WorldPlus: An Augmented Reality Application with Georeferenced content for smartphones – the Android example

Conference Paper · June 2012

READS
33

3 authors:

Sérgio Graça

2 PUBLICATIONS **1** CITATION

SEE PROFILE

João Fradinho Oliveira

C3i/IPP

35 PUBLICATIONS **43** CITATIONS

SEE PROFILE

Valentim Realinho

Instituto Politécnico de Portalegre

14 PUBLICATIONS **16** CITATIONS

SEE PROFILE

WorldPlus: *An Augmented Reality Application with Georeferenced content for smartphones - the Android example.*

Sérgio Graça[1] João Fradinho Oliveira[1] Valentim Realinho[1]

sergiogr@gmail.com jfoliveira@estgp.pt valentim.realinho@estgp.pt

[1] C3i Centro Interdisciplinar de Investigação e Inovação/Instituto Politécnico de Portalegre

Lugar da Abadessa, Apartado 148, 7301-901 Portalegre

ABSTRACT

In the last few years there has been a significant evolution in the mobile devices hardware capabilities, this evolution is very important as it allows for more complex applications and services to be developed for this type of devices. In this paper we describe the concepts and key issues that arise when developing an Augmented Reality system in a localization application for smartphones in general. WorldPlus presents solutions and implements these concepts using the Android smartphone as an example. In addition, WorldPlus allows programmers to develop their own content providers for both an online and an offline mode.

Keywords

Augmented Reality, Smart phone, Android

1. INTRODUCTION

In the last few years there has been a significant evolution in the mobile devices hardware capabilities, this evolution is very important as it allows for more complex applications and services to be developed for this type of devices.

Platforms like IOS (Mobile Operating System from Apple) and Android (Mobile Operating System developed by the Open Handset Alliance with Google Inc. leadership) are examples of platforms that were developed for mobile platforms; they include a set of tools that enable one to take advantage of the hardware capabilities of the new mobile devices.

Whilst mobile localization Augmented Reality applications are not new, in this paper we describe the concepts and key issues that arise when developing an Augmented Reality system in a localization application for smartphones in general, and offer solutions to some of the problems. We present available tools that can be used in ordinary smartphones and implement an augmented reality system using the Android system as an example, where the user can browse the world through the mobile device and find additional information about his surroundings, with these tools. In this paper we review the main concepts of an Augmented Reality based on Localization in Section 2, including application requirements and the description of two sources of online content or points of interest POI that are viewed with our system (Panoramio/photos, Wikipedia/documents). In Section 3 we present the WordPlus application, which describes how different components and services from the OS of mobile devices can be used to gather geo referenced points of interest, and how these points are managed and projected in our system. We show results in Section 4, propose further solutions in Section 5, and conclude in Section 6.

2. BACKGROUND

In this section we review key concepts that need to be modeled in order to build our augmented reality system based on localization, we outline the application requirements, and review two online sources of georeferenced information that will be accessed and viewed with our system.

2.1 Augmented Reality Based on Localization

Augmented Reality is a concept contained in a larger concept that is Mixed Realities, Mixed Reality is a concept that represents a system where various realities are mixed together, usually mixing the reality that we all know, the world where we live, and realities generated by computational processes To facilitate the comprehension of these concepts, and relations between them, Paul Milgram has created a spectrum that clarifies it called Reality-Virtuality Continuum [Mil94a] (see Fig. 1), the spectrum goes from Reality (left) that comprises the world where we live in, with no extra elements

computationally generated, to Virtual Reality (right) where the "world" is completely computer generated and doesn't have any elements from the real world. In the middle of these two is the Mixed Reality that consists in Augmented Reality and Augmented Virtuality, these two AR and AV are opposite concepts, while AR is the real world augmented with computer generated elements, AV is a virtual world augmented with real world components.

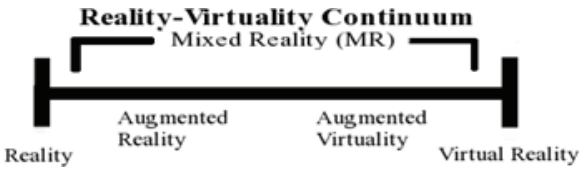

Figure 1: Reality-Virtuality Continuum.

Now that these concepts have been introduced, Augmented Reality based on localization will be described in more depth. Augmented Reality is the mixture between the real world and computer generated elements, this concept has the following elements:

- Caption of real world;
- Computer generated elements;
- In a 3D environment.

AR Reference Model

The cost and effort in developing mobile AR systems is quite high. To reduce these, a number of different software architectures and toolkits have been proposed. A survey of different mobile and wireless technologies and how they have impact AR, including software architectures for mobile AR can be found in Papagiannakis et al. [Pap08a]. The authors compare several models and conclude that there is no single ideal mobile AR system approach but rather different AR systems according to location (indoors or outdoors), type of display (handheld or head mounted displays), content augmentation (static 3D, virtual characters) as dictated by each application domain. Reicher [Rei04a] presents a reference architecture for augmented reality systems organized into logical subsystems as follows: (i) a *tracking* subsystem that in smartphones is typically based on location sensors in the device such as GPS, compass and accelerometer allowing 6 degrees of freedom in displaying a digital object; (ii) an *application* subsystem responsible for the main control flow logic of the application and coordinating communication between other subsystems; (iii) a *world model* subsystem that stores and provides access to a digital representation of the world, including points of interest, 3D objects and metadata about the model itself; (iv) a *presentation* subsystem responsible for all output, including reality view streams, 2D and 3D rendering and audio and tactile outputs; (v) an *interaction* subsystem which gathers and processes any input that the user makes deliberately; and (vi) a *context* subsystem that provides the application with context about the status and situation of the user. We use an architecture that can be mapped to the one presented by Reicher.

AR Browsers for Smartphones

Layar [Lay12a] is the most prominent AR browser designed for smartphone devices. It offers animated 3D rendering, location based tracking, and has a highly flexible API and a useful set of tools for developers. Wikitude Worlds [Wik12a] is a general purpose AR browser with location based tracking, support for 2D images and it is one of the most easy to use browsers for developers. The Wikitude Worlds AR browser is based on the Wikitude API, which is an open source framework for developing users own standalone AR applications on iPhone, Android and some Symbian based devices. Junaio [Jun12a] is a powerful AR browser which includes features to support 3D object rendering, location based tracking, marker and markerless image recognition and a powerful API for developers. Sekai Camera [Sek12a] is another example of an AR browser which is a social augmented reality mobile location-based service. Wikitude is the only one that supports an offline mode which is an advantage where the availability of an internet connection is an issue. Our approach was to support both online and offline mode. We provide an architecture that enables programmers to develop their own content providers for both online and offline modes. For evaluation purposes, we provide in the current version of WorldPlus an offline access of the geo-referenced Wikipedia articles, and an online access of the geo-referenced Panoramio photos.

2.2 Application Requirements

In order to implement an Augmented Reality Localization application on a smartphone, we identified the following five requirements:

1. ***Caption of real world*** – this requirement will be resolved with the camera that comes with the mobile device, this camera will help on capturing a stream of real world images in real time like the AR concept requires.
2. ***Gather Points of Interest*** – these points are places in the world that are represented by a latitude and a longitude, these elements will represent the real world places that the application detects when the user points the camera at them, they will consist basically on a bitmap image/tag for each point of interest with which the user can interact and retrieve additional information about the place represented. These graphic elements will be superimposed in the view of the camera, like a layer on top of the captured real world images of the camera in which the images representing the points are drawn.

3. ***Process Sensors Data*** – modern mobile devices have a set of sensors that can track the position and direction of the device, the accelerometer sensor and magnetic sensor respectively.

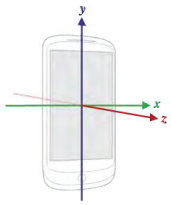

Figure 2: Three axis system in a device.

The accelerometer´s main objective is to map the device's position (axis) in relation to the real world, it measures the force in G's applied in the 3 axis (Figure 2) and provides values that help one determine for example if the device is leaning forwards facing the ground or leaning backwards and facing the sky. In addition, these values can represent if the device is in a portrait position or in a landscape position, enabling an application to decide to perform some processing based on this information, perhaps for flipping the user interface elements so that the user can view them always in a correct position. Furthermore these values will later be used in the calculation of the projection of the points of interest in the device.

The magnetic sensor helps to determine what direction the device is facing, much like a compass in reading which direction is north. This sensor is important to determinate if a given point of interest is in the same direction that the device is facing, in this case the point has a high chance of being drawn on the device's display. The magnetic sensor provides the values in radians that can be converted to degrees, it measures from 1 to 360 degrees. 0/360 degrees being the direction of north.

These two sensors are processed at the same time using methods provided by the Android SDK library, and it returns 3 values:

- **Azimuth** – this value corresponds to the angle between the true north and the direction that the device is facing.
- **Pitch** – this value corresponds to the angle that represents the inclination of the device, if the device is leaning forward or backward.
- **Roll** – this value represents the rotation in the device that helps one determine for example, if the device is in a portrait position or in landscape.

4. ***Localization System*** - the application has to know where the user/device is located on the surface of the Earth, it needs to know this so that amongst other tasks it can detect geo referenced points surrounding the user or get distances from the users to points with a latitude and a longitude. This can easily be achieved by a the GPS system that most devices have already built in, where the system using a method of triangulation with satellites can locate the position of the user on the surface of the Earth. In the particular case of the android platform this localization system uses a mixture of the GPS and network systems.

5. ***Network*** – to gather points of interest, our system will use online services such as Panoramio[1] that need an Internet connection for retrieving information. And as mentioned in the "Localization System" requirement, in the particular case of the Android platform, network protocols help to geographical locate the device in a combination of broadband systems and Wi-fi access points[2], this could be useful, for example, when the GPS service is not available in certain conditions like inside buildings. We point out however that in our experience these alternative localization systems have a much higher error than GPS, serious limiting their use.

2.3 Online Content (Points of Interest)

Our system accesses content / points of interest from two online services: Panoramio for photographs and Wikipedia for documents, both systems hold information that is geo-referenced.

2.3.1 Panoramio

Panoramio is an online service that permits people to submit pictures that are geo referenced by a latitude and a longitude, the pictures are submitted with some additional information like the author of the picture, the title given to the picture by the author, and so on, people then can share their photos with all the users of Panoramio. The objective of using this service in our application is to obtain the pictures taken in the surroundings of the location of the user and show these places to the user with all the information attached to that particular place. When the user turns on the application, it makes a call to the Panoramio API that consists on a REST[3] call for obtaining the points near the user of the application. All the points and their information are obtained formatted in a JSON object that will be parsed in order to retrieve and store in memory all the data that is needed. This type of service is very simple and easy to process with a simple HTTP Request with a query string

[1] Panoramio – Panoramio is an online website that permits the users to submit geo referenced photographs to the system, add additional information about the place where the photograph was taken among other functionalities.

[2] More information about this method of localization can be accessed in the Android SDK Reference Guide in: http://developer.android.com/reference/android/location/LocationManager.htm

[3] REST – Is a method to obtain information from a web server with a simple HTTP Request, much like RPC calls or SOAP but without their formalisms.

containing all the parameters to filter the data. Services using the same method are spreading because of its simplicity. A call at the Panoramio's API is much like the next example.

```
"http://www.panoramio.com/wapi/template/list.html?tag=mountain&width=280&height=140&rows=2&columns=4&orientation=horizontal"
```

2.3.2 Wikipedia

As is common knowledge, the Wikipedia is a website that strives to store articles from all branches of human knowledge, some of these articles are about places for instance historical places, Museums, and much more. Articles regarding places often contain references of latitude and longitude, these articles become points of interest for our application. These places will show on the device's display as images (a tag) with which the user can interact. The Wikipedia is a good source of points of interest because of the shear coverage of articles that it stores. Unfortunately Wikipedia was not designed for fast retrieval of geo-referenced data like Panoramio. In order for information on Wikipedia to be used as a source of points of interest, one needs to download "dumps" that the Wikipedia shares online in various forms, in XML files that store all the articles in a given time stamp, or for example SQL dumps. This way of sharing all that information is not very mobile phone friendly as those dumps are released almost every day and consist of files of a few gigabytes. We filter offline only the georeferenced points of interest of such dump files and make the database file available physically in the mobile for offline querying of GPS coordinates.

3. WORLDPLUS

WordPlus was implemented using one of the platforms that is growing rapidly in users, the Android platform, however we believe that with the concepts described in this paper it should be easy to implement an augmented reality application on other similar platforms like the IOS. In this section we will first present the architecture of the application and describe some of the key classes created for the application along with other classes that were reused from the Android SDK libraries. We will then describe in detail how we create the points of interest and tackle problems such as the size of the data. We then describe how the sensor data is processed, noise removed and finally describe the projection method that enables points to be mapped on to the display.

3.1 Application Architecture

Figure 3 illustrates the main classes of our system and how they interact between themselves. There are two central classes that have the most relations with other classes, the *WorldPlus* and *PoiViewManager* classes, these classes are the foundations of the application. The *WorldPlus* class is the base class of the application, it is the "main" class, responsible for instantiating the necessary classes when they are needed and manipulating them as needed in the cycle of the application by using their methods.

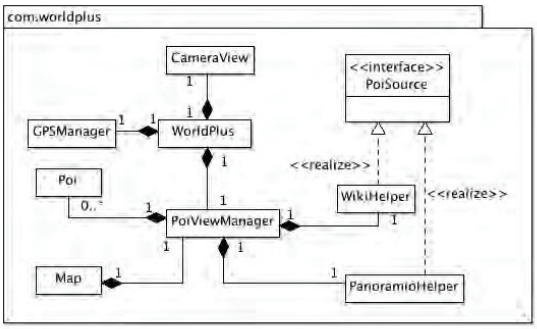

Figure 3: Class diagram of the application.

This class has the following set of tasks:

- Create an instance of the CameraView, the CameraView is itself responsible for displaying the view of the device's camera, real images in real time that are being captured by the camera.
- Create an instance of the GPSManager to get the user´s location at any time.
- Create an instance of the PoiViewManager that is the other central class, and is responsible for managing all that has to do with the points of interest.
- Create an instance of a Sensor Listener that receives notifications from the SensorManager every time that alterations in the state on the sensors of the device are detected. The sensor listener and sensor manager are classes used from the Android SDK libraries.
- Process the data obtained from the sensors, this includes the task of eliminating the noise created by the sensors. Values undergo an averaging procedure (described in section 3.3) before being passed to the method that deals with the calculations of the projection, in this way the projection of points is more stable and prevents points from constantly jumping between positions on the screen due the instability of the sensors.
- "Order" the PoiViewManager to calculate the projection of the points when all the data needed from the sensors is gathered, this includes data from the accelerometer, and from the magnetic sensor more commonly referred as the compass.

The next class discussed will be the PoiViewManager, this class as the name describes, is responsible for managing all the process of dealing with the information contained in the points of interest (Poi – the class that represents a point of interest, this class consist on a set of attributes and interface methods to access them), from the process of loading the information in memory until the points are presented to the user, drawn on the device's display. This class has to accomplish these tasks:

- Load all the information of the points of interest, this consists on gathering the poi's information from the poi source mechanisms that were discussed earlier, like Panoramio or Wikipedia. For this purpose an interface class was created that helps one load ubiquitously different types of poi. Every type of poi source implemented had to implement the methods necessary to load their associated information. In this way, PoiViewManager does not need to know the manner in which the points are loaded, all it knows is that it has to call the same function and the points are loaded in their particular way. This helps in the scalability of the application, with this interface it is easy to implement other forms of gathering points of interest with different processes, and these processes are then transparent to the application.
- Maintain in memory the information loaded.
- Update that information when it is required, for example, when the user walks to another location it is necessary to update the points of interest in memory with the ones gathered in the new location and the previous data become outdated.
- Calculate the positions on screen of the tags that represent the real world places of the points of interest, the projection is described in section 3.4.

The classes *PanoramioHelper* and *WikiHelper*, are the implementations of the sources of poi, these classes implement the methods that are necessary to load the information of the points of interest from their systems, in this case the Panoramio and Wikipedia sites. Both classes implement the interface *PoiSource* that consist of a set of methods to load points information, their implementations are different, as the way in which they load the points is different as described in the previous section. The *GPSManager* class deals with the localization of the user, the geographical location of the device. This class has the task to manage the location providers and to update the location in the application. This class extends the class LocationManager provided by the Android SDK, that has all the necessary methods to manage the localization of the device.

3.2 Gathering Points of interest

The process of developing an application of this kind relies on the use of several services and systems, like the ones listed section 2, that have their limitations, and as a consequence limit the application itself. It is up to the developer to overcome these limitations with a design solution that renders itself still useful. In this section we list the most relevant problems when dealing with online data for a mobile device, along with our design solutions that were implemented. The key problems were: Size of Data; Age of Data; Dynamic Data;

The ***Size of Data*** in any mobile application is crucial, and in the context of this application it has a great impact on it's design. First of all, the data, in the context of WorldPlus, is composed by all the information associated with the points of interest: latitude and longitude, a title that represents the place, images of the place, a description such as historical facts of that place, along with many other characteristics that can be associated with a point of interest and could be important information for the user experience. The objective of the application is to gather points of interest that are in the surrounding environment where the user is located, as we don't know where a user will use the application, it is important to have a large number of points of interest located all around the world so that people from all places in the world can experience the application.

One of the sources of points of interest are the Wikipedia articles as mentioned previously, for a few years now, wikipedia has started to geographically reference some of their articles. One problem arises: how can we process all this data?, Wikipedia has a very large number of articles which means dealing with gigabytes of information in the application, something which is impossible to process with the mobile devices of nowadays, where the space of an application of this kind normally amounts to only a few megabytes. So the challenge is not to store the data in the device, but to create a way in which the application can retrieve this data, filtered in some way, so that it could be processed by these devices and the information presented to the user with a nice fluid experience. Our main interest was on the creation of the AR system itself, we did not solve the problem of transferring and parsing vasts amounts of data from wikipedia, instead we stored in the application parsed content from a few points of interest in order to test basic rendering functionality. In the case of the Panoramio API, we do have direct access to all the available online content. One needs to make the request to the API and process the response, this revealed itself as the most time expensive process in the execution of the application. This process is composed by the request, parsing the data of the response and transfer of the image files which are typically photos of the respective locations and can have arbitrary resolutions.

Age of Data. It is crucial to have data to present in the application, the basic objective of the application, but it is equally important that the information is correct, that it represents exactly the state of the point of interest in the moment that the user requests it's information. For example a file showing train delays. It is important to have a system for point source that is easy to update so the user always sees correct information, and not useless outdated data, the lack of a system that support this issue, could cause failure of some applications.

Dynamic Data In a tourism hotspot, for example Rome, it would be useful to have a system that can show the photos just taken by others in key places of the city, so other people can visit those places too. A person could be submitting a photo and another person nearby could see that photo through the application, for instance to help navigation. But with this dynamic aspect arise some problems that consist on the amount of photos in a given area, e.g. if we were to define a 1 km radius around the location of the user and retrieve all the photos taken inside that radius, the amount of photos retrieved in a place like Rome could still be prohibitive interaction wise. A simpler solution that worked well is to limit the number of points that are computed and drawn with a fixed number, some tests have been done to determine empirically the limit of points of interest that can be computed interactively in the application, these tests will be described in the results section. But another issue arises, how can we choose which points of interest to discard and which ones are the best to present to the user. This can be achieved by determining the distance at which the point of interest is from the position of the user, and then discard the ones that are more far, compute and present just those near the user´s location. This is a straightforward solution to overcome the problem of having to compute too many points of interest but there are more elegant solutions that could be implemented in the future. For example implementing machine-learned ranking[4] techniques that can determine the most relevant points, as used in Information Retrieval[5] systems, such as search engines, etc. Points of interest are ranked by an algorithm that computes a group of special characteristics of the points and determines their relevance to the user, this relevance then determines which points are more interesting/relevant to be presented. The application could profile the user by saving user interests, and then the application could assign different scores to poi categories based on those interests, some categories would have more relevance than others or one could create a way of scoring based on the distance the poi is from the user.

3.3 Sensors

Another problem encountered when developing this application was the noise that the sensors report in their readings. The accelerometer sensor of the devices returns values of the acceleration that are imposed in the device to realize the position of the device in relation to the real world, as stated in the previous section. These values are returned in time intervals of milliseconds, and sometimes they return values that are not consistent with each other, if we put a device in a rest position on top of a table for example, we can see that the sensors don't give always the same value, the values oscillate around the real value. Our solution consists on simply saving the values returned by the sensor in a temporary collection of data and before the projection of the points, an average of the values is calculated. Three read outs are taken and averaged in under 100 ms. With processing units with multiple cores one envisages that Kalman filtering in a dedicated core could be used to further improve results.

3.4 Point Projection

The point projection is the process that maps the real places on the display of the device, using tags that consist of a bitmap that represent the points. Our calculations are based on the work of Matuscheck [Mat11a]. The projection is divided in two different calculations, the horizontal projection and the vertical projection, the first consists on calculating the screen coordinate in the *X* axis (horizontal) and the second consists on calculating the Y screen coordinate. To draw a point in the device's display it is necessary to have an *X* and an *Y* coordinate *P(x,y)*. The horizontal projection is based on two similar but different values, the azimuth that we mentioned earlier, that the magnetic sensor provides, and the bearing, that is the angle formed between the true north and a given point on Earth that the device is facing. Before describing these calculations it is important to define the *angle of view*, the angle of view in this application is defined by a 45 degree angle, in the horizontal and in the vertical projections. In the case of the horizontal projection this angle will define two boundaries, like two "arms" that embrace the field of view of the application and determine what is inside of the field of view and shown to the user. Figure 4 illustrates these concepts. As can be seen the azimuth is the angle formed between North and a point that the device, in this case the little man is facing. The bearing is calculated in the same way but it requires an additional point, but if we assume that the point that the little man is facing is a point of interest we can say that the bearing to that point has the same value as the azimuth. In the projection if the azimuth and bearing to a point is the same or very close, that point is a good candidate to be drawn on the display, it means that the device is facing in the direction of that point. After knowing the two angles needed we can define a horizontal screen coordinate from them, we know that the azimuth is always in the middle of the screen, if the bearing has the same value as the

[4] Machine-Learning Ranking - is a type of supervised or semi-supervised machine learning technique where the objective is to create a ranking model with gathered data.

[5] Information Retrieval – area of study with methodologies for searching documents, information inside documents, etc. In these methodologies there are concepts of scoring special characteristics of the documents that determine their relevance.

azimuth the position of the point will be in the middle of the display.

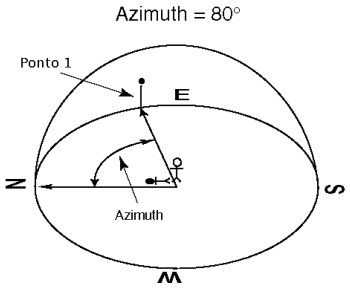

Figure 4: Azimuth calculation (adapted [Phy12a]).

In the case that the bearing to a point is greater than the azimuth then we know that the point is in the right half of the screen, or else, if the bearing is less than the azimuth we know that the point is in the left half of the screen. With this we have a horizontal screen coordinate. Now that we have the horizontal screen coordinate for the point, it is necessary to calculate the vertical coordinate for it, the vertical projection, that consists in calculating the *Y* coordinate that the point will have on the device's display. Like in the horizontal projection, the vertical angle of view is defined with an angle of 45 degrees.

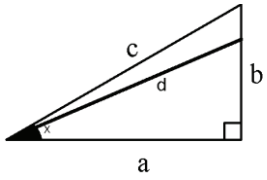

Figure 5: Altitude culling calculation.

Figure 5 shows the triangle *(a,b,c)* that represents half of the vertical field of view. As mentioned before the full vertical angle of view is 45 degrees, the full vertical field of view would have a similar triangle mirrored in the bottom of the one presented.

It is important to define some variables before we calculate the vertical projection, first of all the position of the user and it's device, the user is located in the vertex formed by *ca* facing to *b*, *c* is the upper boundary, *a* is the line that connects the position of the user with the position of the point of interest, *a* represents the distance between those two points. The *b* line represents the height of the point in relation to the height of the position of the user. For example, if the user is at 500 meters high and the point has an altitude of 630 meters *b* will be the difference between them, 130 meters. Representing the position of a point is defined by a line *d* that also represent the angle formed between the position of the user and the position of the point of interest. This is the angle that we need to determine if the point of interest is in the field of view or not, if it will be drawn in the device's display or not. We calculate this angle using the Law of Cosines, this law trivially determines a set of rules that can be used to calculate angles and lengths of the sides of a triangle that has a 90 degree angle.

We have the distance between the two points (*a*), we have the height of the point in relation to the height of the user position (*b*), we know that the angle formed between *a* and *b* is a 90 degree angle, what is missing is the angle formed between the user position and the point of interest position. This is what we are going to calculate next using a rule of the Law of Cosines. The Law of Cosines is an generalization of the Pythagorean Theorem, the base is the next formula.

$$\boldsymbol{d}\text{^}2 = \boldsymbol{a}\text{^}2 + \boldsymbol{b}\text{^}2$$

Here ***d*** is our hypotenuse (squared) and is equal to the sum of the squares of the two sides (***a*** and ***b***). We can now obtain the length of ***d*** that will help to determine the angle that we are looking for, using the next formula.

$\boldsymbol{a} = \boldsymbol{d} * \cos(y)$, where *y* is the angle that we need.

By transforming the equation we obtain the angle.

$$cos(y) = \boldsymbol{a} / \boldsymbol{d} \text{ or,} \qquad y = cos(\boldsymbol{a}/\boldsymbol{d})$$

In this way the angle formed between where the user is looking at and the height of the point is obtained. Finally, and similar to the horizontal projection, we just need to compare angles, the angle ***y*** (calculated just now) and the angle that restricts the vertical field of view represented as ***c***, if the angle *y* is lesser than the boundary angle it means that the point is in the angle of view, if it is greater it means that the point is outside the angle of view and it will not be drawn in the device's display. In this way we can determine if the point is inside the field of view and calculate the exact screen coordinate.

4. RESULTS

In this section we present results from WordPlus, our augmented reality application. Results were carried out on a HTC Wildfire, with a 528 Mhz CPU, 512MB ROM and 384MB RAM. Fig 6, shows how the application is shown to the user. Namely the application consist in showing the user points of interest (poi) that are located near the location of the user, in this case four points of interest are shown (in green), which were gathered from the Panoramio system, the user can interact with these tags and request additional information.

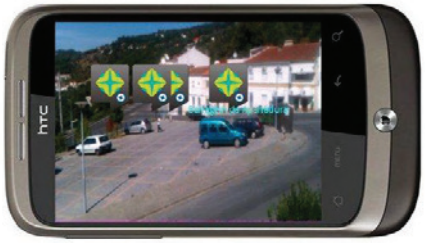

Figure 6: Points of interest (Poi) in WorldPlus.

The user "browses" the world that surrounds him and discovers points/green tags when he directs the camera at them. When the user holds the device with an inclination the points will react to that motion, pushing the points up if the inclination is towards the ground like (Figure 7).

Figure 7: Poi projected with inclination.

Or with an inclination towards the sky pushing the points to the bottom of the screen (Figure 8).

Figure 8: Poi projected with inclination.

We found that the biggest bottleneck for rendering is the projection calculation. In order to determine the maximum number points of interest that the system can compute coordinates and render whilst still being interactive, we performed some tests running the application with different amounts of points of interest in memory at a given time. Namely tests computing 25, 50 and 100 points of interest were carried out. In the first case the application runs smoothly in a range of 10-15 frames per second, the user can have a fluid navigation experience with no "freezing" effect at all. With 50 points the result was a range of 6-10 frames per second, again a good experience is provided, it is possible to move the equipment around and the points are projected almost instantly. With 100 points the result was 4-8 fps, in this case when the user moves the equipment it can take a couple of seconds to update the points on the screen. These tests led us to limit our system to 50 points of interest so that the user can have the best experience with it. On the network/transfer of images side, tests showed that 20 points of interest retrieved from the Panoramio take an average of 9 seconds to load and parse all the data. We note that with the parsed offline wikipedia dump file, these access lags are non-existent as points are loaded into main memory when the application starts.

5. FUTURE WORK

In the future, we would like to obtain more sources of points of interest, there are many services online that contain geographical referenced information that can be used in this type of application. Our application is prepared for scalability, new sources of points are easily implemented with a few lines of code. We would like to create a way for the users to create their own points of interest and submit them so that they can be shared with friends or other users of the service. We would like to find a way to more efficiently parse Wikipedia. One way would be to set up a computer that stores a database with all the information of the Wikipedia articles that are needed in the application, and create some services for accessing that data across the internet. In this way it would be possible to update the information that is used in the application in a transparent manner, on the side of the application it is not known if the information stored in the server is up to date or not, it just makes calls to retrieve the data. All the updates could be achieved by a routine on the server side that picks up the new dumps given by the wikipedia, those dumps could be parsed and the resulting information could be stored in the database that is accessed by the application.

6. CONCLUSIONS

We have presented WorldPlus. An Augmented Reality Application with Georeferenced content for smartphones. We described the requirements, concepts, components, problems and designed solutions using the Android platform as an example. We presented problems for mobile devices such as the size of the retrieved data and dynamic content that are likely to still be research topics in the future.

REFERENCES

[Jun12a] Junaio Home Page http://www.junaio.com/

[Lay12a] Layar Home Page http://www.layar.com/

[Mil94a] Milgram P., Takemura H., Utsumi A., Kishino F., 1994. Augmented Reality: A class of displays on the reality-virtuality continuum..

[Mat11a] Matuscheck J. - Finding Points Within a Distance of a Latitude/Longitude Using Bounding Coordinates:http://janmatuschek.de/LatitudeLongitude BoundingCoordinates. At 2011-08-10.

[Pap08a] Papagiannakis, G., Singh, G., Thalmann, N., 2008, "A survey of mobile and wireless technologies for augmented reality systems2, in Computer Animation and Virtual Worlds, Vol. 19, No. 1, pp.3-22.

[Rei04a] Reicher, T., 2004, "A Framework for Dynamically Adaptable Augmented Reality Systems", PhD thesis, Technische Universität München, library.

[Sek12a] http://sekaicamera.com/

[Wik12a] http://www.wikitude.org/en

[Phy12a] www.physics.csbsju.edu/astro/CS/CS.05.html