Challenges of a Pose Computation Augmented Reality Game Application

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Chiu Ni Wang

Graduate Program in Computer Science and Engineering

The Ohio State University

2011

Master's Examination Committee:

Rajiv Ramanath, Advisor

Jay Ramanathan
Abstract

An Augmented Reality (AR) application combines the virtual world and reality to enhance the user’s perception of the real world. Earlier, augmented reality technology had not found much commercial success because it required bulky gadgets and considerable processing power. But now AR technologies are available and computationally feasible on commodity mobile devices, such as smartphones.

Pose estimation AR applications make use of inbuilt hardware features like the accelerometer, magnetometer and Global Positioning System (GPS) for overlaying images onto the real world. However, the dependence of the working of AR on hardware features gives rise to new challenges, which must be addressed in order to provide augmentation in as seamless a manner as possible. These challenges include real time estimation of the user’s viewport, geo-magnetic sensor noise, the delay in receiving GPS data, and the synchronization of timer threads. Other challenges include those of debugging and testing because realistic GPS feeds and other sensory data are unavailable during development.

Next, most AR applications have the need for common framework-based services, such as tracking, image registration and interface handling. This thesis also explores the feasibility of such a framework, where a reusable platform is coupled with a customizable application specific component to create the AR application.
Finally, this thesis also proposes an architecture for an AR application that lives and evolves inside a cyber-infrastructure, in order to support a wider range of users through the development of a wider range of AR applications.
Dedication

This document is dedicated to my parents.
Acknowledgments

I am grateful to my Advisor, Dr. Rajiv Ramnath and Dr. Jay Ramanathan for their continuous assistance and support throughout my master’s program especially, during the course of my thesis writing.

I am also deeply indebted to Songwhale and CETI for funding the project, which helped me produce this thesis. Songwhale’s CTO and my supervisor, Jon Greenlee has been an amazing boss. He was understanding and thoughtful in allowing flexibility in my hectic schedule as a student. It was a blessing to work with him.

I appreciate the help I received from Jim Power (application development teammate) and Chenjie Wei (research partner) for my work. They have been cooperative, competent and fun to work with.

Lastly, I would like to show gratitude to my family and friends from church. Their goodwill and prayers have helped me through the two years at OSU. Above all, none of this would have been possible without the grace of God.
Vita

March 2002 .................................. Indian Certificate of Secondary Education,
Council for the Indian School Certificate
Examination, India
March 2004 .................................. Indian School Certificate, Council for the
Indian School Certificate Examination, India
2009 .............................................. B.Tech Computer Science and Engineering,
West Bengal University of Technology,
India
2009 to present ............................... Graduate Student, Department of Computer
Science and Engineering, The Ohio State
University

Fields of Study

Major Field: Computer Science and Engineering
Table of Contents

Abstract ........................................................................................................................................... ii

Dedication ................................................................................................................................ iv

Acknowledgments ................................................................................................................... v

Vita ................................................................................................................................................ vi

Fields of Study ........................................................................................................................ vi

Table of Contents ..................................................................................................................... vii

List of Figures ............................................................................................................................ x

Chapter 1: Introduction ............................................................................................................. 1

Chapter 2: Related Work .......................................................................................................... 7

Chapter 3: iOS Architecture .................................................................................................... 11

   iOS SDK ................................................................................................................................. 11

   iOS Technology Layer ......................................................................................................... 12

   The MVC design pattern ..................................................................................................... 13

Chapter 4: Concepts behind Game Acre ............................................................................... 18

   Motivation ............................................................................................................................ 18
Work Flow of the Application ................................................................. 19

The application has three main phases namely, game set up, game in action and end of

game. Each of these phases has been explained in the following sections. ............... 19

Game setup ........................................................................................................... 19

Game In action...................................................................................................... 21

Game Score calculation ....................................................................................... 24

Game design ......................................................................................................... 24

Chapter 5: Challenges ....................................................................................... 30

Challenge #1: Real-time pose estimation of view port is computationally demanding 30

Challenge #2: Geomagnetic sensor noise causes jitter and inaccurate positioning of

virtual objects ....................................................................................................... 32

Challenge #3: Delay in geolocation determination by GPS displaces the central point of

reference ................................................................................................................ 33

Challenge #4: Altitude problem ......................................................................... 35

Challenge #5: Synchronization problems arise if several timers are used to fire events

that share the same variables ............................................................................... 36

Challenge #6: Long periods of inactivity cause the phone to sleep ..................... 40

Challenge #7: Debugging and testing .................................................................. 41

Chapter 6: Architecture Framework ................................................................... 44

Chapter 8: Future work ....................................................................................... 48
Chapter 9: Conclusion

References
List of Figures

Figure 1: Simplified representation of a “virtuality continuum”. Source: 2009 Trendbird Report – Nokia’s Mobile Mixed Reality ......................................................... 2

Figure 2: Augmented Reality example. Source: 2011 Circuits Today Report- Augmented Reality Technology and Paris Metro application......................................................... 3

Figure 3: High level view of the iOS Technology Layer. Source: iOS Developer Library .............................................................................................................. 12

Figure 4: Model View Controller Architecture. Source: Designing Enterprise Application with J2EE platform, Second Edition................................................................. 15

Figure 5: Interface builder workflow .............................................................................. 16

Figure 6: UIKit framework. (Source [36]).................................................................. 17

Figure 7: Overview of the workflow .............................................................................. 19

Figure 8: Bird's-eye view of the game field created in the initial set up ......................... 20

Figure 9: Tracking logic............................................................................................... 21

Figure 10: Repositioning logic .................................................................................... 22

Figure 11: Core Location update ................................................................................. 23

Figure 12: Different moods of the nutrias depending upon the distance (D) ............ 28

Figure 13: Transition between in and out of bounds ................................................... 29

Figure 14: Closer Nutrias have lower height than the farther ones ............................ 36
Figure 15: changeLocations() and updateLocation() tries to access the same
ARCoordinates object when their respective timers fire ...................................................... 38

Figure 16: MapKit view of the device's current location ......................................................... 42

Figure 17: High-level view of the architecture framework .................................................... 45

Figure 18: Component based view of the architecture framework ........................................ 45

Figure 19: High-level system architecture for Cyberinfrastructure (Source [1]) ............... 50

Figure 20: Services within the Cyberinfrastructure architecture for application development (Source [5]) .......................................................... 51
Chapter 1: Introduction

What is Augmented Reality?

Augmented Reality (AR) is a technology that enhances the physical worldview by overlaying information on top of the objects of interest. The augmented information is computer generated sensory inputs, such as sound and graphics superimposed on top of the user’s display view. There are two formal definitions that are widely accepted today: Milgram and Kishino (1994) defined Milgram’s Reality-Virtuality Continuum to explain different forms of digitally enhanced worldview. They describe a continuum that spans from the real world to the absolute virtual world. According to them, augmented reality is a special kind of mixed reality where there is a presence of both the real world and the virtual objects, as opposed to total replacement of the surrounding with virtuality.
Azuma defines Augmented Reality as a technology that has the following three characteristics [14]:

- Combines virtual objects with the real world
- Sense and respond user’s motion/interaction in real time
- Registers virtual objects on top of the object of interest in the real world

There are several other definitions quoted in different papers, publications, theses and websites. But the basic idea behind augmented reality is self-explanatory. It is an extension of reality that is achieved through computing. Note that augmented reality is different from virtual reality. Virtual reality requires a person to be completely immersed in an artificial world where everything is synthetic, whereas augmented reality has a presence of reality where the points of interest are augmented with digital information.
Augmented reality applications built on smartphones are primarily of two types – pose computation and object recognition [13]. Pose computation AR requires only knowledge of the orientation of the camera and the direction in which it points. On obtaining the pose, all virtual elements that lie within the field of view are rendered over a camera image. On the other hand, object recognition AR overlays information only on recognizable objects (viewed by the user) that are recognized or identified by the computer. It uses pre-determined features or markers for tracking and image registration. Thus, an object recognition system is feasible only within constrained environments (such as indoors).

Focus of this work

This Master’s thesis focuses on Pose Computation AR coupled with a game component. The main objective is to address technical challenges imposed by the limited capabilities of a smartphone, such as the noisy reading of the accelerometer and
magnetometer, the delay in receiving GPS data and so on. More details on these challenges will be discussed in Chapter 5. In addition to addressing the challenges, the thesis architecture framework should allow easy customization of an application with AR as the reusable component and customizable part as the modifiability component.

Since this thesis study is based on the AR game application called Game Acre, the modifiability component in this case are the game modules. It consists of game design, rules and storyline. A sub goal of the thesis is to study the game design of AR games in order to enhance the playability.

Terms and definitions

*Marker based & Markerless Augmented Reality*

Marker based AR uses markers like QR (Quick Response) code and Semacode markers to perform real time processing of the image frames captured by the digital camera. It uses encoded information in the marker to recognize the marked objects and superimpose information. On the other hand, markerless AR does not use any special markers but rather it uses the natural features of the object to recognize it and augment it with information.

*Point of Interest (POI)*

Point of Interest are the objects of interest in the real world. For example, in a university campus tour application, all the locations of the buildings on campus will serve
as POIs. POIs are usually associated with some geographical information or embedded features in them so that they can be referenced for tracking or rendering of images.

**AR Browsers**

An AR browser is a browser for finding points of interest (POI). It uses a combination of GPS, accelerometer and magnetometer to locate the user’s position, orientation and heading. After determining the pose of the device, it retrieves information from a data repository corresponding to the location of the device. Finally, digital information in the form of audio, video or graphic image is presented onscreen.

**GeoTagging & GeoLocation**

A Geotag is a spatial information associated with digital content such as photographs, video, etc. It is reference data used to extract information specific to a location. It is mostly used by applications where local data must be filtered based on the user’s current location. For example, a travel guide application retrieves only the POIs around the user’s current location. Geolocation is the geographic coordinate of an object in the real world.

**Tracking and Image registration**

Tracking is the process of identifying the displacement of an object of interest with respect to the user’s line of sight (through the camera). Image registration is the
process of alignment of the virtual object based on the information obtained from tracking.

Thesis organization

Chapter 2 talks about related work and research publications. This is followed by the description of the iOS SDK and explanation of the MVC architecture (Chapter 3). Chapter 4 describes the concepts behind the Game Acre application, its workflow and game design. What entails after that is a detailed discussion on the AR application development challenges (Chapter 5). Chapter 6 explains the architecture framework behind the Game Acre application. Chapter 7 presents future work and its implication to Cyberinfrastructure. This is followed by conclusion in Chapter 8.
Chapter 2: Related Work

Smartphones are getting more powerful with every new model that is rolled out into the market. They have evolved to work almost like a desktop computer [15]. The availability of sensors and hardware features like GPS, camera, magnetometer and accelerometer in smartphones have made it viable to deploy AR-based applications [16]. Although developers have been successful in deploying AR technology in smartphones, there are limitations imposed by the in-built hardware components. They cannot provide performance as robust as dedicated sensors in standalone sensor equipment.

In their paper, Gotow, Zienkiewicz, White and Schmidt have addressed the challenges with AR mobile application [3]. This work is mainly focused on providing lightweight algorithms or solutions to address computationally demanding AR technology such as image registration, sensor noise filtering and content retrieval from a database server. This thesis covers the further levels of development challenges. They include finding the optimal update frequency for tracking a user’s motion and working with a delayed GPS feed. Issues like thread synchronization and interruption of the camera view by the device going into sleep mode due to longtime inactivity are some of the other researched problems. Lastly, debugging and testing of a pose computation AR application is difficult because the test cases require actual physical movements to generate realistic test data.
A lot of research has been done in designing a common platform for building AR applications. Some research is targeted at building a framework for provide rapid content creation and consumption [1][4] while others try to unify different AR systems (e.g. marker based vs. markerless AR) [5]. Another aspect, which has also been a focal area of AR research, is providing a platform for the general public (non-programmers) to generate and share content [4][1]. In all this work, the approach adopted has been to identify the common components of an AR system and provide them in a baseline framework. For example, tracking and image overlay/registration are two of the common software components in AR application.

These common parts are the reusable components that can be composed in a software framework [5][17]. There are several AR authoring tools for developing AR desktop and mobile phone applications, which can serve as a starting point in application development. However, with each requirement additional code must be added to create a complete application [4]. ARToolKit [6] is an example of a low-level computer vision tracking library for marker-based AR applications, while Studierstube[7] is a high level programming library for AR.

Existing popular platforms available for developing pose computation AR applications are Layar, wikitude and Junaio [18]. Since they are GPS-based and have a web infrastructure, they are also called AR browsers. AR browsers operate in the following way:

- The browser first locates the user with the GPS and compass of phone
• Next, it retrieves information and graphics based on those geographical coordinates from online sources.

• The information and graphics are then overlaid onto the camera view of the smartphone’s screen.

The information such as POIs are hosted on a database web server.

**Layar** is an AR browser application, which also provides an open platform for users to publish, discover and search for augmented reality layers. It provides the infrastructure for third party developers to add “layers” i.e. their own content as a sub-application [20].

**Wikitude** provides two kinds of platform- ARML and Wikitude API.

• ARML is similar to Layar in that users can create “worlds” i.e. their own content or POIs.

• Wikitude API provides a framework for developers to create GPS based AR applications. What is different in this API is that applications can be built locally without relying on external server. It does not depend on the Wikitude infrastructure for content management and publishing. The API provides the underlying framework to locally execute the AR component, with the flexibility of using developer-created content.

**Junaio** provides facilities for a wider range of AR application development. It supports both marker-based and marker-less AR applications. The marker-less AR application supported are AR browser applications, similar to Layar and Wikitude’s ARML. In this platform the user-defined augmented information is called a “channel”, as opposed to “worlds” in Wikitude and “layers” in Layar.
Layar, Wikitude and Junaio are proprietary. In contrast, for the development of Game Acre application, we used an open source library called iPhone ARKit. Our purpose was not to simply add content (layers, channels or worlds) to an existing system but to allow customization of the behavior of the application. We achieved this by designing and implementing a customization layer for game development on top of the base AR framework.

Although the iPhone ARKit library does not support 3D image registration and dynamic update of frame of reference with respect to the user’s location update, direct manipulation (such as image size scaling with respect to distance) of the code is possible. Another advantage of using the open source library was that it helped in understanding the technical implementations of AR technology, as a result of which practical challenges and issues faced in the development phase of an AR application, could be addressed and studied. A more detailed discussion on challenges and architecture are covered in Chapter 3, Chapter 5 and Chapter 6.
iOS SDK

The iOS SDK is a software development kit developed by Apple Inc. to allow third-party developers to create native applications for iOS. An integrated development environment (IDE) called XCode is provided, consisting of a complete set of development tools that includes the XCode IDE, an iOS simulator, Instruments (performance tool), interface builder and more.

Most of the frameworks provided by iOS consist of technologies built in Objective-C. Hence, learning Objective-C is one of the essential requirements. Since Objective-C is a superset of ANSI C, one can freely intermix C code with Objective-C code.
The infrastructure provided at each layer is in the form of frameworks, which makes it easy for developers to choose or include a framework according to the type of application they are building. The most important of all frameworks are the Foundation and the UIKit framework, which is provided at the **Cocoa touch** layer (UIKit.framework and Foundation.framework). As described on Apple’s website [25], the Foundation framework provides object-oriented support for collections, file management, network operations, and more. The UIKit framework provides components for implementing the user interface of the application by including classes for windows, views, controls, and the controllers that manage those objects. This layer also provides other components to
access native applications like contacts, and hardware features such as the accelerometer, the compass and so on.

The next layer in the hierarchy is **Media** provides multimedia components. It supports 2D, 3D graphics, video and sound. It encapsulates C-based graphics libraries like OpenGL ES, Quartz, and Core Audio. Hence, it is an advantage for developers who are already familiar with computer graphics and multimedia. This layer also provides an animation engine called Core Animation which is Objective C based.

The **Core Services** layer provides the fundamental system services that all applications use. This include networking, the embedded SQLite database, Geolocation and Threads. An application may not be using these technologies directly; however every other component in the system is built on top of them.

The **Core OS** layer forms the bottom-most layer of the iOS architecture. It encompasses the OS kernel environment, drivers, and basic interfaces of the operating system. The kernel is a variant of Mach kernel found in Mac OS X. Together with the Core Service layer, it provides the fundamental capability of the system. It manages the virtual memory system, threads, file system, network, and inter-process communication.

The MVC design pattern

iPhone programming is based on the Model View Controller (MVC) design pattern. The architecture of Cocoa Touch is built around the MVC paradigm. The separation of different components (model, view and controller) makes it easy to design frameworks
that have reusable components as well as the extensibility to create new or enhanced applications.

The **Model** layer is responsible for the application data and the logic for data manipulation. The objects in this layer should be organized in such a way that they are well planned in terms of classes, interfaces and member variables. It is the component that is mostly going to be reused or extended for new requirements. It is also responsible for protecting the data from external manipulation.

The **View** layer is the presentation layer of application data. It is the interface through which users interact with the application. This layer takes the form of views, windows and controls. The views object must also generate notifications based on user’s interaction with the UI that is handled by the Controller.

The **Controller** layer acts as a bridge between the View and the Model layer. It handles the notifications sent by the Views and trigger the appropriate action or method calls in the Model. In the same way, if there is any change in the data i.e. Model layer, the corresponding controller object will be notified causing it to update the views [22].
Figure 4: Model View Controller Architecture. Source: Designing Enterprise Application with J2EE platform, Second Edition.

XCode provides many templates with ready-to-use views and view controllers. The core classes are UIViewController and UIImageView where UIImageView is used for adding UI elements to the views either programmatically or through the Interface Builder. The UIViewController is subclassed to add custom functionality.
IBOutlet provides a way to manipulate an UI element both programmatically and through the Interface Builder (GUI). IBAction allows action definition of events triggered by the UI elements (see Figure 5).
Figure 6: UIKit framework. (Source [36])

Figure 6 shows the hierarchy of classes available in the UIKit framework. There are some UIViewController classes which work in conjunction with the UIView. For example UITabBarController object is responsible for managing activities triggered by the UITabBar object [36].
Chapter 4: Concepts behind Game Acre

Motivation

Game Acre is an application built for Songhwale LLC, Pittsburgh. The game application is loosely based on a scavenger hunt. The motive of the application is to use the game as a means of brand promotion. The plan is to have virtual elements populated that are specific to a particular brand with the prize of the winner being some merchandise of the brand. For example, if the brand is Pizza Hut then the virtual elements populated could be pizzas, sodas or the Pizza Hut logo floating in the air around the physical elements in the scene with the game rule as “grab 10 sodas and 10 pizzas to win a free meal at Pizza Hut”. Using this game, the application owner can promote different product of a brand and reach out to its customers.

As a developer, the goal is to create a framework where the same application can be customized for different brands with the similar underlying concept. For this thesis study, a prototype was built with a game logic and rules. The virtual elements populated are nutrias and the rule of the game is to grab all the four nutrias within a time of 5-7 minutes. In the game, the nutrias are pests that have been let loose and must be hunted down by the player. The nutrias are deemed as “captured” if the spherical coordinates of the player and the geo-tagged nutrias coincide. This means the player has to physically walk up to the nutrias and “grab” them.
Work Flow of the Application

The application has three main phases namely, game setup, game in action and end of game. Each of these phases has been explained in the following sections.

Figure 7: Overview of the workflow

Game setup
On the launch of the application, the user’s current location is initialized as the center of the game field. Four random geo coordinates are generated for the four nutrias. The geo coordinates must be within the radius of 100 m and no less than 20 m. A countdown timer is initialized to 5 minutes within which the nutrias hunter must catch at least one nutrias or the game will be over.

Since the nutrias are populated at randomly generated coordinates, it may so happen that some of the geo-locations are physically unreachable by the user. In order to overcome this problem we have set another timer for repositioning the nutrias (see Figure 10). A third timer is used which is the main driver of the core application. It is responsible for tracking and updating the user’s motion, location and direction (see Figure 9). The check for the updates is carried out at the frequency of 1/20 seconds so that every small displacement of the user’s screen/phone is captured and image placement or removal on the viewport is smooth.

Figure 8: Bird's-eye view of the game field created in the initial set up
**Game In action**

After the game is set up, the camera is turned on and the user must locate the nutrias by sweeping his phone across the room (around him). As and when there is an update on the GPS location, heading (magnetometer) and accelerometer, the viewport of the camera is checked to see which nutria has come within (or gone out) of the field of view. In other words, some nutrias will be overlaid and some will be removed from the users view. The update location timer tracks this update and has been shown in Error! Reference source not found..

**Figure 9: Tracking logic**
The other timers simultaneously do their own jobs - repositioning of the nutrias and declaring the game over as and when their corresponding timers fire off. The logic for “repositioning” and “game over” has been explained in Figure 10 and Figure 11. The GPS (position and heading) and accelerometer have inbuilt delegates which notify the application if there is an update. During the course of the game whenever there is an update in the geo location, a check is carried out to see if the user’s location coincides with the nutrias’. If a collision is detected, the nutria is deemed as “captured” and a bonus of 1 min is added to the countdown timer (see Figure 11).

Figure 10: Repositioning logic
In addition to the above checks, there is also a check to see if the user is within the game field or not. If the user is detected to be out of bounds then an alert image will pop up warning the user to step back into the game field. At this point no nutrias will be visible, indicating that the game is suspended. As soon as the user walks back in, the alert image is removed and the game resumes from where it left off.

Figure 11: Core Location update
**Game Score calculation**

For the prototype, the score calculation is based on the amount of time left after all the nutrias have been captured. The user’s best three scores are saved by the application. This data is stored in a file so that if the user wants to check the score history before the start of the game, it can be retrieved from the file and displayed in the appropriate format on the score view.

**Game Design**

We have all grown up playing some kind of games or the other. “Games are a form of fun, that gives enjoyment and pleasure” [19]. With the growth of the computers from the first generation to the present time, computer games have evolved alongside from typing a command using a keyboard (as a means of interaction) to the waving of a hand using motion detection.

Augmented reality games can be thought of as a bridge between computer games and real world games. It is a new genre of games in which computer generated sensory inputs are used to lay a game set up in the physical world. In Weiguny’s thesis [8], he describes the new opportunity that AR has provided to create games that take the advantages of both computer games and off-screen classical games.

Augmented reality games are also called *Augmented games*. As described by Weiguny [8], augmented games follow the same design principles as that of the classical computer games but due to the added dimension of intermixing reality with virtuality, there are new aspects that must be considered. There are questions on how to combine the
elements of classical computer games and the physical world to produce a satisfying and entertaining experience of AR games. This thesis discusses a high level view of two design aspects of augmented games – Space, and Input & Feedback.

One of the key elements of an Augmented Game is the game space. It is therefore important to nail down the space requirement for the game. The play field could be restricted based on the AR technology used (for example, if stationary cameras are used for motion detection then the player must be within the field of visibility of the system setup). Or, the game space could be restricted due to the nature of the game as in case of GPS-based applications e.g. the game space in Game Acre has to be outdoors, where GPS can be easily fetched.

Input and Feedback through an interface is another important aspect of game design in augmented games. The interface through which the user interacts can help him better relate to the game world. For instance, in a war game, the player can have a better experience if the interface used to fight is a gun shaped console. Once again, [8] suggests that an interface has the capability to affects the player’s emotional experiences. Therefore, it is important to analyze the available interfaces of a system.

The primary inputs through interfaces are motion, sound and haptics (touch sensitive interface elements such as a button on a touch screen). In response the system gives feedback in the form of visual displays, audio and vibrations. AR games on smartphones use all the hardware features encompassed on the phone. They do not use any head mounted displays or other types of standalone consoles. In other words, they only use their inbuilt GPS, magnetometer, accelerometer, camera and the touch screen for
inputs and feedback. The GPS, magnetometer and accelerometer are used to calculate the position, orientation and heading. The camera provides the video inputs for object recognition in case of marker based AR and uses the physical world as a background for image overlay. The touch screen acts as a display monitor and an interface for haptic inputs.

Game design on smartphone applications can also be approached from a perspective of user-centered considerations. This perspective takes into account the challenges faced by the limitations of the device and the technology. Some of the design challenges are [9] lag between the tracking and rendering of images, low display resolutions causing difficulty in object recognition, field of view restricted by the phone’s interface screen and tracking noise from sensors such as magnetometer.

In Game Acre, the game field is set to be a circle of 100m radius. Since the game is GPS based and is an outdoor game, the whole world is its game field. However, a large game field is not desired since the goal of the game is to grab nutrias. If the game field were too big, the player might feel that it is too much work to catch the all the nutrias and not finish the game. Also, in the pursuit of nutrias, the user might wander too far away.

One of the fun elements of Game Acre was the ‘hot and cold’ concept. If the player was close to a nutria, the image of the nutrias would change to one that showed fright. When it was finally captured, the event was indicated by an image of captured nutrias that was displayed with vibration and audio. All these elements added to the fun experience of the game. The current prototype still has some design flaws; for example the user could not tell from the game that it must be played outdoors. The distance label
on the nutrias did clue to the player that he should walk up to the nutrias to capture them. Most users began under the impression that the nutrias could be made to act in some manner by simply touching the screen.

There are two reasons why the game did not seem intuitive. One is that most AR applications that exist today only have touch screen interactions. For example, in Layar [20], on *touching* the POI (virtual object) on the screen, detailed information would show about that corresponding object. The second reason is really that most users are still unfamiliar with the workings of AR applications.

Just like any other game, one of the most essential criteria of *Game Acre* is to ensure or enhance playability. Weilguny[8] describes playability as “the experience that players have and the way they feel about the game. Playability includes parameters that are hard to define like fun, challenge, satisfaction, enjoyment, achievement and motivation that the player feels.” Learning from the feedback we obtained through the user’s experience, we intend to revamp the user interface of Game Acre. We will be providing more directions during game play, such as navigational arrows, radar displays that show where the nutrias are, and pop up messages. Also, a quick tutorial will be included at the start of the game which will also present the storyline and the rules. Another change includes the placement of nutrias. Currently, the nutrias float in the air, which makes the significance of their supposed distance less intuitive. Finally, as mentioned earlier, the purpose of Game Acre is to use it for brand promotion. Therefore we hope that the reward of a free gift or merchandise would add to the playability of the game.
Given below is the description of a few **Game scenarios**:

‘Hot and cold’ concept:

D = distance of the user from the Nutrias

Figure 12: Different moods of the nutrias depending upon the distance (D)
As the user gets closer to the Nutrias, the image of the Nutrias transitions from a ‘calm’ state to a ‘concerned’ state and finally the ‘captured’ state.

Acre Alert

\[ D = \text{distance from the initial starting point to the user’s current location} \]

Figure 13: Transition between in and out of bounds

If the user steps out of the 100m boundary game area then an alert image will pop up. At this point no Nutrias will be visible. The application will remain in this state as long as the user is out of bounds.
Chapter 5: Challenges

As mentioned earlier, one of the main aims of this thesis is addressing the challenges faced during the development of the Game Acre application. The challenges are described below along with a detailed explanation of how we addressed them.

Challenge #1: Real-time pose estimation of view port is computationally demanding

Augmented reality is all about displaying information at the right time and at the right place. This means that with the change of position, orientation and heading, image transformation and/or rendering must be reflected on the camera preview almost instantly (i.e. in real time). Therefore, computation of the view port must be done almost every fraction of a second in order that the field of view stays synchronized with the phone’s orientation. Note, however, in reality, the sensory inputs from the phone’s accelerometer and magnetometer are constantly changing by small values even if the user is not moving physically. This is because, it is almost impossible for the user to hold a phone (or any object) completely still. In addition, noise inputs from the surrounding environment introduce errors in the readings of the sensors. Thus, even as the computation must be carried out frequently, care must be taken to filter out sensor noise.

Tracking for updates in the position, orientation and heading must be quick enough for the user to not be able to detect jitter in the images overlaid on the view from the
camera. The question is how often must system check for updates? For example, would there be a problem if the system updates the camera view (frame) at a very high frequency? What would be the optimal frequency rate to avoid wastage of CPU cycles and energy? To sum up, we need to find the optimal value of the calibration frequency such that the real time image placement and removal will appear smooth, minimize potential errors, while conserving computation.

Certainly, the upper bound of the view port recalculation rate is the frame rate (frames per second - fps) of the phone’s camera. By setting the update checks to any frequency higher than that is a clear wastage of CPU cycles because any changes will not be reflected on the camera view as it operates at a lower fps than the assigned value.

The camera on iPhone 4G has a frequency of 30 fps. Hence the upper bound of the update frequency in Game Acre is 1/30 sec. On having set the upper bound, the optimal frequency was then obtained through an experimental study. Two users were asked to play Game Acre set at different frequency levels. The frequency levels ranged from 1 sec to 1/30 sec. The users were asked to select the minimum comfortable level at which they experienced no jitter on the camera view.

The result obtained through the experiment was 1/20 sec. This period of time relates to the perception of motion where the brain needs to comprehend reality rather than just the persistence of an image. Wertheimer in 1912 explained the false association of motion perception and persistence of vision [34]. Human vision is not exactly the same, as light passing through a lens in the camera. Visual data obtained through the eye
requires further processing by the brain to comprehend reality. By finding the optimal frequency at which the frame of reference is computed, CPU cycles are not wasted.

**Challenge #2: Geomagnetic sensor noise causes jitter and inaccurate positioning of virtual objects**

The magnetic sensor (compass) provided by iPhone works similar to the magnetic needle in a compass. The accuracy of this sensor can be affected by stray magnetic fields in the surrounding environment. The noisy inputs could come from nearby computer and television monitors, other electronics components and so on [27]. Sensor readings can even be affected by the interference from things as small as the magnets in the earbuds in the phone. There are three most common sources of magnetic fields: permanent ferromagnets (for example, those in speakers and buzzers), induced fields within ferromagnetic materials (such as sheet steel) and electromagnetic fields (essentially generated by electric circuits) [2].

In a smart phone, the magnetometer measures the earth’s ambient magnetic field. The above mentioned magnetic interference could cause severe problems for AR applications or any application that relies on detecting orientation using the magnetometer. The lowest value of horizontal magnetic field strength to be detected by a smartphone compass is 10 μT (in northern Canada and in Russia) [2]. Thus, for an accuracy of 0.05 radians or 3 degrees in compass heading readings, the tolerable error in estimating geomagnetic field is no more than 0.5 μT. Although the calibration software running within any e-compass is able to remove interference to achieve accuracy better than 0.5 μT, it is still not error-
free enough to provide accurate readings. The general algorithms used to remove noise interference are pattern recognition and smoothing filters such as Savitzky-Golay [35]. Since a noise filter algorithm uses regression or statistical methods, they are computationally intensive. Especially when they are required to be executed for every heading update, they are computationally infeasible for a smartphone. A lightweight and portable algorithm has been proposed in the paper [3]. It is an extension of Finite Impulse Response filters with added statistical analysis for data exclusion and outlier analysis. The algorithm uses statistical methods that are computationally expensive during the initialization phase. However, after that the calculations may be optimized to run in constant time by keeping track of only the current sum and variation.

That said, for the Game Acre application, we did not need to map nutrias onto a particular object in real space. Hence, the error rate is tolerable. However, for future extensions of the application that require greater precision, we may consider the lightweight algorithm mentioned above.

*Challenge #3: Delay in geolocation determination by GPS displaces the central point of reference*

GPS satellites continuously transmit information (its location and current time) to the earth surface. The GPS receiver on obtaining information from different satellites performs some calibration to determine the user’s location. As a result of this process, GPS devices normally take about a minute or two to get the location from the satellites [28]. Thus, any application that depends on GPS must deal with an initialization delay
before the correct location of the user is found. Note that an extended version of GPS called assisted-GPS (AGPS) finds the user’s approximate location using cell towers and wi-fi networks [33], and thus tries to minimize delay at the expense of accuracy.

For GPS based iPhone applications the Core Location framework is used. The framework allows us to determine the device’s current location and heading. Although iPhone uses AGPS it still take some time before the correct location is determined. Thus, the first call made by the locationManager (delegate) always returns (0,0). In order to overcome this limitation, the call to obtain the geolocation should be made in advance instead of waiting until the need arises. The workaround against such a problem is application specific. If the application requires the geo co-ordinates as soon as it begins then the start up time can be delayed by using an activity monitor. Else, (0,0) could be used for a brief moment and then updated as and when the didUpdateLocation() function is called.

In the Game Acre application, the call to get the user’s location was made in advance (at the start of the application). This solution was applicable because the spherical coordinates were required only for the second view (the camera view) in the application (the first view of the application is the home page so we took advantage of initiating the GPS call during the display of that page). Having the correct location at the launch of the camera view was important for the Game Acre application because the user’s spherical coordinates were used as the origin around which the nutria were populated (with their actual coordinates being selected randomly). The above solution prevented erroneous out of bounds (i.e. outside game field) reports, as well as application jitter.
**Challenge #4: Altitude problem**

The GPS takes even longer to return the correct altitude value. As a result of this the frame of reference may be disrupted. As explained earlier, the iPhone uses three ways to find the user’s location: local wi-fi signals, cell towers and satellites. However, the altitude value may only be obtained from the GPS satellite. If there is a strong signal reception then the altitude value may be returned quickly but if the signal is weak the phone may not find the value until the satellite signal has been received several times.

Again, the solution approach to this problem is application specific. If the actual altitude value is required to display the augmented information then the application can use the most recent altitude value returned by the location manager. Else, we could use the same approach as mentioned in challenge #3 where we try to obtain the GPS data early on. On the other hand, if the altitude is required merely for pose estimation of the device (smartphone) then the application can use zero as the datum and add to each virtual object a height that is proportional to its distance from user’s current location. This is the solution used in Game Acre, and described below:

```plaintext
double base = centerLocation.altitude; // for Game Acre base = 0
double newAltitude = (base - 20) + 
([item.geoLocation distanceFromLocation: centerLocation] / 2.5);
```

The formula used above was derived by adjusting the datum (base) and the distance-height relationship. The reason why we chose distance-height relationship was to achieve
an elevated view of virtual objects so that complete overlapping of object (images) would be avoided. The described effect is shown in Figure 14.

Figure 14: Closer Nutrias have lower height than the farther ones

Challenge #5: Synchronization problems arise if several timers are used to fire events that share the same variables

Timers are generally used to trigger events that do not involve the user’s interaction. A timer waits until a certain interval has elapsed and fires an event. In game applications, there are generally several actions that must occur at different times. For example, in a game, an image can be set to fade after every 5 minutes and an alert message to pop up every 7 minutes. A separate timer must be used for each periodic event. Each timer executes on its own thread; hence when several timers are used, multiple threads will be forked.
The use of several timers often leads to synchronization issues. If the course of actions triggered by several timers manipulates the same resources/variables then it may give rise to race conditions. Two threads modifying the same resource can affect each other in unintended ways [29]. For example, the changes made by one thread could be overwritten by another thread. Such interference may be hard to track and debug. Thus, if the application crashes or causes performance impedance due to a corrupted resource, the root issue would be challenging to detect. Moreover, in some cases, errors may be generated that do not manifest themselves until much later. Further, addressing the errors post facto usually require a significant overhaul of the underlying coding assumptions.

As with any synchronization problem, the solution is to identify critical sections in the code and use in-built or user-defined synchronization functions to prevent inconsistencies. In addition to that it is important to remember to check the values of the variables (for example to ensure that the variables are not null) and their states (for example, released) before performing any operations on them. Once a thread releases a lock, another thread waiting in the queue may gain access to a shared variable/resource that does not exist anymore. This could cause the application to crash. Above all, the most important step in assuring thread safety is having a good program design that minimizes the number of shared resources and their interaction.

In the guidelines for thread safe design provided by Apple, we are asked to find the right balance between safety and performance. For example, the code could be designed to avoid threads altogether [30], or by having each task manipulate only its own local copy of data so that synchronization would not be necessary. Note that using locks and
other synchronization techniques can also impede performance. If there is a high contention among the threads for a specific resource then the threads may end up waiting for a long time before they can perform any useful actions. On the other hand, creating local copies of data has storage overhead as well as consistency issues.

In the Game Acre application, three timers were used. Each time a timer is fired, the corresponding thread manipulated the shared variables holding the virtual object’s location and the image overlays. Given the iPhone framework, the NSTimer class is used to create timer objects. The iOS sdk provides in-built synchronization functions, which enabled the protection of the critical section (CS) of code.

Given below is an illustration of one such scenario where two threads try to access the same object:

Figure 15: changeLocations() and updateLocation() tries to access the same ARCoordinates object when their respective timers fire
/*  * Repositioning  * Move all the elements in ar_coordinates.  */  
-(void)changeLocations:(NSTimer *)timer {
@synchronized(self) {
   NSLog(@"Moving Objects!");
   if (!ar_coordinateViews || ar_coordinateViews.count == 0 ||
       ar_coordinates.count == 0) {
       return;
   }
   NSUInteger size = [ar_coordinates count];
   [ar_coordinates removeAllObjects];
   for (NSUInteger i = 0; i < size; i++) {
      CLLocation *tempLocation;
      ARGGeoCoordinate *tempCoordinate;
      tempLocation = [GameSetUp getRandomLocation:centerLocation];
      tempCoordinate = [ARGGeoCoordinate
                          coordinateWithLocation:tempLocation];
      tempCoordinate.title = @"Neil Ave";
      [ar_coordinates addObject:tempCoordinate];
   }
}
}

//Function to capture the device's every movement  
-(void)updateLocations:(NSTimer *)timer {
@synchronized(self) {
   //"prevent" the phone from locking when app is running
   UIApplication sharedApplication].idleTimerDisabled = YES;
   //update locations!
   if (!ar_coordinateViews || ar_coordinateViews.count == 0 ||
       ar_coordinates.count == 0) {
       return;
   }
   }
As we can see from Figure 15: changeLocations() and updateLocation() tries to access the same ARCoordinates object when their respective timers fire simultaneously. In the code, we have used the in-built synchronization function to manage concurrent access of the object. We have also provided a check to see if the shared object(s) still exists before performing any kind of manipulation.

Challenge #6: Long periods of inactivity cause the phone to sleep

The camera is essential for the proper functioning of augmented reality application. By default, when the digital cameras are held open for a long time, the device automatically goes into sleep mode. Before the smartphone goes into sleep mode and auto-locks the screen, it takes a snapshot of the state of all the running application, so when the user unlocks his or her phone the same application resumes from where it left off. For some applications, this default feature may be bothersome but for others it may not be.

Game Acre is an application in which time plays an important role. The game rule requires the player to catch all the nutrias as fast as they can. The default nature of auto-locking is not desired here as it could frustrate the player who is close to winning a game, or worse, lead to potential misuse of the system by the player. The player could
intentionally run to the approximate location of the next nutria to be grabbed while the phone is in sleep mode thus saving some time and falsely gaining a higher score. Therefore, to avoid these complications, the Game Acre application forced the device to remain ON throughout the lifetime of the game.

**Challenge #7: Debugging and testing**

The testing and debugging of applications that use the GPS and the CoreLocation framework can be quite a challenge because such applications are not able to test code that handles location updates unless the application is in actual use, and the user is moving around. The iPhone simulator is not of much use because it does not provide support for hardware features like GPS, accelerometer, magnetometer and camera. For debugging, applications can be deployed on the phone directly in the debug mode but it is still a problem because the debugger tool requires the phone to be plugged into the computer at all times.

In order to overcome this hurdle, we need a library/module that will simulate fake GPS inputs to the application without any actual physical movement. There are two libraries that are available as plug-ins for iPhone, namely, FTLocation Simulator and iSimulator.

FTLocationSimulator is an open source project hosted online on GitHub ([https://github.com](https://github.com)). It may be used to simulate core location updates in the iPhone simulator. Essentially, customized core location feeds can be fed to the application. These artificial feeds can be prepared by generating a KML file that defines a pre-defined route
using Google Earth [31]. Besides that it can also simulate the blue dot\(^1\) on the Google map, which corresponds to the GPS current location.

![MapKit view of the device's current location](image)

**Figure 16**: MapKit view of the device's current location

**iSimulate** is a combination of an iPhone application and a library that is added to a development project. The application must first be installed on the phone, which acts as the agent for sending core location updates to the application for testing on the iPhone simulator. This method makes it possible for the simulator to support multi-touch events, accelerometer data, as well as GPS location information.

In the case of Game Acre, neither of the mentioned custom simulators was used. Since the prototype that was developed had only simple features with the simple game rule of grabbing nutrias, fake core location updates were created locally.

\(^1\)The blue dot indicates to the user its current location on the Google map. If the user moves, the blue dot also appears to move on the map simultaneously.
Locations were simulated by reading the coordinate values of the nutrias every 60 seconds. In the future, if there is a need to simulate more realistic GPS feeds, FTLocationSimulator will be used, since it provides the flexibility of creating a custom KML file. Although iSimulator is capable of providing fake GPS feeds, it will still be inappropriate for Game Acre because of the need for the availability of CoreLocation data on the phone. For example, in Game Acre if we need to test the out of bounds check and see the alert pop up on the iPhone simulator, the developer must take the iPhone atleast 100 m away from his desktop. This means that the developer himself cannot see the alert pop up on the desktop when he is out of bounds. Hence, using iSimulator, the phone cannot be at a distance further than a few feet away in order to be able to see the changes/updates reflected on the iPhone simulator and the XCode debugger.
One of the main goals of building Game Acre is to use it for brand promotion or marketing campaign. The idea is to help small businesses and companies in digital advertising via an augmented reality game. Therefore, in order to enable the framework to achieve the above goal, the software architecture should be designed in such a way that it will allow easy customization (modifiability).

The architecture framework that has been designed for Game Acre consists of two main layers – AR framework and game layer (see Figure 17). The AR framework is made up of the fundamental components (tracking and image registration) that are required for developing any AR application whereas the game component encompasses the game logic and custom graphics. For customization, requirements are gathered from the clients/businesses before developers proceed to the next phase. The product for the final application is developed after incorporating the custom requirements to the AR framework.
Figure 17: High-level view of the architecture framework

Figure 18: Component based view of the architecture framework
As mentioned earlier, the AR library used for development of Game Acre is an open source project called iPhone ARKit. It has of three main modules- AR object, AR view controller and Data (see Figure 18).

- AR object is used for defining the attributes of a custom virtual object. For Game Acre, the attributes include geo coordinates (Latitude, Longitude), distance label (for displaying distance between the user and nutria), inclination and azimuth. It is basic information used by the tracking system to render the images on the camera view.

- AR view controller holds the main logic for pose computation (tracking) and image rendering.

- Data module serves as a repository for the graphics and the coordinate values of the virtual object.

The game layer encompasses the game component of the application. It consists of the game set up, game-in-action, and game score modules. All the three modules have been explained in Chapter 4. As of now, we have built the architecture framework with a small user base. With this simple framework, a string constant file is used to reflect changes in the initial game set up and the graphics images in the game layer. However, for the new game rules, new functions/methods must be written as “glue logic” to connect different components.

The inbuilt MVC architecture for iPhone programming has made it easier to design our architecture framework that supports modifiability. The view component allows easy addition of new views or modification of the existing ones. Most of the UI elements that
our custom applications will have will be the same. If customization involves a change in the game logic then additional functions will be added in the model component. And the view controller will perform the linking of the view and the model.

Currently, the framework design supports the development of local applications. In the future version it will be extended to server based system where content management and services will be hosted on the Internet. By generalizing the framework, we want it to be able to serve as a Cyberinfrastructure for wide variety of AR applications and be of use to all kinds of users. A more detailed discussion on Cyberinfrastructure will be covered in Chapter 7.
Chapter 7: Future work

Implications for Cyberinfrastructures

Augmented reality has been a long-term area of research. Its applications were mainly confined to the laboratories because of the need for expensive custom hardware. However, in recent times, commercial, commodity devices such as smartphones have become viable AR application deployment platforms. These devices have inbuilt hardware features such as a camera, GPS, accelerometer and a magnetometer. The consequent commercial potential of AR technology has opened up new opportunities for developers to create complex systems that integrate digital information with the environment. Schmalstieg and Reitmayr [12] described in their paper the new concept of a semantic world model for augmented reality derived from direct analysis of the infrastructure required for ubiquitous computing. In this model, future work may be targeted at building a knowledge environment based on cyberinfrastructures [10] (also called the Cyberinfrastructure Information Ether) where the future internet will not be confined to desktops and laptops but will use “smart” objects spread out in the physical environment such as on roads, home, office, playground, etc. As proposed by Mark Weiser [21], these smart objects could take the form of pads, tabs and boards. They could also be contemporary devices such as mobile phones, digital audio players, radio
frequency identification tags, GPS and interactive whiteboards that will communicate with one another.

The vision of a Cyberinfrastructure is captured in the Atkins report [10][11] and stated as “Applications are enabled and supported by the Cyberinfrastructure, which incorporates a set of equipment, facilities, tools, software, and services that support a range of applications. Cyberinfrastructure makes applications dramatically easier to develop and deploy…. The Cyberinfrastructure also increases efficiency and quality and reliability by capturing commonalities among application needs, and facilitates the efficient sharing of equipment and services.“ Following this view, we present a Cyberinfrastructure for augmented reality applications where different kinds of users can participate in content creation, sharing, publishing and consumption, and AR application development.

In the research done at the Nokia Research Center for Mixed Reality Solutions [1], Belimpasakis, You and Selonen proposed an infrastructure for rapid content creation for all kinds of users including developers and non-developers (normal users). The framework design discussed in the paper [5] is based on flexible and modular software components that can be reused between different kinds of AR applications. By combining the two ideas mentioned above, we have derived a cyberinfrastructure framework that enables rapid content creation as well as provides services for creating AR applications.
The cyberinfrastructure shown in Figure 19 is responsible for managing various AR content datasets. The datasets include both user generated content and external content hosted by third party servers. The users are provided with web interfaces to communicate with the system for content creation, publishing, sharing and consumption. For normal users and small business owners they can simply link their web accounts hosted on different sources like Flicker, Twitter, Facebook, etc. to create AR content and share it with others. However, for developers, an application programming interface (API) is provided so that custom AR applications can be created.
As we say above, besides hosting data on the web, the cyberinfrastructure also supports services for development of different kinds of AR application. As shown in Figure 20 the framework is divided into four functional areas, namely, Services, Information, World and User. The Services component represents the external services (such as printing) that are not a part of the cyberinfrastructure. The World component describes the modeling of the physical space; for example, in a marker-less AR application, natural features of the real objects and optical tracking is used. The User
component defines how a user interacts with an AR system. And the Information component handles the way information will be accessed from the database server.

In the future, we plan to extend Game Acre to allow game creators to author their own game settings. In the current prototype, nutria positions are generated randomly in three-dimensional space, which requires nutria to be repositioned if they are found to be unreachable. We also plan to extend Game Acre to support multiple players. This requires a server to store game configurations as well as to carry out the synchronization of events across the players. For example, nutria caught by the players must be reflected on every player’s camera view in near real time.

Finally, this project has been part of a larger effort in developing engaging massive scale applications on distributed cyberinfrastructures. Lessons learned from the Game Acre implementation will feed into this larger research program.
Chapter 8: Conclusion

Every new technology comes with its own set of challenges and limitations. Augmented reality (AR) is no exception. It is easy to envision a world where digital information is available at any place and any time but to make it a reality, a complex system must be created. Since AR is still in its nascent stage, researchers are still working towards providing AR technology that can be used to develop light weight applications on compact commodity devices like phones, pads, and tablets. The challenges are not confined only to the imperfections of AR technology but also related to the limitations of these commodity computing devices.

An ideal augmented reality application is one where the world is used as an interface, all objects in the surrounding are recognizable in real time and virtual data or digital information is superimposed exactly on the object of interest. However on the practical side, there will always be some lag in the determination of spatial awareness, context sensitivity and registration of images. Unlike humans, most objects in the surrounding will be unrecognizable, and there will be high cost of system maintenance and content management. These are just a few challenges enlisted to illustrate the complexity of such a system.

In spite of imperfections, this technology is still pursued because of its promise in aligning with the intuitive nature in which humans would like to communicate with the

53
environment. It is easier for man to interact with the digital world using sight, sound and touch rather than having to wear gadgets or be confined to a limited field of vision. What we are moving towards is a future of seamless integration of virtual (digital) and real world, hence the pursuit will go on.
References


[22] https://www.datasprings.com/resources/articles-information/iphone-sdk-getting-started-example-code


determine-orientation/

gps-in-mobiles/


[34] Wertheimer M. Experimentelle Studien über das Sehen von Bewegung. Zeitschrift
für Psychologie 61, pp. 161–265. 1912

Least Squares Procedures". Analytical Chemistry, 1964