarchy without hassle.

Figure 3.1 shows the view hierarchy for our application.

In order to render a view on screen, one must define an application window that is

a subclass of UIWindow and that will take care of rendering every visible view on the iPhone's screen. By definition, a UIWindow must be the higher node in any view hierarchy.

Once the application's window is defined, one should know that there is a serious restriction on the iPhone SDK up to the version 3.2 which prohibits to insert a camera view (UIImagePickerView) as a subview to a UIView. Instead, it must be directly put into a UIWindow. Since we want to use the camera in our application, we have no choice but to put it on top of our view stack. This is semantically very wrong, but there are no alternatives left here by Apple.

Furthermore, the camera view itself is locked, so we can't help but use a view called "Camera Overlay" to display any view on top of it. That is ugly and clearly opposed to the concept of our application, but so far there is no other way to do it.

As a result, the camera is always rendering in the background of our application, and everything else is rendered on top of it.

Now we have a window and a background with a camera view, and we want to be able to switch between a Map View and an Augmented Reality View. To do so, we implement a Main View that will display either of them according to the iPhone position. In order to avoid redundancy, all the data of the application is kept in the Main View Controller and dispatched in its subview, be it a Map View or an Augmented View. The same goes for the acceleration and location data.

To present the forecast of a stop, a "modal view" is opened by the Main View Controller. The display of a modal view will replace the content of the actual window by a temporary view. When dismissing this view, the previous content of the window is loaded again and the application continues.

This modal view is a table view itself composed of view cells that are respectively subclasses of UITableView and UITableViewCell. Those view cells are used to display the forecasts list for a bus stop.

3.3.2. Positioning within 6 degrees of freedom

In our application, we need to locate the camera of the iPhone within 6 degrees of freedom:

- Latitude

- Longitude
- Altitude
- Azimuth
- Roll angle
- Pitch angle

The GPS directly gives us the Altitude, Latitude and Longitude of the user whereas the compass gives us his Azimuth. This directly takes care of 4 out of the 6 degrees.

But thanks to the 3-axis accelerometer, the roll angle and pitch angle can easily be computed by the mean of simple trigonometry as shown in Figure 3.2

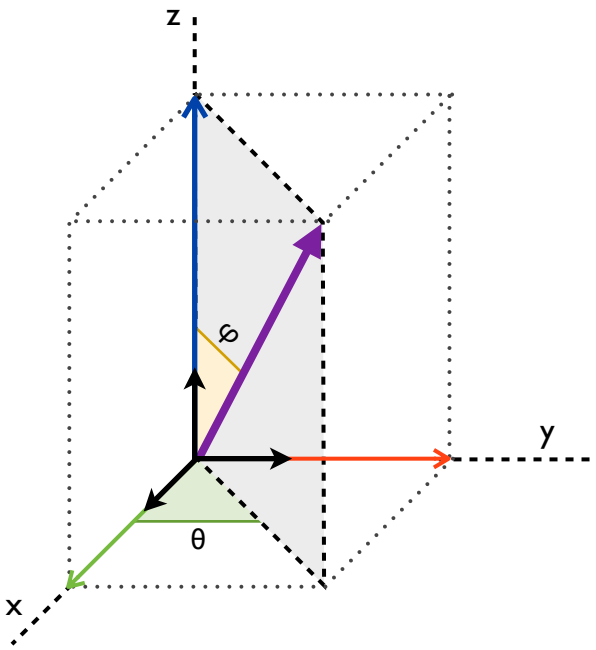


Figure 3.2.: Projection giving the Roll and Pitch angles, respectively θ and ϕ

3.3.3. 3D Projection

Now that we know the position of the camera relatively to the object we want to display, we need to project our knowledge on the Augmented View.

First, we project a Map on the plane defining the ground. The iPhone has an API to perform 3D projections, by the mean of layer transformations. To apply a transform to a layer, one must provide the transform matrix corresponding to the desired projection.

In this case, the transformation will be a rotation of $\phi = \pi/2$ on the X-axis and another θ corresponding to the Azimuth on the Y-Axis, followed by translations t_y and t_z on the Y and Z axis to give depth to the view, which gives the following matrix:

$$\begin{pmatrix} \cos\theta & 0 & -\sin\theta & -\sin\theta/e_z \\ \sin\phi\sin\theta & \cos\phi & \sin\phi\cos\theta & \sin\phi\cos\theta/e_z \\ \cos\phi\sin\theta & -\sin\phi & \cos\phi\cos\theta & \cos\phi\cos\theta/e_z \\ 0 & t_y & t_z & t_z/e_z \end{pmatrix}$$

where e_z is the distance of the user from the projection plan. Since we have $\phi = \pi/2$, the transformation matrix becomes:

$$\begin{pmatrix} \cos\theta & 0 & -\sin\theta & -\sin\theta/e_z \\ \sin\theta & 0 & \cos\theta & \cos\theta/e_z \\ 0 & -1 & 0 & 0 \\ 0 & t_y & t_z & t_z/e_z \end{pmatrix}$$

Once the map is projected on the proper plane, we can project the buttons representing the bus stops. First, we need to compute the distance and the azimuth of the stop relatively to the camera.

To compute the distance d, we use a simplified version of the great-circle formula as shown in Equation 3.1 where ϕ_s and λ_s are the Latitude and Longitude of the standpoint (camera position), ϕ_f and λ_f the ones of the forepoint (bus stop location) and R is the average radius of Earth ($\approx$ 6371km).

$$d = R \times \arccos\left(\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos(\lambda_f - \lambda_s)\right) \tag{3.1}$$

To compute the azimuth, we use another formula

4. On the Server Side

4.1. Architecture

In order to retrieve information about nearby stops, the client must request a remote source of information by calling a server. The server that has been implemented in our case acts as a middleware between the public transport providers (Västtrafik and SL) and the client. By doing so, requests to the different providers are transparent to the client.

The server processes a request for nearby bus stops as follows:

1. it receives a request with coordinates (Latitude and Longitude)
2. it finds the 10 nearest bus locations from the given coordinates in its SQLite database
3. it fetches the forecast for each of these stops in an independent lightweight process
4. once each forecast is obtained, it parses the result into JSON format
5. finally, it sends a reply to the client containing the JSON data

In order to handle high quantities of lightweight processes, the best solution was to use Erlang. Erlang is a functional programming language especially designed to handle high concurrency and distributed systems. A large part of the application is therefore implemented in Erlang in order to take advantage of this.

Figure 4.1 shows in details the request processing on an UML sequential diagram, the grey area being our server.

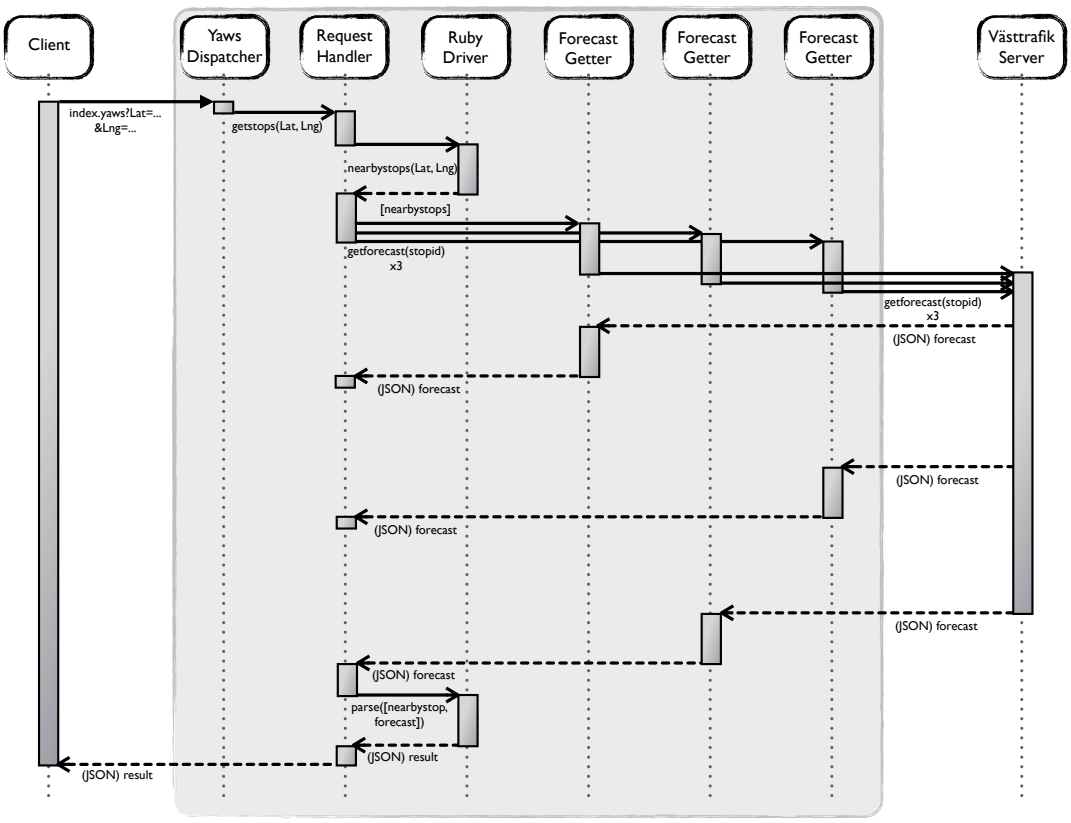


Figure 4.1.: Simplified UML Sequential Diagram of a request

The application is hosted on a Debian machine (Ubuntu) with a Yaws webserver. Yaws is the Erlang alternative to Apache, and as such offers high availability and high scalability.

But to make it easy to maintain and update, the more complex tasks are implemented in Ruby. Ruby is a modern dynamic scripted language, and has been designed to make the developers happy. As such, it is a powerful tool to implement complex algorithms painlessly.

Figure 4.2 shows the details of the server's architecture.

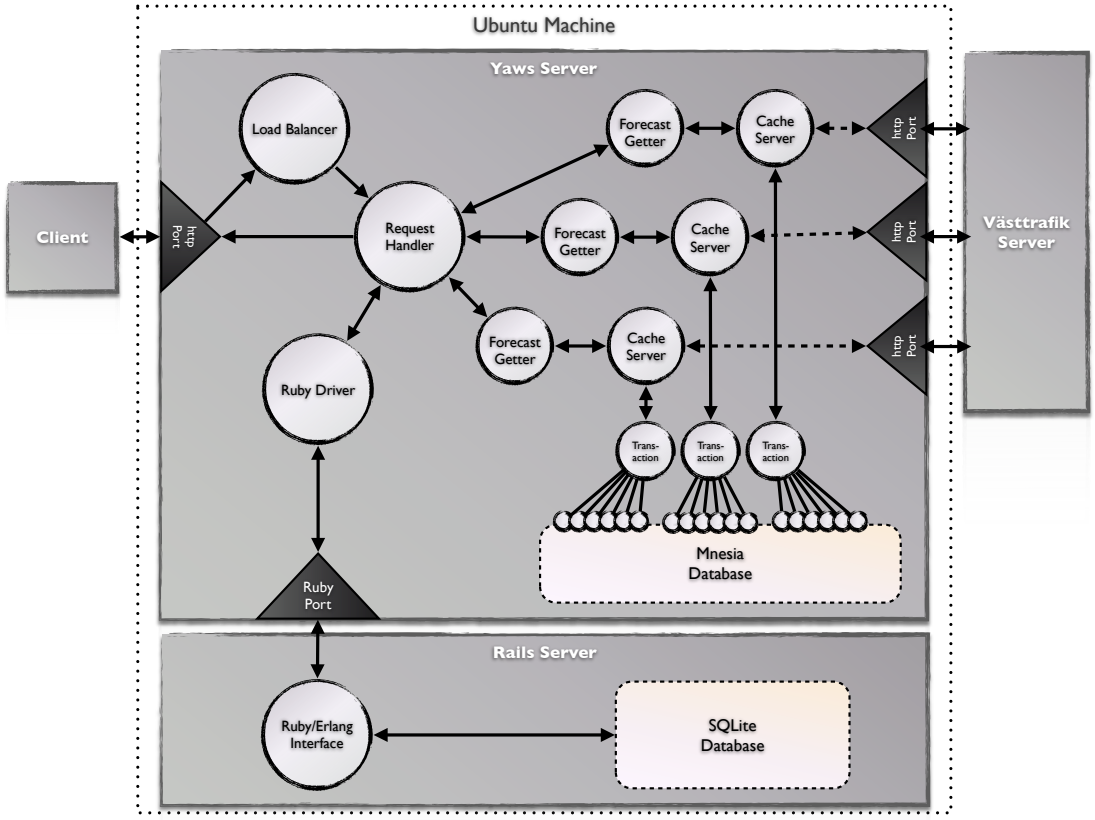


Figure 4.2.: Scheme of the Server Architecture

4.2. Implementation

4.2.1. Erlang

Erlang is used to dispatch the lightweight processes, to perform the distant requests to Västtrafik and Storstockholms Locaktrafik, and also to take care of data caching by the mean of a Mnesia database.

4.2.2. Ruby

Ruby is used to implement the SQL request to the database and to parse data. It is also used to maintain the database. The SQL database contains information on all the bus stops for each provider. Each entry in the database has the following attributes:

- stop_name : the name of the stop
- lat : the Latitude of the stop
- lng : the Longitude of the stop
- provider_name : the name of the stop's provider (Västtrafik or SL)
- stop_id : the id of the stop as defined by its provider

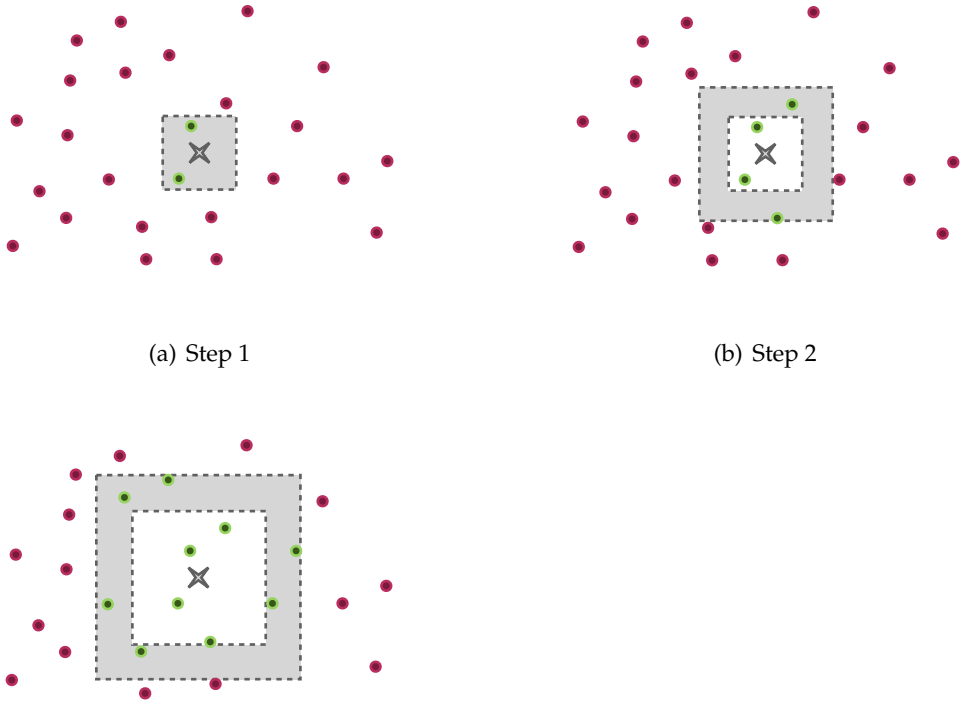
This database is populated by a background process that fetch data from the providers on monthly basis (it is not that often that new stops are build or that old one are destroyed).

The "lat" and "lng" attributes are the key entries allowing geographic search in the database, but finding nearby points from a database can be a tricky problem. In our case, we perform queries iteratively as shown in Figure 4.3 until a satisfying number of stops are returned.

Once we get a list of stops within an arbitrary large area, we need to sort those stops by distance in order to keep only the 10 closer ones. The distance d between the user's location and a stop is computed according to a simplified version of the great-circle distance formula given in Equations 4.1 and 4.2, where ϕ_s and λ_s are the Latitude and Longitude of the standpoint, ϕ_f and λ_f the ones of the forepoint, with $\Delta\hat{\sigma}$ defining the central angle of those two points and r being the average radius of Earth ($\approx$ 6371km).

$$\Delta\hat{\sigma} = \arccos\left(\cos\phi_s \cos\lambda_s \cos\phi_f \cos\lambda_f + \cos\phi_s \sin\lambda_s \cos\phi_f \sin\lambda_f + \sin\phi_s \sin\phi_f\right) \quad (4.1)$$

$$d = r \times \Delta\hat{\sigma} \quad (4.2)$$



(a) Step 1

(b) Step 2

(c) Step 3

Figure 4.3.: Scheme of the SQL queries on the Ruby side

4.3. Results

Everything works fine.
(give numbers)

5. Conclusion

5.1. On the AppStore

The first release of the application in its revision number 0.9 was uploaded under the name ”Hållplats Väst”on the AppStore on March the 8^{th} and was ready for sale on March the 13^{th}.

After one month on the store, more than 400 units were downloaded.

5.2. Further Considerations

5.2.1. on the Client

Implement image recognition.

5.2.2. on the Server

Extend to more cities in Sweden or even Europe.

5.3. Summary

References

[Cau 92] Caudell, T.P. and Mizell, D.W: *Augmented reality: an application of heads-up display technology to manual manufacturing processes*, Res. & Technol., Boeing Comput. Services, Seattle, January 1992

[Sut 68] Sutherland, I. E. , *A head-mounted three dimensional display*, Proceedings of the December 9-11, 1968, fall joint computer conference, part I, San Francisco, California, December 1968

[Mil 94] Milgram, P., H. Takemura, et al., *Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum.* SPIE Proceedings: Telemanipulator and Telepresence Technologies . H. Das, SPIE. 2351 : 282-292, 1994

A. Some Code