Aida Hailemichael

Augmented reality using JavaScript

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

13 May 2013

Helsinki Metropolia University of Applied Sciences

Author Title Number of Pages Date	Aida Hailemichael Augmented Reality using JavaScript 39 pages + 1 appendix 13 May 2013
Degree	Bachelor of Engineering
Degree Programme	Bachelor of Media Engineering
Specialisation option	Web Application Development
Instructors	Kari Salo, Principal Lecturer Erik Pöysti, Executive Director

The project goal was to provide a mobile application, for a venue called Kulturhuset Karelia which is located in Tammisaari Finland. In this paper, the concept of Augmented Reality technology is briefly discussed along with the mobile application for the venue. The utilisation of JavaScript for creating Augmented reality content for mobile Augmented reality browser is also demonstrated.

The application was created by using Architecht API which is Jacvascript library based on the Wikitude world browser. Wikitude world browser is nowadays one of the most popular Augmented reality browsers for mobile phones.

The development of the application was to demonstrate the possibility of utilising simple web technologies such as HTML, JavaScript and CSS for developing Augmented Reality application for mobile Augmented reality browsers. The application was implemented by using Wikitude ARchitecht API for creating the Augmented Reality features, together with additional JavaScript libraries JQuery mobile for creating the user interface and Google Maps API for creating navigational capability of the application.

Keywords	Augmented reality, AR browser, ARchitect, JavaScript

Contents

1 Introduction

It is not long ago that augmented reality applications were behind closed doors at their experimental stage. Today however, they have reached a point that they can get the attention of an average user. The significant change towards this new application resulted from the development in mobile computing. Although many restrictions in the technical part are still immense, more new innovations for implementing fresh ideas into the reality have emerged.

Importance of Augmented reality technology in the future have been predicted through many researches. The current state of the technology does not allow to exploit the full potential of what is generally known as Augmented Reality. However, each step towards increasing the current computing power will bring the technology on step closer to accomplish its idealistic features. Also the competitiveness nature of the commercial use the technology is an additional ingredient to speed up its development stages.

The main goal of this thesis is to demonstrate the ability of augmented reality applications to interact with dynamic data by utilising basic web technologies such as HTML, CSS and JavaScript. However, in the project part of this thesis PHP another scripting language have been used. The final application is developed for a venue called Kulturhuset Karelia located in town of Tammisaari in Finland. The application is for example used to showing upcoming events, getting direction to the venue and checking availabilities of ticket. The application retrieves data dynamically from a database and lets the user to interact with it.

2 Overview

In the past it has been a bit ambiguous to determine one phenomena as Augmented Reality(AR) or not. The residue of this ambiguity still lingers among the technologies developers. [2] Eventually AR was first given a clear definition as being a variation of virtual environments (VE). VE puts the user in a computer generated world by preventing the user from interact with the real world. Unlike VE, AR supplements the reality rather than replacing it. AR combines the real world with computer-generated information. Hence, AR can be seen as a blend of real and virtual reality. [1; 2]

Even though, the exact demarcation for a technology to be AR or not is still not clear. However, to categorize a technology as AR it has to:

- Combine real and virtual.
- Provide interaction and tracking of objects in real time.
- Provide real time data interaction.
- Provide recognition of images or objects.
- Operate and used in 3D environments.

This chapter reviews a brief history of AR, as it is the basic building block for this thesis. It emphasizes on the crucial points to the development of AR technology. It goes further and explains how it has been developed by giving some examples and what is new in the technology. The aim of the chapter is not to explain each and every research done on AR, but rather to give a general understanding of what AR is.

2.1 Brief history of Augmented Reality

The history of Augmented Reality (AR) dates back before the term Augmented reality existed. In the year 1962 Morton Heiling designed the Sensorama as one of the immersive, multi-sensory technology that had visual, sound, vibration and smell. The machine gives the sensation of being in the actual place when watching a movie. This technology was the first point in history when the idea of AR was introduced to the world. Following this idea after six years later, the first AR and VE system was developed by Ivan Southerland who used an optical see-through head mounted display and two different trackers. Due to the limited computing power of computers at that time, only limited wireframes of drawings could be displayed in real time. In 1975 a system

called Videoplace, created by Myron Kruener allowed users to interact with virtual objects for the first time. [1, 7-8]

In 1992 Tom Caudell and David Mizell coined the term Augmented reality. Some say that this was the year that the AR was born. They developed a system for their R&D project to simplify the production of Boeing aircraft mainly overlaying the position of certain cables that has to be placed in the building process. [1, 8]

Shortly after the name Augmented reality have been introduced in 1995 NaviCam was created by Jun Rekimoto and Katashi Nagao. The system used a powerful workstation that had a camera attached on the mobile screen. The system introduced the use of markers. Markers are what the computer identifies as the place where digital information is to be presented. The system used colour- coded markers by processing live video feed from the camera and placing context based information on top of the live video feed. [6]

Even though the term AR was coined, there was no kind of definition to present the technology. In 1997 Ronal Azuma who presented the first survey in AR, produced a widely accepted definition using three fundamental characters of AR.
The three fundamental characters being:

It combines real and virtual.
It is interactive in real time.
It is registered in 3D. [25]

In the same year the Touring machine which was the first Mobile Augmented Reality System (MARS) presented by Steve Feiner. The system provided the user with a freedom to move into different places.

Coming to the year 2000 the first application to utilize Global Positioning System (GPS), vision based tracking and digital compass was presented by Bruce Thomas [8, 5]. Three years later, the first AR system called the indoor AR guidance, which runs on a personal hand held device and used windows mobile port ARToolKit, was developed. This system was the earliest step of AR technology towards the current implementation of AR in Smartphone. [3]

Because of their flexible functionalities, mobile phones became an ideal tool for production of AR technologies. From the year 2004-2006 series of experiments on, the capa-

bilities of mobile phones in rendering digital information in real time has been done [1, 13]. In 2008, Mobilizy launched Wikitude world browser, an application that combines GPS and compass data with Wikipedia entries. The application overlays information on the real time camera view of an Android smart phone. [7]

The history of AR from being just an idea to becoming a tangible application, shows that the future development of the technology coincides with that of the computing capabilities of mobile phones. It shows that this technology has just began its journey and in the future the development, depending on future technological scopes, it might change its course to a different direction.

2.2 Current trends in Augmented Reality

For a technology which is still in the development stages, most of AR applications have been developed for research purposes. Due to the rapid growth of smart phone's technologies, the idea of using AR becomes clear. AR has been used to aid people in visual guidance in assembly, maintenance and training. Today however, it has transformed into a widely and highly visible platform for delivering rich media experiences. For instance, in print media AR has been used to link 3-dimensional graphics and videos to printed publication. [4]

Current information retrieval and crowed sourcing applications are rapidly being adopted by the mainstream market. The market for AR applications also has been growing with the estimated download of more than 400 million in 2014 [18]. Being platform independent, AR browsers are more viable for developing sustainable application. Thus, many companies developing new application and framework, make it possible to create and publish AR content with minimum level of programming knowledge. [4]

Current smart phones, with the combination of fast computer processing unit (CPU), graphics hardware, large screen, high resolution cameras and sensors such as GPS, compass and gyroscopes, which makes them the perfect tool to experience AR. In previous times researchers have shown how computer vision based AR applications can be implemented on mobile phones. Recently, commercial applications such as AR browsers show the use of GPS and compass sensor data to experience AR. [5;4] Today AR has been used in many sectors using mobile phones to deliver the application.

Applicable areas of Augmented Reality

Even though there are many ways to utilise AR technologies, one of the ways to use AR is for urban exploration. For instance Nokia's city lens, Wikitude and Junaio are a few of AR applications used to explore cites. Instead of using a regular map, an application on the phone using those AR applications would benefit the user from their visual display of the real world. Whether it is a restaurant or a pub that the user is looking for, or just to know the surroundings of the new city one can look into the mobile phone and see what is around.

For indoor application for instance in museums, instead of using the regular audio guide while visiting a museum, it is much more interesting to use AR application. A good example would be an AR application developed by using Junaio mobile Augmented reality browser to display visual information about any particular art in museum. This application was used in the British museum using indoor navigation system. [9]

Educational application of AR can be seen in medical sectors especially during surgical training. One case for this scenario is the solution called ProMIS from CAE, which is used to train surgeons by giving them different approaches for scenarios that they are learning. The system makes the training more dynamic and simple for the student, while cutting cost for having many simulation tools. [11]

Application of AR is not limited on the above mentioned areas, it has been used by big commercial companies for advertisement purposes. Amazon, Logo, Nissan, Toyota and many big companies have been using AR technologies to advertise their products. For instance, Lego used a system in retail stores to display the readymade product which is inside the box. [10]

3 Augmented Reality Browsers in depth

AR browsers, in addition to providing with visual information to the user they also provide the same functionalities as any common mobile browser could provide. For a mobile phone to be capable of running AR browser, the minimum requirements can be camera and GPS or either one of them depending on the type of browser. Some browsers use only camera and screen for instance, and an application might use an

image as a trigger to overlay the additional information. In other cases the application only utilizes GPS as a trigger to publish the virtual information. [2, 22]

As any other generic browsers, AR browsers use HTTP (Hypertext Transfer Protocol) agents and consume content from the web. Generic browsers have standardizes mark-up languages like XML, XHTML and CSS which gives them the ability to render and consume any web content regardless of their being different application. The lack of generally agreed standardised mark-up languages in AR browsers prevents them from rendering and consuming the same web content. It means that when a developer or content publisher wants to publish something, the content has to be targeted for only one specific AR browser. [4, 25]

The next chapter focuses more on mobile AR browsers, particularly on content publication using available tools. It also discusses a specific browser and its functionalities as well as development environments from the developer's point of view.

3.1 Basics of Augmented Reality browser

In order to fully understand mobile AR browsers some basic AR concepts will be discussed. The following terms are defined because of the wide range of AR technology and to narrow the area of its focus.

`Reality view`: this represents the live video feed from the mobile phone's camera. The video is similar to that of the regular camera application in the phone. The video is utilized in creation of the real world surrounding of the user to overlay the virtual object, which in combination create the augmented view.

`Tracking and registration` : this refers to the method for overlaying the virtual object with 3-dimensional co-ordinate in the reality view. Object tracking is done either with location sensors such as GPS and compass or image recognition system or sometimes it is done using both systems.

`Point Of Interest(POI)`**:** refers to a single data item which is associated with geographic location ,LLA(Longitude, Latitude, Altitude) or visual markers to be ren-

dered by the AR application. The output content to be publication is not part of the POI instead POI is a link to the output content.

`Virtual object`: refers to the digital content such as 3-dimensional model, 2-dimensional image, icon or text, that can be rendered by the AR application and overlaid on the reality view.

`Channels, Layers and Worlds` : all are the final publication of a related POI and associated virtual objects, which most of the time provide access to all the contents of a single content publisher. Or in other words those can be web applications that can be browsed using AR browser.

`Marker based AR` : for overlaying virtual objects using image recognition method, two different systems can be used marker-less and marker based. Where marker based AR system uses natural feature detection or 2-dimentional matrix, to identify the exact real world object such as book covers, posters or land marks in order to recognize objects. After the object is being recognised, the virtual information is usually displayed on top of the marker or the natural feature when seen on the reality view.

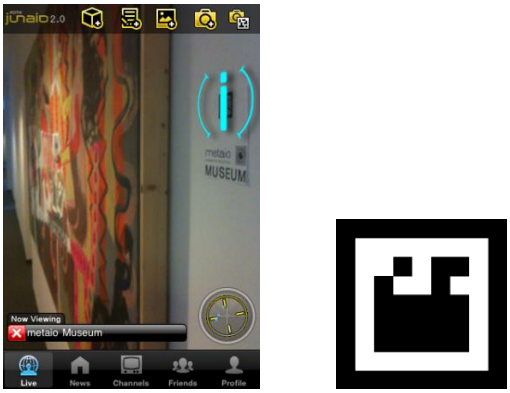

Figure 1. Implementation of 2d matrix code by Junaio. [4]

In the case of using either location detection or image recognition, figure 1 illustrates a common use of 2-dimensional matrix as a trigger to overlay the virtual information. The application was used by Junaio which is a mobile AR browser to give additional information in the museum.

`Location based AR` : is a tracking system based on geo-location, information obtained from device's location sensors. Unlike optical tracking systems, location based tracking is less accurate and only works in outdoor environments. [4, 2-4]

Figure 2. AR browser example. [14]

Figure 2 presents a simple AR browser. It demonstrates what all AR browsers can have in common when describing the anatomy of AR browsers. Number 1 indicates a traditional map view which sometimes used during navigation instead of the camera view. Radar which is represented by number 2 presents visual location guidance to where the POI is located and guides the user in which direction the mobile device should be directed. Number 3 represents the virtual object that visually indicate what POI represents in the application. In this application cars are displayed but in some AR browse services POI can be represented by what is called info bubbles for instance, knife and fork sign for food or house for infrastructure. The information bar indicated by number 4 illustrates a short description about a single user selected POI represented by virtual object or info bubble. The user might be directed to a web page, a text or a file, if the information bar is further selected depending on the browser functionality. Number 5 indicates the number of POI that are available in the given radius. The user can define the search range by setting the distance in which the result of the search is required by making it more relevant to the current location. [2, 25]

3.2 Examples of mobile Augmented Reality browsers

Junaio

Junaio is one of the famous AR browsers currently used. It was originally developed as an AR social networking browser by a German company Metaio. Its original features were targeted to enable users to post 3-dimensional scenes to a virtual world and share it among the users. After its second revised realise, it fully transformed to an AR browser. Despite its changes it still maintains the social touch.

Junaio's features are the combination of 3-dimensional object rendering, location based tracking, and both marker and marker-less image recognition. Also for development purposes Junaio offers developers its API (application programming interface) called Junaio Glue, which is used to anchor 3-dimensional objects to marker or marker-less pattern. Developers can incorporate the API into their applications to include some of Junaio's features into their application. Junaio also offers features to publish and host services for its browser. Those features include simple POI, sound and video, 3-dimentsional and animation, proximity detection, natural feature tracking and along with for indoor application it provides LLA(Latitude, Longitude, Altitude). [2, 44-45]

Figure 3. Junaio's Channel structure. [15]

As shown in figure 3, a typical channel in Junaio is comprised of three components as table 1 shows below.

Table 1. Components of a channel.[Idea from 15]

AREX XML	Defines type of channel and refers to the static return of the 3rd party server
AREX JS	JavaScript part which is responsible for interaction and animation of objects.
HTML 5	Enables implementation of additional user interface elements like buttons.

Layar

Similar to Junaio Layar is also among the well known AR browsers, it even comes pre-installed on mobile phones, for instance, on Samsung Galaxy phone. Within a short period of time Layar has become the dominant AR browser in the world. The application presents 3-dimensional object rendering and location based tracking features. Layar also provides API for developers and offers 3rd party content management tools and hosting companies for content publisher to publish their own content. The platform has five components that can be found in any basic Layar platforms. Thos components include the Layar reality browser, Layar server, Layar Publishing site, the 3rd party service providers and Layar content sources. Layar content sources includes for instance Youtube.com and Fliker.com.[16; 4, 6]

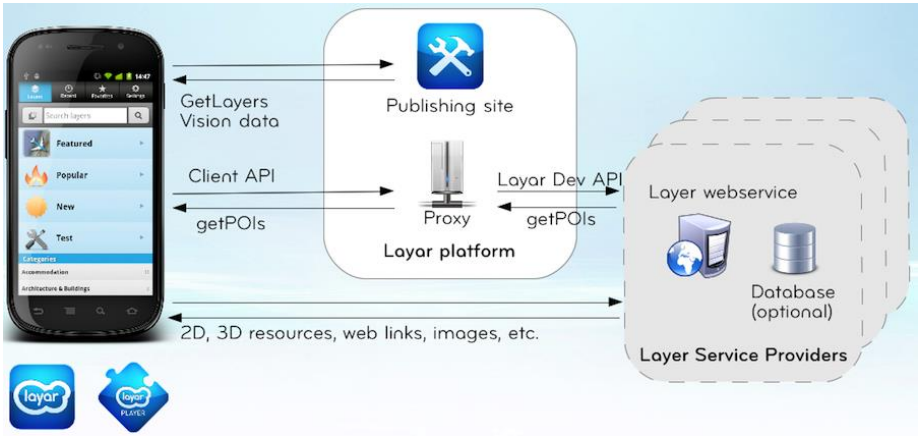

Figure 4. Work flow of Layar platform.[16]

The work flow shown in figure 4 shows how information is processed when a user is interacting with Layar platform.

3.3 Wikitude and its development tools

The very first AR browser to be launched in the world was Wikitude. It is also the easiest application to create content with. Wikitude provides location based tracking, image recognition and 3-dimensional object recognition services. To develop content and publish it in Wikitude world, the basic requirements are a mark-up language such as KML or ARML, web Services and an API or using Google collaborative maps interface.

As mentioned previously, AR browsers lack well adopted standardised mark-up languages. Before going any further into the next topic, it is crucial to know which mark-up languages Wikitude is using to define POI on the map to utilise them on the browser. [20]

3.3.1 Keyhole Mark-up Language

KML (Keyhole Mark-up Language) is XML based file format used to display geographic data using images and coordinates. It has been used by Google Maps and Google Earth.

The KML elements that Google map is utilising includes:

- Placemark with <name> elements
- Icons
- Points
- Folders
- HTML elements such as: image, list, table, etc.
- KMZ which is a compressed format of KML
- LineStrings and Polygons
- Styles for polylines and polygons.

Placemark

Placemarks used to mark a position on the surface of the earth. Their main role is to symbolize and display POI. They also govern the interaction of the user with geographic features such as points, lines or polygons. Information about each feature is available through each placemark's feature.

For instance the object placemark which is used to mark a place or position on the surface of the earth can be implemented in KML as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
   <Placemark>
      <name>Simple placemark</name>
      <description>Attached to the ground. Intelligently place itself
      at the height of the underlying terrain.</description>
      <Point>
         <coordnates>-122.0822035425683,37.42228990140251,0
         </coordinates>
      </Point>
   </Placemark>
</kml>
```

The code above demonstrates the use of the element placemark in the KML file format to mark the position represented by the coordinates (-122.0822035425683, 37.42228990140251,0) on the earth's surface using a commonly used yellow pushpin as an icon. Since KML is based on XML standards, the document used xml as a header to define standard. [17]

Icon

Represents an image to be used as an icon for the placemark.

```
<Icon></Icon>
```

The icon tag is represented in the KML file as the code shows above.

Wikitude developers have used KML to create wikitude worlds and experience AR. By using Google Earth and acquiring the KML file, any one with the ability to use computer can create wikitude world. To finally publish the KML file into the Wikitude world, the user can go to the developers site and fill out the create world form and submit it. [2, 71]

3.3.2 Augmented Reality Mark-up Language

ARML (Augmented Reality Mark-up Language) is a descriptive KML based data formatting, specifically targeted for mobile AR applications. It was first defined by Mobilizy the same company that created Wikitude, to facilitate developers to create content for AR browsers. ARML implements the concept of KML and AR browser. Since KML is a

very rich standard, it covers the fundamental tags for location based application. However, ARML reduces the use of KML tags and adds some additional features to meet the context of AR. [19]

A simple ARML file consists of two parts, one which defines the content provider for instance Wikipedia the other defines POI. An example of an empty ARML file is shown with the code bellow.

```
<?xml version="1.0" encoding="UTF-8"?>
   <kml xmlns="http://www.opengis.net/kml/2.2"
      xmlns:ar="http://www.openarml.org/arml/ext/2.2"
      xmlns:wikitude="http://www.openarml.org/arml/ext/2.2">
      <Document>
         <ar:provider>
         </ar:provider>

         <Placemark>
         </Placemark>
      </Document>
   </kml>
```

The code above shows an empty ARML file comprised of its two basic parts and tags. The content provider section which contains information about a single content provider which includes its name, description, logo, webpage that can be further shared among POIs of the content provider. The second section which starts with <Placemark> tag is linked to a single content provider and comprised of essential AR data such as: location, name, description, phone number, email, etc. [18]

Table 2. ARML required parameters. [24]

Name	**Required**	**Description**
ar:provider	Yes	Identifies the content provider. Each provider must have unique identifier.
ar:name	Yes	Name of the content provider used to display content provider on the settings and bookmark menu of the browser
wikitude:icon	Yes	Icon is used to represent the POI in when seen from the reality view of the wikitude browser
ar:provider(within placemark)	Yes	References to the content provider definition.
name	Yes	Name of POI. Displayed as POI title.

point/coordinates	Yes	Coordinated are given in the format of longitude, latitude, altitude. Altitude is optional and is must be given in meters.

Even though there are a lot of attributes and tags that are needed to create Wikitude worlds, table 2 contains attributes that are required by the ARML standard. [18]

3.4 ARchitect

Another way to publish AR content in Wikitude is to use the ARchitect API. ARchitect enables developers to utilise simple web technologies such as HTML, CSS and JavaScript to create AR application for Wikitude. Moreover, ARchitect makes it possible to add more additional mobile optimised libraries. For instance, it is possible to add JavaScript library like jQuery mobile. ARchitec's features takes AR content development technology into a new level by using simple web technologies which means developing AR content is no different than developing any simple web content.

In ARchitect geo-objects are defined to be displayed at a specific location in AR view, but they are not represented by info bubbles, rather they are represented by shapes also known as drawables. Manipulation of geo-objects are done using JavaScript and they can be manipulated to trigger events as well. [26]

ARchitect itself is a JavaScript library constituted of classes to define its features. After creating an AR application or content using the ARchitect API the application can be accessed using the ARchitect engine which is incorporated on the Wikitude world browser mobile application. Before over viewing all the classes and their functionalities, some general concepts that are relevant to the classes will be explained.

SDU (Scaled Distance Unit)

The size of an object in AR applications cannot be defined by pixels or other measurements used in 2-dimensional applications for instance, in WebPages images can be defined using pixels. The size of an object inherently depends on the distance from the viewer to the object. A good example would be the representation of POI on a map, when POI used in a simple map application is represented by a logo defined using pix-

els. But in AR applications the logo size depends on the real representation of the size of the logo in the real world, and its distance between the viewer and the object.

Since AR applications cannot use pixels or other similar measurements because they are defined in static and unable to be scaled accordingly to their inherent. Therefore Wikitude ARchtect introduces a new measurement unit called SDU, which is used to define the size of an object and its inherent distance.

Drawables are shapes that can be drawn to a certain geographical location. Drawables that are attached to geo-objects and Trackable2Dobjects can be defined using SDU. For geo-objects SDU defines the screen size of an object as its height and location from the viewer. For instance 1 SDU represents an objects with a height of 1 meter and located 10 meters away from the viewer. A drawable with width 5 SDU is 5 meters wide and 10 meters away from the user. For trackable 2D Objects, SDU depends on the size of the track able object itself. [21]

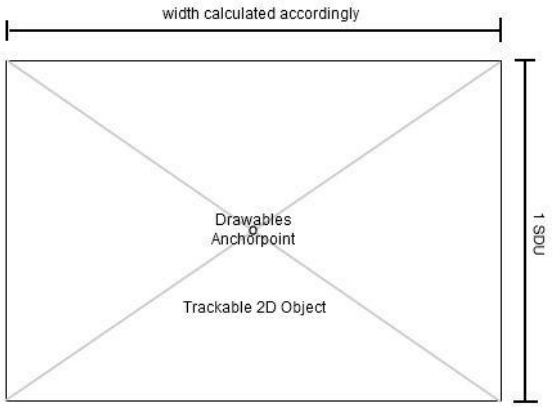

Figure 5. Drawable attached to trackable2Dobject.[21]

Figure 5 illustrates a drawable's anchor point which is set to be the centre of the trackable2Dobject. SDU is defined by the height of the image and the width is to be calculated accordingly.

DBS (Distance Based Scaling)

If a user using AR application with a drawable moves away from the geo-object location, the size of the drawable will get smaller and smaller and finally reached a point where it cannot be visible on the screen. Or in another case, if the user comes closer to the geo-object location, the size of the drawable will be scaled to a bigger size and can

cover the whole screen. In order to avoid such conditions, ARchitect introduced DBS system, which controls drawables by preventing them not to excide certain limits of their original form.

DBS has two values to define the size of a drawable, they are cutoff_min and cut-off_max. Cutoff_min sets the vale for a drawable's visibility on the screen to preventing it from enlarging, whenever the viewing distance gets closer to the geo-objects location. Cutoff_max does the opposite of cutoff_min, by setting the drawables maximum shrinking size.

A function of $x^{(0.2)}$ is applied to objects to scale between cutoff_min and cutoff_max. The effect of this particular function allows the user to experience a smooth change in scaling of drawables size. [21]

ARchitectObject

This class is the base class for objects created in ARchtect API. Since it was created for inheritance purposes, it cannot be used directly. ARchtitectObject has a property to destroy objects. This property is used by developers to free the memory space occupied by the object during the creation of new objects. Setting destroy() method's property value to true. This imposes an irreversible action by deleting all the properties and methods of an object by removing it from the AR scene.

Subclasses of ARchitectObject include:

- ActionArea
- Animation
- Location
- Drawable
- ARObject
- Sound
- ImageResource
- Tracker

Subclasses of ARchitectObject

In this section subclasses of ARchitectObject class will be discussed briefly. These subclasses are mainly used to produce an application which is going to be run on the ARchitect engine on Wikitude world browser.

ActionArea

ActionArea is used to define a geographical area where actions are executed and triggered up on entrance or exit of the area. ActionArea represents a 2-dimensional area, which is a location defined by only latitude and longitude without altitude, and OnEnter() and OnExit() functions will be executed whenever a user is entering or exiting the ActionArea. The borders of the ActionArea are considered to be inside the service range, so OnEnter() function will be triggered even if the user is not fully present on the defined action area.

ActionArea has a sub class called ActionRange which defines a circle around a certain location. ActionRanges are ranges of locations used to trigger events to be executed on the entrance and exit of a certain circle of location.

```
var actionRange = new AR.ActionRange(location1, 500);
```

The code above shows the implementation ActionRange class to create an ActionRange that covers a radius of 500 meters centering location1. Its parameters are described as follows:

`Location`	defines the center of ActionRange
`Radius`	Value of ActionRange's radius in meters
`Option`	functions like OnEnter and OnExit

ActionArea has another method which is used to check whether a certain location is in the defined action area. The method is isInAre() and it returns a boolean that is either true or false. Its parameter is the GeoLocation that needs to be checked.

Animation

This class allows developers to incorporate an animation feature into their application, by giving them a control over the change in numeric values or properties of an object. Since it is an abstract class it should not be instantiated directly. Its subclasses are AnimationGroup and PropertyAnimation.

AnimationGroup is constituted of a group of animations, which are executed parallel or sequentially. A parallel animation group runs all the animations so that they will all start

at the same time. A sequential animation group executes the animations orderly as they appear on the array starting from the first and goes accordingly to the last.

Animation's properties are described as follows:

`Type`	Defines the type of animation group
`Animations`	Array of different animations
`Options`	used to define additional properties OnStart and OnFinish functions are used.

PropertyAnimation class controls the continuous change of a numeric value or property of an object. Which is based on the animation's starting value, ending value and duration of the animation. The mode in which PropertyAnimation is changing the animated property's values can be determined using Easing curves. Basically Easing curves define the nature of an animation and they are composed of constant values. An example of Easing curve type can be:

```
AR.CONST.EASING_CURVE_TYPE.EASE_IN_OUT_ELASTIC
```

For reference purposes the list of Easing curves can be found in the official site of Wikitude ARchitect documentation.

Location

The location class refers to a description of a general location of POI in the augmented view. As the previous classes, it is also a super class to GeoLocation and RelativeLocation classes and it should not be instantiated directly.

GeoLocation class is comprised of latitude and longitude with an optional altitude property. Where the value of the altitude is not defined by the user, the default value for altitude will be set to unknown in the system. However, in reality, it is set to the user's current altitude. The coordinate information can be gathered from Google map or Goole Earth.

RelativeLocation class describes a location in relation to another location or the location of the user. Relative location of a place can be found using northing and easting in relation to its reference location. Additional or optional use of altitude during relative

positioning can be gained as differential value of altitudes. The value of RelativeLocation changes whenever the referencing point is changed.

ARObject

ARObject represents a geographical coordinate in the real world using geo-objects or trackable2DObjects. The class ARObject should not be instantiated directly. ARObjects are represented in the AR application by Drawables which are attached to them. Since ARObjects are used to represent geographical coordinates as GeoObjects which are referenced to a location in the real world. Thus a simple GeoObject :

- has at least one location
- Drawables can be associated with it
- triggers can be associated with it
- can be enabled and disabled

Trackable2DObject is a customizable ARObject which represents a virtual object that is bound to a particular target in a given pattern set. The Trackable2DObject itself is represented by Drawable. The target is where Drawables are projected when it becomes visible on the camera view.

A simple Trackable2DObject :

- is linked to one tracker and its specific target.
- can be represented by Drawables.
- can have trigger events.

Drawable

Drawable is used to represent ARObjects on the reality view. It serves as a building block for all geographical representation of GeoObjects. It should not be instantiated directly as it has subclasses. The subclasses are Drawable2D and Model.

Right-handed coordinate system which is a local coordinate system used by Drawables is characterised as:

- The origin of the coordinate system is in the location of the Object the Drawable is attached to.
- The x axis is pointing right
- The y axis is pointing up
- The z axis is pointing out of the screen. [21]

In the list *right* and *up* representation depends on the type of ARObject the Drawables are representing. For instance if it is a GeoObject that Drawables are representing, the reference *right* and *up* are defined relative to the surface of the earth and the Drawable is pointing towards the viewer. But if it is a Trackable2Dobject, they are defined in relation to the trackable object itself.

Drawable2D is a class that represents all Drawables that represent all GeoObjects as a 2-dimensional object. The main functionalities of Drawable2D class are opacity and rotation scaling. A GeoObject having multiple geographical location can be represented by duplicates of a single Drawables2D.

Drawable2D is also a super class to Label, Circle, ImageDrawable, where the Label class used to represent GeoObjects as a text. Circle represents GeoObjects with plain circles, and ImageDrawabls uses images to represent GeoObjects. [21]

Table 3. Label class parameters

Parameter	**Property**	**Description**
Text	String	Text representing the label.
Height	Float	Height of the label.
Option	Object	Additional parameters to be setup.

Context

Context is a static class that includes general functions such as onLocationChanged(). Its instance will automatically be created when Wikitude world is started. This class helps developers to identify the current location of the a user.

Methods used in this class are:

- destroyAll() which destroys all object,
- onLocationChanged() which detect location change of the user by using parameters such as latitude, longitude, altitude and accuracy,
- onScreenClicked() which is used to detect the click made by the user where the click is used to click an empty space on the screen,
- startVideoPlayer() which can be used to launch the native video player and play video when video finishes playing the next video will start playing.

```
function locationChanged(lat, lon, alt, acc) {
            AR.context.onLocationChanged = null;
        }
```

The code above shows implementation of onLocationChanged () method.

4 Design and implementation

This chapter will demonstrate the practical work of this thesis which focuses on the software implementation. The integral requirements of this thesis stated that the final implementation would be a simple ARchitect based AR application that would retrieve data dynamically using JavaScript libraries and PHP.
The project consisted of the following:

- HTML5 as a mark-up language
- ARchitect library as AR tool
- JavaScript libraries to manipulate data
- PHP as a server side language to retrieve data
- MYSQL for management of data
- CSS would also used for styling purposes
- Wikitude world browser to publish the final solution and for browsing

The final application would allow users to access information about a events that are held on the town culture house. The application is intended to show how ARchitect API can be used to interact with dynamic data and present it using AR browser. In this application the user can:

- Access events and their related information such as time and price and its related information.
- Locate the venue and calculated distance.
- Direction to the venue plotted on the map.

The application is available only to those who have smart phones with wikitude augmented reality browser installed. First the user gets the link to the application from ether the venue's website or poster or from the list of architect worlds that are listed on

the Wikitude browser. After this the user can access the application through the browser's ARchtect engine.

Figure 6: Application in use

The main reason for choosing ARchitect API to demonstrate the ability of AR applications to access dynamic data is because it is easy to develop without requiring very sophisticated programming languages. Implementation of the project would demonstrate how ARchitect API is capable of providing a means to develop dynamic web service and interact using AR browser.

4.1 Dynamic data interaction

The whole part of the project is based on the idea of dynamic data interaction. Looking into what is meant by dynamic data interaction and what technologies are involved form the basis for the project. The main focus will be on the methods and properties used to create this application.

Dynamic data interaction in this context represents the interaction of the application with data that is stored in a database. The application is a web based application which runs on web server. To further clarify how dynamic data interaction works let's look into simple web pages and web applications since the idea behind them is the same.

Web pages using HTML as a mark up language send request to web servers and get the response and publish it. That is what a simple web page can do, there is no further interaction between the page and the server. This workflow shows that a web page that uses only HTML is a static web page. Figure 7 below demonstrates the workflow of a simple static web page.

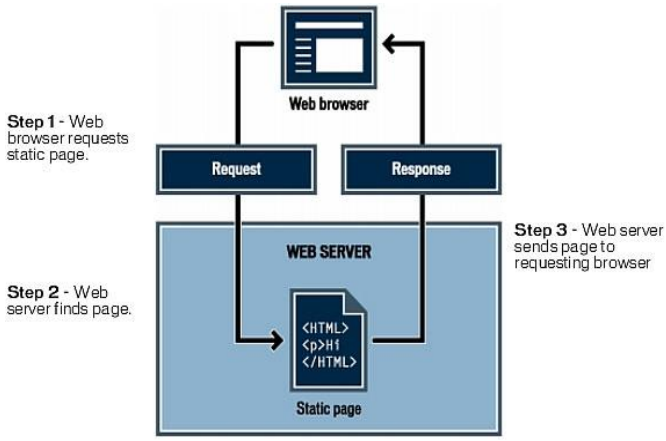

Figure 7. Processing of static web page. [22]

Sometimes additional interaction is needed between the web page and the server for instance a web application that enables a user to search for information can be an example for dynamic web page. In that case additional scripting languages are included in to the HTML file or server side scripting languages that are capable of generating HTML pages. Those additional scripting languages are responsible for dynamic features of the web page. Figure 8 shows the workflow of dynamic web pages.

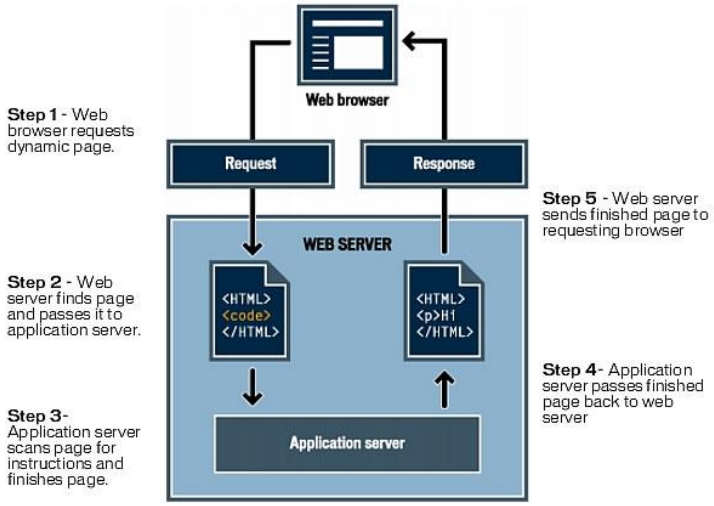

Figure 8. Process of dynamic web page. [22]

In this project PHP and JavaScript, where PHP is utilised carryout a server side scripting and JavaScript has utilise to handle the client side scripting, are utilised to create the dynamic interaction features in the application.

Tools and software

Development of the application is mainly done on windows computer and tools used during production is text editor, server, database and mobile phone with Wikitude world browser installed.

Selection of programming languages is done according to the area of the technology. The project development revolves around HTML5, styling using CSS, JavaScript, MYSQL for database schema and PHP. The use of open libraries is to avoid doing what has already been done and speed up the development process.

JavaScript libraries that are used in the project are and they are event driven.

- jQuery mobile for look and feel of the application
- AJAX for retrieving and data from database
- Google map API for using map and draw direction.
- ARchitect

4.2 Implementation

The application is designed to operate in a specific place. So when a user starts the application, the first thing that the application does is check the location of the user.

```
function locationChanged(lat, lon, alt, acc){
            AR.context.onLocationChanged = null;
            myGeoLocation = new AR.GeoLocation(lat,lon);
            actionarea();
        }
```

The code above shows a function that defines the basic functionalities of the application. It locates the current location of a user by using ARchitects GeoLocation class

parameters latitude and longitude and assigns it to a variable 'myGeoLocation'. Then it implements the function actionarea().

```
function actionarea(){

            if(myGeoLocation != null){

                var L5 = new AR.GeoLocation(59.975935,23.438757);

                var actionrange1 = new     AR.ActionRange(L5,2000,{onEnter:
onenter, onExit: onexit});

                var inarea = actionrange1.isInArea(myGeoLocation);
                if(inarea == false){
                    document.getElementById("cc").style.display =  "none";
                    document.getElementById("records").style.display = "none";
                    document.getElementById("lod").innerHTML =
                        "<?=lang('front.NIServiceArea')?>";
                }
```

The code above shows the function actionarea. First it checks if the user's location is true and then it assigns the center of the action area, as 'L5' in this project is the location of the venue. Then it defines the action range circle with 2 kilometers as a radius and two options as OnEnter() and OnExit() functions. Then it checks if the user is in the defined action range using isInArea() method. IsInArea() method checks if a certain location is inside the defined action area and returns true or false. In the code above the return of the isInArea is checked and if it is false, it means the user is outside the defined action area. In that case a message informing that the user is not in the defined area where he or she should be in order to access the application will be displayed.

The OnEnter() function is triggered if the user enters the defined area. It also sets the page content. In this application there is a list of events with their corresponding timestamp which will also be displayed.

List of upcoming events contains images and occurrence date and time of the next four upcoming events based on current date and time. In this page the events are listed in chronological order and once the occurrence of the event has passed, that particular

event will be removed from the list and replaced by a new upcoming event. This is achieved by implementing the Ajax() function which is illustrated in the code below. It handles all the transaction of data. Data is requested through the value of *url* and the data type is defined by dataType variable. If the request succeeds, then the success function will present the data to be plotted.

```
$.ajax({
                    url: readUrl,
                    data: "",
                    dataType: 'json',
                    success: function( response ) {

                        for( var i in response ) {

                            response[i].imgsrc = imgsrs + '/' +  response[i].Image;
                            response[i].updateLink = read1Url2 + '/' + response[i].Id;
                            response[i].deleteLink = delUrl + '/' + response[i].Id;

                        }

                        //clear old rows
                        $( '#records' ).html( '' );

                        //append new rows
                        $( '#readTemplate' ).render( response ).appendTo( "#records" );


                    }
                });
```

OnExit() function will be triggered when the user leaves the action area. In this application function returns the message which is displayed when a user is not in the action area.

The second action which leads to the retrieval of information about details of the upcoming events can be accessed by clicking the image of the event. In the details page the user can get the price, short description, time, availability of the ticket and title of the event. This is done using the same methods as previously used to retrieve the list of events.

Implementing the GeoObject class is to used to represent the location of the venue. In this section the GeoObject is represented by Imagedrawable class as shown by the code below.

```
var  img = new AR.ImageDrawable(image, 6, {enabled: true});
```

The image used as a imageDrawable is an image of the venue. When the image is visible, events such as enterFOV() and exitFOV() will be triggered. enterFOV() is a function that is executed when the image is visible on the camera view. In the application enterFOV() method will display a text message containing further information about the venue.

Figure 9. GeoObject

Figure 9 shows implementation of the GeoObject Class. It shows the image of the venue as a GeoObject and the texts under the image are the result of the enterFOV() function.

Another method which is triggered when the GeoObject is not visible on the camera view is exitFOV(). When this function executed, another text message will be displayed. Events can be of any sort, in this project utilisation of text messages is done just for demonstration purposes.

The other part of this page contains a text with the value of a distance between the user and the venue. This distance is calculated using the distanceTo() function and the following code.

```
var dist = (Math.round(myGeoLocation.distanceTo(L5)))/1000;
```

The code above shows implementation of distanceTo() function which calculates the distance between *myGeolocaion* and *L5*.and return the distance in meters. Math.round() is used to round up the decimal points and division in 1000 is applied to convert the rounder up result in kilometers.

Figure 10. Distance to location.

Figure 10 illustrates the distance of a user who is 0.34km away from the venue.
Without using distanceTo() function to calculate the distance between the two locations, the same result would have been achieved by implementing the Haversin formula. [24]

$$a = \sin^2(\Delta\varphi/2) + \cos(\varphi 1).\cos(\varphi 2).\sin^2(\Delta\lambda/2)$$

$$c = 2.a\tan 2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R.c$$

where:

- φ is altitude
- λ is longitude
- R is earth's radius(R=6,371Km)
- d is the required distance

Following the mathematical formula a simple JavaScript or any other scripting langue can be implemented to calculate the distance between two points located on the earth's surface.
The final functionality of the developed application would be to draw a direction on the map. The direction would be drawn from the user's current location to the venue. This navigation would help the user to find the venue easily. This is implemented using Google map API.

Google map JavaScript API

This API enables to incorporate the Google map and its functionalities into other applications. In this project Google map for mobile is utilised. The API is free and can only be used to create free applications. In order to use this API, an API key which can be acquired through Google's API access panel must be obtained. Google uses the key to track utilisation of API. For instance, if the API's quota excides the usage limitation, Google might ask to purchase the API.

Constructors that are used for development of AR application are as follows.
Markers are location identifiers represented customisable icons. Icons and be customized by utilising setIcon() constructor. Marker options objects have properties that specify the properties of a marker. Those properties are:

- position indicates initial location of the marker
- map the map object where marker is placed

The setMap() constructor is called to specify a map to add the marker and it has to be specified inside the Marker constructor otherwise the marker will not appear on the map. To interact with markers, events can be passed to them. For instance, the 'click' event is set to be true by default which will allow the marker to be clickable.

InfoWindow

Info windows are located floating above the map to display information about a specific location on the map. They have a content area and they point into the location that they are representing. Figure 11 shows implementation of InfoWindow() constructor.

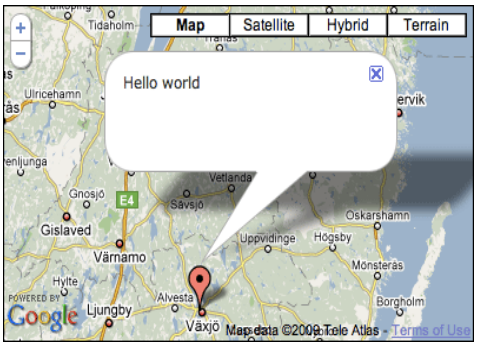

Figure 11. InfoWindow.[25]

InfoWindow contains parameters such as:

- `content` contains content to be displayed on the InfoWindow
- `Position` defines the location of the anchor point using latitude and longitude.
- `maxWidth` defines the maximum width of the InfoWindow the height can be variable

Content can be passed to the setContent() constructor to set the content explicitly. InfoWindows attached to Marker use the location of the Marker as an anchor point.

MapType

Type of maps are a collection of different maps that are available while using Google maps API. The MapType object is used in order to carry information about different

maps. There are different types of MapTypes that can be utilised by developers including custom MapTyps and basic maps with styledMaps. However, the basic MapTypes are:

- ROADMAP default road map view
- SATELLITE Google Earth satellite view
- HYBRID Mixture of roadmap and satellite view
- TERRAIN Physical map based on terrain information

Direction request

Direction request is done by using DirectionService object. To initiate the DirectionService DirectionService.route(), the method has to be called within DirectionRequest object and passed to it. DirectionRequest has parameters such as:

- Origin starting location
- Destination end location
- Travel mode mode of transportation use for calculating direction

Displaying DirectionRequest's response can be done using DirectionResult object which holds query of results. Another option to render the result would be to pass the result of DirectionRenderer. DirectionRenderer draws a polyline between origin and destination and places markers on both locations. The placement of markers can be customised to meet the developers requirements. [25]

Implementation of Google Map API

At the beginning the application, utilising ARchitect library defines current location of the user and this current location will be used as an origin to get the direction, while the location of the venue will serve as a destination.

```
var myLatlng = new google.maps.LatLng(lat,lon);
var location2 = new google.maps.LatLng(59.975935,23.438757);
```

The code above defines myLatlng being the user's current location and location2 being the venues location.

Defining the map object and assigning its parameters are done as shown in the code below. The code below container of the map in the HTML is defined and the options are passed using myOptions object.

```
var map = new google.maps.Map(document.getElementById("map_canvas"),
                myOptions);
```

MyOption is defined with parameters. The following code shows that the map is centred at the user's current location, 13 times zoomed out and roadmap type of map is used.

```
var myOptions = {
                    center: myLatlng,
                    zoom: 13,
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                };
```

Info window is implemented in the application to represent the location of a user.

```
var infowindow = new google.maps.InfoWindow({
             maxWidth : 10, content: "<?=lang('front.yloc')?>"
                            });
```

In the code above, an info window is defined with maximum width 10. The content is telling the user the marked point is his/her current location.

As the code below describes, the request for calculation of navigation is done before the origin of the navigation is the user's current location, and the destination is the venue. The additional information would be the type of mode of transportation which in this case is set to walking mode. The request is defined inside the infowindow() function.

```
var request = {
                   origin: myLatlng,
                   destination: location2,
                   travelMode: google.maps.TravelMode.WALKING
                };
```

DirectionService is implemented as shown below. The function checks if a valid response has been made. In this case the status of the response is ok which means that a valid response has been made.

```
directionsService.route(request, function(response, status) {
                    if (status == google.maps.DirectionsStatus.OK) {
                        directionsDisplay.setDirections(response);
                    }
                });
```

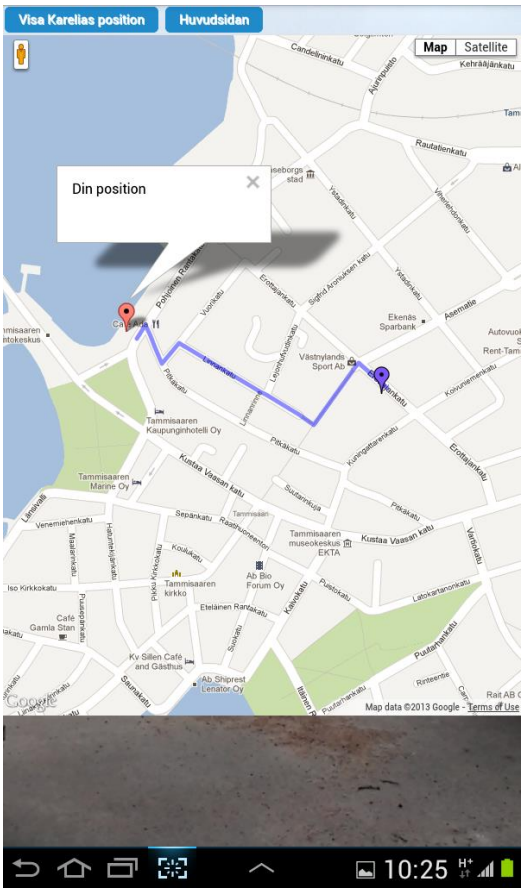

Figure 12. Direction to Kulthuset Karelia.

Figure 12 demonstrates the direction drawn from the user's current location towards the venue's location. Using the indication on the map the user will be able to fine the venue.

4.3 Testing

The test hypothesis is that the AR application enables the user to interact with a dynamic data. The main reason for carrying out this test is to establish and identify potential defects of the ARchitect API. Its objective is to identify AR related problems that are related to the functionalities of the designed application.

Requirements for the test

- Mobile phone with :
    - Wikitude world browser at least version 7.0 installed

    - GPS access
    - Mobile broad band or Wi-Fi
- Being at the location where the application functions.

Table 4. Specification for the mobile phone used:

Phone	HTC Desire S S510e
Display	3.7'' WVGA
Camera	5M
Operating System	Android 2.3.5
GPS	Enabled
Wi-Fi/mobile data transfer	Enabled

Table 5. Software used setting and specification:

Software	Wikitude world browser incorporates the ARchitect engine
Version	7.1
Unit of Length	Meters
Default view	Camera

Three places were chosen to conduct the test. The first one is at the venues exact location, the second at the border of the defined action area and the third is outside the defined action area.

The first test was done at the first location. The user launched Wikitude world browser and entered the URL, because the application is located by using Wikitude's incorporated Architect engine. After that the user starts to access all the functionalities of the application and play around with it for a while. The next part was to go to the second position and repeat the same scenario as in the first location. Once the user was done with the second location, he went to the third location and carried out the same scenario.

Results

In the first case the user was able to access all the functionalities of the application and everything worked as it should. However, there was a delay in retrieving the dynamic data. This could have resulted in less processing power of the mobile phone and the speed of mobile data roaming. It was noticed that after the data was loaded for the first

time, the delay in retrieval the dynamic data was not experienced, when the application was used for the second or the third time.

In the second case the user was located at the border and was able to access all the application's functionalities. No problems were experienced.

In the third case the user was located at the third location. Since the third location is outside the defined action area, there was only one functionality that the user could get. It was a message stating that the user is outside the defined area and in order to access the application he/she should be in the defined area.

5 Conclusion

Augmented Reality technology has come to a point where it can be utilized in anyone's daily activities. Since the technology is still on its early stage, it can open doors to many possibilities in the future.

During the actual development phase of the application there were no serious problems. Since the company who published the implemented ARchitect library is constantly updating it. This could enable application developers or content publishers to incorporate additional functionalities into their applications.

The main goal of this project was to demonstrate the capability of an augmented reality application to interact with a dynamic data by developing the AR application using simple web technologies such as HTML, CSS and JavaScript.

As a result an application was developed and it was capable of interacting with a dynamic data. The user was able to access data from a database and also able to interact with it. This shows that development of AR applications or AR content does not necessarily require knowledge of more sophisticated programming languages.

References

1 Kipper Gregory, Rampolla Joseph. Augmented reality, An emerging technology Guide to AR. Massachusetts, Elsevier Inc. 2013.

2 Lester, M. Professional Augmented reality Browsers for smartphones: Programming for Junaio, Layar and Wikitude. West Sussex: Jhon Wiley & Sons Ltd; 2011.

3 Wagner Daniel, Schmalstieg Dieter. First steps towards handheld augmented reality [online]. Vienna, Austria: Vienna University of Technology; 2003. URL: https://www.ims.tuwien.ac.at/publications/tuw-138074.pdf. Accessed 12 May 2013.

4 Butchart Ben. Augmented reality for smartphones: A guide for developers and content publishers[online]. JISC Observatory; 1 March 2011.
URL: http://observatory.jisc.ac.uk/docs/AR_Smartphones.pdf. Accessed 12 May 2013.

5 Billinghurst Mark, Thomas Bruce H. Mobile Collaborative Augmented Reality. In: Alem Leila, Huanng Weidong, editor. Recent trends of mobile collaborative augmented reality systems. New York: Springer; 2011. p.1-20.

6 Nagao Katashi, Rekimoto Jun. The world through the computer: computer augmented interaction with real world environments. UIST '95 Proceedings of the 8th annual ACM symposium on User interface and software technology [online] 1995: 29-36.
URL: http://dl.acm.org/citation.cfm?id=215639. Accessed 12 May 2013.

7 Perry Simon. Wikitude: Android App with Augmented Reality [online]. Digital Lifestyles; 23 October 2008.
URL: http://digital-lifestyles.info/2008/10/23/ wikitude-android-app-with-augmented-reality-mind-blowing/. Accessed 12 May 2013.

8 Carmigniani Julie, Furht Borko. Augmented Reality: An overview. In: Borko Furth, editor. Handbook of Augmented reality. New York: Springer; 2011. p.3-46.

9 Messieh Nancy. Augmented reality browser, Junaio, wants you to 'scan the world' with its latest upgrade [online]. The Next Web; 29 August 2011.
URL: http://spider21.wordpress.com/2013/04/03/
application-areas-for-augmented-reality/. Accessed 12 May 2013.

10 McDonough Meghan J. Lego's Augmented Reality App shows you the what's inside the box [online]. Laptop Magazin: LAPTOP; 15 September 2011. URL: http://blog.laptopmag.com/legos-augmented-reality-app-shows-you-whats-inside-the-box. Accessed 12 May 2013.

11 CAE. ProMIS: The Hybrid laparoscopic simulation that combines Hands-on and virtual learning [online]. CAE Healthcare; 2013. URL: https://caehealthcare.com/home/eng/product_services/product_details/promis. Accessed 10 May 2013.

12 Levett Jhon. Mobile Augmented reality App Downloads to pass 400 million annually by 2014 as App Stores and AR- enabled SmartPhones Surde, Juniper Report finds [online]. Juniper research; 5 January 2010. URL: http://juniperresearch.com/viewpressrelease.php?pr=176. Accessed 10 May 2013.

13 Siltanen Sanni. Theory and applications of marker-based augmented reality [online]. Espoo: VTT Technical Reaserch Center of Finland; 2012. URL: http://www.vtt.fi/inf/pdf/science/2012/S3.pdf. Accessed 12 May 2013.

14 Nita Ilinca. Layar 3.0 mobile augmented reality browser launched [online]. Unwired View; 03 December 2009. URL:http://www.unwiredview.com/2009/12/03/layar-3-0-mobile-augmented-reality-browser-launched/. Accessed 12 May 2013.

15 Metaio. How Channels work [online]. Metaio developer Portal; 2013. URL: http://www.junaio.com/develop/docs/what-are-channels/. Accessed 20 April 2013.

16 Layar. Developer Documentation [online]. Layar official website; 2013. URL: http://www.layar.com/documentation/browser/layar-platform-overview/. Accessed 10 May 2013.

17 Google. Kehole Markup Language [online]. Google Developers; 24 February 2012. URL: https://developers.google.com/kml/documentation/kml_tut#basic_kml. Accessed 20 April 2013.

18 Open ARML. ARML 1.0 Specification for Wikitude [online]. openarml.org; 23 April 2012. URL: http://openarml.org/wikitude4.html. Accessed 20 April 2013.

19 ARML Standards Working GroupAvailable. ARML 2.0 SWG [online]. Open Geospatial Consortium; 2013. URL: http://www.opengeospatial.org/projects/groups/arml2.0swg. Accessed 12 March 2013.

20 Wikitude. Developer[online]. Wikitude.com; 2013. URL: http://www.wikitude.com. Accessed 10 May 2012.

21 Wikitude GmbH. Wikitude ARchitecht v2.0 API Documentation [online]. Wikitude.com; 2012. URL: http://www.wikitude.com/external/doc/alr/index.html/. Accessed 29 April 2013.

22 eTutotials. Dream Weaver Online Help [online]. eTutorials.org; 2013. URL:http://etutorials.org/Macromedia/Dream+Weaver+Online+Help/Getting+Started+with+Dreamweaver/Understanding+Web+Applications/How+a+web+application+works/Processing+dynamic+pages/. Accessed 12 May 2013.

23 Veness Chris. Calculate distance, bearing and more between Latitude/Longitude points [online]. Movable Type Scripts; January 2010. URL: http://www.movable-type.co.uk/scripts/latlong.html. Accessed 20 February 2013.

24 Google Developers. Google Maps API JavaScript API v3 [online]. 27 July 2012. URL: https://developers.google.com/maps/documentation/javascript/. Accessed 20 February 2013.

25 Ronald T. Azuma. A Survey of Augmented Reality [online]. Malibu, CA: Hughes Research Laboratories; 1917. URL:http://www.cs.unc.edu/~azuma/ARpresence.pdf. Accessed 20 March 2013.

Source Code

1. First page

```
<script type="text/javascript">

    var  myGeoLocation = null;
    var  readUrl = "index.php/karelia/read",

    function locationChanged(lat, lon, alt, acc){

        AR.context.onLocationChanged = null;

        myGeoLocation = new AR.GeoLocation(lat,lon);

        actionarea();
    }

    AR.context.onLocationChanged = locationChanged;

    function actionarea(){

        if(myGeoLocation != null){
            var L5 = new AR.GeoLocation(59.975935,23.438757);
            var     actionrange1=new
AR.ActionRange(L5,2000,{onEnter: onenter, onExit: onexit});
            var inarea = actionrange1.isInArea(myGeoLocation);
            if(inarea == false){
                document.getElementById("cc").style.display=
                            "none";
                document.getElementById("records").style.display
                            = "none";
                document.getElementById("lod").innerHTML =
                    "<?= lang('front.NIServiceArea') ?>";
            }
        }else {
```

```
        document.getElementById("lod").innerHTML="Your      geo-
location can't be identified";
        }
    }

    function onenter(){

        document.getElementById("lod").style.display = "none";

        $( function() {


            $.ajax({
                url: readUrl,
                data: "",
                dataType: 'json',
                success: function( response ) {

                    $( '#records' ).html( '' );

                    $('#readTemplate').render(response
                              ).appendTo( "#records" );
                }
            });

        });

    }

    function onexit(){
        document.getElementById("lod").innerHTML="<?=
          lang('front.NIServiceArea') ?>";
    }

</script>
```

2. Single event selection

```
<script type="text/javascript">


document.getElementById("pp").innerHTML = location.href;
function redirectmap(){

    window.location = "<?php echo $pageLng; ?>";
}
function redirecthome(){

    window.location =
  "http://users.metropolia.fi/~aidah/wikitude/index.php
/karelia";
}


</script>
```

3. GeoObject Implementation

```
<script>
    function enterFOV() {
      document.getElementById("statusElement2").innerHTML
              = "<?= lang('front.Tisthepicture') ?>";
      docment.getElementById("statusElement3").innerHTML  =
              "<?= lang('front.kareliablow') ?>";
    }

    function exitFOV() {
       docment.getElementById("statusElement2").innerHTML
              = " ";
    }

    function locationChanged(lat, lon, alt, acc) {
        AR.context.onLocationChanged = null;
        var L5 = new AR.GeoLocation(59.975935,23.438757);
        var myGeoLocation = new AR.GeoLocation(lat,lon);
        var
     dist=(Math.round(myGeoLocation.distanceTo(L5)))/1000;
```

```
            var
        image=newAR.ImageResource("upload/karelia.jpg");


            var img = new AR.ImageDrawable(image, 6, {enabled:
                                true});

            var myGeoObject = new AR.GeoObject(L5, {drawables:
        {cam: img}, triggers: { onEnterFieldOfVision: enterFOV,
        onExitFieldOfVision: exitFOV }});
            document.getElementById("statusElement").innerHTML
        = dist+"km "+"<?= lang('front.TPicture') ?>" ;

        docment.getElementById("messageElement").style.display="no
        ne";
        }

        AR.context.onLocationChanged = locationChanged;

        </script>
```

4. Direction calculating script for Map page

```
<script type="text/javascript">

       function locationChanged(lat, lon, alt, acc) {

            AR.context.onLocationChanged = null;

            var myGeoLocation = new AR.GeoLocation(lat,lon);
            var directionDisplay;

            var directionsService = new
        google.maps.DirectionsService();
            var map;
            var myLatlng = new google.maps.LatLng(lat,lon);
            var location2 = new
        google.maps.LatLng(59.975935,23.438757);

            if(myGeoLocation !=null){
```

```
                directionsDisplay = new
      google.maps.DirectionsRenderer({suppressMarkers: true});
      var myOptions = {
                    center: myLatlng,
                    zoom: 13,
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                };
  var map = new
google.maps.Map(document.getElementById("map_canvas"),
                myOptions);
  directionsDisplay.setMap(map);
  var marker = new google.maps.Marker({
                          position: myLatlng,
                          map: map,
                         title:"Your location"
                         });
var imag =
'http://maps.google.com/intl/en_us/mapfiles/ms/micons/purple-
dot.png';
var marker2 = new google.maps.Marker({
                          position: location2,
                          map: map,
                         title:"Destination",
                         icon :imag

                         });
var infowindow = new google.maps.InfoWindow({
                  maxWidth : 10,
                   content: "<?=lang('front.yloc')?>"
                              });
infowindow.open(map, marker);

                var request = {
                    origin: myLatlng,
                    destination: location2,

                    travelMode: google.maps.TravelMode.WALKING
                };

directionsService.route(request, function(response, status) {
                    if (status ==
google.maps.DirectionsStatus.OK) {
                        directionsDis-
play.setDirections(response);
                    }
                });
       marker.setMap(map);
 document.getElementById("messageElement").style.display="none";
           }

       }
```

```
    AR.context.onLocationChanged = locationChanged;
</scrip>
```