Empowering Our Warfighter: Using iPhones for Situational Awareness

Steven Borkman, Michael Hoffman
Dignitas Technologies
Orlando, FL
sborkman@dignitastechnologies.com,
mhoffman@dignitastechnologies.com

Gregory Peele, Jr.
Applied Research Associates
Orlando, FL
gpeele@ara.com

ABSTRACT

Meeting mission objectives while ensuring Warfighter safety is a difficult balance to strike. One way to meet this balance is by leveraging new technologies which can deliver the necessary data to make quick, life saving decisions. Today, mobile handheld devices, including smart phones, are abundant in our society and offer many features that can aid the Warfighter including GPS location, integrated maps, and augmented reality. However, for more advanced applications these devices need to be coupled with highly accurate terrain models which support services such as designating areas blocked by line of sight or accurately reporting changes in the environment.

Until recently, full terrain services were either too computationally/resource expensive to operate on handheld devices or operated on data too coarse to provide significant benefit to the frontline soldier. The Army's STTC has invested in the Layered Terrain Format (LTF), which is specifically designed to be a terrain simulation engine providing high fidelity terrain representation and services for devices with limited resources. LTF provides the necessary foundation to build and deploy situational awareness applications on mobile commercial hardware.

To prove our concept of providing detailed situational awareness on mobile platforms we selected both the iPhone and Android devices based on their price, capabilities, availability, and overall popularity. We leveraged, and further developed the LTF baseline to meet the needs of a mobile, high resolution situational awareness device.

In this paper we discuss the overall applicability of portable devices to the Warfighter, describe our solution to the problem, discuss the interesting quirks in developing for different mobile platforms, and describe the future capabilities that can be achieved with mobile devices supporting Situational Awareness, planning, and communications.

ABOUT THE AUTHORS

Steven Borkman is a Senior Software Engineer at Dignitas Technologies with over 11 years of experience developing software for the simulation community. Mr. Borkman currently serves as the project lead for the High Fidelity Runtime Database Engine and was the co-lead architect of the Layered Terrain Format. Mr. Borkman holds a Bachelor of Science degree in Computer Science from the University of Central Florida.

Michael Hoffman is a Software Engineer at Dignitas Technologies with over five years of experience. He holds a Master’s of Science degree from the University of Central Florida. He served as the lead developer of the Tactical Terrain Analysis iPhone application. Prior to HFRDE he spent many years as a developer on OneSAF, focusing on models and behaviors development. He also worked on many research topics including the DARPA Urban Challenge, where he provided a training environment for the robot.

Gregory Peele, Jr. is the Principal Investigator for the Layered Terrain Format at Applied Research Associates, Inc. He has five years of experience with terrain database production, geospatial information systems (GIS), modeling and simulation, and computational geometry, and served as one of the two original software architects of the Layered Terrain Format in 2007. Mr. Peele holds Bachelor of Science degrees in Mathematics and Computer Science from the University of Central Florida.

Empowering Our Warfighter: Using iPhones for Situational Awareness

Steven Borkman, Michael Hoffman
Dignitas Technologies
Orlando, FL
sborkman@dignitastechnologies.com, mhoffman@dignitastechnologies.com

Gregory Peele, Jr.
Applied Research Associates
Orlando, FL
gpeele@ara.com

INTRODUCTION

A clear and complete operational picture is critical to the success of our military forces. This picture serves as the basis for all key decisions made, from peacekeeper to combat unit. In order to meet mission objectives while at the same time ensuring their safety, it is imperative that they are utilizing equipment that will deliver the desired information both efficiently and effectively. While the need for new technology is apparent, the ability to quickly and economically produce the necessary tools has been challenging.

Today, mobile handheld devices, including smartphones, are abundant in our society and offer the Warfighter many beneficial capabilities, including GPS location, integrated maps, and augmented reality. The technology in these devices, in terms of both hardware and software, are advancing in a near-daily basis with no signs of slowing down. These advances are being driven by not only a competitive consumer market but very large, successful, and innovative companies (e.g., Apple, Google, and Microsoft).

A decade ago, the Army’s FBCB2 (Force XXI Battle Command Brigade and Below) represented a huge leap forward simply by providing vehicle GPS locations on electronic maps for M1 commanders. Now, UAVs and vehicle-mounted sensors provide constant streams of data, including high-resolution LIDAR/LADAR data. However, it is difficult to push such data forward to the front line Warfighters in a way that provides critical information without substantial data overload. DARPA's Urban Mapping capability provided one solution for this by providing a 3D visualization of such data, while TIGR focused on gathering key information based upon location, event, and associated data. Unfortunately, such data must be viewed or browsed in the context of desktop platforms.

These devices, however, are not mobile enough to aid the individual combatant. Nor do they provide the needed capability to generate derived terrain analysis results based on advanced geometric algorithms. In short, despite an abundance of complex and potentially useful geospatial data, the front-line Warfighter still has no practical way of translating the information into increased real-time situational awareness. To meet this objective, the device needs to be coupled with an accurate terrain model.

Until recently, full terrain services were either too computationally expensive to operate on handheld devices or operated on data too coarse to provide significant benefit to the frontline soldier. The Army's STTC has invested in the Layered Terrain Format (LTF), which was specifically designed to provide these terrain services on mobile devices. LTF provides the necessary foundation to build and deploy situational awareness applications on mobile devices.

LAYERED TERRAIN FORMAT OVERVIEW

LTF was originally developed to support the OneTESS simulation system’s requirement for geopairing on small man-portable devices. Geopairing is a novel concept for live force-on-force training, replacing the “laser tag” model found in other live training systems (MILES) with an electronic bullet (e-bullet) fly-out in a correlated virtual environment. As a result, geopairing requires more from terrain environments than the typical use case for Modeling and Simulation (M&S) terrain representations: extremely high-resolution three-dimensional terrain and feature representation that correlate as closely as possible with the real world. To add further complexity, the e-bullet fly-outs are calculated on mobile devices worn by the trainees, with limited resources and processing power compared to desktop workstations.

Key Design Principles of LTF

LTF was designed to meet these unique requirements. It was designed to be composable, light-weight, accurate, dynamic, and efficient. The follow provides a brief overview of each of the architectural principles.

Composable Layered Solution
LTF is built on modular layers of data. Each layer models a specific component of the terrain. By storing each component in a separate layer, it allows optimized algorithms and data structures to be developed for each unique environment feature. For example, the terrain skin (the ground) and terrain features (buildings, trees, street signs, etc.) are fundamentally different data. It is difficult to imagine an optimal storage technique or traversal algorithm that could handle both disparate data types.

Perhaps the most important benefit of the layered architecture is composability. Data layers can be loaded dynamically and allow for extremely flexible models which, at runtime, can be augmented to meet the user's simulation needs. On a system with limited resources (e.g. mobile device), LTF can be composed with only the essential components, as a result saving valuable resources.

Dynamic Environment
In the real world, the environment around us changes frequently, either by natural (earthquake) or man-made (building demolished) forces. In the combat zone, this is especially true. A virtual environment needs to react to these changes to be able to correlate with the real world. LTF is a fully dynamic environment and has the capability to alter its contents to reflect changes in the environment.

Accurate
LTF is a high resolution terrain system which has to correlate with the real world to the extent possible. Unfortunately, any terrain model is only as good as its source input. Because of this, LTF was designed to work with innovative data collection techniques such as LIDAR. LTF natively operates on 1-meter spaced grids, with elevation values within decimeter accuracy to the real world.

LTF Content
LTF currently consists of the Terrain Manager (the core management layer and interface), and several content layers: Terrain Skin, Terrain Volume Features, Ground Cover Features, and Feature Attribution. Each of these layers was designed to be efficient, lightweight, and accurate. LTF is still an actively developed product and the content layers will grow in the future as more types are supported.

In short, the compact nature, performance optimizations, and composable aspects of the LTF make it an ideal candidate for mobile applications. LTF has proven its ability to meet the performance requirements of real-time operations on hardware devices with much less computing power than current generation smartphones. This section was meant to serve as a brief introduction to the LTF format, for a more detailed description of it, please see the paper "An Optimized Synthetic Environment Representation Developed for OneTESS Live Training" (Borkman, Peele, and Campbell, 2007).

MOBILE DEVICES

Mobile Device Background

In the past few years, there has been an explosion in the mobile smartphone marketplace. The consumer cellular phone market evolved in 2002 when RIM introduced the BlackBerry. The BlackBerry, which was optimized for wireless email access, excelled in the corporate world, but did not make a major dent in the consumer marketplace. The consumer marketplace changed, however, in 2007 when Apple introduced the iPhone.

The iPhone, in conjunction with iTunes and the App Store, revolutionized the communication world. The iPhone not only incorporated common cellular phone capabilities (phone calls, text messaging, and voice mail) but also a mobile media player (music and videos) and internet capabilities (web browsing and email). The hardware included a multi-touch screen, GPS, camera, and compass. The App Store contains over 200,000 third party applications developed specifically for the iPhone of which users worldwide have downloaded over 2 billion apps (Chen, 2009). The apps available from the App Store run the gamut from productive to entertaining. One thing was clear; the iPhone was a major revolution in the mobile computing industry and set the trend for future technologies.

In 2008, Google also entered the smartphone market. Google differentiated itself from the other smartphone competitors by building only the operating system, called Android, and not the hardware. Google made Android available for hardware manufacturers, and by the end of 2009, there were 18 different phone models using Android worldwide (Technologizer, 2010). The Android operating system supported many advanced capabilities including: media playing, GPS, accelerometers, multi-touch, Bluetooth, and multitasking. One thing that clearly separated Android from the iPhone was that it was truly open, meaning that developers have access to all core capabilities of the phone and deliver applications without going through a centralized application store.

The market for smartphones continues to grow with over 42.7 million people in the U.S. owning smartphones as of January 2010, according to comScore, Inc., an industry leader in measuring the digital world (comScore, 2010). As the market increases so too does the number of users that can now develop and download mobile applications at will. This simple, yet innovative concept is driving sales of phones, thereby causing multiple manufactures and distributors to change their business strategies towards increasing the amount of available applications. As each new generation of smartphones is produced the choices for commercial off the shelf (COTS) products that can be utilized for government use is increased. By acquiring COTS products, the military can circumvent research and development efforts, thereby saving money. Furthermore, the use of COTS products often reduces acquisition times, thereby yielding faster integration into the field. The use of existing mobile devices gives soldiers the needed capabilities that historically were only available at the end of a long and complex future program.

The Tactical Terrain Analysis App

As a part of the Future Force Warrior project, LTF was successfully ported to a Nomad ruggedized PDA to provide terrain analysis and line-of-sight (LOS) information. Our experience of porting LTF code to an ARM based mobile platform, sparked interest in researching the ability to utilize smartphones for a similar capability.

Building on the lessons learned during the Future Force Warrior, project we successfully built the Tactical Terrain Analysis (TTA) app for both the iPhone 3GS (iPhone SDK 3.1) and HTC G1 (Android SDK 1.5). The TTA app provides situational awareness capabilities to the user by leveraging the capabilities of the device and LTF.

The main user interface for the application uses Google maps in order to show geo-typical reference information such as satellite imagery, road networks and other points of interest (see Figure 1, left image). The interface allows for all of the platform specific haptic interactions to control the map and navigate through features and options. On the iPhone this includes advanced capabilities such as finger swiping (to pan the map) and double-tap/multi-touch pinches (to zoom in and out of the map). At the time, the Android SDK did not support multi-touch, so buttons were added for navigation.

In order to use LTF, its source code had to be ported to both platforms (Android and iPhone). LTF is written in C++ which is different than either platform’s native development language. Because of this, there were issues involved in porting to these devices covered thoroughly in the Android/iPhone Development Comparison section.

After a database is selected and loaded in the app, an outline is overlaid on the map that highlights the areas where LTF information is available. All of the LTF reasoning services only function in the outlined area.

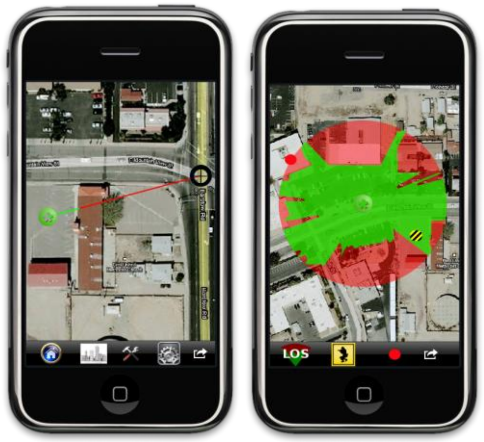

Figure 1. LOS capabilities of the iPhone app. The left image shows a point-to-point LOS query that is blocked by a building. The right view shows a 360 degree field-of-view LOS with various areas of clear (green/lighter shade) and blocked (red/darker shade) LOS.

In addition to the map, the user is presented with several environment reasoning and manipulation capabilities to choose from. The core capability of the TTA app is its line of sight (LOS) service. There are two LOS services available: point-to-point (PTP LOS) or field-of-view LOS (FOV LOS). LOS queries can be executed from either the user’s GPS location or a selected location on the map. The target location is placed by selecting and moving the annotation across the touchscreen.

PTP LOS queries consist of a line between two discrete points (see Figure 1, center image). Here the green star icon represents the source location and the crosshair image represents the target location. The elevation of the start and stop points are chosen by the user. This flexibility allows the user to use the PTP LOS in many different situations. For example, users on the ground are trying to find cover and concealment locations from an elevated sniper. The user can run LOS queries from the elevated source position to positions on the ground to help determine areas which provide cover.

The Field-of-view LOS service allows the user to define a planar arc to query for LOS. It executes a series of centric point-to-point LOS queries of the same length starting from the same source location and offset from the previous query by a user defined angle. Like the point-to-point LOS service, the source annotation can be either a chosen location or the user's GPS position. The fan's field of view attributes (orientation, depth and angle) are all configurable by the user. The results of a fan LOS query (see Figure 1, right image) are displayed in a similar manner as the point-to-point LOS, where the green region represents clear LOS, and the red region is blocked LOS. In many ways the typical user will find the field-of-view LOS fan to be more useful than the point-to-point query. For example, a user needs to determine the best location to place a sensor in a dense urban environment. The user could run a series of field-of-view LOS queries in areas of interest to find the optimal location with clear LOS.

Both LOS services use the correlated LTF database to calculate their results. LOS rays are checked against both the terrain surface and the terrain features. In figure 1, the point-to-point LOS is clear (green line) until it intersects a building (remainder of the line is red). This example also demonstrates the correlation between the real world (the Google Map's imagery) and the LTF.

From the outset, our plan was to prototype the base LOS capability on both the iPhone and Android platforms. From there, in order to reduce duplication of effort and to focus developing new features, we would down select to a single platform. Due to the overall community interest of the iPhone and also the better integration of LTF native code (discussed in more detail later), we chose to focus new development solely on iPhone.

By the end of the evaluation stage of development, the application on both devices had similar capabilities. The major focus from here was to provide tools which would enhance the user's understanding of numerous LOS results, as well as complement other LTF services, on the iPhone.

Feature Footprints

The next capability added was the visualization of the LTF feature content. Before this, besides the database extents overlay, the user was unable to visualize the correlation between Google Map's imagery and respective LTF databases. In order to give the user an accurate portrayal of the LTF feature contents, we add the feature footprint overlay service (see Figure 2).

For each feature (e.g. building, tree, etc.) on the terrain database, a blue outline is rendered at the appropriate locations. In most cases, both the LTF and the Google Map collection of features correlated, while in other instances a feature footprint might be shifted or missing completely in either system. These miscorrelations can occur for many reasons including the angle of inclination of the satellite imagery collection and the date of which the data (for both LTF and Google Map) was collected.

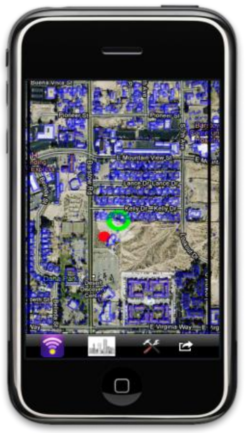

Figure 2. iPhone capture of map with LTF building footprints turned on. This demonstrates the correlation between Google Maps and LTF.

Dynamic Environment

Since the environment in the real world is not static, over time changes will degrade the correlation between a terrain database, which is built from source collected at a finite snapshot in time, and the actual world. This is apparent at locations where the LTF and the Google Map's interface miscorrelate. Additionally, due to the affects of combat operations such as munition detonations or various military construction projects, terrain modifications are inevitable, as are correlation errors.

However, since LTF is a dynamic environment model, it has the necessary capabilities to react and reflect environment changes. Although a full implementation of dynamic capabilities was outside the scope of this project, we were able to introduce a couple of capabilities as a proof of concept.

The first dynamic environment capability developed was the ability to create new building features from the iPhone. This capability can be useful for many reasons. Imagine a situation where a soldier on patrol notices a new building has been erected since the LTF terrain source was captured. Using the TTA app, the soldier is able to add the building on the fly. He/she would do this by first entering "draw mode" on the TTA app. From this mode each of the building's exterior vertices are entered at the appropriate geospatial locations using the touchscreen. Finally, the user enters the building height (in meters) and selects to save the feature. Once the feature is created, it is committed to the LTF database and is fully integrated into the system. The new feature works in the system just like all of the pre-existing features, it is visible when feature footprints are displayed and capable of blocking LOS calls (see figure 3). In the future, we envision the app to be able to communicate with the TTA apps of other teammates. Therefore in this example, the feature created by the patrolling soldier would not only be updated in their LTF database, but also sent out to the entire team or up the command chain for further analysis on the operational affects.

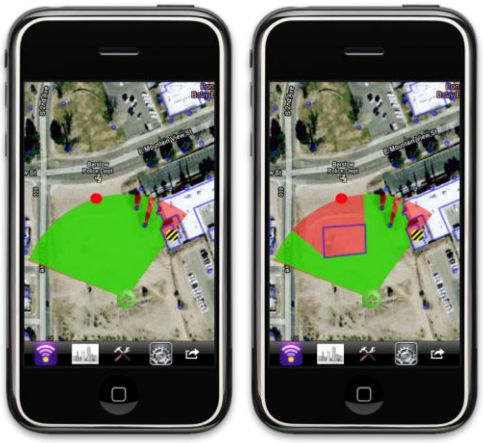

Figure 3. iPhone captures of the dynamic feature capability. Left image shows an LOS fan that is mostly clear. A new building was created in the area (right). Now the LOS fan query is blocked by the new building.

The "add building" service was used to prove the dynamic editing capability of the feature layer. But like features, the terrain surface is also capable of modification. One common occurrence in the battle zone is terrain cratering. The TTA app allows users to generate craters in the LTF terrain database.

The interface for creating craters is nearly identical to the feature creation interface. In a similar manner to feature creation, the user creates new craters in "Draw Mode." The center of the crater is inputted on the map with the touchscreen. The user then is prompted for the depth and radius of the crater. Like dynamic features, craters are completely integrated into the LTF and affect LOS queries. Craters may also prove useful for other decision-making purposes. For example, route planning of a convoy may need to change (due to impassible roads) or crater information can be used to determine trends in insurgent activities by tracking IED (improvised explosive device) detonation locations.

Terrain Elevation View

The top-down Google Map view portrays topographical information quite well, but is not an effective way to portray terrain elevation information. While testing the cratering service, this became quite apparent when we were unable to visualize any of the terrain deformations. To compensate an additional view, the height map display, was developed.

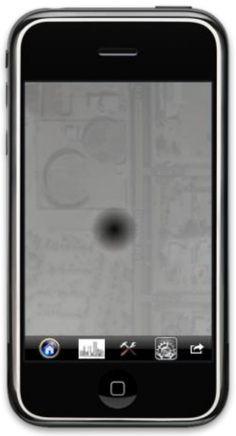

Figure 4. Screen capture of height map display on the iPhone 3GS. The height map displays elevation changes in the area, with darker regions signifying lower elevations and lighter areas signifying higher elevations. In this example, a crater was created dynamically and is visible in the height map.

A height map is a gray-scale raster image used to display the elevation of the terrain. Terrain elevation is shaded in a gray-scale range, with pure black signifying the lowest elevation in the range, and pure white signifying the highest elevation. Figure 4, shows a screenshot of the height map view. In this example, a crater was created using the dynamic cratering capability. The image clearly shows that the terrain elevation was altered as a result of the crater. Since the crater lowers the terrain, its location is much darker than the surrounding area.

Military Map View

Google Maps currently lacks high resolution imagery for areas of military interest (Iraq and Afghanistan for example). It also requires constant network connectivity (Wi-Fi or 3G) for map updates, a situation that may not be guaranteed in the field. For this reason, an alternate map view using the Compressed ARC Digitized Raster Graphics (CADRG) format was introduced into the application (see Figure 5). CADRG is a general-purpose product, comprised of maps and images derived directly from numerous digital sources through filtering, compression and re-formatting to support various weapons, C3I theater battle management, mission planning, and digital moving map systems. CADRG is a standard defense format and closely resembles printed military maps. Since CADRG closely resembles standard military maps, the Warfighter will have a natural comfort level with the view.

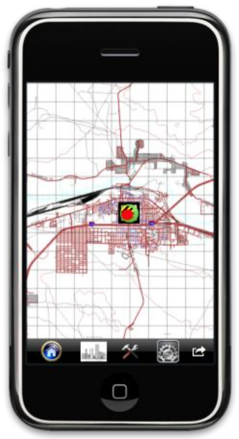

Figure 5. Screen capture of the CADRG view on the iPhone 3GS. The CADRG view can be used as an alternative view to Google Maps in areas where high resolution Google Map data is non-existent.

Additional Usability Enhancements

Additional usability enhancements built specifically to take advantage of the iPhone's hardware were added to improve the user experience. For example, to quickly clear the screen and erase any previous LOS results, the "Shake-to-Clear" capability was introduced. By simply shaking the device, all overlay information (including LOS results) is erased.

Shortcuts were also added for screen navigation. As a user navigates around the map, they can move away from the LOS display. To quickly return to the LOS results, they can use the pan and zoom capability. By selecting the source or target icons on the toolbar, the screen is updated to focus on the selected corresponding location. If both annotation buttons are selected (using a multi-touch gesture) the map window will zoom/pan to display both the source and target LOS annotations. An additional capability was added to snap the source or target LOS location to a selected spot by first touching the appropriate LOS button and then selecting a point on the map. Both of these LOS annotation movements enhance the user's ability to position the LOS field of view, thus enabling quicker querying capabilities.

App Configuration

Included with the app is a preferences menu to help configure the system for a user's needs. The preferences menu consists of all of the definable application variables. Preferences can be set for a variety of capabilities, including LOS. Important LOS variables include: fan step (degrees between each LOS query in a fan), source and elevation heights (offset from terrain skin for the locations of the ray), and LOS mode (sets to use point-to-point or field-of-view LOS queries). Other preferences include the ability to choose the map type (Google Map or CADRG), to show the elevation map (and its transparency level), use the compass for field-of-view orientation, or to turn on capabilities like GPS positioning.

.

ANDROID/IPHONE DEVELOPMENT COMPARISION

At program inception our goal was to prototype an application on both the Android and iPhone platforms. We used the most current (at the time of development) APIs for both the Android (NDK rev 1, SDK 1.5) and iPhone (SDK 3.1). We deployed the apps to a HTC G1 (Android) and iPhone 3GS.

The TTA app consists of two layers: the user interface, which is developed in the platform's native language, and the LTF code, which is in C++. The Android platform utilizes Java for its core development language, and the iPhone uses Objective-C. Both platforms are capable of executing code developed in C++, but it is not their preferred language.

Complicating matters, devices for both platforms have ARM architectures. This requires the use of a cross-compiler when code development occurs on a typical Intel-based workstation.

Android Development

Cross-compiling C++ source code, for NDK rev. 1, proved to be a difficult and complicated process. Since Android interface code has to be developed in Java, a JNI layer needs to be developed to integrate C++/Java code.

Unfortunately, the C++ compiler provided in Android NDK rev. 1 did not fully support the C++ standard, such as I/O streams. Because of this, to get LTF to build with the NDK compiler would have meant major re-writes of LTF source, which was not possible. Without the NDK, it would be impossible to connect Java and C++ through JNI.

With the ideal solution determined to be impossible (using that version of the NDK), we moved on to a less desirable solution. Unlike the iPhone OS, the Android OS allowed for the execution of background apps. We determined that we could create an LTF backend application (operating in the background) that connected to the main Java application through inter-process communication (sockets). Utilizing a third-party cross-compiling toolchain (from Code Sourcery) we successfully built an ARM compliant LTF executable. With this workaround in place, we successfully developed the TTA Android app.

iPhone Development

Compiling native code for the iPhone was significantly easier than the Android counterpart. The Apple compilers are based on GNU C/C++ compilers. Objective-C is a superset of C/C++, meaning that the Objective-C compiler successfully compiles C, C++, and Objective-C code. C++ code can be called directly from Objective-C applications.

Xcode, unfortunately, uses its own build environment forcing projects which use standard makefiles to be ported into Xcode projects. However since LTF uses CMake to manage the build system and CMake can produce Xcode project files without modification, the level of effort normally required was diminished. With Xcode project files in place, compiling native code was simple.

Although porting native code to the iPhone was straight forward, the same cannot be said in general for iPhone development. The first issue is that all development must take place on an Intel-based Mac computer, and be compiled using Xcode. This can become an obstacle, with most companies using either Windows and/or Linux systems as their development workstations.

Another obstacle is Objective-C. Android uses Java as its native development language, a modern, popular language used in development of many popular applications. IPhone development, however, is done in Objective-C, a C based language with heavy Smalltalk influences. Objective-C uses interesting (and perhaps unusual) syntax notation that is quite unfamiliar to C/C++/Java developers. As a result, there is certainly a bigger learning curve to iPhone development compared to Android development.

Configuration management of the iPhone development environment also proved to be difficult. Minor changes of either Xcode or the iPhone OS happened on a near weekly basis. Our app was deployed to numerous devices for testing and demonstration purposes. Each device was managed independently by their owner and updated to different OS versions on an ad-hoc basis. There were even instances where we bought multiple iPod touches on the same day, and they were delivered with different OS versions. Regardless of the reason why the OS was different, if the OS is newer than those supported by our current version of Xcode, an update to Xcode is needed. Unfortunately, Xcode does not have an update feature, which meant an entire new instance of Xcode has to be downloaded (nearly a gigabyte of data) and installed. Depending on network speed and traffic, developers have reported download times in the several of hours.

The process of deploying an app on Android is simple and straightforward, Apple’s deployment process, on the other hand, is anything but simple. There are multiple forums and videos online dedicated to the process. Even Apple's developer website contains a web based "Development Provisioning Assistant" tool to help the developer. In order to run an application on an iPhone or iPod, a developer must get a provisioning profile and a development certificate on their device and their development Mac. After completing the necessary steps to link a device (iPhone or iPod Touch) with an Xcode project, they have to enable it to be used as a development device. Depending on the type of developer account that was setup, the provisioning profile expires every 90 days forcing each device to be reunited with the Xcode project periodically.

Development Conclusions

Clearly, at the time of our initial foray into mobile app development, the Android platform was too immature for real native code development. On the other hand, beyond the possible headaches involved in porting a C++ project over to Xcode, the iPhone platform worked with native C++ code seamlessly. The Android NDK is under constant development with several new releases since our effort, so native code development may now be significantly easier.

However, iPhone development is not without its own headaches. Difficult configuration management, Objective-C, and the necessity for a Mac all are hurdles to overcome.

On a hardware/software level, the iPhone and Android platforms are similar, and will most likely continue to be competitive with one another in the future. In addition, as the Android SDK/NDK continues to evolve, it will eventually support the entire breadth of the C++ language. In the end, which platform is right for the military is going to come down to how unrestricted the underlying system is. Android is a truly open platform where apps can be delivered to the user through the Android Market and/or installed directly on the device. Unless a special agreement is made between Apple and the U.S. Military, all iPhone apps (outside of the limited development deployments allowed) have to be acquired through the App Store. All of these apps are subject to Apple approval and must follow certain guidelines before they are available for download.

BENCHMARK COMPARISON

One of the hallmark qualities of LTF is its LOS performance, which was designed to operate on lower-powered, mobile platforms. Although it was easy to tell that the LOS performance on the iPhone 3G was acceptable and responsive, it was still interesting to see how it compared to LTF performance on desktop platforms. Desktop LTF performance numbers were captured in an earlier publication, “An Optimized Synthetic Environment Representation Developed for OneTESS Live Training” (Borkman, Peele, Campbell 2007). Those tests were conducted on a workstation with the following specs (Table 1).

Table 1. Linux Desktop Specs

CPU	Intel Pentium D EM64T
CPU Speed	3.00 Ghz (HT on)
RAM	2 GB DDR
OS	Kubuntu Linux 7.04 (i386)

Two 9 km^2 terrain databases were produced for testing purposes, both from Barstow, California. The grids were set to one meter post spacing, with 10 and 100 meter culling grids. The representative terrain contains over 1,255 volumetric features including trees and buildings. Table 2 displays the results from variously placed LOS queries. Each length of the query ray was selected based on an analysis of weapon ranges for OneTESS. The actual distances traveled by the algorithm are less due to LOS blockage by either the terrain or a feature. For each query distance a total of 10,000 different rays were randomly generated for the terrain, and each ray was queried 1,000 times.

Table 2. Linux Desktop LOS Query Results

Query Ray Length	Actual Distance Traveled	Rep. Terrain Query Time	Dense Terrain Query Time
150 m	76.8 m	12.5 μs	67.8 μs
300 m	121 m	13.2 μs	70.9 μs
500 m	160 m	14.6 μs	81.5 μs
1000 m	235 m	16.9 μs	83.9 μs
1800 m	348 m	18.0 μs	104.9 μs
2000 m	375 m	18.4 μs	115.1 μs

At the end of the TTA application’s short development cycle, a series of similar benchmark scenarios were executed on the iPhone. For each query ray distance, a single random location was selected. Then the LOS query was executed 1,000 times. In an effort to compare and contrast the iPhone simulator on the Mac and the iPhone 3GS, performance metrics where collected for both. The timing results are shown in Table 3 and 4. In every scenario the iPhone Simulator, which utilizes an Intel architecture and the host system’s resources, outperformed the iPhone by a factor of 20. This contrast in performance is due to the difference between the iPhone test environment and deployed hardware (i.e. iPhone, iPod). In some instances the differences can introduce unexpected issues such as when deploying an application that performs well on the simulator during hours of testing but not on the device. Although the query times on the iPhone are higher, the delay from query to displaying results is minimal. In the near future, as both the devices hardware and software capabilities increase, the TTA application will have the ability to produce faster and more complex queries on mobile platforms.

Table 3. iPhone Simulator LOS Query Results

Query Ray Length	Actual Distance Traveled	Dense Terrain Query Time
150 m	139.5 m	195 µs
300 m	193 m	221 µs
500 m	232.3 m	229 µs
1000 m	399 m	261 µs

Table 4. iPhone 3GS LOS Query Results

Query Ray Length	Actual Distance Traveled	Dense Terrain Query Time
150 m	114.7 m	4055 µs
300 m	198.9 m	4129 µs
500 m	266.2 m	4504 µs
1000 m	352.4 m	5287 µs

FUTURE WORK

The current TTA application is only the first phase in an ongoing effort to provide to the Warfighter a state of the art situational awareness device on commercial hardware. At this point it is a novel approach to conducting very high-resolution LOS routines on a correlated environment, a service itself that would greatly benefit the Warfighter. However, the potential of the app is far greater.

The Warfighter is seldom alone in combat. Their eventual success, and safety, is generally a product how their team functions. Communication is typically the main driving factor of team success. Therefore, adding Intra-team communication services would be a key component for a true situational awareness device.

The possibilities of new capabilities built upon the communication infrastructure are exciting. One important capability needed is real time unit position at an individual level. This would give the user a new perspective on the theater of operation. But the capabilities could go far beyond that. For example, when a member of a combat unit has intelligence about the exact location of an enemy combatant, that information could be placed on the map and broadcasted to the team. The user could take a photo of the combatant's location and which can be geo-tagged with geospatial information and shared. All of these features and more can easily be built on top of LTF and TTA foundations.

CONCLUSION

Over the last decade, geospatial data capture capabilities have outpaced the ability to get that data to the people most in need of the information, the Warfighter. The TTA Analysis app reverses that trend. It allows the Warfighter to leverage detailed geospatial information "on the go", going beyond just a map of the world by providing derived terrain analysis on a correlated, virtual environment. Since the LTF system is easily loaded onto the device, requires little processing power, and maintains a small memory footprint, the TTA application is able to increase the user's situational awareness by augmenting the amount of environmental information in hand. Although it is currently a prototype capability, we envision that it will continue to grow into a tool that will be able to provide our Warfighter with a clear and complete operational picture and aid in their safety and eventual success.

ACKNOWLEDGEMENTS

This work was made possible by the U.S. Army's STTC, through SBIR Topic A08-013 "High-Fidelity Runtime Database Engine." The government POC for this work is Mr. Julio de la Cruz. The work also built on capabilities developed for the OneTESS program.

REFERENCES

Borkman, Peele, Campbell (2007) "An Optimized Synthetic Environment Representation Developed for OneTESS Live Training," *Interservice/Industry Training, Simulation and Education Conference. Orlando, FL November 26-29 2007*.

Campos, Borkman, Peele, Campbell (2008) "Towards Cross Domain Terrain Services", *Interservice/Industry Training, Simulation, and Education Conference. Orlando, FL December 1-4, 2008*.

Chen, Brian (2009). *Apple's App Store Hits Six Digits; How Many Apps Do You Need?*. Retrieved from http://www.wired.com/gadgetlab/2009/appstore/

ComScore (2010). *ComScore Reports January 2010 U.S. Mobile Subscriber Market Share*. Retrieved from http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Reports_January_2010_U.S._Mobile_Subscriber_Market_Share

FMEpedia. (n.d.). *Compressed ARC Digitized Raster Graphics*. Retrieved from http://www.fmepedia.com/

Open Handset Alliance (n.d.), Retrieved from http://www.openhandsetalliance.com

Technologizer (2010). *Google: Expect 18 Android Phones by Year's End.* Retrieved from http://www.technologizer.com