SAPIENZA
UNIVERSITÀ DI ROMA

FACULTY OF
INFORMATION ENGINEERING, INFORMATICS AND STATISTICS

Master of Science in

ARTIFICIAL INTELLIGENCE AND ROBOTICS

Augmented Reality Framework for the 3D Interactive Mobile Representation of the Ancient Forum of Nerva

Supervisor:
Prof. Marco Fratarcangeli

Co-Supervisor:
Prof. Tommaso Empler

Candidate:
Giovanni Murru

Academic Year 2011/2012

Alla mia famiglia

Sommario

In questo lavoro di tesi presentiamo lo sviluppo di un framework su piattaforma mobile per l'estrazione, attraverso un sistema di realtà aumentata, di informazioni riguardanti il settore dei beni culturali.

Informazioni relative a siti archeologici possono essere ottenute attraverso l'interazione con modelli grafici tridimensionali proiettati nell'ambiente reale.

Il framework è interamente sviluppato usando librerie multipiattaforma. Viene dimostrata la flessibilità del nostro framework attraverso la realizzazione di un'applicazione mobile per la visualizzazione aumentata e interativa di una ricostruzione storica del foro di Nerva, uno dei Fori Imperiali di Roma.

Abstract

In this work we present a framework for mobile handheld devices developed using the augmented reality technology with the objective of presenting digital information in the context of cultural heritage.

Information related to archaeological sites is presented by means of interactive three-dimensional computer graphics models, projected on the real environment.

The framework is entirely devised and built using free and cross-platform components. We demonstrate the flexibility of our framework by designing a mobile application for a real case scenario, namely the augmented visualization of a historically reliable 3D reconstruction of the ancient Forum of Nerva, one of the Imperial Fora of Rome, in Italy.

CONTENTS

LIST OF TABLES

INTRODUCTION

Augmented Reality can be thought as the technology that will bring our real world experience to a next step by extending our perception with virtual elements of different nature.

Imagine a technology that can augment your sight, your hearing and even your sense of smell, touch and taste. With such technology human perception will enter in a new era where borderlines between what is real and what is computer generated will quickly fade out.

The application of such technology ranges over a wide variety of subjects: the robotic, military and medical fields are just few of them.

Augmented Reality will create a new kind of human-computer interaction, where the advent of mobile smart-phones, AR browsers and the recent advances in computer vision and artificial intelligence will very likely play a central role in making this happen sooner than we think.

In 2012 Gartner, the world's leading information technology research and advisory company, released a special report evaluating the maturity of more than 1,900 technologies [Gartner, 2012].

In the paragraph *The Human Way to Interact With Technology* the researchers describe our world as a place where people are demanding a more natural way of interacting with technologies, including the Augmented Reality among those ones that will make this happen in a near future (see Figure 1).

The possibility to augment a real scene by superimposing virtual objects is supported by the existence and the development of scientific research fields such as computer graphics, computer vision and artificial intelligence.

Computer vision is fundamental for understanding the structure and the spatial position of real objects with respect to the environment, artificial in-

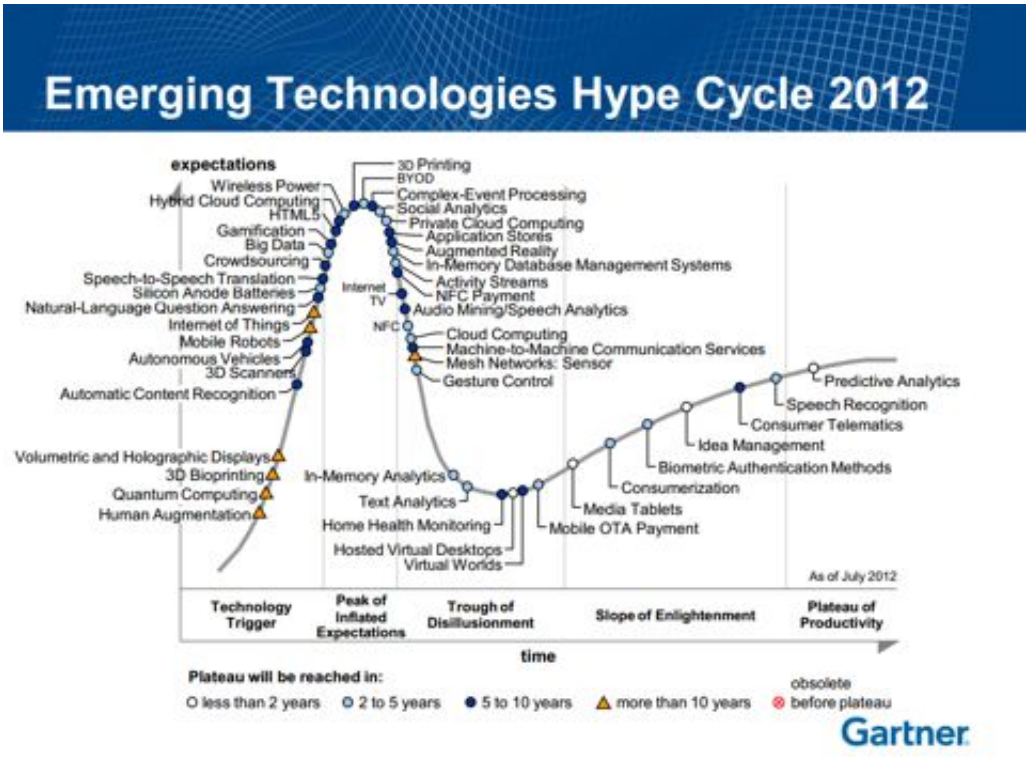


Figure 1: Gartner's 2012 Hype Cycle for Emerging Technologies

telligence is necessary to tag and semantically understand these components of the real world, while computer graphics techniques are essential to translate everything that has been mentioned in something understandable by the human sight.

In this thesis we address the problem of developing a framework for the mobile platform using GPU programming techniques with the purpose of creating realistic three-dimensional models integrated by visual augmentations in the context of cultural heritage.

In particular we used our framework for developing a mobile application that runs on iOS devices, presenting a 3D reconstruction of the ancient forum of Nerva, one of the Roman's Imperial Fora. The work of design and project has been accomplished in collaboration with the faculty of Architecture, Valle Giulia of Rome.

The application projects three dimensional virtual models of the ancient forum of Nerva and of the surrounding environment in real information boards located near the ruins of the forum. The user can interact with the virtual

world using the touch screen of an iPhone, iPad or iPod Touch (see Figure 2).

The choice of iOS mobile platform is motivated by its large distribution in the mobile market, especially in the tablet segment. Furthermore hardware differences on these devices are limited, providing a sort of standardized platform.

As a matter of fact iOS and Android are the today's most used platforms in the market of smartphones and tablets. Together they share more than the 85% of this market. For this reason we chose to develop our application using multi-platforms libraries for the rendering and the augmented reality layers, building a framework of functions that will simplify a future porting of the application on the Android OS.

We conducted a research work on the fields of architecture, archaeology, computer graphics and augmented reality, by establishing a collaboration with experts in these areas. In addition the artistic aspect involving this philological effort must be pointed out too.

Figure 2: iOS Devices: from the left to the right: the iPod Touch, the iPad and the iPhone

Structure of the Thesis

In the first chapter we briefly explain the theoretical background at the basis of our work, going deeper in the concept of augmented reality system and presenting the necessary tools for the development of such system. A section is dedicated to the importance of the pose estimation in the context of the augmented reality systems, explicating the problems of tracking and reconstruction. The importance of using the programmable pipeline of OpenGL is highlighted, clarifying the difference with the old fixed pipeline. We also make a concise introduction to the GLSL shading language, that was actually used for writing the shaders, a special kind of programs that run on the GPU.

In the second chapter we present the state of the art in the augmented reality field emphasizing the current solutions in the mobile platform. Different works of recent literature are described showing the main differences of the various approaches. A section is dedicated to works relevant for the augmented reality in the context of cultural heritage, presenting the current solutions available in the market.

The implementation steps involving the design of the system are explicated in the third chapter. Initially we talk about the problem of interaction with the augmented reality interface, explaining our system for the selection of and interaction with the virtual objects. We point out the choice of multiplatform libraries such as OpenSceneGraph and Vuforia, describing in detail how and why these libraries are used and integrated in our framework. We explain the optimal guidelines to create the best trackable and detectable image targets, the markers used to project the virtual objects. Moreover the main functionalities of the application are presented.

In the final part of the chapter we delve into the detail of implementation elucidating how modeling of the virtual objects was realized. We explain in detail the design of the main functionalities of the framework such as the augmented video playback. Then we describe the steps involving preparation, structure and optimization of the 3D graphical models for the mobile application. Furthermore we explain in detail how some of the shaders we used, such as the normal mapping, work. At the end we do a brief discussion about the flexibility of our framework.

Chapter four is dedicated to experimental results obtained after several

tests that we performed on our framework. At the end of the chapter we do a brief discussion of the hardware platforms used in our tests showing the main differences in terms of performance between the different chipsets mounted on these devices.

Finally in the conclusion we present the results of our work, make a discussion about the positive and negative aspects encountered during the development of our framework, and propose future works and improvements.

CHAPTER 1

THEORETICAL BACKGROUND

Before going on the details of the implementation we'd rather like to talk about some of the theoretical concepts at the base of our work. In first analysis we will explain what is augmented reality underlining the possibilities of this technology and its evolution in time. Then we will briefly talk about the problem of pose extraction in the context of AR systems presenting some known technique in literature and we will explain how the output of this process is used to produce the mathematical tools for superimposing the virtual scene in the real world using OpenGL.

A section is dedicated to the importance of the programmable pipeline, emphasizing the main differences with respect to the old model of OpenGL characterized by the use of a fixed pipeline. We introduce to the reader the OpenGL shading language explaining the concept of shader and the power of these small programs able to run on graphics processor units.

In the final part of the chapter we explain in detail the procedure we we used for creating the shaders, showing in detail the concepts and the math at the basis of one of the most common illumination model of computer graphics history, the Phong illumination model.

The problematics of rendering in mobile platforms are also investigated, underlining the issue of limited resource and explaining the importance of multi-threading and GPU programming.

1.1 What is Augmented Reality

Augmented Reality is a technology that allows the user to extend the real world around him through superimposition of virtual objects.

Augmented Reality does not completely replace reality but rather supplements it with virtual (computer-generated) objects. Ideally it would appear to the user that virtual and real objects coexisted in the same space [Azuma, 1997].

Augmented reality (AR) is the technology to create a "next generation, reality-based interface" and is moving from laboratories around the world into various industries and consumer markets [van Krevelen and Poelman, 2010].

Today's smartphones technology has given a speed up to augmented reality and new forms of human-computer interaction are arising.

Key components essential to build a complete AR system have remained the same since augmented reality concept's birth in the 1960s. In general an augmented reality system must be equipped with display, tracker, graphics computer and appropriate software.

How augmented reality is presented to the user is a fundamental aspect to take in consideration. There are basically three ways to visually present an augmented reality [van Krevelen and Poelman, 2010].

- *Video see-through* is probably the most used and works replacing the virtual environment with a video feed of reality and overlaying AR objects upon.
- *Optical see-through* leaves the real-world perception to the user and displays only the AR overlay by means of transparent mirrors and lenses.
- The third approach, is still an expensive technology that works projecting the AR overlay onto the real objects resulting in projective displays.

Video see-through are the cheapest and easiest to implement. Since reality is presented as a digital video stream, it is easier to mediate or remove objects from reality. This includes removing or replacing fiducial markers or placeholders with virtual objects. Furthermore there is no problem for contrast and brightness between virtual objects and the real environment that can be matched in an easier way.

Using external sensors it is possible to evaluate light conditions of the real world to smoothly blend the computer generated content with the video stream. A disadvantage of the video see-through approach is that the digital representation of reality may have a low resolution and a limited field of view. Moreover user may be disoriented in case of offset between viewer's eyes location and camera position [van Krevelen and Poelman, 2010].

Optical see-through displays have the main advantage to leave real-world resolution intact and solve the problem of parallax because there's no eye-offset due to camera positioning. Optical techniques are in some way safer because user can still see if the power fails. They are ideal for military and medical purposes, however there are some disadvantages caused by the holographic technology such as low brightness and bad contrast, and for this reason they are less suited for outdoor use. Moreover some important limitation in the field of view are caused by the mirrors and lenses used to generate the virtual images that for this reason may result clipped.

Virtual retinal displays or retinal scanning displays (RSDs) solve the problems of low brightness and low field-of-view in (head-worn) optical see-through displays, however this kind of technology is still in development and for now it is only monochrome [van Krevelen and Poelman, 2010].

Based on the assumed location between the viewer and the real environment, three main categories of displays can be identified:

- *head-worn displays*, that are attached to the head (e.g. video/optical see-through head-mounted display (HMD), virtual retinal display (VRD), and head-mounted projective display (HMPD)).

- *hand-held displays*, which must be held by hand (e.g. hand-held video or optical see-through displays, hand-held projectors, smartphones).

- *spatial displays*, that are placed statically in the environment (e.g. screen-based video see-through displays, spatial optical see-through displays, and projective displays).

An augmented reality system should be able to track user position with respect to the real environment so that it can correctly superimpose the virtual objects into the real scene.

Tracking of user's movement should be complete in the whole six degrees of freedom: (X, Y, Z) for position and (yaw, pitch, roll) for orientation.

For tracking we can use different type of sensors such as:

- Mechanical, ultrasonic and magnetic sensors
- Global positioning systems (e.g. GPS, A-GPS)
- Radio sensors (e.g. RFIDs)
- Inertial sensors (e.g. Accelerometer, Gyroscope)
- Optical sensors (e.g. Algorithms based on vision tracking using fiducial markers or marker-less),
- Hybrid systems, that combine use of magnetometers with gravitational tilt sensors and gyroscopes

In addition we have to construct a model of the environment that should be accurate for the tracking purpose. Thanks to AR new concept of user interfaces and interaction have been conceptualized. The WIMP (windows, icons, menu, pointing) does not suit well in AR systems. Gaze tracking, speech and gesture recognition are just few of the alternative ways used to control augmented reality applications.

Another interesting class of sensors are those that gather informations about user's biological state in order to understand the emotional state of the user and adapt the application behavior accordingly.

At the beginning military, industrial and medical were the main field of application for augmented reality systems, however recently also commercial and entertainment sectors showed interest for this technology.

Navigation assistance providing informations such as routes, highway exits, fuel prices is another possible scenario of AR system application.

Another interesting setting for this technology is the guided touring, such as the ArcheoGuide project proposed in [Gleue and Dähne, 2001] and [Dähne and Karigiannis, 2002], that reconstructs a cultural heritage site of Olympia in Greece, allowing visitors to view and learn about ancient architecture.

1.2 Pose Estimation Problem in AR Systems

Objects in real and virtual worlds must be properly aligned with respect to each other, otherwise the illusion that the two worlds coexist will be compromised. Many applications, such as medical ones, demand very accurate registration. Consider for example a biopsy application in which the surgeon has to take a sample of tumor. If the virtual object is not exactly where the real tumor is, the surgeon will probably fail the biopsy. For this and many other applications an augmented reality system laking of accurate registration is not acceptable [Azuma, 1997].

Computer vision is extensively used in AR systems mostly in relation to video tracking. Computer visions' methods are usually articulated in two stages: tracking and reconstructing/recognizing. Tracking algorithms must interpret camera images in a consistent way using feature detectors, edge detectors or other image processing methods. For this purpose it is essential the use of fiducial markers, interest points or optical images that must be detected and isolated in the camera images.

Tracking techniques can be essentially categorized in two classes: those based on feature detection and those using a model of the tracked object's features. This is a requisite for establishing a connection between the 2D image features and the 3D world frame [Carmigniani et al., 2011].

The problem of registration is directly related to the pose estimation problem. Pose algorithms are essentially used to find the transformation that yields the best correspondence between the target and camera models' predicted locations and the known image features [Hoff et al., 1996].

A possible marker-based solution is the one that use video tracking capabilities to calculate in real-time the real camera position and orientation relative to the markers. Once the position is known, we can place a virtual camera to properly overlay the 3D computer graphics model onto the markers.

Markerless tracking is a complex task involving image processing algorithms for natural feature detection with the goal of recovering camera position and orientation. Model-based tracking approaches seems to be the most promising among the several vision techniques used in AR systems. This tracking approach works identifying image features from the object model and aligning the 2D image data with the 3D model.

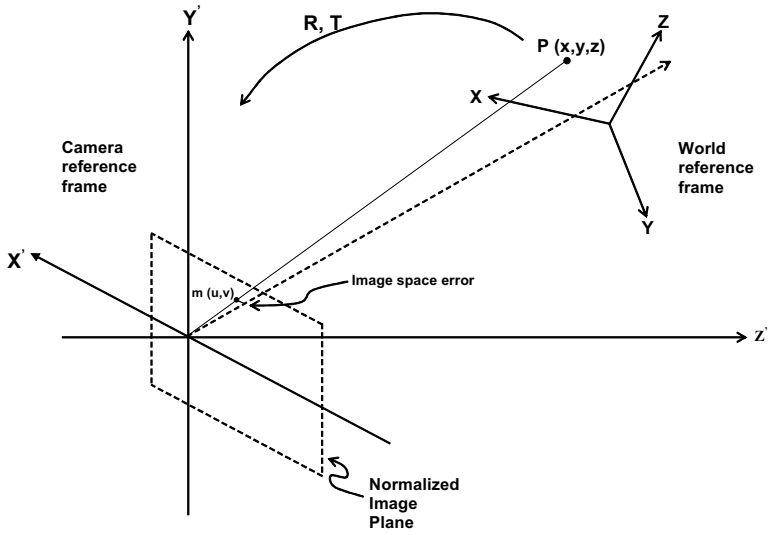


Figure 1.1: Point constraints for the camera pose problem.

A possible approach used to compute camera pose is to minimize the distances between the projection of the model lines and the most likely matches found in the image, then tracking becomes a simple optimization problem which can be solved with known methods such as iterative reweighed least squares.

Another possible solution to the pose estimation problem is the use of visual servoing. In the work [Ababsa and Mallem, 2008] Ababsa and Mallem, two researchers working at the French University of Evry, proposed a robust camera pose estimator combining 2D/3D points and line tracking.

In the second stage of reconstruction/recognition the data obtained in the tracking phase is used for reconstructing objects in the real world coordinate system. Assuming a calibrated camera and a perspective projection model, determination of the camera pose can be achieved using point and line features. Assume a set of at least 3 non-collinear reference points $\mathbf{p_i} = (x_i, y_i, z_i)^t$ in the world reference frame, the corresponding coordinates in the camera-space are given by:

$$\mathbf{q_i} = R\mathbf{p_i} + T \tag{1.1}$$

where R and T are respectively a rotation matrix and a translation vector.

Let L be a line belonging to the 3D object, and let this line be described by its end points P_1 and P_2, then the corresponding line in the image plane is

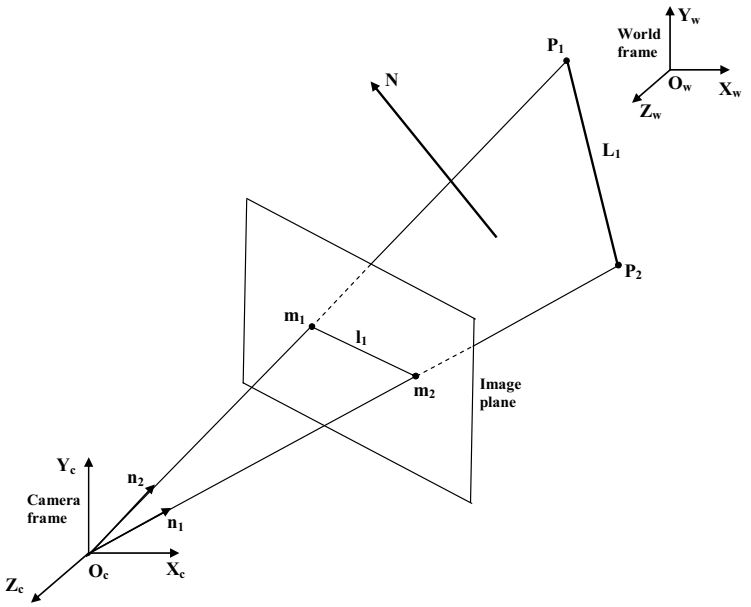


Figure 1.2: Line constraints for the camera pose problem.

identified as shown in Figure 1.2 by the projected end points $\mathbf{m_1}$ and $\mathbf{m_2}$.

The unknown camera parameters that have to be determined for the pose estimation problem are six: three for rotation and 3 for translation. Hence the solution to the pose problem can be found if at least 3 features correspondences (points or lines) are available [Ababsa and Mallem, 2008].

In the context of mobile augmented reality applications, most of the times tracking is achieved with reference to a planar object. Hence tracking systems as the one described in [Lee et al., 2012], can find the pose without an iterative process, increasing computational efficiency in the whole system.

Once the object features are detected and tracked from the camera image, the corresponding 3D points are extracted from the projective mapping, then the camera pose is estimated by means of this 2D-3D projective relationship.

The pose is afterward used to update the Model-View matrix in OpenGL, which is computed as the product between the View matrix M_{view}, which maps coordinates from world space to eye (or camera) space, and the Model matrix M_{model}, employed in the coordinate transformation from object space to world space.

$$M_{modelview} = M_{view} \times M_{model} \tag{1.2}$$

In practice Model matrix's purpose is to move object in the world frame, while View matrix sets camera position and orientation.

OpenGL combines these two matrices into a single matrix of the type expressed in Equation 1.3, in which (m_0, m_1, m_2), (m_4, m_5, m_6) and (m_8, m_9, m_{10})

$$M_{modelview} = \begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix} \tag{1.3}$$

are three element sets dedicated to Euclidean and affine transformations, such as rotation or scaling. The last column of the matrix (m_{12}, m_{13}, m_{14}) is used for translation transformation. The element m_{15} is the homogeneous coordinate, used for projective transformation.

The projection matrix is used to define the viewing frustum, the three-dimensional region which is visible on the screen and also determines how the virtual scene is projected on the screen. The type of projection can be orthographic or perspective. Usually, in AR systems, perspective representation, such as the one in Figure 1.3, is preferred to the orthographic one because it creates more realistic looking scenes.

Orthographic, or parallel, projections do not involve perspective correction,

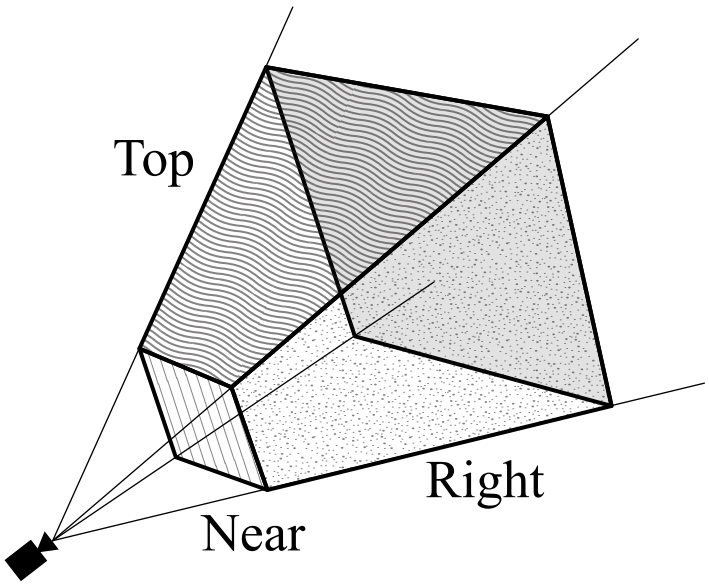


Figure 1.3: Graphical representation of a perspective viewing frustum

meaning objects on the screen will have the same size no matter how close or far away they are. For this reason orthographic projection is preferred in CAD applications rather than AR systems. Everything that lies outside the viewing frustum is clipped out from the rendered scene.

The projection matrix can be expressed in function of 6 parameters associated to the left, right, top, bottom, near and far planes composing the frustum. The projection matrix for a perspective frustum is expressed in Equation 1.4.

$$M_{projection} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{1.4}$$

where l, r, t, b, n and f stand for the left, right, top, bottom, near and far plane's parameters belonging to the viewing frustum.

1.3 OpenGL ES: from Fixed to Programmable Pipeline

OpenGL® ES is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems: including consoles, phones, appliances and vehicles. It consists of well-defined subsets of desktop OpenGL, creating a flexible and powerful low-level interface between software and graphics acceleration.

Before talking about the passage from fixed to programmable pipeline that characterized the natural evolution of OpenGL, we have to explain what the term pipeline refers to. We can define the pipeline as the sequence of events or steps called from the moment the application tells OpenGL to draw a particular object on the screen to the moment in which the object is actually drawn.

Older versions of OpenGL ES used fixed rendering pipeline which was easy to use but with significant limitations.

Introducing a light in a scene using the fixed rendering pipeline was easy just as defining the desired kind of light by its position, strength and perhaps

some other attributes. In fact OpenGL ES by taking this information automatically did all the math figuring out how to shade objects and draw them accordingly. Furthermore fixed pipeline insulated programmer from a lot of things and permitted with simple API calls to rotate, scale and move objects. However programmer was not be able to control when and where the code belonging to all this magic was to be executed.

OpenGL ES 2.0 introduced the programmable pipeline, giving to programmers the ability to take advantage of the great capability of a modern graphics processor unit (GPU), that is specifically optimized and designed for executing floating point operations, a very common practice in the computer graphics context.

Much of the OpenGL ES 2.0 pipeline is outside programmer's control, however there exist the possibility to write some specific programs called *shaders* that are properly designed to run on the GPU. Shaders are small pieces of code written using the GL Shading Language (GLSL). There are two kind of shaders the programmer can write: *vertex shaders* and *fragment shaders*. A fundamental aspect to consider about shaders is that their source code is bundled in the application and compiled at runtime. This is done to preserve device independence.

The shader that runs first on the GPU is called vertex shader and its goal is to compute vertex position in space given a rotation, a scaling or another given transformation. Vertex shader program is executed for each vertex belonging to the object that OpenGL has to draw. Hence if the object has 10 thousand vertices the program is executed 10 thousand times for each draw call.

The other GLSL program we have to write is called fragment shader and as its name suggests is executed for every fragment, that we can picture as possibly drawn pixels. The fragment must consider all the variables that could affect the pixel's final color. Fragment shader is not executed for the pixels that are not candidate to be drawn. Even considering that, fragment shader still is executed a very high number of times.

To keep our 3D application efficient and responsive we have to consider several performance aspects when writing the shaders and keep their code as small and as clean as possible.

Both the shaders haven't a return value but there are two important variables that have to be set. Vertex shader, once calculated the final position for

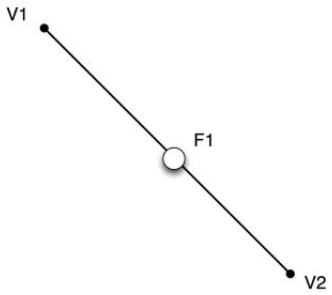


Figure 1.4: Interpolation of F1 with respect to vertices V1 and V2.

the vertex, has to set the system variable *gl_Position* to indicate its location in Cartesian coordinates (x, y, z, w) for drawing. After performing shading calculations involving light and material properties in order to properly determine pixel's color, the fragment shader has to set the final product of its computation in the system variable *gl_FragColor*, using a common (r, g, b, a) quadruple.

GLSL use basic data-types as bool, float and int, and data structures are limited to fixed length vectors (vec2, vec3, vec4) and matrices (mat2, mat3, mat4). Some few special data structures are available for managing textures and there's the possibility to create custom structures like in C.

Shaders have no direct access to the application's main memory and all the data they need has to be explicitly sent to the GPU by the application code. There exist two main types of data we can send from application code: *attributes* and *uniforms*. Attributes are data structures containing a unique value for each vertex, for example the color of the vertex may be a attribute because it is different for every vertex. Every time the vertex shader runs, the pipeline will provide it with the correct value that corresponds to the vertex that the shader is executing for.

There are situations in which attributes assume a uniform value for each vertex: in this case it is better to use *uniforms*. Uniforms are the other kind of data we can pass from application code to the shaders. The main difference with attributes is that uniforms can be used in both vertex and fragment shaders. Another important fact is that uniform will have a single value for every vertex and fragment the shaders are executed for.

In addition to uniforms and attributes there are *varyings* that are special variables used to pass data from the vertex shader to the fragment shader.

Since fragments are usually more in number than vertices and there isn't a biunivocal correspondence between them, the problem to how figure out which is the correct value to pass to a varying variable that has been set in the vertex shader is solved using interpolation techniques.

Take for example Figure 1.4. Say that we want to pass the color to the fragment shader using a varying set in the vertex shader. Let V1 be blue colored and V2 be red, then fragments between V1 and V2 will assume interpolated color values based on these two surrounding vertices. The varying set in the vertex shader will assign correct value based on interpolation: for example F1 will assume a violet color, average of blue and red colors.

Once vertex and fragment shaders have been written they can be compiled and loaded on GPU using simple OpenGL calls or using third-parties libraries such as OpenSceneGraph, the open source library we used.

1.4 The iOS Mobile Platform Architecture

As shown in Figure 1.6, according to Net Market Share, iOS is the most used mobile operating system as concern the tablet and smartphone market[1]. The operating system is developed by Apple Inc. and runs on iPhone, iPod touch, and iPad devices.

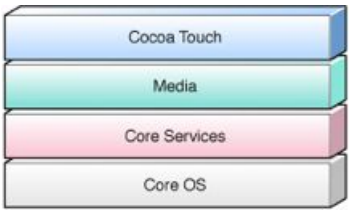


Figure 1.5: Layers of iOS

The structure of iOS is designed as a set of layers, which are shown in Figure 1.5. At the lower layers of the system we find fundamental services and technologies on which all applications rely, while at higher-level layers there are more sophisticated services and technologies.

[1]Data was collected from the browsers of site visitors to a network of over 40,000 websites all over the world. Unique visitors are counted every day and data is aggregated after aproximately 160 million unique visits per month.

Mobile/Tablet Operating System Market Share

July, 2012

Operating System	Total Market Share
iOS	65.74%
Android	20.16%
Java ME	9.11%
BlackBerry	1.86%
Symbian	1.61%
Windows Phone	0.70%
Kindle	0.49%
Samsung	0.10%
Bada	0.09%
Windows Mobile	0.06%
BREW	0.03%
LG	0.02%
HUAWEI	0.01%
ZTE	0.00%

Figure 1.6: Mobile/Tablet Operating System Market Share

The Cocoa Touch layer incorporates the key frameworks for building iOS applications. In these frameworks we find easy access to technologies such as multitasking, touch-based input, and many high-level system services.

Apple strictly recommends a Model-View-Controller (MVC) design pattern that assigns objects to one of three possible roles: model, view or controller. The pattern defines also the way these three kind of objects interact with each other. Adopting this pattern brings numerous benefits, among them the separation of user interface from data management.

Grand Central Dispatch (GCD) is a technology developed by Apple Inc. to optimize and improve concurrent code execution on multi-core processors.

GCD API supports the asynchronous execution of operations at the Unix level of the system. The API can be used to manage interactions with file descriptors, signals, or timers.

GCD provides and manages FIFO queues to which the application can submit tasks in the form of block objects. Blocks submitted to dispatch queues are executed on a pool of threads fully managed by the system. No guarantee is made as to the thread on which a task executes. GCD offers three kinds of queues:

- Main: tasks execute serially on your application's main thread
- Concurrent: tasks are dequeued in FIFO order, but run concurrently and can finish in any order.
- Serial: tasks execute one at a time in FIFO order

GCD automatically creates three concurrent dispatch queues that are global to the application and are differentiated only by their priority level. The application requests these queues using the dispatch_get_global_queue function. Serial queues are used to ensure that tasks are executed in a predictable order.

1.5 Modeling the Illumination

World appears in beautiful colors and shape thanks to one of the most fascinating phenomenon of physics, which is light. Light is what make objects appear as they are. Colors are just an exhibition of its electromagnetic radiation's power. As we all know light is fundamental for our sight in order to properly work and in computer graphics is maybe the most important part to take in consideration for representing virtual objects in the most realistic way. Light is fundamental for this discipline and many models across the history of computer graphics have been elaborated to simulate this phenomenon.

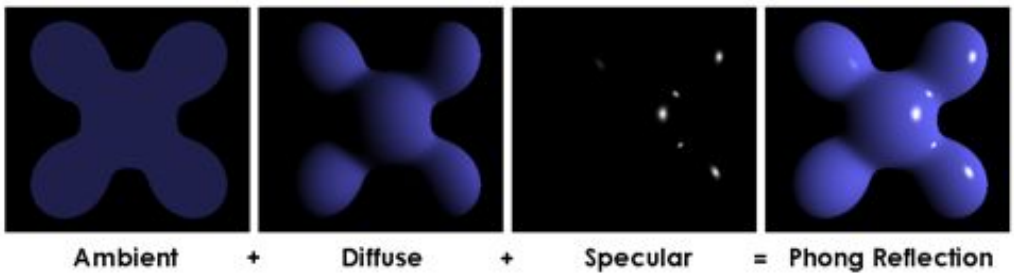


Figure 1.7: Visual illustration of the Phong equation (from Wikipedia).

One of the most notorious models was elaborated by Bui Tuong Phong, a pioneer of computer graphics. The researcher proposed an illumination model which describes the way light is reflected by surface. Phong illumination model describes light as combination of ambient, diffuse and specular reflections. Each type of light component is represented by a color which is encoded using the RGB color model. However, this doesn't mean that these components of

light actually exist in nature, we just think of them as an abstraction of the effects that light can produce when casting on different materials.

We have to introduce some vectors and terminology in order to describe Phong's illumination model.

- **Normal Vector:** a vector **n** that is perpendicular to the surface and directed outwards from the surface
- **View Vector:** a vector **v** that points in the direction of the viewer
- **Light Vector:** a vector **l** that points towards the light source
- **Reflection Vector:** a vector **r** that indicates the direction of pure reflection of the light vector

All vectors are assumed as normalized to unit length and can be graphically represented as in Figure 1.8. How objects react to light emission is characterized by material's properties, that determine how much of a given input light intensity is reflected. Phong model captures these properties using some reflectivity coefficient vectors for ambient, diffuse and specular lights.

The final color belonging to a fragment of the 3d object is computed using the following standard equation:

$$I_f = I_a + I_d + I_s \tag{1.5}$$

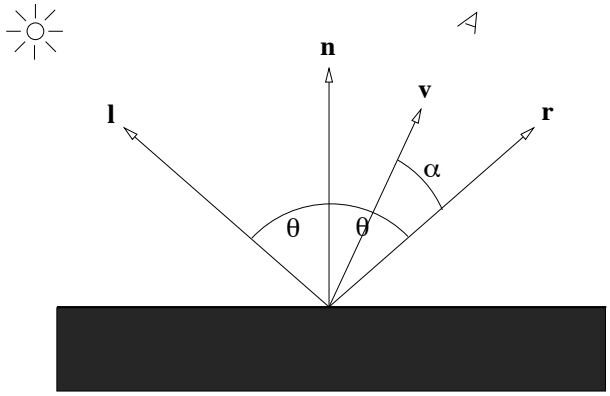


Figure 1.8: Vectors involved in Phong's illumination model.

in which I_a, I_d and I_s are respectively the ambient, diffuse and specular components of illumination.

The ambient term, shown in equation 1.6 is not influenced by any kind of variable as the light direction or surface normals, and that's why it is considered the constant component of the illumination model.

$$I_a = (L_a * M_a) + (L_{ma} * M_a) \tag{1.6}$$

Ambient component is in fact used to simulate the radiant effect, which happens when light is reflected and scattered into all different directions averagely brightening up the whole scene. Ambient light alone makes objects appear flat and bi-dimensional, that's why other components like diffuse and specular have to be introduced.

When diffuse light touches the surface of an object, it scatters and reflects evenly across that surface. When computing diffuse light component, we need to consider attenuation and light position, those variables that were ignored with ambient lighting. The light position will be used by the shader to calculate it's direction $\mathbf{l}$ with respect to the object's pixel fragment, and the angle between light vector $\mathbf{l}$ and normal vector $\mathbf{n}$ shown in figure 1.8 will affect pixel shading. Light position will also affect the strength of the light at that fragment point.

$$\gamma = \max(0, \mathbf{n} \cdot \mathbf{l}) \tag{1.7}$$

All these parameters are considered in the computation of the Lambert Factor γ, expressed in equation 1.7, which is the ultimate cause of self shadow effect in three-dimensional objects and is given by the dot product between the normal vector $\mathbf{n}$ and the light vector $\mathbf{l}$. In fact, since these two vectors are normalized, the scalar product between them is equal to the cosine of the angle between them. The scalar product results to be negative in the case that this angle is greater than 90 degrees, in which case the value of Lambert Factor is set to zero to express the fact that the point cannot be reached by light, as shown in figure 1.9. Lambert Factor is used to compute the diffuse component of illumination I_d expressed in equation 1.8 as the product between the diffuse component of light, the diffuse component of material and the Lambert Factor.

$$I_d = L_d * M_d * \gamma \tag{1.8}$$

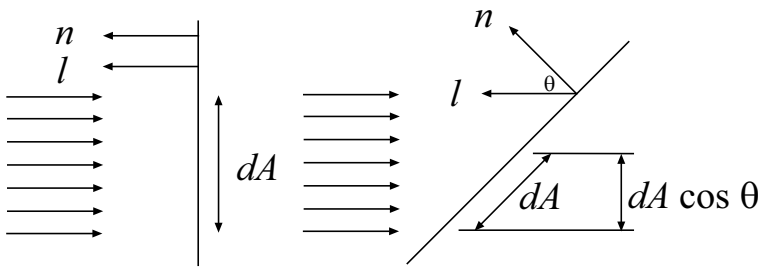


Figure 1.9: Lambert factor in diffuse reflection.

Finally there's the specular component of illumination. The rendering of specular light is influenced by the angle between the viewer and the light source. Its intensity of reflection depends on the material the object is made of and on the strength of the light source which contains the specular light component.

To further explain this term we have to introduce some new definition as:

- The shininess of an object f, which is a factor indicating how much the material will shine in reaction to a source of direct light.
- The eye vector $\mathbf{e}$, which is equal to the vector indicating position of the vertex, with inverted direction and normalized length.

Let be as usual L_s and M_s the specular components relative to light and material. Then the reflection vector $\mathbf{r}$ is computed using the built-in function *reflect* of GLSL, the OpenGL shading language. A possible implementation is expressed in 1.9.

$$\mathbf{r} = 2 * (\mathbf{n} \cdot \mathbf{l}) * \mathbf{n} - \mathbf{l} \tag{1.9}$$

$$\alpha = (\max(\mathbf{r} \cdot \mathbf{e}, 0.0))^f \tag{1.10}$$

This expression is then used for the computation of specular factor α expressed in 1.10, which is finally used to compute the total specular factor I_s.

$$I_s = L_s * M_s * \alpha \tag{1.11}$$

In summary we compute the ambient, diffuse and specular components of light and then we sum up them as in 1.5 to obtain the real final color of the fragment. Usually this sum is scaled down by a factor called attenuation

of light that is computed in function of the distance between fragment's and light's position.

1.6 Rendering In Mobile Platform

Mobile platform market is constantly increasing and by now the computational capabilities of some of these devices are strong enough to permit three-dimensional applications such as games to run smoothly.

1.6.1 Limited Resources

However besides their constant improvement, mobile platforms such as the smartphones are in general limited in resource when compared to desktop or notebook solutions.

The proportion of main memory an application can use is probably one of the most important limitations, especially in the graphical context. In particular mobile operating systems such as iOS restrict the memory consumption that applications are allowed to make before receiving a memory warning and being killed.

Generally virtual memory model does not include disk swap space, limiting applications in the amount of memory they have available for use. The overall system performance may be seriously degraded and potentially cause the system to terminate the application if large amounts of memory are used. Therefore, reducing the amount of memory used by a mobile application should be a high priority.

1.6.2 Multi-threading and GPU programming

Rendering is also limited by the Central Processing Unit (CPU) and the Graphics Processing Unit (GPU) performance. Limits on the number of vertices and triangles that the application can render in real time are evident and strongly depends on the processing units. During the development of our application we found consistent differences between the performance exhibited by iPad 2, equipped with the dual-core Apple A5 processor, with that one shown by the single core Apple A4 processor mounted in iPhone 4 and iPod Touch 4G.

For this reason the use of a multi-threaded approach and GPU is crucial to optimize the performance of the application.

The introduction of OpenGL ES 2.0 for Android and iOS, the two major competitors in the mobile market segment, has enabled programmers to fully control and program the mobile GPUs, lightening the load on the CPU which can be rather dedicated to tasks that do not involve graphical rendering.

CHAPTER 2

STATE OF THE ART

Augmented Reality (AR) is a technology that allows a user to see an extension of the real world, with virtual objects superimposed upon. It is a supplement to reality and does not entirely replace it. Ideally a user should be able to see a common space where virtual and real objects coexist [Azuma, 1997]. However a lot of problematics may arise in order to achieve this level of result, nonetheless many attempts were made in this direction. AR is a specific example of what Fred Brooks calls Intelligence Amplification (IA): using the computer as a tool to make a task easier for a human to perform [Brooks, 1996].

We have to make a distinction between different augmented reality technologies. In general the AR technology should have the following 3 characteristics:

- Combine real and virtual
- Interactive in real time
- Registered in 3D

In this section we analyze some of the most interesting works at the state of the art in this technology, focusing our attention on its application in the mobile platform and presenting some existent solution on the context of cultural heritage.

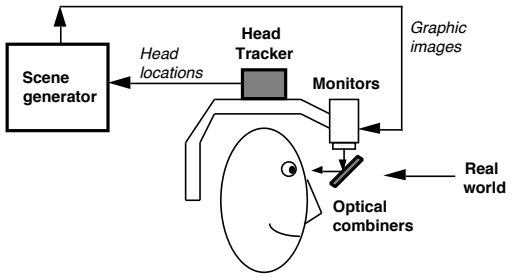


Figure 2.1: Optical see-through HMDs conceptual diagram

2.1 Archeoguide Project: not really mobile

There are various ways to create an augmented reality system. One of the most popular approaches requires the use of Head-Mounted Displays (HMDs) (see the figures 2.1 and 2.2). An example of its use is given in [Gleue and Dähne, 2001], where the design and implementation issues associated with the development of a mobile device for the Archeoguide project are discussed.

Archeoguide project's goal is to provide information through augmented reality system to visitors of cultural-heritage sites, in someway something similar to what has been developed in this work of thesis. One of the problem that may arise using this approach is the need of carrying an extra device beyond the HMD. Infact visitors are equipped with a mobile computer and a HMD display unit. In particular position and orientation tracking is accomplished using a camera and an electronic compass both mounted with the display unit, measuring the user's current field of view. The mobile device constantly tracks visitors' position through the aid of a (D)GPS receiver.

However as found in [Dähne and Karigiannis, 2002] this approach presents

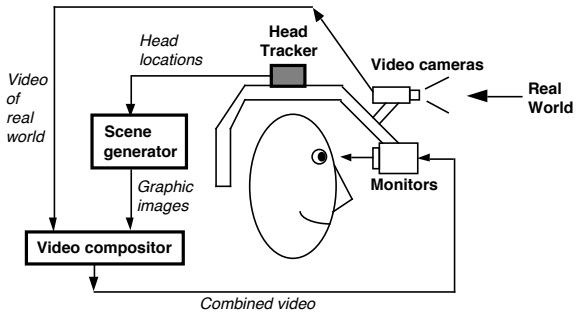


Figure 2.2: Video see-through HMDs conceptual diagram

several problems such as weight and dimension of the complete system that are too big for a comfortable use. Furthermore excessive hardware costs are a serious problem for a possible commercial setting. Issues like the poor visibility in direct light condition of the LCD display and problems like inadequacy of webcam to outdoor conditions are other complications that made us lean towards simpler hardware solutions.

2.2 Using Mobile Devices for AR

Today we are almost surrounded of mobile devices which integrates almost any of the described sensors, that in case such as camera sensor are specifically designed for working in outdoor conditions too. One of the best and in some way standardized mobile platform currently available is for sure the iOS device family. The high quality of the IPS retina display, which is very bright and visible also in high light conditions, the amount of sensors built-in in such devices led us to the choice of this mobile platform. Devices like iPhone and iPad have a complete set of high quality sensors such as proximity, a 3-axis gyroscope, ambient light, accelerometers, magnetometer and assisted GPS and are hence well suited to the development of augmented reality systems in a scenario similar to Archeoguide.

In a more recent work [Honkamaa et al., 2007], an augmented reality system able to project 3d models in a real scene was developed by a team of researchers, differing from Archeoguide where the system was only able to superimpose 2d augmentations. The team of researchers considered two main cases. In the first case the application leaves to users the duty to interactively place the virtual object modifying orientation and scale to the environment. In the second case the task is simplified using GPS information of the viewer's location and prior knowledge of the object's global placement. After the positioning is performed the object is kept in place using camera motion estimation. One limitation of the camera estimation approach used in this work is the assumption that viewer is standing in place while viewing the augmented scenary, so that the application needs to track only three degrees of freedom (camera rotation around each axis). Again here, because of the needed computational power, a mobile computer, which is compact but not versatile as a hand-held device like iPhone or iPad, has been used for the project.

Another work [Pasman and Woodward, 2003] treats about implementing an augmented reality system on a PDA device. However because of computational power limitation the PDA is mainly used just as a display device demanding rendering and tracking on a server computer machine. Recent advances on mobile platform hardware specifications have permitted to obtain good results also using a standalone application as our implementation does.

2.2.1 Hand's Palm Pose Estimation

One example of alternative augmented interaction on mobile devices is the one presented in [Seo et al., 2008], where researchers describe the realization of a mobile application for augmenting virtual objects on the user's palm. The pose of the palm is estimated using natural features contained in a general hand so there's no need to have markers. As stated by the researchers the user can interact with the virtual object thanks to a tracking algorithm that detects the motions of the hand using the fingertips. Moreover user's perception of the virtual object is enhanced using a tactile glove interface. However when running on mobile devices the application resulted to be slow in terms of frame rate because of the limited computational power of the mobile device.

The main difference with a system such as the one used in our application is that in this AR system the hand is described by a more generic model of natural features so that almost every hand can be recognized and tracked, while in our application specific and detailed description of the objects to be tracked are provided.

2.3 Mobile Applications in the Context of Cultural Heritage

In this section we talk about three applications available on the Apple *App Store* that are in a certain manner similar to our product.

2.3.1 Rome MVR

Rome MVR is an iPhone application that permits to visit Rome across the ages and see what it looked like in the various periods of time. Rome MVR

is not really a true augmented reality application because it does not extend reality, but replace it instead.

In all likelihood the application works using GPS, compass and gyroscope information, by replacing the images coming from the iPhone's camera with a virtual reconstruction of the ancient Rome. The application allows to view the condition of archaeological sites, urban areas or monuments in different historical periods of Rome, fading between the 3D reconstruction and the image coming from the Camera.

As already underlined this application does not really use augmented reality, but it just implements a sensor-based intelligent navigation system for a virtual world. The illusion of image registration is achieved using fade effect between the real and the virtual.

2.3.2 RomeView

RomeView is another application focused on providing information of cultural heritage for the city of Rome and for its museum. The application permits to receive details about the city and its history. Using localization services as the GPS, the user can explore an informative map receiving informations and multimedia content about the nearest monuments in the city.

The application is also integrated with the museum of Rome for "augmenting" the experience of the user during the visit. However this is not a real augmented reality experience since the user has to precisely select the picture or the artwork is looking at from a map of the museum. The only guise of augmented reality in this application works using only GPS (maybe the digital compass) data with all the related problems of registration. In fact the virtual objects are not correctly aligned with the real images coming from the camera, resulting in inadequate integration of what is real and what is virtual.

2.3.3 Voyajer

Voyajer is an application that allows the visualization of 3D contents in real time of the most important monuments of the archaeological sites of Rome. Even this application does not use an authentic augmented reality system, but as like as Rome MVR by using GPS, compass and accelerometers to track user

position and orientation, aligns the virtual reconstruction of the Ancient Rome to the user's point of view.

2.4 Mobile Applications using AR

As stated in [Azuma, 1997] we can say that an application uses augmented reality if it combines real and virtual objects providing 3D registration and real time interaction. This is a point that makes the difference when comparing our application to those described in the previous section. We are proposing a different kind of interface for the fruition of digital content in the context of cultural heritage. Unlike traditional AR devices, smartphones combine all necessary technologies for augmentation in one small device [Zornitza et al., 2012], therefore we believe that this platform will drive the growth of this technology.

Currently the majority of mobile applications that are using AR technology are employed in the sectors of advertisement and games. As regard the sector of cultural heritage and tourism there is still a lack of solutions. Moreover many tourists still prefer more traditional sources of information, such as, for example, paper-based guidebooks.

An AR application that is collecting positive ratings in the AppStore is the new digital catalog by IKEA, a company notorious for selling ready-to-assemble furnitures. The application allows smartphone interaction with the 2013 paper catalog of IKEA. Pointing the camera in specific pages of the catalog the user can receive extended informations on the products visible on the page by means of augmented reality. In particular three type of informations are provided:

- Photography content: the application shows an album of photographies showing the product from different perspectives.
- Video content: the application shows a videoclip advertising the product.
- AR 3D Model: the application shows an animation of a 3D model of the product using augmented reality. The model is registered to the page of the catalog and the user can move the smartphone to see the augmented view of the product from different perspectives.

CHAPTER 3

IMPLEMENTATION

In this chapter we talk about the technologies used to realize the mobile AR framework. We describe encountered issues and choices taken during design and development.

3.1 Interacting With Virtual World

Augmented reality gives us the possibility to extend with virtual objects what we usually percept. These virtual objects can be seen as vehicle for information. Hence an augmented reality system needs to provide some kind of interface with both real and virtual objects. The classical WIMP (Windows, Icons, Menu and Pointing) style of human-computer interaction is not well suited for this kind of application. Experimental interfaces for interaction in augmented reality systems include mobile trackballs, gyroscopic mice, haptic UIs, gesture recognition and many other interfaces that tends to be more natural for the user.

3.1.1 Interacting with the Touch Screen

The introduction of touch technology in the mass market was almost contemporary with that one regarding a number of other natural interaction interfaces such as gesture recognition, speech recognition or more complex motion sens-

ing input devices like the Kinect introduced by Microsoft in 2010. All these new human-machine interaction technologies are a breeding ground for the development of new augmented reality systems.

An attempt has been recently made by Google, an American corporation that provides Internet-related products and services such as Internet search, sector in which today Google is the undisputed leader of. The multinational corporation presented a research and development program, namely *Project Glass*, to develop an augmented reality HMD with the purpose of building a device able to provide Internet-based informations hands free and fully integrated in the real environment.

Our framework is designed to run on mobile devices such as iPhone, iPad or iPod Touch, that are mainly controlled through the touch interface. The framework developed for the augmented reality links the Vuforia subsystem with the OpenSceneGraph rendering engine. In order to interact with the virtual objects, select them and receive the appropriate linked information, a line segment intersector has been used to detect precisely the identity of the virtual object the user is touching on the screen.

In practice we scan the whole scene-graph tree associated to the virtual scene testing for intersection with the projected line generated by the point touched by the user in the screen, then we take the identity of the object that results to be in the foreground and we return it to the commands controller to take in consideration an appropriate action.

Making augmented reality with smartphones means to deal with handheld display devices with all the related problems. One of the issues that may arise is that target or fiducial marker is often distant from the viewer making impossible to interact with the virtual scene using occlusion detection techniques. Vuforia system enables the programmer to use such procedure to operate virtual buttons, which can be thought as virtual objects attached to the real scene that can be operated by the user through their partial occlusion. A typical scenario of application can be described by a user keeping the phone in the left hand while operating the virtual button with the right hand. After testing this approach in our application we decided to avoid the use of such type of user interaction because of the problematics associated with the use of distant targets.

Such approach would appear natural when using a HMD display but the

same cannot be said when hand-held devices are used. As a matter of fact when touch screen is available user will touch the object in the touch screen more likely than positioning a hand between the camera and the virtual environment.

3.1.2 Information Board Interaction

To create a better user experience we designed a specific information board for interacting with the application. The information board contains a map of the area in which the forum of Nerva's ruins are located, and other sensible image targets. When the user points her mobile device on the board different kind of information appears based on the target. Specifically a three dimensional model of the forum of Nerva will appear positioned within the map of the fora, showing an augmented view of it. The user can then touch the virtual forum through the touch screen and retrieve information about the forum of Nerva, such as it's history, photographies, and other kind of information.

3.1.3 Interacting with the virtual forum of Nerva

The area around the forum is divided in several sensible areas so that different actions are proposed depending on where the user is touching. One possible action is to see a detailed view of the touched part. In this case a new view is presented to the user showing a high quality rendering of the interested part.

The user can interact with the detailed view zooming in and out the model through a pinch. Rotating the object (roll-pitch-yaw) is performed with touch using an arcball [Shoemake, 1992] and the scene can be reset to default with a double tap. If the detailed view contains multiple views, user can swap between them using a swap in the lower part of the screen.

3.1.4 Virtual Video Streaming

Another fiducial marker is used to invite the user to see a video. When the user points her phone on the fiducial marker a virtual TV appears showing a video animation about the forum of Nerva. The user can interact with the virtual TV using the touch screen. She can put in pause and resume the video streaming. Furthermore she can go full-screen if she's interested in seeing the

video with more details. The virtual TV is designed taking in mind the concept of a virtual animated icon.

3.2 Choice of Multi-Platform Libraries

Before starting the real development of the mobile augmented reality system, aspects such as the choice of libraries able to manage the augmented reality and the rendering process are a critical aspect to consider especially in consideration of the future portability of the framework in the different realities of the market. For this reason we opted for privileging multi-platform libraries that can be used in the two major mobile operating system platforms: Android and iOS.

Several alternatives have been taken in consideration, finally converging to what we thought were the best choices available in the market today. As regard the augmented reality subsystem the Vuforia library developed by Qualcomm was the primary choice since open source systems like the well known ARToolKit are currently not freely available for mobile platform. Vuforia has demonstrated to be a mature product and it is released for free, even for commercial purposes. Furthermore it is enough easy to configure and can be natively integrated on both Xcode and Eclipse development environments. The only negative aspect of Vuforia is that Qualcomm keeps lower levels functions such as the tracking and pose extraction subsystem closed source. However the robustness of this library and the use of standard and portable C++ code is enough to justify its selection.

One of the aspects that negatively influenced the choice of multi-platform libraries was the necessity for some of them, such as the rendering engines Marmalade and Unity, to use a cross-platform deployment process. At the beginning, this may appear as a good strategy to speed-up the development process because scripting languages like javascript can be used, but when programmers want to be in control of every aspects and optimize system specific aspects, libraries working at lower level such as NivehGL and OpenSceneGraph are a better solution.

NinevehGL is a 3D rendering engine built right on top of OpenGL ES using Objective-C, the native language of iOS. NinevehGL is a very promising rendering engine, however it is still a young project and it is not multi-platform.

The OpenSceneGraph is an open source high performance 3D graphics

toolkit that can be used for developing applications in fields like visual simulation, games, virtual reality, scientific visualization and modelling. The OpenSceneGraph is well established as the world leading scene graph technology, used widely in visual-simulation, space, scientific, oil-gas, games and virtual reality industries. Moreover OpenSceneGraph is multi-platform and more mature than NinevehGL.

As you will certainly understood OpenSceneGraph was the choice for managing the rendering subsystem of the augmented reality framework, motivated also by the fact that library interaction is managed through portable standard C++ code.

3.2.1 The Open Scene Graph

OpenSceneGraph uses a data structure, namely a scene graph, to represent the spatial and logical relationship of the graphical scene. This data structure can be assimilated to a hierarchical graph in which we can have a collection of graphics nodes starting from a root node located at the top level.

The scene graph contains a number of group nodes each of which can have any number of child nodes, and a set of leaf nodes. A typical scene graph does not contain direct cycles and isolated nodes. Group nodes can contain any number of children and are used to share common operations and data.

A node can have multiple parents, in which case the node is said to be *instanced*. This can be used for multi-pass rendering. [Wang and Qian, 2010].

For iOS we choosed to configure OpenSceneGraph for using OpenGL ES 2.0. The support is experimental and some extensions are not supported. In particular, since OpenGL ES 2.0 relies on the use of the programmable pipeline all the graphical effects, the lights, the materials and the drawing it self have to be manually managed using shaders.

OpenSceneGraph includes a set of APIs to manage the shaders, which can be attached to any node in the scene graph achieving different effects for the complete scene. Uniforms and attributes can also be set using OpenSceneGraph libraries.

Although all the rendering setup was realized using OpenSceneGraph, the rendering update is called through Vuforia, the augmented reality subsystem. The choice to give Vuforia the control of the update is motivated by the in-

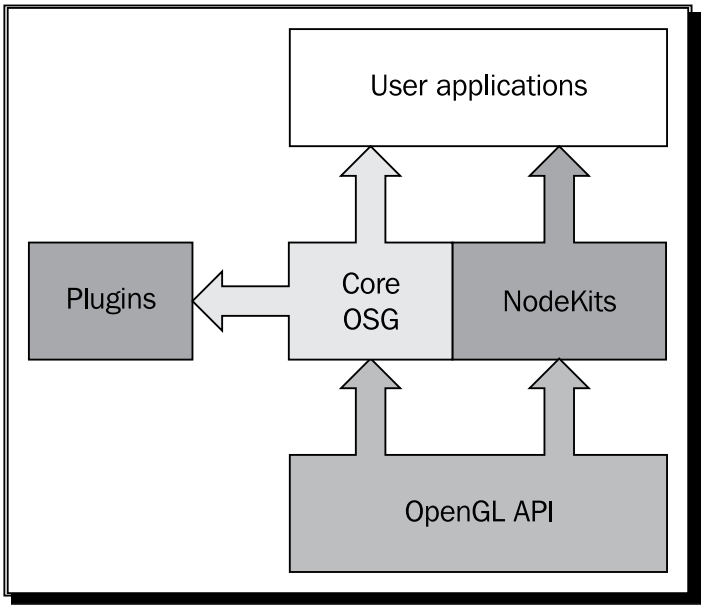


Figure 3.1: The OpenSceneGraph architecture

ability to use the rendering call of OpenSceneGraph.

The other possibility was to use a CADisplayLink, a timer object that allows the application to synchronize its drawing to the refresh rate of the display. We tested this approach and the result was very bad, hence we delegated the render update function to renderFrameQCAR, a function that is called by QCAR (Vuforia) when it wishes to render the current frame on the screen.

CADisplayLink is used in the detail view where rendering is performed without augmentation. In this case in fact the synchronization between the augmented reality layer and the rendering layer is not needed since no augmented reality is involved.

3.2.2 Vuforia™ by Qualcomm

The augmented reality framework is built upon the Vuforia library by Qualcomm. Vuforia can manage 2 different type of targets:

- AR Tags, marker based.
- Image Targets, marker less.

In our application we choose the second type which is more flexible and appealing. The library allows creation of image targets with a simple web interface

which automatically classify the target accuracy with a rating from 1 to 5 stars. The accuracy is based on how many relevant features can be extracted by the Target Management System hosted on the developer portal QDevNet. In practice the developer uploads an image for the trackables she wants to track and then downloads the relative target resources bundled in a dataset, which can be loaded in the application through a simple API call. The downloaded

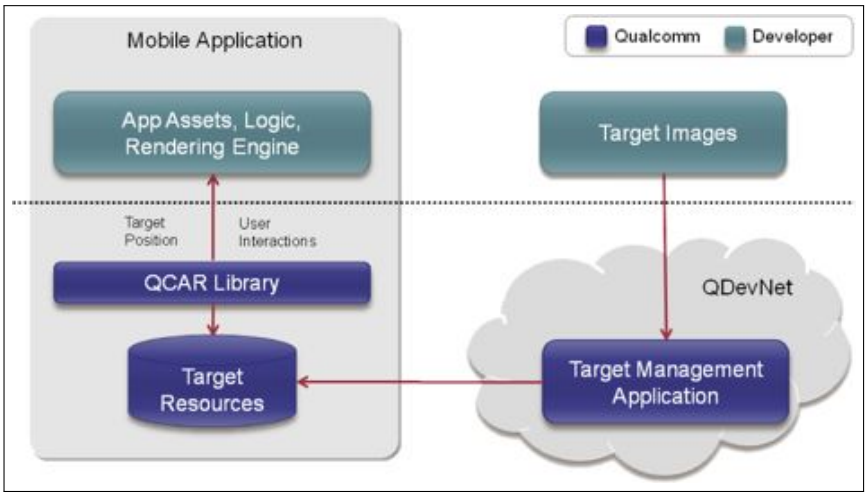


Figure 3.2: QDevNet process overview

dataset contain an XML configuration file that allows the developer to configure certain trackables features and a binary file that contains the trackables database. These assets are bundled in the application and used at run-time.

Our AR application which is based on the Vuforia SDK, must manage the following core components:

Camera The camera component manage capture of each frame and transmission of data to the tracker. The application only has to tell the camera singleton when capture should start and stop. The camera frame is automatically delivered in a device dependent image format and size suitable for OpenGL ES rendering and for tracking. This conversion also includes down-sampling to have the camera image in different resolutions available in the converted frame stack.

Tracker The tracker component contains algorithms for detecting and tracking real world objects in camera video frames. Unfortunately these algorithms are proprietary and not accessible to the developer. However

results are stored in a state object that is used by the video background renderer and can be accessed from application code. A single dataset can contain multiple targets. Although the tracker can load multiple datasets, only one can be active at a time. Nevertheless we designed a system to scan multiple datasets in order to simulate an automatic dataset switch detection.

Video Background Renderer The video background renderer component renders the camera image stored in the state object. The performance of the background video rendering is optimized for specific devices.

Application Code Our application initializes all the above components and for each processed frame, the state object is updated and the render method is called. The application queries the state object for newly detected targets and updates its logic with the new input data. Then it renders the augmented graphics overlay.

3.3 Structure of the system

During the development of the application we followed the Model-View-Controller (MVC) approach, suggested by Apple documentation. MVC design pattern introduce guidelines on which role among the three categories of *model*, *view* and *controller* should be assigned to objects, defining also how the communication between these objects must be performed.

Benefits are that application tends to be more reusable and the interface is well defined. The management of the application in delegated to a Main view controller. When the application is launched the Main view controller will initialize all the sub components, namely the Commands view controller, the AR view controller and the OSG view controller.

The Commands view controller is the class demanded to manage all the controls available in the application, from the buttons to the virtual object selection. This controller captures every command issued by the user and often delegates the execution of an appropriate action. For example, in the case of virtual object selection the computation of which object has been selected is delegated to the OSG view controller.

The AR view controller manages the lower part of the augmented reality system. It's main function is to control the AR_EAGL view, a special view used to display and control the lower part of the rendering system as the call to the rendering update function, which is delegate to the OSG view controller.

The AR view controller is also in charge of initializing, pausing and resuming the augmented reality layer, whose flow is governed by QCARUtils, a class devoted to perform low level interaction with the Vuforia AR layer.

The OSG view controller manages all the rendering calls using the OpenSceneGraph libraries and is responsible for setting up the shaders, loading the 3D object models in memory and managing 3D animations.

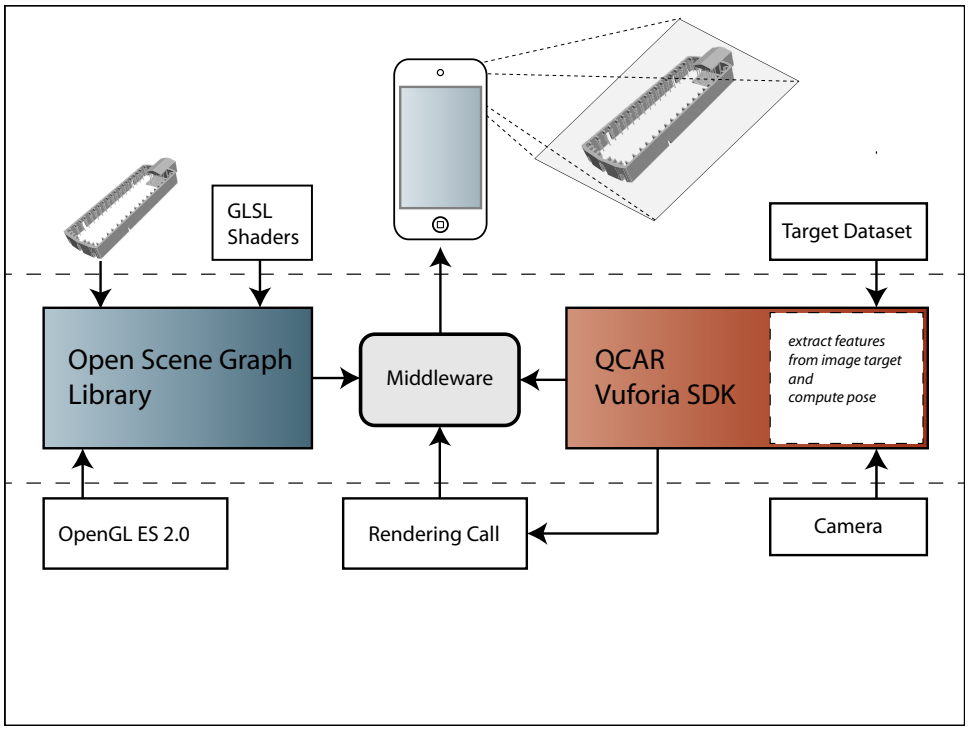


Figure 3.3: Framework Architecture

3.4 Image Target Management System

Tracking system relies on a marker less approach that uses feature tracking and vision algorithms to estimate the pose of some predefined image targets.

Image targets description is encapsulated in datasets that can be created using the Vuforia Target Management System, available online in the Qual-

comm developer portal. Target resource contains the Dataset configuration XML file that allows to configure certain trackable features and a binary file holding the trackable database. In order to be trackable images must follow some simple rules:

- They should be rich in detail
- They must have optimal contrast. Image should not appear flat and homogeneous.
- Repetitive patterns like a grassy field, the façade of modern house with identical windows, a checkerboard and other regular grids and patterns should be avoided.

Once built the datasets is loaded from the QCARUtils class during the application initialization. Only one dataset can be active at a certain moment, however a dataset can contain multiple targets.

Once dataset and all the 3D AR models are loaded in memory the application is authorized to test for the presence of the image targets in the camera field of view. The test is performed at the beginning of each rendering cycle.

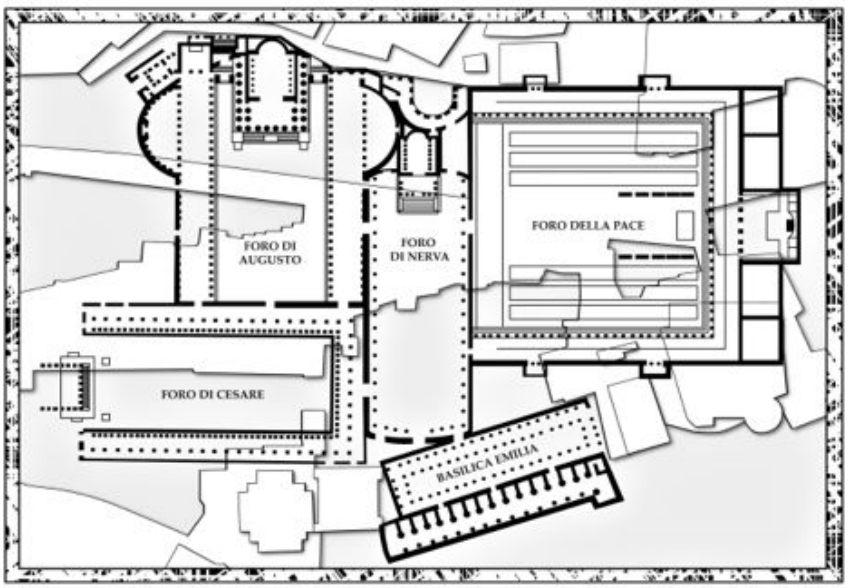


Figure 3.4: Target image

When the targets are detected opportune matrix transformations such as translation, rotation and scaling are performed in order to correctly set the initial status of the 3D model. After these operations are completed the 3D models are added to the scene graph, based on application status.

Before submitting everything to the pipeline for each target trackable the pose is estimated and the current projection and model view matrices are computed from that. Camera projection and model view matrices are hence updated with the new values.

For our application we created a custom target image (see Figure 3.4) representing a plan of the imperial fora around the forum of Nerva. The image is designed to be understandable without the AR layer. The augmented reality will serve as vehicle for providing information about the three-dimensional reconstruction of the area. The image target will be printed in a information board that will be placed near the ruins of the Forum of Nerva.

3.5 Functionalities of the Application

The application permits to visualize an augmented view of the ancient forum of Nerva. The 3D reconstruction has been realized in collaboration with architects and historians of the Architecture Faculty of Sapienza, the well-known University of Rome. The reconstruction is based on the work [Viscogliosi, 2010] written by a maximum expert in the field, the historian Alessandro Viscogliosi.

The augmented view is superimposed to a sensible map located near the ruins of the ancient forum of Nerva (see Figure 3.6), so that the visitors can admire how the forum looked in ancient times.

The three-dimensional model of the forum of Nerva is interactive and the user can select parts of it using the touch screen and retrieve associated information.

It is possible to enable or disable a satellite 3D map onto the basis of the forum of Nerva to better understand its location. By tapping on the virtual map the user is redirected to a interactive satellite visualization of the area around the forum of Nerva underlining its archaeological area and indicating the information board's position and the location of "Le Colonnacce", the ruins of the forum.

We also implemented a function for playing videos (see Figure 3.7) in the

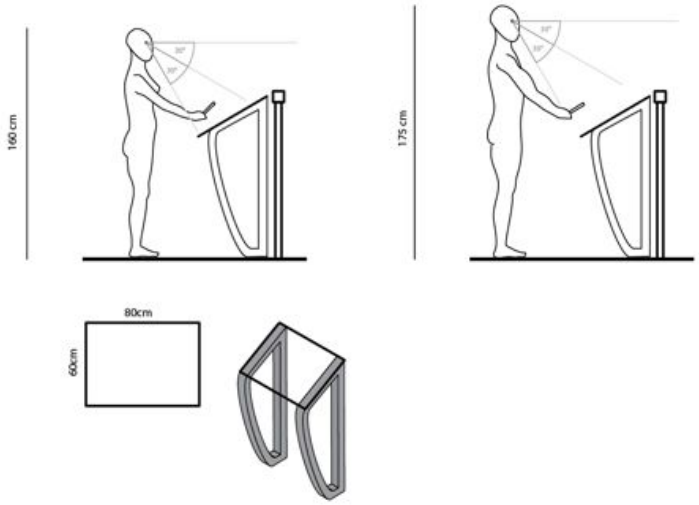


Figure 3.5: Schema illustrating the use and design of the information board.

augmented reality scene. A virtual TV has been created and superimposed on a specific image target. The user can interact through the touch screen with the virtual TV. Specifically one tap on the virtual TV will play the video or pause it. A double tap will bring the video in full screen mode.

Video streaming on augmented reality was achieved by using the ffmpeg library in conjunction with OpenSceneGraph. The ffmpeg is a free software project that includes libraries and programs for handling multimedia data.

The library has been cross-compiled for the ARM architecture in order to work with the OpenSceneGraph ffmpeg plugin. We preferred the use of ffmpeg because it is open source and it provides support for many types of video extensions (avi, flv, mov, ogg, mpg, wmv, mp4, 3gp and many other formats) and additionally allows video/audio streaming from http and rtsp.

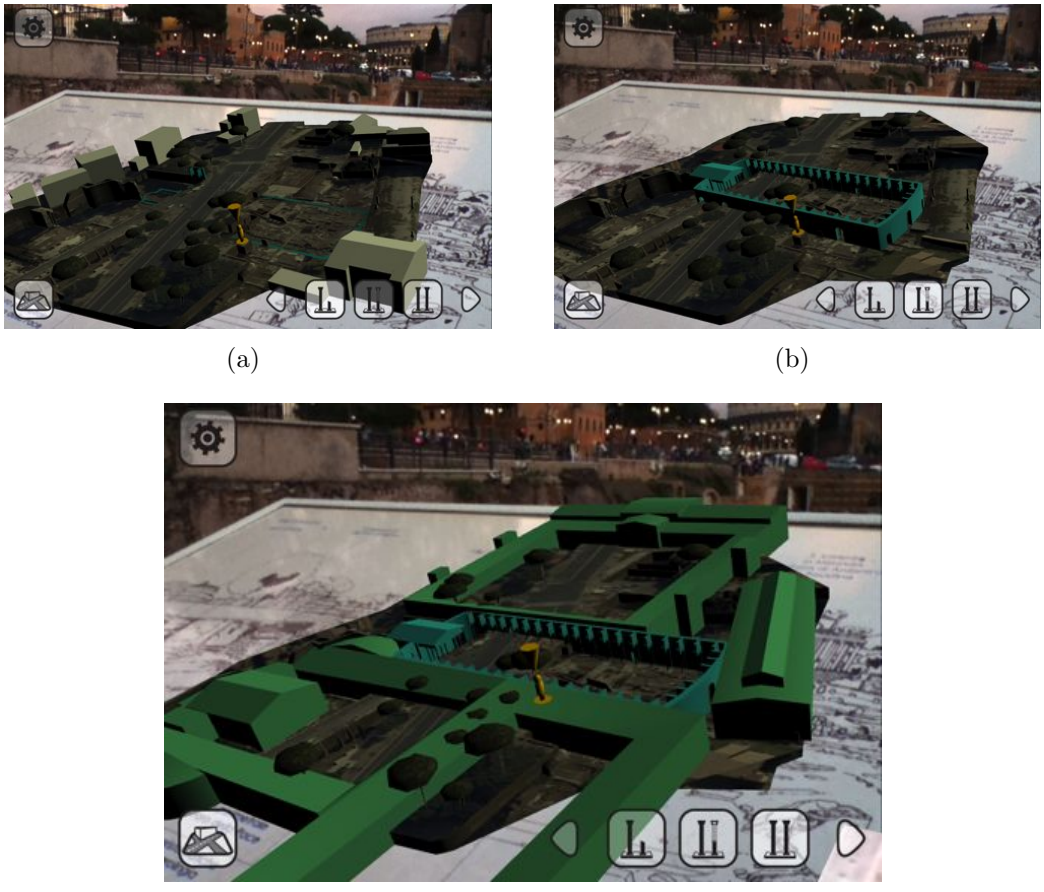
In practice the video is loaded using the OSG ffmpeg plugin and at every rendering cycle the texture of the object corresponding to the virtual screen is updated with the current frame of the video.

3.6 Modeling the Virtual Objects

Modeling of the virtual objects such as the forum of Nerva has been realized in collaboration with students of Architecture Faculty Valle Giulia, of Sapienza, University of Rome.

One of the biggest problems encountered during model development was to find a compromise between the aesthetic and the performance aspects. A high work of optimization was done in the first part for reducing the very high number of vertices and faces constituting the original model of the Forum of Nerva created in origin by the architects. This model, having more than 5 million vertices, was in fact realized with another purpose in mind, such as the one of high quality rendering that you can achieve with desktop computational power.

On the other hand mobile platform are very resource limited and part of the initial work was dedicated to find an approximate upperbound for the number of vertices and faces. Furthermore we found out huge gaps for these numbers between devices such as iPhone 4, using the Apple A4 single core cpu, and



(a) (b)

(c)

Figure 3.6: The states of the app: (a) the view of the ruins, (b) the reconstruction of the Forum of Nerva, (c) the forum of Nerva with the other fora around.

Figure 3.7: Views of the AR Video on the map located near the ruins. On the left a virtual animated icon. On top a virtual TV screen.

iPhone 4s, using the Apple A5 dual core cpu.

The original model was initially optimized using 3D Studio Max optimize function, resulting in two models : one thought for the A4 chipset with lower quality having around 10 thousand vertices and the other one created for the A5 chipset with higher quality having around 45 thousand vertices.

The result was not very good in relation to the number of vertices, hence after discussing with architects we decided to realize a new model specifically optimized for the mobile platform. Another motivation that leads us to this choice was based on historical arguments. In fact after consulting with an expert in the history of architecture, we realized that the original model had a wrong basis structure based on wrong reconstruction. There exist in truth different reconstruction of the forum of Nerva, differing from the number of columns, position of the entrances and of the walls.

We decided then to follow the reconstruction model described in [Viscogliosi, 2010]. In this work the historian presents a architectural plan of the forum of Nerva, shown also in Figure 3.8.

Software used for modeling include Autodesk 3D Studio Max, Cinema 4D and MeshLab.

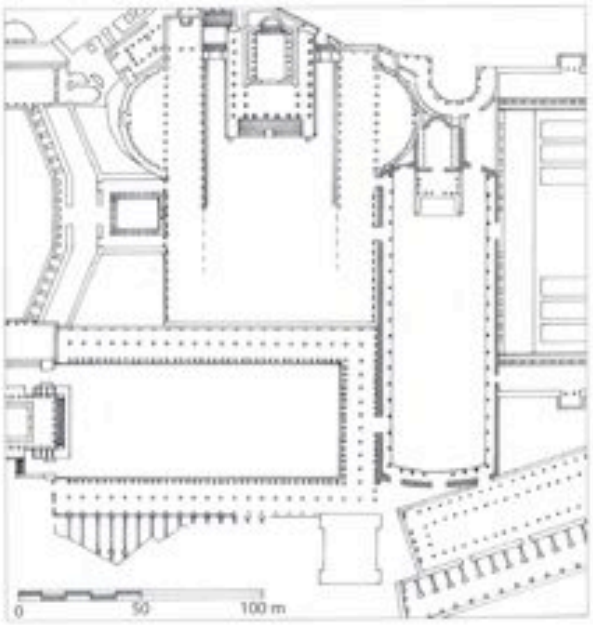


Figure 3.8: Reconstruction of the forum of Nerva illustrated in [Viscogliosi, 2010]

3.7 Loading the 3D models

The question on how to encode 3D models is an open context in computer graphics. In general there's no such a thing as a better format, however some of these formats are open in a sense that their content is not encrypted and fully accessible even with a simple file editor.

The fact of being open speeds up important aspects like testing and debugging. This was one of the main aspects that lead us to the choice of the well known OBJ geometry format, developed by Wavefront Technologies. Furthermore OBJ is almost universally accepted and fully compatible with the OpenSceneGraph library.

The model composing the scene is split in several OBJ files to facilitate its control and transformation. Following the scene-graph approach every OBJ file is loaded in memory and stored in a node. Some of the nodes are grouped together based on their meaning. For example there is a group characterizing the forum of Nerva, another one characterizing the virtual TV and so on.

In order to avoid an unresponsive status for the application, all the models are loaded in memory concurrently using a secondary thread. Threads are managed using the already described Grand Central Dispatch (GCD) tech-

nology. Most of them are pre-loaded in memory when application is starting, others are loaded on demand.

The model of *Colonnacce*, the only remains of the forum of Nerva, is saved in a separate group in order to have an efficient interaction.

3.8 Designing the User Interface

The User Interface is designed to be flexible and compatible with the different graphical resolution the iDevices are equipped with. Following the *iOS Human Interface Guidelines*[1], we designed a simple and clean user interface with three status buttons for observing different reconstruction of a cultural property such as the forum of Nerva. The interface is thought to be universal so that the framework can be used for other cultural properties.

The user interface has been designed with the principle of aesthetic integrity, consistency and direct manipulation described in the previously cited document. The application adapts its interface based on the orientation of the device and on its status.

3.9 Shaders

Shaders, as already presented in the previous chapters, are small programs that runs directly on the GPU. The effects achieved for a 3D model at time of rendering strictly depends on this small piece of software.

Shaders determine how the objects will be rendered in screen. We realized different shaders depending on the characteristics of the objects portions. Some objects may have textures, others have texture for normal mapping and others may have both. In order to optimize the rendering process different shaders for each different category have been created.

All the shaders implement the phong illumination model and share the following uniforms:

- *u_LightPos*: it is a vector of 4 elements (X, Y, Z, W) indicating the light position.

[1]See at iOS Developer Library the document `http://developer.apple.com/library/ios/DOCUMENTATION/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf`

- *u_LightAmbient*: it is a vector of 4 elements (R, G, B, A) indicating the ambient color of the light source
- *u_LightModel*: it is a vector of 4 elements (R, G, B, A) indicating the model color of the light source
- *u_LightDiffuse*: it is a vector of 4 elements (R, G, B, A) indicating the diffuse color of the light source
- *u_LightSpecular*: it is a vector of 4 elements (R, G, B, A) indicating the specular color of the light source
- *u_MatAmbient*: it is a vector of 4 elements (R, G, B, A) indicating the ambient color of the material
- *u_MatDiffuse*: it is a vector of 4 elements (R, G, B, A) indicating the diffuse color of the material
- *u_MatSpecular*: it is a vector of 4 elements (R, G, B, A) indicating the specular color of the material
- *u_MatShininess*: it is a float value indicating the shininess of the material

The uniforms are configured at initialization time but they can also be modified in the render loop. The shaders used for textured objects include further uniforms for the texture and for the normal map. All these uniforms are used in the fragment shader to decide the final color of the fragment for the object the shader is executed for.

When a OBJ model is loaded in memory, it is saved in a node that in reality is a group of nodes constituting all OBJ model's drawable parts.

In context like normal mapping, the shader needs to know the tangents, binormals and normals associated to the object whose normals are going to be modified. These are informations that must be saved in attributes because they are different for every vertex composing the object. They are not uniform.

In order to set these attributes in the shaders we need to scan the group graph associated with the OBJ model and search for the geodes (the leafs) which are drawable entities containing the geometry information. For each of the geode we compute the tangent space using OpenSceneGraph, and extract tangents from its structure. Tangent array is then linked to the appropriate

shader attribute (*a_tangent*). Normals are included in the OBJ and so they are read from the file. Binormals are computed as cross product between normal and tangent arrays inside the shader.

We now explain in brief how the fragment and vertex shaders presented in Figures 3.9 and 3.10, and in general all the written shaders that are based on Phong's illumination model work.

In the vertex shader we compute the position of the vertex and we save it on the special variable gl_Position. Then we save some informations in varyings, variables that are shared with the fragment shader. Specifically we compute the eye vector, the light vector and the we retrieve the texture coordinates.

When normal mapping is active we also modify the light vector and the eye vector based on tangents, normals and binormals vectors. Light vector and

```
precision highp float;

varying vec3 lightVec;
varying vec3 eyeVec;
varying vec2 texCoord;
attribute vec4 a_tangent;


uniform vec4 u_LightPos;
uniform bool u_AR_FixedLight;


void main(void)
{
    gl_Position = ftransform();
    texCoord = gl_MultiTexCoord0.xy;

    vec3 n = normalize(gl_NormalMatrix * gl_Normal);
    vec3 t = normalize(gl_NormalMatrix * a_tangent.xyz);
    vec3 b = cross(n, t);

    vec3 vVertex = vec3(gl_ModelViewMatrix * gl_Vertex);
    vec3 vLight;

    if(u_AR_FixedLight)
        vLight = vec3(gl_ModelViewMatrix * u_LightPos);
    else
        vLight = u_LightPos.xyz;

    vec3 tmpVec = vLight - vVertex;

    lightVec.x = dot(tmpVec, t);
    lightVec.y = dot(tmpVec, b);
    lightVec.z = dot(tmpVec, n);

    tmpVec = -vVertex;
    eyeVec.x = dot(tmpVec, t);
    eyeVec.y = dot(tmpVec, b);
    eyeVec.z = dot(tmpVec, n);
}
```

Figure 3.9: Vertex Shader for Textured Normal Mapping using Phong Illumination Model

```
precision highp float;

varying vec3 lightVec;
varying vec3 eyeVec;
varying vec2 texCoord;
//light
uniform vec4 u_LightAmbient;
uniform vec4 u_LightModel;
uniform vec4 u_LightDiffuse;
uniform vec4 u_LightSpecular;
// material
uniform vec4 u_MatAmbient;
uniform vec4 u_MatDiffuse;
uniform vec4 u_MatSpecular;
uniform float u_MatShininess;
//textures
uniform sampler2D u_TextureMap;
uniform sampler2D u_NormalMap;
uniform float u_InvRadius;

void main (void)
{
    float distSqr = dot(lightVec, lightVec);
    float att = clamp(1.0 - (u_InvRadius * sqrt(distSqr)), 0.0, 1.0);
    vec3 lVec = lightVec * inversesqrt(distSqr);//normalize(lightVec);
    vec3 vEyeVec = normalize(eyeVec);

    vec4 base = texture2D(u_TextureMap, texCoord);
    vec3 bumpNormals = normalize(texture2D(u_NormalMap, texCoord).xyz *
        2.0 - 1.0);
    //vec4 vAmbient = u_LightAmbient * u_MatAmbient;
    vec4 vAmbient = (u_LightAmbient * u_MatAmbient) +
                    (u_LightModel*u_MatAmbient);

    float lambertTerm = max( dot(lVec, bumpNormals), 0.0 );
    vec4 vDiffuse = u_LightDiffuse * u_MatDiffuse * lambertTerm;

    float specular = pow(clamp(dot(reflect(-lVec, bumpNormals), vEyeVec),
        0.0, 1.0), u_MatShinines);
    vec4 vSpecular = u_LightSpecular * u_MatSpecular * specular;

    gl_FragColor = vAmbient*base + vDiffuse*base + vSpecular;

}
```

Figure 3.10: Fragment Shader for Textured Normal Mapping using Phong Illumination Model

eye vector are modified as in the equations 3.1 and 3.2,

$$\begin{pmatrix} lightVector.x \\ lightVector.y \\ lightVector.z \end{pmatrix} = \begin{pmatrix} dot(lightVector, t) \\ dot(lightVector, b) \\ dot(lightVector, n) \end{pmatrix} \tag{3.1}$$

Where *lightVector* on the right hand of equation 3.1 is initially computed as difference between the vectors representing light and vertex positions in the model view reference frame, while *vertexVector* of equation 3.2 is the vertex position in the model view reference frame obtained as product between the model view matrix and the vertex position of the 3d model.

$$\begin{pmatrix} eyeVector.x \\ eyeVector.y \\ eyeVector.z \end{pmatrix} = \begin{pmatrix} dot(-vertexVector, t) \\ dot(-vertexVector, b) \\ dot(-vertexVector, n) \end{pmatrix} \tag{3.2}$$

In the fragment shader at the beginning we compute the attenuation of

light based on the distance of the light from the fragment. In particular we use the Light Radius method, in which an imaginary sphere centered on the light is used as influence area. The attenuation factor decreases along the radius and becomes equal to zero on the edge of the sphere, and outside of course.

If normal mapping is active we modify the normals using the corresponding texture. After normalizing the eye vector and adjusting the light vector based on the distance of the light, we finally compute ambient, diffuse and specular terms. At the end of the shader we set the fragment color using the special variable gl_FragColor as sum of the ambient, diffuse and specular components scaled by the attenuation factor.

In the case of texturing these components are multiplied with the correspondent texture color associated to the fragment.

3.10 Data persistence

In order to increase the flexibility of the application framework, all the shader settings, 3D model names and other relevant content is stored in XML files. By modifying the XML files we can adapt the application for another forum or another cultural heritage site. The information about image targets is stored in XML files generated by the Vuforia TMS web interface. Their data feature representation is stored in a binary file which is referenced by the XML.

The data as the video, image and textual content related to the forum of Nerva is bundled on the application, however it is possible to use sources like the web for storing and updating this information. We plan in the future to use a cloud system such as dropbox in order to introduce a flexible update of the target images.

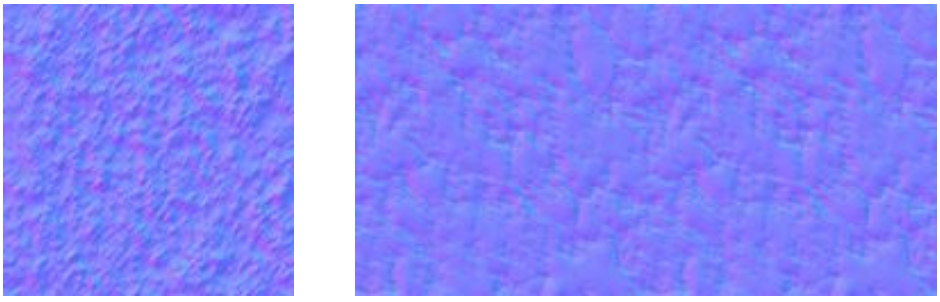


Figure 3.11: Example of textures used for Normal Mapping

CHAPTER 4

EXPERIMENTAL RESULTS

One of the most prominent factors in augmented reality is the measure of robustness with respect to light variations, deformations and target's occlusion. The application has been tested in various light conditions such as artificial spot light or the midday' sun light that caused several highlights to partially occlude the target. We noticed that, gap of performance between opposed trial runs, increased when using targets that *Vuforia*'s TMS classified with 1 or 2 stars. Best performance is obtained with targets classified between 3 and 5 stars. Testing was important to evaluate how to build the best image targets for an outdoor application.

Another important factor for performance is the complexity of the 3D model representation of the ancient forum of Nerva. Due to the limited but different hardware specifications in which the application was going to run, we performed tests in all the current iOS mobile devices running iOS 5, the latest operating system developed by Apple at the moment of this writing. Difference in performance between devices running on Apple A5 cpu (such as iPad 2nd generation and iPhone 4s) and those running on Apple A4 cpu (such as iPod Touch 4th generation and iPhone 4) was consistent and evident in terms of FPS measure.

To obtain a smooth frame rate during real-time rendering we had to consistently limit the number of vertices of the complete scene graph. As we can see in table 4.1 the performance of the single core cpu A4 is very limited, while the

State of the App	Vertices/ Faces	Dual Core A5	Single Core A4
State 1	3269/4034	40-50 fps	9-15 fps
State 1 (no map)	1296/1358	60 fps	25-28 fps
State 2	9570/14267	28-30 fps	6-7 fps
State 2 (no map)	7597/11591	30-32 fps	8-10 fps
State 3	9791/14577	26-30 fps	5-7 fps
State 3 (no map)	7818/11901	30-31 fps	8-10 fps
ARVideo	12/7	60 fps	39-41 fps

Table 4.1: FPS performance comparison between the single core A4 and the dual core A5.

one of the dual core cpu A5 is good enough for a smooth use of the application. This difference in performance is also motivated by the fact that A5 has also a dual core GPU that is much faster than the single core GPU integrated in A4. As we can see the number of vertices and faces are the bottle neck of our application.

We used techniques such as normal mapping and texturing to maintain a good level of quality in spite of the model's low complexity.

Application was tested several times in place at different hours of the day with different light conditions proving a good usability.

We paid attention finding a trade off between user touch interaction responsiveness and quality of real time rendering.

4.1 Outdoor testing

Outdoor tests were performed near the ruins of the forum of Nerva, in Rome. A site inspection was done for searching the best spots where tourists used to walk and stop to observe the ruins. The site inspection highlighted three possible zones as candidates for the target placement. An aspect that we took in consideration for the choice of the sites was the presence of two information boards in two of the sites (see Figures 4.1 and 4.2).

The first approach was to capture photographies of the two information boards and use them as image targets for our application. Then we processed the images for extracting features by optimizing local contrast and we computed the datasets using the Vuforia TMS. Afterward we tested the application

Figure 4.1: First Site used for Tests

Figure 4.2: Second Site used for Tests

pointing the iPod Touch camera on the real information boards. This preliminary test was successful and its result can be seen in the screen's snapshot shown in figure 4.3. The target was immediately recognized and it was very accurate in tracking and in managing the possible occlusion. It is relevant to mention that shadows of the railings projected on the information board did not have a considerable impact on the target tracking and recognition.

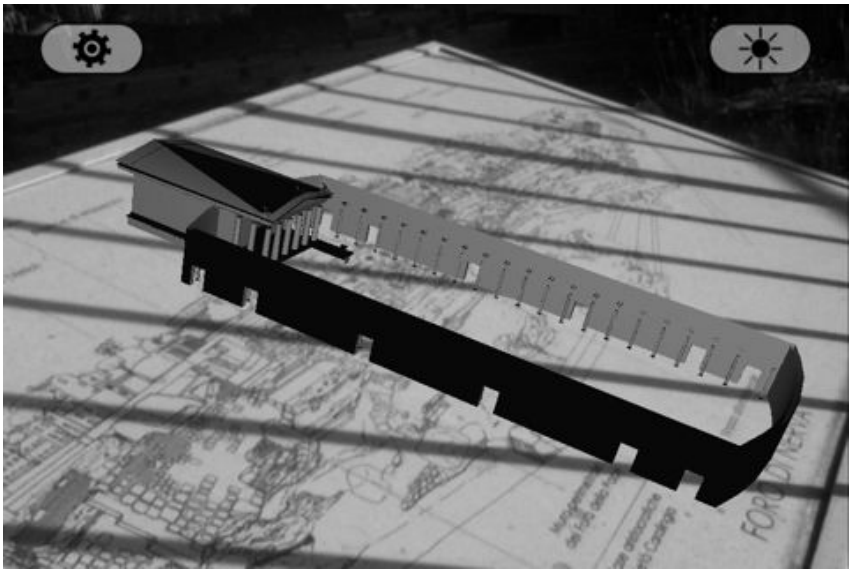


Figure 4.3: Screenshot of the first application test in outdoor conditions.

After discussing with the architects and the historian we decided that the best approach was to design a new information board including our specific image targets. These information boards should be designed to provide a kind of information that can be intuitively extended by the augmented reality layer and should be easy to understand. Ideally they should not contain any kind of written information so that people from every nation can understand. The written information could then be augmented through the application which will be ideally issued using every possible and known language in the world.

4.2 Hardware Architecture

The augmented reality mobile framework has been tested in almost all the Apple devices running iOS 5:

- iPad 2nd Generation
- iPhone 4
- iPhone 4s
- iPod Touch 4th Generation

In addition we are planning to test the application in the new iPad. The main device used for the development is an iPod Touch 4th Generation equipped

	iPad *(3rd gen)*	iPad *(2nd gen)*	iPhone 4s	iPhone 4	iPod Touch *(4th gen)*
Capacity	16 GB, 32 GB, 64 GB	16 GB, 32 GB, 64 GB	16 GB, 32 GB, 64 GB	8 GB	8 GB, 32 GB, 64 GB
Display Size	2048x1536 264 ppi 9.7"	1024x768 132 ppi 9.7"	960x640 326ppi 3.5"	960x640 326ppi 3.5"	960x640 326ppi 3.5"
CPU & GPU	A5X dual-core cpu quad-core gpu	A5 dual-core cpu dual-core gpu	A5 dual-core cpu dual-core gpu	A4 single-core cpu single-core gpu	A4 single-core cpu single-core gpu
RAM	1GB	512MB	512MB	512MB	256MB

Table 4.2: Hardware comparison between iDevices

with 8GB of SSD and 256 MB of RAM. As we can see from the table 4.2 the iPod Touch can be considered a lower bound for the device family. Nonetheless, considering a future porting of the application to Android OS, iPod Touch is closer in performance to the middle segment of smartphones running this operating system.

The main difference between the iPod Touch and the other iOS devices is the absence of the A-GPS and of the digital compass. This is negative for augmented reality applications because many times these sensors can be of great help for the registration problem, however the design of our framework does not rely on such sensors but is completely based on vision algorithms.

Two other important differences on iPod Touch are the limited RAM which is only 256 MB and the limited processor power. In fact the CPU and the GPU have only one core, limiting the multi-threaded characteristics of the application.

As we can observe in table 4.3 all the iOS devices are equipped with a complete set of sensors:

- *MEMS Gyroscope:* it is a electronic sensor used to measure orientation and speed of rotation on the 3 axis X-Y-Z. It works using the angular momentum principles.

- *Accelerometers:* it is used to measure the acceleration of the device in

the 3 axis X-Y-Z.

- *A-GPS*: the Assisted GPS is a location system that improves under certain conditions the GPS satellite based positioning system. It uses the cellular network to locate the satellites in poor signal conditions.
- *Digital Compass:* also known as magnetometer it is used to measure the direction's strength of the Earth's magnetic field.
- *Proximity:* available only on iPhone it is a sensor able to detect the presence of nearby objects without any physical contact.
- *Ambient Light Sensor*: it is a sensor used to measure light intensity in the environment.

  All these sensors can be used in future to improve the registration of virtual objects in the real environment. Unfortunately the main device used during the development of the application, the fourth generation of iPod Touch, lacks of A-GPS and Digital Compass, and for this reason the development was oriented using an approach based only on vision algorithms included in the Vuforia SDK.

	iPad *(3rd gen)*	iPad *(2nd gen)*	iPhone 4s	iPhone 4	iPod Touch *(4th gen)*
Gyro	YES	YES	YES	YES	YES
A-GPS	YES	YES	YES	YES	NO
Accel.	YES	YES	YES	YES	YES
Compass	YES	YES	YES	YES	NO
Proximity	NO	NO	YES	YES	NO

Table 4.3: Sensors comparison between iDevices

CHAPTER 5

CONCLUSION

In this final chapter we do a brief discussion about the results of our work describing the methodology, and underlining positive and negative aspects encountered in the project.

5.1 Results

An augmented reality framework has been designed for the specific purpose of building applications to provide accurate information on the context of cultural heritage and architecture. In this particular field our approach differs from other solutions currently available in the market that we were able to test. In fact most of these applications claim to use augmented reality, but in truth are just virtual reality systems able to provide geolocalized content.

In this thesis we tried to give a contribute to the development of digital technological interfaces such as augmented reality to influence relevant aspects of the cultural heritage environment. Using three-dimensional animations integrated with the real world we give a better way for the delivery and understanding of cultural information to people interested in the history of the Roman Imperial Fora.

We established an open talk between different areas of expertise like IT and Architecture, with the objective to achieve a better understanding of the critical requirements in delivering and managing information for the cultural

heritage science.

Augmented Reality technology has in fact the potential to influence many aspect of the cultural heritage environment.

The system has been realized using mobile devices, considering the fact that they are the perfect platform for this kind of application because they are easy to carry, already in the market and economically advantageous. Furthermore they integrate all the different kind of sensors that an augmented reality system needs. The only limitation can be identified in the computational power, however mobile technology is in rapid evolution.

We used cross-platform libraries for the development of our system so that a future porting to the Android platform will be easier. The application has been tested with different light conditions in indoor and outdoor environments.

5.2 Discussion

During the development and the design of our work we came up against different challenges, as the one regarding the research for the best trade-off between the application performance and the artistic result of the representation. Mobile smartphones and tablets are in fact limited platforms with bounded and disparate hardware capabilities, especially concerning the computational power. A negative aspect arising from these limitations is defined by the complexity of creating and designing a product capable to run on hardware that is rapidly evolving, such as the one the mobile platform is characterized by.

A question that may emerge in this direction is whether the support for devices such as the iPhone 4 and the iPod Touch should be taken into account or ignored. Keeping a considerable low level of polygons didn't help to improve enough the performance of these two devices that are equipped with a single core CPU and GPU.

Considerable efforts have been devoted to optimization of the threedimensional representation of the reconstruction, however potential annihilation of the artistic result inflicted by the limits of such devices should also be considered, keeping in mind the ambition of historical-architectural documentation that the application should lead up to.

This was the main topic of discussion with the historian of architecture, who addressed the importance of giving accurate informations relevant to the

historical context, suggesting the omission of those informations that were not accurately verified. The accuracy of representation is also given by the level of detail characterizing the reconstructed model of the forum of Nerva and of the surrounding environment. In this case it is crucial to understand the significance of each one of the forum's parts, in order to avoid oversimplifications that can easily lead to an incorrect representation in view of architecture and history.

It is clear that, at the moment, an augmented real time rendering of a detailed reconstruction of the forum of Nerva is unfeasible, however the augmented reality technology can improve and facilitate the human perception and understanding of the historical evolution in world cultural heritage sites, such as the area of the ancient Imperial Fora of Rome.

The augmented reality technology should serve as a door for retrieving informations in a more natural and intuitive way. In this fashion the virtual layer has been designed to be simple and interactive. By touching the point of interest users can gain access to relevant content conveyed by means of other media types like images, video and photographies.

5.3 Future Works

The work of this thesis has been submitted to two conferences well known in the sector of computer graphics and augmented reality, applied to the cultural heritage and archeology fields.

The fist conference, VAST 2012 is the 13th International Symposium on Virtual Reality, Archaeology and Cultural Heritage, and will be held at Brighton, UK, on 19-21 November 2012. The second one, CAA 2013, is an annual conference that has a long-standing reputation of showcasing new and innovative developments in the application of digital technologies to various aspects of cultural heritage and ancient world studies [1].

Recently we realized the porting of our framework to iOS 6, that was released in October. In future we plan to continue the development of the application dedicated to the forum of Nerva with the goal of improving its functionalities. If the application is going to be a valid product, we will use

[1] See `http://www.vast2012.org` and `http://www.caa2013.org/`

our framework for developing other applications dedicated to the other Imperial Fora. To do that we are in collaboration with personnel in the sector of architecture and archeology which will give us the necessary information to improve the application from the artistic and historical point of view.

Furthermore improvements of the AR subsystem can be realized using sensor fusion to improve the solution of the pose detection problem. Sensors like GPS, compass, gyroscope and accelerometer could be useful for "detaching" the augmented reality layer from the image target.

We know that after calibrating the pose using an image target at a specific known location, it is possible to project the object at the ground level or in a specific relative position with a simple transformation. Most of the times the problem arising in this situation is that you can't freely observe the object projected on the ground without loosing the target and hence the necessary information for extracting the correct relative pose. The real challenge is to compensate the movements of the mobile device using only the information coming from motion sensors in order to unlink the augmented layer from the feature information coming from the target image. How to realize such motion sensor fusion is a possible field of future research for the improvement of our application framework.

RINGRAZIAMENTI

Con piacere chiudo questa tesi ringraziando tutte quelle persone che hanno sempre creduto in me, sostenendomi durante le difficoltà e incoraggiandomi ad andare avanti anche nei momenti difficili.

Ringrazio i miei genitori per avermi messo al mondo e cresciuto con impegno e sacrificio. Grazie per avermi dato e negato quando serviva. Per avermi concesso la libertà di scegliere, la libertà di conoscere, la libertà di decidere per il mio futuro. Senza il vostro sostegno non ce l'avrei mai fatta. Un grazie di cuore a mio fratello Angelo per essermi sempre stato vicino e per avermi sopportato in questi mesi di tesi. Grazie a tutta la mia famiglia.

Ringrazio tutti gli amici, che mi hanno accompagnato con esperienze uniche nelle mie passioni e nei miei interessi, credendo in me e nelle mie capacità.

Un ringraziamento particolare va a tutte quelle persone che hanno collaborato alla realizzazione di questo progetto. Ringrazio quindi i professori Marco Fratarcangeli e Tommaso Empler che mi hanno permesso di realizzare una tesi in un campo affascinante e innovativo, consentendomi di conoscere e collaborare con esperti del settore come il prof. Alessandro Viscogliosi, a cui dedico uno speciale ringraziamento per i suoi utili consigli nel campo storico-architettonico. Un grosso ringraziamento va a Giacomo Zilocchi per il suo contributo nel lavoro di progettazione e modellazione architettonica.

Appendices

APPENDIX A

APPLICATION MANUAL

Welcome to Nervar. Nervar will help you to get, in a innovative way, cultural and historical information about the ancient Forum of Nerva, one of the Imperial Fora of Rome.

A.1 How to get started

You can get Nervar in different ways. The easiest way is to use a QR code reader and point the camera of your iPhone, iPad or iPod Touch toward the QR code printed on the information board located near the ruins of the Forum of Nerva. You will be automatically redirected to the AppStore for downloading the application. In alternative you can directly search for Nervar in the AppStore. In any case you can download and install the application very easily.

After Nervar starts up you will see images coming from your backside camera live on your screen. Just point the camera of your mobile device towards the information board and the virtual world will magically appear on it. You can also download and print our target to use the application indoor. You can find them in our website `http://nervar.altervista.org`.

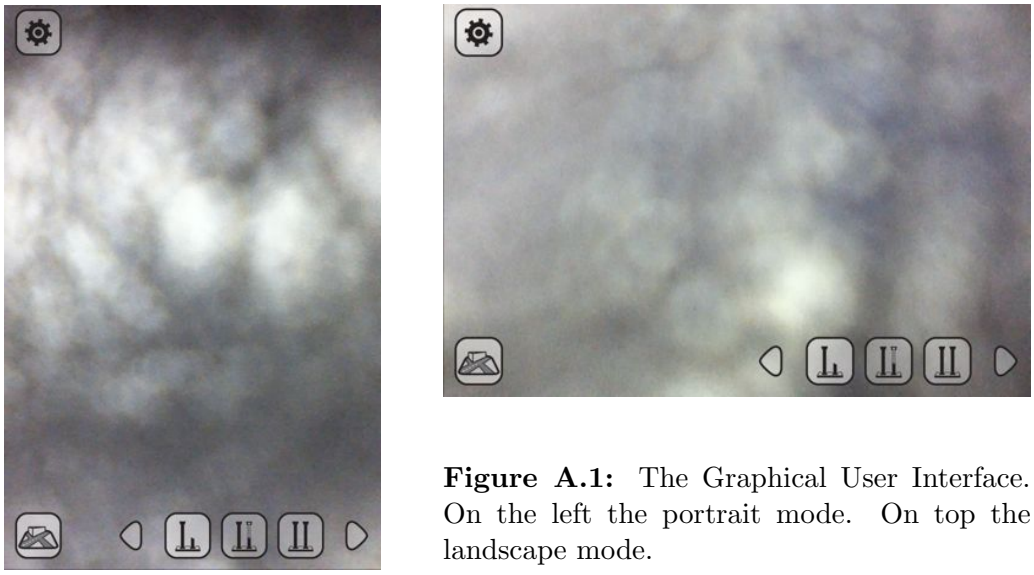


Figure A.1: The Graphical User Interface. On the left the portrait mode. On top the landscape mode.

A.2 User Interface

Nerva works in both landscape and portrait mode.

When you rotate your device buttons will be automatically rearranged as shown in figure A.1. The button located on the upper-left part of your screen is the settings button. On the bottom-left there's the maps button and on the bottom-right you can find the status buttons for the augmented reality. When a button is selected, its color becomes more opaque.

A.3 Interaction

The three status buttons shown on figure A.2 are located at the bottom right of the screen and can be used for getting different visual information about the forum of Nerva:

1. Ruins and Plan of the Forum of Nerva.
2. Reconstruction of the Forum of Nerva.
3. Reconstruction of the Forum of Nerva and other fora around it.

When you press the first button, you can observe the ruins of the forum of Nerva above its plan, so that it's easy to understand what parts of the Forum of Nerva are still present today (i.e. The *Colonnacce*).

Figure A.2: The three status button

The second button augments the reality by projecting the complete reconstruction of the forum of Nerva on the information board. The third button adds a volumetric reconstruction of the fora around the forum of Nerva, in order to better explain it's location during the Roman Imperial Period.

Furthermore you can easily navigate among the three states using the left and right arrows.

The yellow man-shaped indicator that you can see in the augmented reality represent your position with respect to the virtual world.

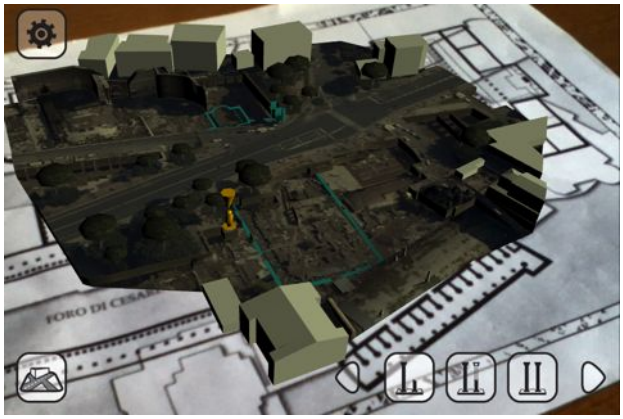
In the second state you have a reconstruction of the forum of Nerva highlighted in green color. You can interact with it using a tap. When you touch the forum of Nerva a menu with different choices appears on the screen allowing you to explore different aspects of architecture and history of the forum.

Each selection contains different visual and historical content so that you can learn more about the forum of Nerva.

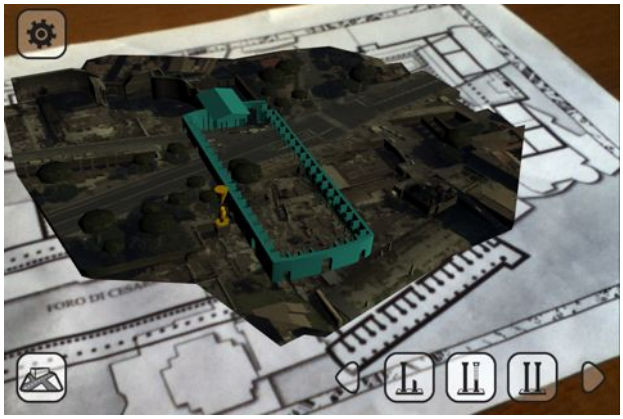
Figure A.3: The maps button

The maps button (see Figure A.3) located at the bottom-left of the screen allows you to extend the augmented reality with a 3D satellite representation of the area around the forum of Nerva, enabling you to better understand forum of Nerva's position. To activate maps just tap on the button. To disable it tap on the button again.

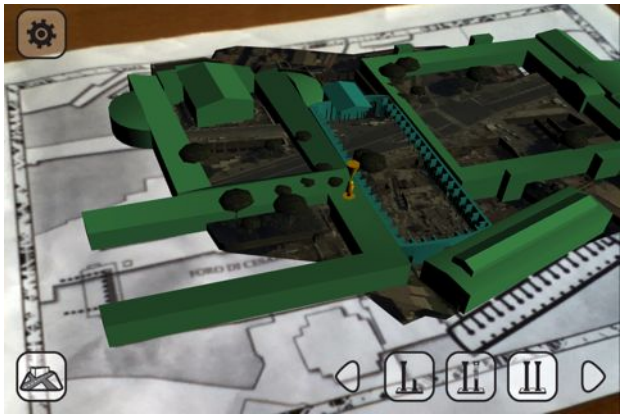
In figure A.4 we can see the three states of the augmented reality. In particular figure (a) shows the ruins and plan of the Forum of Nerva, figure (b) shows a reconstruction of the Forum of Nerva, while figure (c) shows a reconstruction of the Forum of Nerva with respect to the other fora around it.



(a)



(b)



(c)

Figure A.4: The app states: (a) Ruins and Plan of the Forum of Nerva. (b) Reconstruction of the Forum of Nerva (c) Reconstruction of the Forum of Nerva and other fora around it.

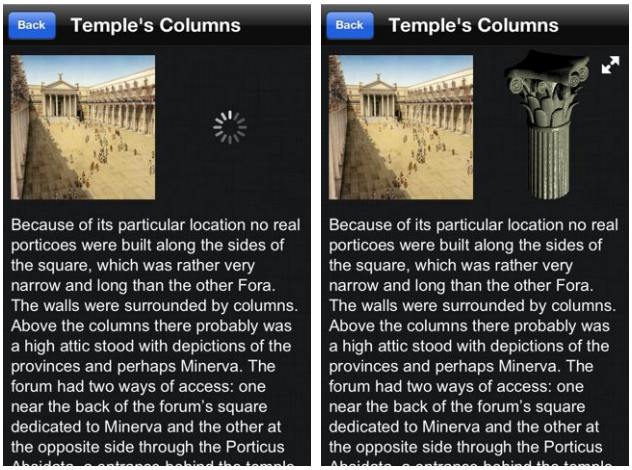


Figure A.5: The visual textual information window

A.4 Visual Textual Information

When you touch the forum of Nerva and select one of the possible choices, a window like the one shown in Figure A.5 is presented on your touch screen, showing photographies, textual information and an interactive 3D model. On the upper part of the screen you will have a 3D accurate reconstruction of some parts of the forum of Nerva (e.g. a detail of the column, see Figure A.6).

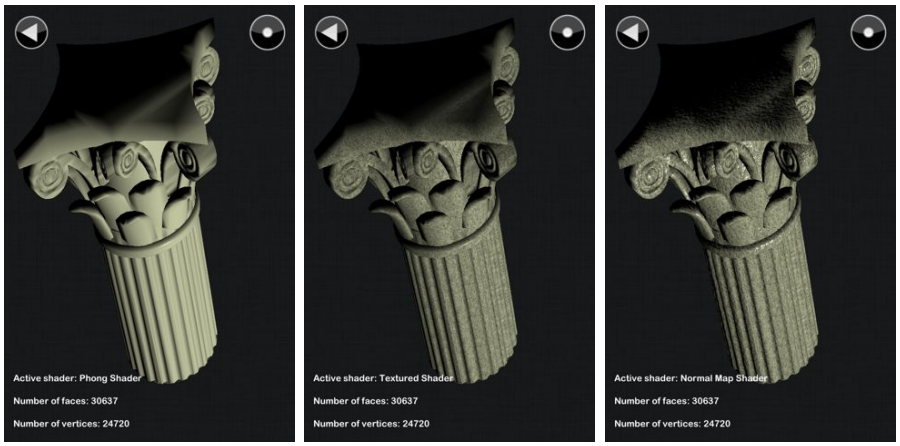


Figure A.6: A detail of the column of the forum of Nerva, using different shader effects. From left to right: Phong, textured, normal map

You can interact with the 3D model by rotating it. If you want to observe it in more detail just tap on the full-screen button. There you can have more information about the model, rotate it with your finger, zoom it by pinching and choose different visual effects: bump mapping, texture, and simple Phong.

A.5 Maps

When you tap on the virtual satellite maps on the augmented reality a window showing an interactive detail view of the area will popup, giving you the possibility to see your position with respect to relevant part of the forum of Nerva, such as the ruins of the forum.

The archaeological area of the forum of Nerva is overlaid on the map with a black stroke.

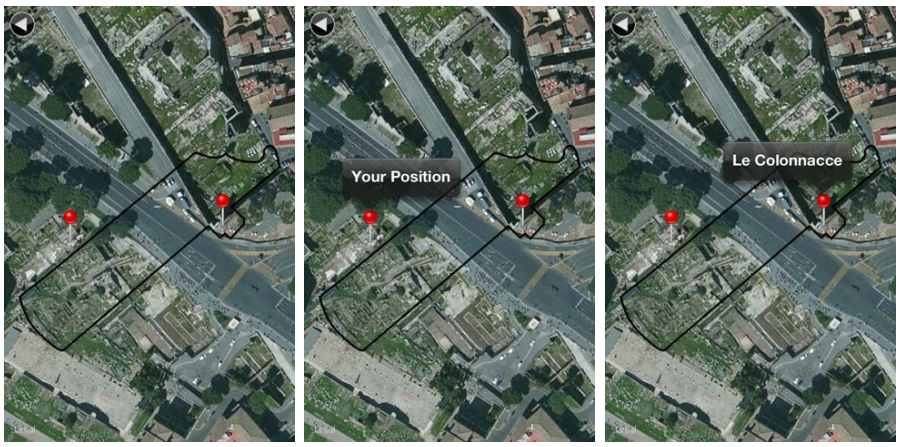


Figure A.7: The map function. In black the archaeological area of the forum of Nerva.

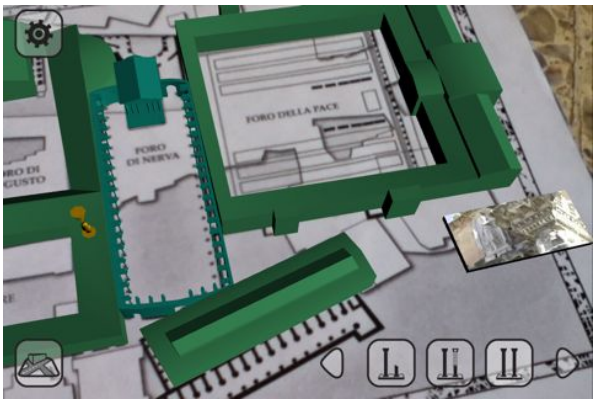
A.6 AR Video

Nervar enriches your experience using multimedia content of different kind. AR Video is one of them. AR Video can be presented in the form of virtual animated icons and of virtual tv screen. Animated video icons are fully integrated in the augmented reality and usually you can find them near the virtual reconstruction (see Figure A.8(b)) . They allow you to explore video content with a single tap.

AR Video can be presented also as a virtual tv screen so that you can choose to watch a live video directly on the augmented reality in a new different way (see Figure A.8(c)). You can interact with the Video AR by tapping it to pause and play. If your desire is to watch the video in fullscreen just double tap the AR video virtual screen (see Figure A.8(a)). In fullscreen mode you

don't need to keep your phone pointed on the information board to watch the video.

(a)



(b)

(c)

Figure A.8: The ARVideo functionality: (a) the fullscreen mode, (b) the virtual animated icon, (c) the virtual tv screen

APPENDIX B

THE ANCIENT FORUM OF NERVA

The developed augmented reality application framework was tested in the context of cultural heritage, specifically with the goal to provide educational and architectural informations about the ancient Forum Transitorium, commonly known as Forum of Nerva.

B.1 History of the Forum of Nerva

Writings by Suetonius, a historian of ancient Rome, say construction of this ancient forum was due to Domiziano. According to Martial, a Latin poet, the forum was built around 85-86 AD and was first used during 95-96 AD, at least one year before its known official opening dated around 97 AD by the emperor Nerva.

The forum of Nerva is also called forum transitorium because of its function of transit between the Subura, an area of the city of Rome, and the Roman Forum, the center of Roman public life. The forum of Nerva transformed and monumentalized the occidental part of Argiletum, an ancient street of Rome which used to link the Republican Forum to the Subura quarter. The Forum was located in a narrow space among the Forum of Augustus, The Forum of Caesar and the Temple of Peace.

B.2 Architectural Reconstruction

Because of its particular location no real porticoes were built along the sides of the square, which was rather very narrow and long if compared to the ones on the other Fora. The walls were surrounded by columns. Above the columns there probably was a high attic stood with depictions of the provinces and perhaps Minerva. The forum had two ways of access: one near the back of the forum's square dedicated to Minerva and the other at the opposite side through the Porticus Absidata, a entrance behind the temple facing the Subura quarter.

The temple had six Corinthian columns on the front with diversely spaced intercolumniations. There was a dedicatory inscription carved on the architrave and on the frieze. As reported by some findings the frieze in the Temple of Minerva was supposed to be decorated with bucrania (oxen's skulls) and sacrificial tools.

The position of the Forum of Nerva is by some means protected by the other Fora and this fact can be a reason for its preservation after the fall of the Roman Empire. However in the same period the original paving was replaced by a compact cobbled one.

B.3 The End of the forum of Nerva

Around 9th century the forum was abandoned and its use was dedicated to habitation: a hut and two-level houses (domus solaratae) were built in the area. The largest one had a four-arch portico along the new route. Later on, the level of the street was raised many times and the habitations were modified or torn down. These residential structures were abandoned around 12th century.

In 1609 Pope Paul V tore down the forum of Nerva's temple with the purpose of using its marbles for the construction of the fountain of Acqua Paola on the Janiculum hill.

B.4 Between the Ruins and the Myth

Today the only remains of the forum of Nerva are the *Colonnacce* that were partially buried around the 1930s. These colums still show part of the figured frieze which represents the myth of Aracne, who was a great mortal weaver

who boasted that her skill was greater than that of Athena, goddess of wisdom and strategy. The myth tells that Athena offended by Arachne's arrogance, set a contest between the two weavers. According to Ovid in the book Metamorphoses, the goddess was so envious of the magnificent tapestry and of the mortal weaver's success that she destroyed the tapestry and loom and slashed the girl's face. Ultimately, the goddess turned Arachne into a spider.

Figure B.1: The only remains of the Forum of Nerva: The Colonnacce

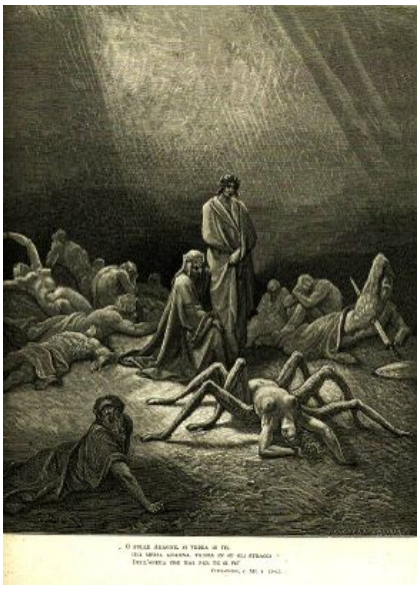


Figure B.2: Detail of a Gustave Doré illustration (1832 - 1883), Divina commedia, Purgatorio, canto XII, The myth of Arachne and Athena

BIBLIOGRAPHY

[Ababsa and Mallem, 2008] Ababsa, F. and Mallem, M. (2008). Robust camera pose estimation combining 2d/3d points and lines tracking. In *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pages 774 –779.

[Altair4, 2012] Altair4 (2012). Rome MVR. `http://www.altair4.com`.

[Azuma, 1993] Azuma, R. (1993). Tracking requirements for augmented reality. *Commun. ACM*, 36(7):50–51.

[Azuma, 1997] Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385.

[Bosché et al., 2012] Bosché, F., Tingdahl, D., Carozza, L., and Van Gool, L. (2012). Markerless vision-based Augmented Reality for enhanced project visualization. *Gerontechnology*, 11(2):69.

[Brooks, 1996] Brooks, Jr., F. P. (1996). The computer scientist as toolsmith ii. *Commun. ACM*, 39(3):61–68.

[Bruns et al., 2007] Bruns, E., Brombach, B., Zeidler, T., and Bimber, O. (2007). Enabling mobile phones to support large-scale museum guidance. *IEEE MultiMedia*, 14(2):16–25.

[Carmigniani et al., 2011] Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., and Ivkovic, M. (2011). Augmented reality technologies,

systems and applications. *Multimedia Tools and Applications*, 51:341–377. 10.1007/s11042-010-0660-6.

[Chou and ChanLin, 2012] Chou, T.-L. and ChanLin, L.-J. (2012). Augmented reality smartphone environment orientation application: A case study of the fu-jen university mobile campus touring system. *Procedia - Social and Behavioral Sciences*, 46(0):410 – 416. 4th World Conference on Educational Sciences (WCES-2012) 02-05 February 2012 Barcelona, Spain.

[Choudary et al., 2009] Choudary, O., Charvillat, V., Grigoras, R., and Gurdjos, P. (2009). March: mobile augmented reality for cultural heritage. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 1023–1024, New York, NY, USA. ACM.

[Corvallis, 2012] Corvallis (2012). Rome View. `http://www.corvallis.it`.

[Dähne and Karigiannis, 2002] Dähne, P. and Karigiannis, J. (2002). Archeoguide: System architecture of a mobile outdoor augmented reality system. *Mixed and Augmented Reality, IEEE / ACM International Symposium on*, 0:263.

[FalloutSoftware, 2009] FalloutSoftware (2009). Opengl lighting or how light sources work (long, in-depth tutorial). `http://www.falloutsoftware.com/tutorials/gl/gl8.htm`.

[Fischer et al., 2008] Fischer, J., Haller, M., and Thomas, B. (2008). Stylized Depiction in Mixed Reality. *International Journal of Virtual Reality*, 7(4):71–79.

[Gao, 2008] Gao, J. (2008). Hybrid tracking and visual search. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 909–912, New York, NY, USA. ACM.

[Gartner, 2012] Gartner (2012). Gartner's 2012 hype cycle for emerging technologies identifies "tipping point" technologies that will unlock long-awaited technology scenarios. `http://www.gartner.com/it/page.jsp?id=2124315`.

[Gleue and Dähne, 2001] Gleue, T. and Dähne, P. (2001). Design and implementation of a mobile device for outdoor augmented reality in the archeoguide project. In *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*, VAST '01, pages 161–168, New York, NY, USA. ACM.

[Google, 2012] Google (2012). Project Glass. `https://plus.google.com/+projectglass`.

[Hartley, 2012] Hartley, M. (2012). Google merges digital and physical worlds with new image-based projects. `http://tinyurl.com/c8o8cod`.

[Hoff et al., 1996] Hoff, W. A., Nguyen, K., and Lyon, T. (1996). Computer-vision-based registration techniques for augmented reality. In Casasent, D. P., editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 2904 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 538–548.

[Honkamaa et al., 2007] Honkamaa, P., Siltanen, S., Jäppinen, J., Woodward, C., and Korkalo, O. (2007). Interactive outdoor mobile augmentation using markerless tracking and gps. *Proceedings of the Virtual Reality International Conference VRIC Laval France*, pages 285–288.

[Illusionetwork, 2012] Illusionetwork (2012). i-Mibac Voyager. `http://www.illusionetwork.com`.

[iPhoneDevelopment, 2012] iPhoneDevelopment (2012). iPhone Development Tutorials. `http://iphonedevelopment.blogspot.it/`.

[Khronos, 2012] Khronos (2012). Opengl es and glsl specifications. `http://www.khronos.org/opengles/`.

[Lee et al., 2012] Lee, A.-H., Lee, S.-H., Lee, J.-Y., and Choi, J.-S. (2012). Real-time camera pose estimation based on planar object tracking for augmented reality environment. In *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, pages 516 –517.

[LightHouse3D, 2012] LightHouse3D (2012). Glsl tutorials. `http://www.lighthouse3d.com/`.

[Papagiannakis et al., 2005] Papagiannakis, G., Schertenleib, S., O'Kennedy, B., Arevalo-Poizat, M., Magnenat-Thalmann, N., Stoddart, A., and Thalmann, D. (2005). Mixing virtual and real scenes in the site of ancient pompeii: Research articles. *Comput. Animat. Virtual Worlds*, 16(1):11–24.

[Pasman and Woodward, 2003] Pasman, W. and Woodward, C. (2003). Implementation of an augmented reality system on a pda. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 276 – 277.

[Seo et al., 2008] Seo, B.-K., Choi, J., Han, J.-H., Park, H., and Park, J.-I. (2008). One-handed interaction with augmented virtual objects on mobile devices. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '08, pages 8:1–8:6, New York, NY, USA. ACM.

[Shah et al., 2012] Shah, M. M., Arshad, H., and Sulaiman, R. (2012). Occlusion in augmented reality. In *Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference on*, volume 2, pages 372 –378.

[Shoemake, 1992] Shoemake, K. (1992). Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of the conference on Graphics interface '92*, pages 151–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Shoemake, 1994] Shoemake, K. (1994). Graphics gems iv.

[van Dam, 1997] van Dam, A. (1997). Post-wimp user interfaces. *Commun. ACM*, 40(2):63–67.

[van Krevelen and Poelman, 2010] van Krevelen, D. and Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *The International Journal of Virtual Reality*, 9(2):1–20.

[Viscogliosi, 2010] Viscogliosi, A. (2010). Il foro transitiorio. In *Divus Vespasianus. Il bimillenario dei Flavi.*, pages 202–208. Mondadori Electa.

[Vlahakis et al., 2002] Vlahakis, V., Ioannidis, N., Karigiannis, J., Tsotros, M., Gounaris, M., Stricker, D., Gleue, T., Daehne, P., and Almeida, L. (2002). Archeoguide: An augmented reality guide for archaeological sites. *IEEE Comput. Graph. Appl.*, 22(5):52–60.

[Wagner et al., 2008] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. (2008). Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 125–134, Washington, DC, USA. IEEE Computer Society.

[Wang and Qian, 2010] Wang, R. and Qian, X. (2010). *OpenSceneGraph 3.0 Beginner's Guide*. Packt Publishing.

[Wang and Qian, 2012] Wang, R. and Qian, X. (2012). *OpenSceneGraph 3.0 Cookbook*. Packt Publishing.

[Zornitza et al., 2012] Zornitza, Y., Dimitrios, B., and Christos, G. (2012). Overview of smartphone augmented reality applications for tourism. *e-Review of Tourism Research (eRTR)*, 10(2).