A Case Study of Augmented Reality for Mobile Platforms

Gabriela Tinti Vasselai
Dalton Solano dos Reis
and Paulo Cesar Rodacki Gomest
Department of Systems and Computing
FURB – Universidade Regional de Blumenau
Blumenau, SC, Brazil
e-mails: gabriela.vasselai@gmail.com
{dalton, rodacki}@inf.furb.br

***Abstract*—This work describes the proposal of an architecture that evolves the concept of augmented reality with geolocation to mobile device platforms. The architecture allows the development of applications that draw points of interest on screen, represented by arrows and panels that directs to each point of interest's location. It uses OpenGL ES 1.0 library to draw the points of interest. The applications' user interaction can be made by moving the device in such a way that activates compass and accelerometer sensors. The device's location determines how far are the points of interest. This work also presents development resources that allow developers to use camera, sensors and geographic coordinates on applications inside the devices simulators. At the end we present an application example along with performance results running on an Android platform's simulator and also on a Android device.**

I. Introduction

Through Augmented Reality (AR) is possible to combine the real world with virtual objects such as images, sounds or touch. This concept can be applied in diverse areas such as construction [1] and games [2], bringing the concept more and more to people's life and to mobile technologies that they use at a daily basis. Therefore it is necessary to study mobile technologies that meet the needs of technical applications of AR, such as the registration of objects and rendering in real time.

Devices such as Head Mounted Displays (HMDs) are the dominant technology for AR applications [3]. However, these devices still have large optical constraints (limiting the scope and focus of vision), technical constraints (limited resolution and unstable images) and the human factor constraints (due to device's large size and weight) [3] . Since mobile devices allow to the user greater control of space in which he will have the augmented reality visualization for it can be moved in any direction [3]. On the other hand one must consider the limitations of mobile devices such as less processing, smaller memory and battery life [4], which offer bigger challenge to real-time immersive rendering applications.

For the record of virtual objects' positions in real environment, some strategies can be used, among which we can mention the use of markings [5]. Another strategy commonly used in devices with low processing power of images is the geographic location, also called geolocation. In this case, the virtual objects are registered in a particular geographical position and by means of GPS and motion sensors applications can measure where the virtual object should be shown [3]. This strategy has a strong dependency on the accuracy and response time of the sensors, which can directly impact the applications that need to generate a response in real time. Another challenge is related to the balance between application's response time and high battery consumption of the device that motion sensors generate.

In general, the platforms for mobile devices such as iOS [6] and Android [7] have GPS support on their devices, allowing the use of geolocation services for the registration of virtual objects. Also motion sensors such as accelerometers and magnetometers (compass), are available on these platforms in order to assist the tracking of virtual objects as the device rotates along it's azimuth, pitch and roll axes.

On mobile devices, such axes consist of the three-dimensional cartesian coordinate system of the physical device and serve as reference for the measurement of rotational movements captured by the accelerometers, as illustrated in Figure 1.

Besides having the hardware limitations described above, mobile devices offer a big challenge for the development of applications with AR. For instance, the development tools usually include a device simulator, but it does not simulate the device's camera features and motion sensors, making the simulation less useful for developers of AR applications.

The present paper proposes the development of an architecture for applications with AR within mobile devices, using the device's video camera to capture images of the actual environment, the GPS features to record the virtual objects and the motion sensors to track the movement of the device. This architecture also includes tools that enable the development of AR applications through the simulator, capturing images from a camera connected to the developer's computer and simulating the movement of the sensors via the computer's mouse.

This paper is organized as follows: Section II presents related work in augmented reality with geographic location.

Fig. 1. Azimuth, pitch and roll axis.

Section III details the architecture of the application presented in this paper, explaining the proposed solutions and the technologies involved. Section IV describes the simulation features developed by the authors to develop mobile augmented reality applications with the use of device simulation. Section V focuses on demonstrating and analyzing the results from tests done on the application architecture. Finally, Section VI presents final conclusions on the project and directions for future work.

II. Related Work

In this research, we have not found related works that would support the development of mobile applications with AR utilizing device's simulators in order to evaluate AR related features. Thus, we have focused on works related to mobile AR architectures with geographic location. For the selection of related works, we have considered projects for mobile devices that also have some kind of strategy for registering objects through geographical coordinates and tracking objects through sensors.

The Layar framework project [8] enables application development with AR through the project's server. It provides a software that has to be installed on the mobile device that does not require customization by the application developer. The virtual objects are superimposed on images from the camera, and are rendered in real time according to the geographic location of the device. The mobile software also has resources for 3D virtual objects rendering, which may be accompanied by sound effects, providing a better user interaction experience.

Magnitude [9] is an academic open source project, which aims at providing practical services of AR by the students of INSA Toulouse School of Engineering, creating a modular and reusable framework for other applications of AR. Its architecture has a mobile application for Android platform, and also a Java Enterprise Edition (JEE) application server that informs the most appropriate points of interest.

The Wikitude [10] mobile application presents points of interest near the device's location through the overlapping information over the camera image in real time. It has compatibility with various mobility platforms such as Symbian, Android and iOS. This project provides an API for developers along with tools for creating augmented reality layers integrated to mobile applications. The Wikitude application has a mobile browsing functionality called Wikitude Drive [11], which guides vehicle's drivers using the concept of AR within a mobile device.

The OMTP BONDI [12] project aims at developing of APIs for major mobile platforms, using the resources of sensors, GPS and camera and providing such resources through the device browser. The strategy adopted in this project was the independence of the mobile platforms, adopting the use of technology inherent in the web browsers. Thus, application developers need not to worry about which platform they are developing, as far as they work with AJAX and browser capabilities.

The case study of this paper is similar to the Magnitude project because both have an application server that provides the most suitable points of interest. The main difference between the two projects is that Magnitude relies on Android platform specific libraries for designing the virtual objects, while the present work uses OpenGL ES which is available in major mobile platforms.

III. Proposed Architecture

The proposed architecture has a component-based organization in order to abstract the hardware layer from the layer that works with the graphical interface. Figure 2 shows it's main components along with their relationships.

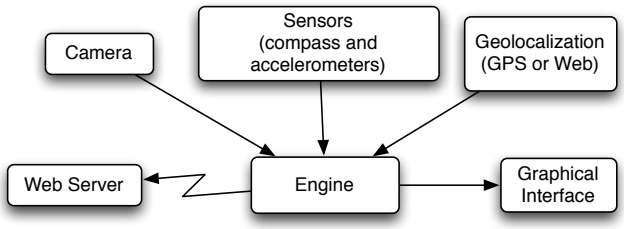

Fig. 2. Proposed architecture.

The Engine component has the responsibility to perform all calculations on the application, preparing the positioning of each virtual object (or point of interest) that will be displayed in the graphical user interface. When there is a change in geographical location, the engine calculates the distance of each point of interest in meters according to the WGS84 standard [13]. The National Geospatial-Intelligence Agency (NGA) maintains the World Geodetic System (WGS) which is the standard system used to define the irregular shape and size of the Earth [13].

When there is any change in compass and accelerometers sensors, the engine gets the values of azimuth, pitch and roll and calculates the angles of each point of interest according to the device orientation. Figure 3 shows the life cycle of this component through its states.

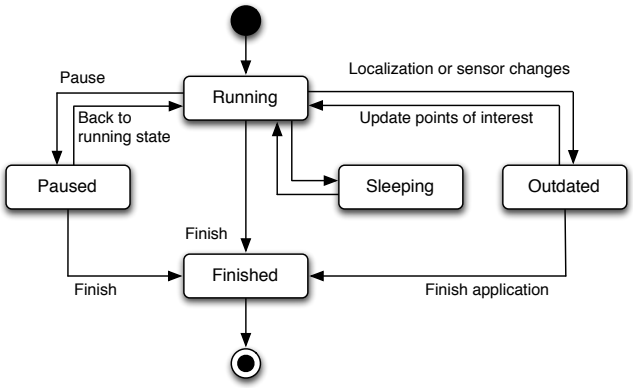

Fig. 3. Application's life-cycle.

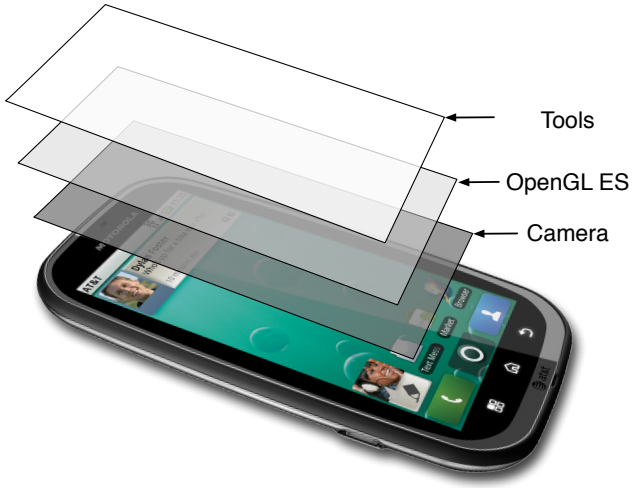

Fig. 4. Interface layers.

The need to implement a state machine was mainly motivated by the concern with battery consumption and response time of the motion sensors and GPS. The "Running" state indicates that the Engine has calculated the position angles and points of interest in accordance with the latest information from the sensors, avoiding the occurrence of recalculations until there is some change in the sensors. Upon receiving such changes in sensors or GPS, the engine changes to the "Outdated" state, indicating that points of interest must be recalculated. This condition also arises when new points of interest (that need to calculate its position) are received. The "Paused" state prevents the engine from receiving new information from sensors and GPS while the application is stopped or in background mode, thus reducing the device's battery consumption.

The graphical interface has three overlapped layers in order to create the impression of Augmented Reality (Figure 4) . The innermost layer is responsible for reproducing the images received by the camera, the middle tier layer uses a three-dimensional OpenGL ES space that has the points of interest and the third layer draws the configuration tools graphical symbols, such as radar tool graphical elements.

The outermost layer, responsible for the rendering of tools symbols, allows users to configure the space of augmented reality, adjusting the range of points of interest that will be drawn. This layer has a radar tool with the four main compass points (north, south, east and west), showing the direction and distance of points of interest that are within a radius of greater range.

The outermost layer has translucent background and the tools interface elements were placed near the screens corners in order to enhance the user experience.

The layer that draws the points of interest utilizes the OpenGL API for Embedded Systems 1.0 [14] library. This library is used by major mobile platforms for 3D rendering capabilities. The configuration used in the OpenGL ES layer has a three-dimensional space to draw the points of interest in depth through the resources of rotation, translation and scale. Through the Engine calculations that the information of angle and position of each point of interest are available for this layer.

Among the available features for drawing objects in OpenGL ES, the orthographic projection vertex data drawing was chosen for this work. Although this approach uses the device's main memory, it is supported by all devices that have OpenGL ES. Another feature available in OpenGL ES is the Vertex Buffer Object (VBO) drawing, which makes use of vertex buffers in the graphics processing unit (GPU). This strategy is not supported by all devices because it does not use the device's main memory.

The choice of using OpenGL ES for this layer was characterized by two main factors, the first is portability among various mobile platforms and the second is the performance obtained by OpenGL ES in relation to other available design strategies.

Because it is portable on major mobile platforms, OpenGL ES has some limitations comparing to OpenGL, originally used in desktop computers. The biggest difference between OpenGL and OpenGL ES is in the immediate mode with glBegin and glEnd calls, which are not implemented in the ES version. Therefore it is only possible to draw primitives using vertex arrays. The second major difference is that OpenGL ES do not support floating-point numeric type. It uses fixed-point numeric type for vertex coordinates and attributes in order to provide better support for embedded systems.

Among the limitations in OpenGL ES, the most important concerning the implementation of augmented reality is the lack of support for text drawing, either in 2D or 3D spaces. This feature, in the present work, could be used to determine the name of the points of interest and also the distance from

the user to a point of interest. Consequently, investigation of existing strategies to meet this need was necessary.

One strategy is the Springtext API found in [15]. It was necessary to adapt the Springtext implementation in order to draw text in a dynamic way, since the points of interest coordinates could change as the device moves. The strategy, common to OpenGL applications, consists in drawing the characters from the ASCII table in an array, creating a texture for each character. To compose a word, the letters textures are combined to create the texture of the entire word.

The architecture also includes the functionality of virtual objects selection through touch events on the device's display. To this end, the Ray Picking algorithm was employed [16]. The algorithm transforms the two-dimensional display touch point in a straight line in the OpenGL ES three-dimensional space, allowing the calculation of collision of touch gesture with any object in the virtual three-dimensional space.

As a way of representing each point of interest, the graphical interface draws two flat objects in a three dimensional space: an arrow and a panel. The arrow indicates the direction and distance of a point of interest relative to the current device location.The arrow becomes visible when the device's camera is pointed towards the ground, as depicted in Figure 5.

Fig. 5. Device's camera pointing downwards.

The panels are displayed when the camera is directed forward, in an horizontal orientation and the device is being held at shoulder height (Figure 6). Each panel shows the name of it's corresponding point of interest.

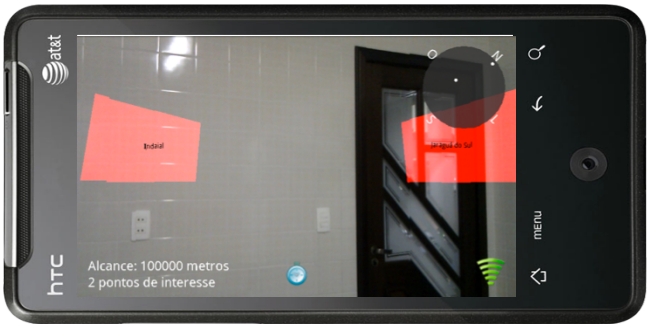

Fig. 6. Device's camera pointing horizontally.

The Web Server component was created to provide points of interest closest to the device's current geographic location. This server implements a JEE Servlet [17] that receives HTTP "get" requests in CSV format and returns the more appropriate points of interest. The Web Server component selects the points of interest on a database that contains information of each point, such as latitude, longitude and name. When communicating with the Web Server component, the device sends its geographical location and also the maximum range radious in meters, so the Web Server can fetch the most appropriate points. The application running on the mobile device connects to the Web Server via the cellular data network or wireless network configured on the device.

IV. Development Tools

The platforms for mobile devices such as iOS, Android and Blackberry have tools to assist the development of applications so the developer generally does not need to purchase a device to test the application being developed. Nevertheless the device simulators have limited resources on the simulation of some hardware resources, such as cameras and sensors. When developing applications with AR, the use of such resources is crucial, making the simulator an incomplete tool for testing this kind of application.

Since the architecture proposed in this paper works independently of the used mobile platform, not having the need to purchase one device for each of the major mobile platforms has become essential for the development process. Thus, the use of simulators with simulation features of the camera, sensors and geographical location became even more relevant.

In the architecture described here, the camera components, sensors and geographical location abstracts all interaction with hardware devices, so that other components need not determine which device is running the application or if it is running in a device simulator. This verification is done at runtime, it identifies whether the application is running in the simulator, thus carrying APIs simulation, or whether it's running on a device, thus carrying the APIs that interact with the mobile device's hardware.

For camera simulation we developed a socket server that must be executed on the developer's computer. This server gets a connection with the cameras that are connected to the computer through the JMS architecture [18] and allows the developer to choose which camera the images for simulation will be obtained. The images are then made available through a socket port. The camera component within the application's architecture get the images at a configurable refresh rate and provides the camera layer to the simulator screen.

For the simulation of sensors it was used the SensorSimulator software [19]. It has a graphical interface that allows the change of sensor values via mouse and provide such values in a socket server. This server provides the values of the sensor's motion in a socket port through a specific communication protocol. The Sensor component within the proposed architecture obtains the sensor values and triggers change events in the Engine component, therefore calculating the new positions of points of interest.

The simulation of geographical location was developed directly in the client application and a screen for configuration of latitude and longitude is provided to the user (developer). It is also possible to simulate a walk through using the computer keyboard arrow keys.

V. Test Results

In order to evaluate the proposed architecture for mobile AR applications, development and testing were done in Android platform, one of the leading emerging platforms for mobile devices. The tests considered the response times of both the engine and the graphical interface, because an AR application should calculate and render points of interest quickly to create the immersion for the user. We also tested the processing power of the simulator compared to an Android device. Tests were performed involving different amounts of points of interest to be calculated and drawn on the screen. The quantities were 2, 4, 8, 16, 32 and 64 points of interest. For each point of interest where drawn two objects in OpenGL ES layer: the arrow and the panel.

TABLE I
Simulator FPS

Number of Points of Interest	Engine's average FPS	Graphical interface's average FPS
2	502.65	22.78
4	283.74	13.03
8	91.51	4.50
16	15.69	3.75
32	5.48	2.43
64	2.61	1.11

All tests were performed in the same way using both the simulator and the Android device. The HTC Desire device was used. It consists of a smartphone with all the features needed for testing, such as accelerometers, GPS, video camera, among others [20]. The application has different codes for hardware interaction (camera, sensors and geographic coordinates) when running on the device and in the simulator. Therefore this interaction is not measured in the tests and should not influence the following analysis of results. The first set of tests was performed on the simulator, generating data in frames per second (FPS) that is available in Table I. One must observe that the value of the FPS in the graphical interface is way below the ideal.

TABLE II
Device FPS

Number of Points of Interest	Engine's average FPS	Graphical interface's average FPS
2	4,273.17	106.25
4	1,897.56	58.22
8	1.498,11	49,44
16	979.24	27.96
32	792.00	17.08
64	286.22	12.32

The second set of tests was run on the HTC Desire device and not only presented an improvement in the performance of the engine as there was a considerable improvement in the performance of the GUI. It can also be seen in Table II that the number of objects has a direct impact on the performance testing application.

Based on two measurements of the engine data it was possible to generate the graph in Figure 7 which represents the performance obtained by measuring the FPS data in the vertical axis and the number of points of interest in the horizontal axis. It became very noticeable the difference in application performance between the simulator and the HTC Desire device.

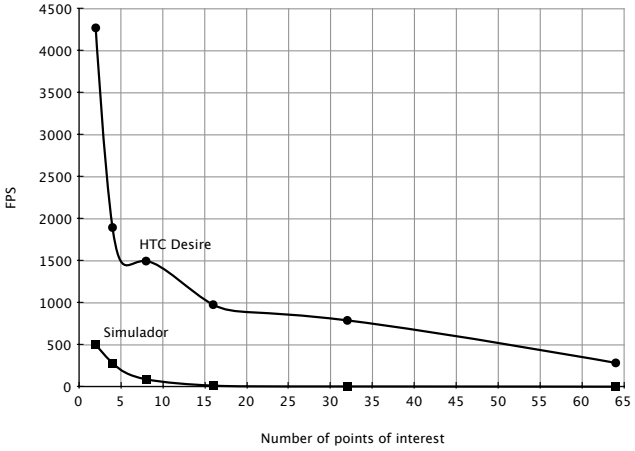

Fig. 7. Engine's FPS

It was also possible to generate a graph showing the performance of the graphical interface (Figure 8) measured in the tests.

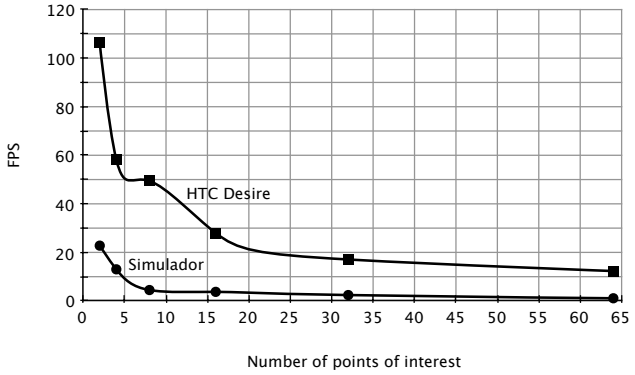

Fig. 8. Graphical interface's FPS

The results obtained by performance tests in the Android simulator are below the results of the HTC Desire, both in the execution of the engine and the rendering of the graphical interface. For the Engine processing, the architecture showed a high performance even with 64 points of interest. Moreover, the design of points of interest in the graphic interface had an acceptable performance up to 8 points of interest, demonstrating that the strategy of drawing the OpenGL ES still

needs optimization. We can also conclude from the tests that when is necessary to evaluate the performance of applications developed for the Android platform with existing development tools, the measurement should not be based solely on the results of the simulator tests.

After analyzing the test results it was necessary to investigate why the graphical interface, more specifically the OpenGL ES layer, showed poor performance.

TABLE III
DEVICE FPS – SECOND MEASUREMENT

Number of Points of Interest	Graphical interface's average device FPS
2	601.14
4	405.12
8	252.20
16	148.06
32	82.57
64	49.19

An analysis was made in the text drawing strategy in OpenGL ES. The used strategy forces the program to load a texture for each character in ASCII table, so this could be one of the factors that affect the performance of the graphical interface.

For performance measurement, initially, we removed all the functionality for text drawing . Let us e call this measurement "second measurement". The results can be seen in Table III.

The results presented in the second measurement were much higher than those obtained in the first measurement, showing that the design strategy of dynamic texts used in this work has a significant impact on application performance as depicted in figure 9.

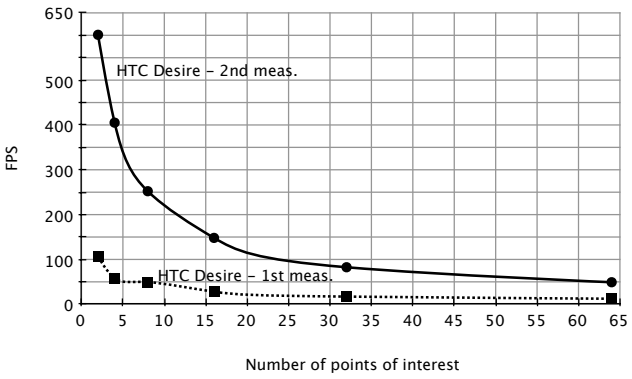

Fig. 9. Graphical interface's FPS

The present work isn't focused in specific OpenGL ES matters, it is not necessary to propose another solution for text drawing. On the other hand, we could replace the OpenGL ES text drawing with another solution that would be best suited for use in augmented reality. Thus we propose the use of images and textures for the virtual objects represented. The OpenGL ES API has the functionality to upload images of objects in the form of textures and draw them in certain locations on the screen. Conceptually this concept meets our design needs.

Another solution to improve the performance of OpenGL ES functionality is by using Vertex Buffer Objects (VBO). The basic idea of this feature is to provide memory space, called buffers, which are available through identifiers [21]. The VBO provides control trough buffer objects mapping and defines the type of each buffer, for instance, vertices, colors and indices. This allows graphics to optimize the management of internal memory choosing the best type of memory (such as cached/uncached system memory or graphics memory) which will store the buffers [22]. Because VBOs allow a better management of device's memory and have more optimizations with respect to the use of the device's GPU, we chose to investigate its use in augmented reality.

TABLE IV
DEVICE FPS – THIRD MEASUREMENT

Number of Points of Interest	Graphical interface's average device FPS
2	677.72
4	429.29
8	261.42
16	154.52
32	101.34
64	90.91

The augmented reality application prototype required a lot of changes with respect to the GUI component to suit the use of VBO on all objects drawn on the screen. The performance test uses the application version in which the texts design strategy had already been removed, so the VBO solution came to increase the previous solution's performance that was observed in the second measurement. Let us call this experiment as the "third measurement". The corresponding results can be seen in Table IV.

Comparing the performance measurements made in the first and second version of the application, with the latest measurements, we could generate the chart depicted in figure 10.

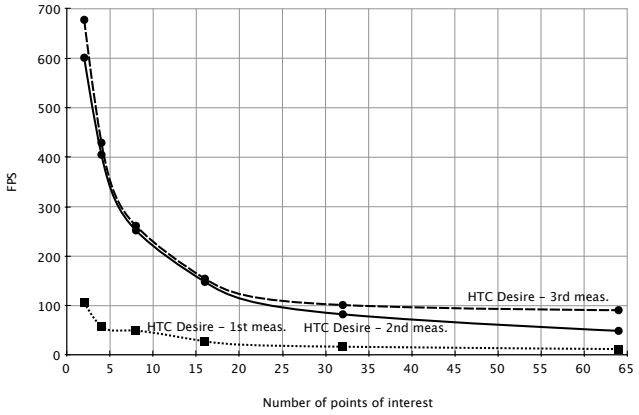

Fig. 10. Graphical interface's FPS

The results presented in the third measurement were higher than those obtained in the first and second measurements, showing that use of the VBO had positive impact on the

graphical interface's performance. The second and third measurements show good results regarding the improvement of the mobile application's performance, not only presenting an appropriate FPS rate for augmented reality applications, but also proposing a solution that is rather complete, allowing the replacement of conventional OpenGL text drawing strategies to textures with VBO.

VI. Conclusion

This paper presented an application architecture with AR using geographic coordinates for the registration of virtual objects, called points of interest. The impression of augmented reality is ensured through the use of video camera, in which virtual objects are drawn over the images from the camera. The interaction between reality and virtuality occurs through the movement of the device, triggering the compass and accelerometers. The paper also presented a set of tools and features that allows the AR application developer perform tests using the mobile platforms device simulators.

The features here are similar to those of related work. Among them we can highlight the determination of points of interest by geographical coordinates, accelerometer and compass; reality visualization through the camera and the visualization of virtual objects rendered in 3D system; and the retrieval of points of interest through a web server.

This case study verified that it is possible the use of the resources available in major mobile platforms such as accelerometers, compass and video camera in AR applications. It was also shown that the use of OpenGL ES can be used to render the immersion through a three-dimensional space, even though its performance and optimization should be focused.

This work confirmed the need for simulation tools that have support for developing applications with augmented reality. We demonstrated in this paper that the simulation of camera and sensors can be made through a connection socket between the simulator and the computer used for development. Based on performance testing conducted, we can conclude that the Android platform simulators still need major improvements to achieve the performance achieved by devices.

The design strategies within the OpenGL ES used in this study were analyzed in order to provide the augmented reality user the best possible experience concerning the mobile device technologies commercially available nowadays. Through the use of VBO, it was possible to increase the performance of the graphical interface to a rather robust rate, allowing up to 64 design points of interest on the screen with 90.91 FPS. It can also be noticed that the initial design strategy aproached in this work caused the greatest impact on performance of the graphical interface, so it was removed from work. Its replacement was done through the use of images in the form of textures with VBOs.

As extensions of this work, we can highlight the rendering of textures with images that represent the points of interest, along with the possibility of using animations and even 3D models. Another possible future work would be to implement touch event handling in the object of immersion, simulating a touch on the virtual object for changing, for instance, its shape or position.

Finally, this paper presented an innovative concept of interaction with the virtual reality via mobile devices, with a case study of emerging platforms. As result, this work presented a functional architecture that stands out not only by immersion, but also for its unique feature of allowing the execution in the device simulator, eliminating the need of deploying the application to a device for testing purposes.

References

[1] R. R. Amim, "Realidade aumentada aplicada a arquitetura e urbanismo," Master's thesis, Engenharia Civil – COPPE/UFRJ, Rio de Janeiro, 2007.

[2] B. Reis, P. Borges, L. A. Vasconcelos, J. M. Teixeira, V. Teichrieb, and J. Kelner, "Markermatch: an embedded augmented reality case study," in *Proc. of the XII Brazilian Symposium on Virtual and Augmented Reality - SVR 2010, Natal, RN, Brasil*, 2010.

[3] O. Bimber and R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*. Natick, MA, USA: A. K. Peters, Ltd., 2005.

[4] C. Gatzidis, V. Brujic-okretic, and F. Liarokapis, "Developing a framework for the automatic generation and visualisation of 3d," in *Urban Areas on Mobile Devices, 10th Symposium For Virtual And Augmented Reality, Joao Pessoa*, 2008, pp. 13–16.

[5] J. Kent, *The Augmented Reality Handbook - Everything You Need to Know about Augmented Reality*. Emereo Pty Limited, 2011.

[6] Apple. (2011) ios dev center - apple developer. [Online]. Available: http://developer.apple.com/devcenter/ios/index.action

[7] Android. (2011) Android 3.0 platform highlights. [Online]. Available: http://developer.android.com/sdk/android-3.0-highlights.html

[8] Layar. (2010) An overview of the layar platform. [Online]. Available: http://layar.com/create/platform-overview/

[9] Magnitude. (2010) Project magnitude. [Online]. Available: http://www.magnitudehq.com/

[10] WIKITUDE. (2010) Wikitude world browser. [Online]. Available: http://www.wikitude.org/

[11] WIKITUDE-Drive. (2010) Wikitude drive. [Online]. Available: http://www.wikitude.org/en/category/02_wikitude/wikitude-drive

[12] D. Rogers, "Bondi augmented reality," in *Mobile AR Summit at Mobile World Congress 2010, Barcelona*. PEREY Research and Consulting, 2010. [Online]. Available: http://www.perey.com/MobileARSummit/Position_Papers.html

[13] NGA. (2010) World geodetic system. [Online]. Available: https://www1.nga.mil/ProductsServices/GeodesyGeophysics/WorldGeodeticSystem/Pages/default.aspx

[14] Kronos-Group. (2011) Open gl es overview. [Online]. Available: http://www.khronos.org/opengles/

[15] A. Developers. (2010) Spritetext – api demos. [Online]. Available: http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/graphics/spritetext/index.html

[16] H. Zhao, X. Jin, J. Shen, and S. Lu, "Fast and reliable mouse picking using graphics hardware," *Int. J. Comput. Games Technol.*, vol. 4, pp. 1–7, Jan. 2009.

[17] ORACLE. (2011) Java servlet technology. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/servlet/index.html

[18] ——. (2011) Oracle streams advanced queuing user's guide and reference 10g release 2 (10.2). [Online]. Available: http://download.oracle.com/docs/cd/B19306_01/server.102/b14257/jm_create.htm

[19] Open-Intents. (2010) Sensor simulator for simulating sensor data in real time. [Online]. Available: http://code.google.com/p/openintents/wiki/SensorSimulator

[20] HTC. (2011) Htc desire - specification. [Online]. Available: http://www.htc.com/pt/help/htc-desire-hd/

[21] A. Munshi, D. Ginsburg, and D. Shreiner, *OpenGL(R) ES 2.0 Programming Guide*, 1st ed. Addison-Wesley Professional, 2008.

[22] I. Williams and E. Hart. (2011) Efficient rendering of geometric data using opengl vbos in specviewperf. [Online]. Available: http://www.spec.org/gwpg/gpc.static/vbo_whitepaper.html