Visualizing Electromagnetic Spectrum Phenomena in Augmented Reality

Michael Longtin
Robert Hernandez
Lockheed Martin RMS
Orlando, FL
michael.longtin@lmco.com, robert.d.hernandez@lmco.com

Richard Schaffer
Lockheed Martin RMS
Burlington, MA
richard.l.schaffer@lmco.com

Mark Wager
Lockheed Martin RMS
Fort Worth, TX
mark.e.wager@lmco.com

ABSTRACT

The Tactical Decision Kit (TDK), developed by the Office of Naval Research (ONR), contains hardware and software components that allow users to rapidly create ultra-realistic geo-specific terrain models and visualize them as holograms on the Microsoft HoloLens via an application called SandTable.

An early experiment in the development of the TDK involved the incorporation of a radio-frequency (RF) propagation model called Sandbar into the SandTable application. This enabled several electromagnetic (EM) phenomena such as probability of detection and received signal strength to be visualized on the SandTable's terrain models in the form of color-coded semi-transparent overlays. Early demonstrations of this capability have garnered a tremendous amount of interest from multiple groups within the military, as it provides a unique way to visualize RF phenomena which are normally invisible, allowing insight into a unit's RF footprint as well as its ability to intercommunicate.

During the development of the EM overlay capability within the SandTable, various challenges were encountered associated with running computationally intensive models on a non-tethered device with a relatively weak CPU. Novel techniques were developed in order to overcome these challenges, including shifting some of the computational burden from the CPU to the GPU, and offloading other computations to a powerful remote PC. This paper details the development of the spectrum operations capability within the SandTable application, and explores the techniques that were employed to overcome the aforementioned challenges.

ABOUT THE AUTHORS

Mr. Michael Longtin is a senior software engineer at Lockheed Martin Co. Michael has been working in the field of augmented reality, developing a terrain database generation system that creates terrain databases. Michael has also developed a mixed-reality sand table application for the Marines, an application that allows trainees to visualize three-dimensional terrain models for scenario creation and mission rehearsal. Mr. Longtin holds a Master of Science degree in Electrical Engineering from the University of Maine.

Mr. Richard Schaffer is a Lockheed Martin Fellow and Principal Investigator at Lockheed Martin Rotary and Mission Systems (RMS). He leads the Human Immersive Simulation Lab at RMS's Advanced Simulation Centers. Richard received his S.B. degree from the Massachusetts Institute of Technology and has over 34 years of experience in modeling and simulation research and development. His areas of research have included distributed simulation, environment modeling, immersive simulation and augmented reality. In 2010 he received the NTSA's lifetime achievement award.

Mark Wager is a software engineer at Lockheed Martin Co. Mark has been researching and developing various augmented and virtual reality research projects that focus on using the GPU to perform large parallel calculations in real-time. He has been working with the ONR to develop an augmented reality SandTable application that visualizes electromagnetic overlays on a 3D terrain. Mark has also developed a virtual reality application that utilizes virtual reality haptic force feedback gloves. Mark holds a Bachelor of Science degree in Computer Engineering from Texas A&M University.

Robert Hernandez is a software engineer at Lockheed Martin Co. with a Bachelor of Science degree in Computer Science from the University of Central Florida. He is assisting in the research and development of augmented reality solutions with ONR for tactical and real-time planning. Robert has integrated the SandTable application with various protocols to receive live data from units and receivers deployed on the field to paint a more accurate view of the battlefield through the use of electromagnetic overlays generated in real-time.

Visualizing Electromagnetic Spectrum Phenomena in Augmented Reality

Michael Longtin
Robert Hernandez
Lockheed Martin RMS
Orlando, FL
michael.longtin@lmco.com, robert.d.hernandez@lmco.com

Richard Schaffer
Lockheed Martin RMS
Burlington, MA
richard.l.schaffer@lmco.com

Mark Wager
Lockheed Martin RMS
Fort Worth, TX
mark.e.wager@lmco.com

INTRODUCTION

The recent proliferation of augmented reality (AR) devices across the modeling and simulation domain has ushered in a migration from traditional desktop computers toward immersive environments. Users are no longer confined to desktops or limited by two-dimensional monoscopic displays. This technological breakthrough has opened up a new realm of possibilities.

One such AR device, the Microsoft HoloLens, was selected for inclusion in the Tactical Decision Kit (TDK), developed by the Office of Naval Research (ONR) (Young, et al, 2019). The TDK contains hardware and software components that allow users to rapidly create ultra-realistic geo-specific terrain models and to visualize them as holograms via an application called the SandTable. This application was designed to be used as a substitute for physical sand tables that have been used for decades in the military. Typically, large sand tables (approximately 4' x 12') are shaped by hand to conform to a specific region. Yarn is laid on top to represent terrain features such as roads, rivers, and grid lines. Paper or cardboard symbols represent units and equipment. Marines gather around the table and use it for mission planning.

The computerized SandTable application serves as a substitute for physical sand tables by instantiating them as computer-generated holograms. For all practical purposes, a holographic sand table functions as a physical sand table; several Marines can stand around it, manipulate unit symbols, and visualize the hologram as if it were a physical sand table. Virtual sand tables have the advantages of being ultra-portable and much less tedious to create and maintain. The TDK has been deployed to all 24 Marine Corps infantry battalions and is being used to facilitate mission planning, mission rehearsal, and after-action review.

VIRTUAL REALITY VS. AUGMENTED REALITY

"Virtual reality" (VR) devices completely immerse users in a digital world; objects in the real world are not visible. They can track the user's position and head orientation, allowing the scene's camera to follow the user's actual head movement. Examples of VR devices include the Oculus Rift and the HTC VIVE. Most VR systems consist of a head-mounted display (HMD) physically tethered to a computer, which severely limits users' movements. To maintain a reasonable frame rate, PCs which drive VR devices typically feature high-end graphics hardware capable of rendering high-resolution images at 30 frames per second or more.

"Augmented Reality" (AR) devices allow computer-generated content to "coexist" with the real world (Johnson, 2016). AR devices need to be much more sophisticated than VR devices because they require an understanding of the physical world around them. Surfaces in the room are scanned by depth cameras. Multiple passes of the scan are stitched together and merged over time to create a 3D mesh representing the physical objects in the room (Ashley, 2016). The mesh models allow computer-generated content to be placed on tabletops, for example, and also allow computer-generated content (a.k.a. "holograms") to be obscured by physical objects. Examples of AR devices include Microsoft's HoloLens and Magic Leap's "Magic Leap One." Most, but not all AR systems are untethered, allowing the user to move about freely, unrestricted by a cable.

In 1994, Paul Milgram first introduced the concept of a "reality-virtuality continuum" that describes mixed reality as anything between the real environment (no synthetic content) and a completely virtual environment (all synthetic content). Note that today, the term "mixed reality" is sometimes used synonymously with AR (Milgram, 1994).

It's important to note that untethered devices tend to have far fewer computational resources than their tethered counterparts. This is because they must be relatively light to enhance comfort. They must run on a self-contained battery, which cannot be large due to weight restrictions. Consequently, the CPU and GPU cannot have an active cooling system due to the reduced power supply requirements. Likewise, the amount of RAM, which uses power, is typically limited. This significantly reduces the computational bandwidth of MR devices, which poses significant challenges for MR application developers.

MR developers must be cognizant of the resource limitations of target devices and tailor their software designs accordingly. While the SandTable application is not limited to running on one particular MR device, it was required to run on the tetherless Microsoft HoloLens due to its inclusion in the TDK. This paper explores the software design decisions made to ensure the SandTable application performs well on a device with limited computational resources.

THE SANDTABLE APPLICATION

One of the primary components of the Tactical Decision Kit is an application called "Interactive Tactical Decision Games," or ITDG, which is a web-based application that allows users to place military symbols, mission graphics, and annotations on a 2-dimensional map. It runs on any device that has a web browser, and scenarios may be shared among participants via a WiFi network. Before the SandTable became part of the TDK, many users found it difficult at times to discern ridges from valleys by looking at the 2D maps. Also, Marines were using physical terrain models called "sand tables" for mission planning. These large models were time-consuming to set up and were not portable.

The SandTable application allows users to visualize 3-dimensional terrain models as holograms. This allows users to realize the benefits of physical sand tables without being encumbered by their limitations. In addition, the 3-dimensional terrain models assist the users with unit placement by giving them a true perception of terrain elevation. Furthermore, when run on see-through augmented reality devices, multiple participants can collaborate with each other, maintaining eye contact, which greatly facilitates mission planning (Young, et al, 2019).

The SandTable application also allows users to scale the terrain to a 1-1 or near 1-1 scale, so it can be used for pre-mission walkthroughs, helping users familiarize themselves with exercise areas before traveling there (see Figure 1).

Figure 1 – Using the SandTable for Mission Rehearsal

A third common use case for the SandTable is as a common operating picture (COP), where it ingests live or pre-recorded unit positions and renders them on the terrain model. Icons can be color-coded (representing which unit they belong to, for example), or annotated with useful information indicating call name, signs, speed, direction, etc. See Figure 2 - SandTable's AAR Capability.

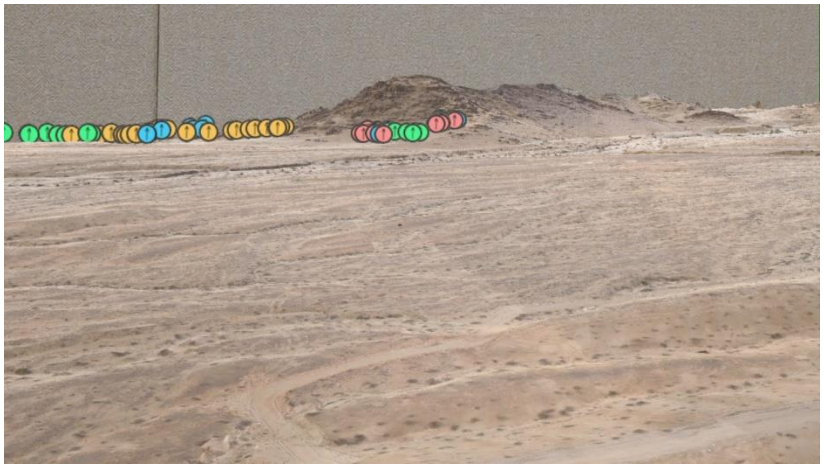

Figure 2 - SandTable's AAR Capability

The SandTable communicates with a series of servers running on a PC via a WiFi connection. These servers provide essential information to the SandTable application and include

- ITDG Scenario Server
  Allows the SandTable to sync with scenarios being developed in ITDG. Changes made to scenarios in either application will be automatically be updated in all other ITDG and SandTable instances.

- Gateway Server
  Serves as the interface to external data feeds, allowing the SandTable to display live unit position data.

- Military Symbology Server
  Dynamically generates standard MIL2525C symbol bitmaps for holographic rendering.

- Collaboration Server
  Handles inter-device communication that allows for multi-user collaboration

See Figure 3 - SandTable System Components.

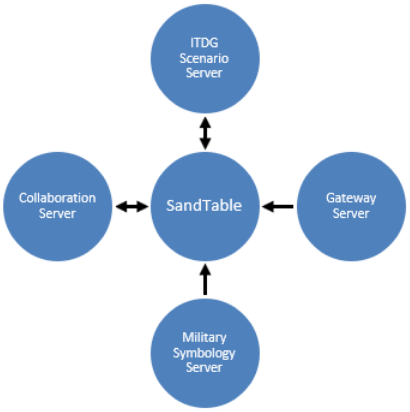

Figure 3 - SandTable System Components

A NEW USE CASE

Radio propagation (RP) models simulate how electromagnetic emissions behave in the environment. Typically, inputs to these models include transmitter positions, power, frequency, and bandwidth. The output is typically described as a data table that describes various EM characteristics for each of a series of grid squares, each covering a specific area. For example, the first two columns of the table might consist of latitude and longitude, representing the center of a

grid square. Other columns might represent characteristics of the emitted signal that exist if a receiver were placed within that grid square. These characteristics are defined by the RP model itself; each model may support different characteristics. Examples of these characteristics may include signal-to-noise ratio, path loss, signal intensity, probability of detection, etc. Typically, data tables are not very human-friendly; it can be difficult to interpret the results of the RP model by looking at an array of numerical values. Consequently, RP outputs are typically rendered as a series of grid squares overlaid on a map, providing geographic context. These grid squares are filled with colors that reflects values, typically ranging from blue (cold or low) through red (hot or high) (see Figure 7 for an example of this).

One early experiment in the development of the TDK involved rendering the results of a RP model called Sandbar in the SandTable application. Early demonstrations of this capability have garnered a tremendous amount of interest from multiple groups within the military, as it provides a unique way to visualize phenomena which are normally invisible, allowing insight into a unit's RF footprint as well as its ability to intercommunicate, for example.

The first attempt at rendering a graphical representation of Sandbar results involved running the model on a high-powered PC, then using the resulting data table to render colored grid squares on the terrain. While this worked well, it wasn't very practical because the user would be required to use Sandbar's user interface on a PC to specify the transmitter locations and parameters, run the model, then transfer the resulting table to the MR device for rendering. A much better approach would involve tightly coupling Sandbar and the SandTable, allowing the user to manipulate transmitter positions and parameters and see the Sandbar results immediately.

Unfortunately, Sandbar is a PC-based application that cannot readily be executed on MR devices, most of which do not run full-fledged Windows operating systems. This necessitates a formal porting of the Sandbar model to the MR device itself. While this may be feasible for tethered MR devices that have resources of a full-fledged PC at their disposal, it is not feasible for untethered MR devices due to their limited resources (CPU, GPU, and RAM).

TECHNICAL CHALLENGES

MR applications running on the Microsoft HoloLens have limited memory and processing power at their disposal. Two strategies were employed to overcome these limitations:

1. Distributed Computing
   Memory and processor-intensive components of the SandTable application are offloaded to a server application hosted on a powerful laptop. This application is called the "SandTable Server" and helps ease the computational burden on the HoloLens. Additionally, it allows other applications (such as ITDG) to fetch the results using Representational State Transfer (REST) Application Program Interface (API) calls. The SandTable Server performs the majority of the calculations needed to generate the Electromagnetic (EM) overlays, then passes the intermediate results to the SandTable to be rendered.

2. Compute Shaders
   Besides shifting the burden of the computational load to a more powerful PC, the SandTable application also exploits the power of the GPU via the use of *compute shaders*, effectively shifting the burden to another processor that is more capable than the CPU at performing certain types of calculations.

Distributed Computing

In order to offload some of the more computationally intensive tasks to more suitable hardware, a PC-based application called the "SandTable Server" was created. See Figure 4 - SandTable System Components with SandTable Server, which depicts the previous system components diagram with the SandTable Server added (in red).

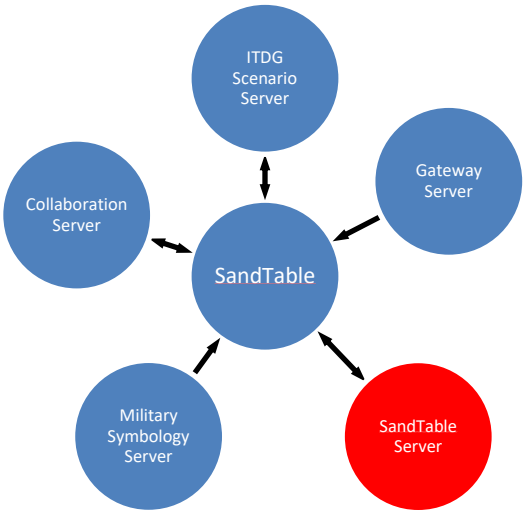

Figure 4 - SandTable System Components with SandTable Server

Conceptually, when the need to add a new capability to the SandTable arises, and that new capability requires significant computational resources, an engineering decision is made whether to run the capability locally on the MR device or offload it to the SandTable Server, which would improve execution time and extend battery life.

The SandTable application offloads the following tasks to the SandTable Server:

- Route Planning
- Intervisibility Overlays
- Electromagnetic Overlays

Compute Shaders

A compute shader is a program that's executed on a GPU rather than a CPU. GPUs were initially designed solely to render graphics and, when they were first introduced, could do only that; the graphics processing pipeline was very rigid. As time went on, however, GPUs became more sophisticated and allowed programmers to define their own implementations of portions of the graphics pipeline. For example, a programmer can write a custom *shader* that changes the appearance of the pixels rendered on the screen.

Over time, developers realized that shaders could be used for purposes other than rendering pixels on a screen. Initially, exploiting shaders in this way was slightly awkward because the supported data structures were designed to process pixels and images. For example, a developer might resort to masking and bit-shifting techniques to encode arbitrary data into a Texture2D, one of the few initially supported data structures used with shaders. Inputs were encoded in textures, the shader would perform operations on that, and place the results in textures, floating point arrays, or other data formats supported by GPUs. Developers began writing shaders to perform general calculations rather than solely rendering pixels on a screen. Support was added for invoking these shaders arbitrarily, and the result was *compute shaders*.

As time progressed, GPU manufacturers made it easier for developers to define their own data structures that could be passed between CPUs and GPUs, making compute shaders even more practical. However, it's important to be cognizant of the limitations of GPUs and consider these before deciding whether to use them. Tasks which are well-suited to run on GPUs have the following characteristics:

- Many parallelizable, independent work items.
- Avoid algorithms that maintain state information.
- Minimal branching.

Running algorithms that do not meet these criteria are likely not good candidates for running on GPUs, and would defeat the purpose of running on GPUs.

In the SandTable application, compute shaders are used in several areas where appropriate, both in the SandTable Server and in the client-side SandTable application. This significantly enhances the runtime efficiency of extremely computationally intensive operations. Without harnessing the power of GPUs, the system would run so slowly that it would be virtually unusable.

The following paragraphs describe how distributed computing and compute shaders were used to benefit the SandTable application.

Route Planning

Marines using the SandTable expressed the desire for a route planning capability. This allows the user to specify a series of waypoints on the terrain and automatically generate a route that passes through these waypoints in order, while circumventing non-traversable obstacles and minimizing exposure to the enemy. SandTable developers briefly considered incorporating the route-planning algorithm into the SandTable application, but quickly came to the realization that this was not feasible because

1. Such algorithms typically require a lot of RAM to represent navigation meshes, and
2. They typically are CPU-intensive, having to consider potentially billions of possible paths in order to find the one that best fits the search criteria.

This was the initial inspiration for the SandTable Server. It loads the same terrain model as the MR device, which guarantees that the generated results can be incorporated into the SandTable terrain without modification with perfect correlation. The SandTable Server exposes REST services that allow external applications to request routes. The resulting routes are then sent to the requesting application as a response to the REST invocation. See Figure 5 – Route Planning via the SandTable.

Intervisibility Overlays

The Marines also requested the ability to render intervisibility overlays within the SandTable. Such overlays visually represent areas on the terrain that can be seen by the enemy, and optionally, a blue force observer.

In Figure 6, red portions of the overlay represent which areas hostile units can see, blue represents friendly, and purple represents areas both friendly and hostile units can see.

Figure 5 - Route Planning via the SandTable

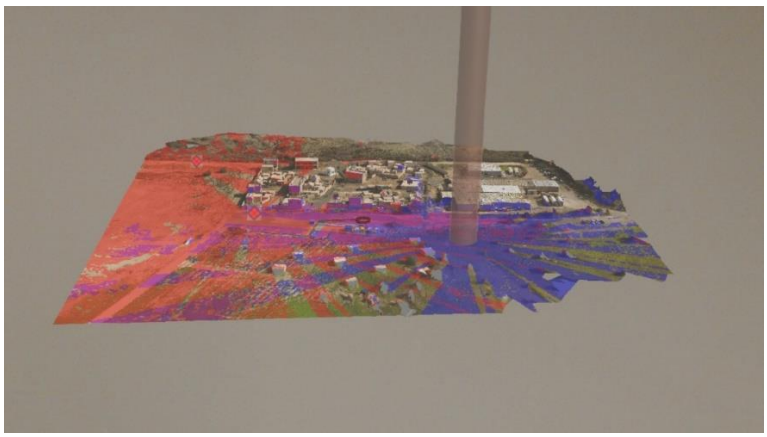

Figure 6 - Intervisibility Overlay

To generate the intervisibility overlay, the majority of the calculations are done on the SandTable Server, and the final results are sent to SandTable for rendering. The terrain mesh is split into several tiles, due to mesh size limitations. The SandTable (or ITDG) then sends to the SandTable Server the positions of each of the units with a JSON packet as part of a REST POST call. The SandTable Server then places Line-Of-Sight (LOS) observers to represent these units that begin scanning the terrain around them.

Each of the LOS observers has several cameras that scan the terrain as they rotate, while continuously applying their scan results to a render texture. This scanning process occurs for each observer, and all observers are processed simultaneously. The contributions to the visibility results of each observer are continually folded into a "result" texture via a compute shader. The scanning task is particularly well-suited for execution on a GPU because it contains no state, it's parallelizable, and has no branching. When the LOS observers have finished scanning, the result texture is made available to clients via REST services.

At this point, the SandTable application running on a MR device takes those 2D textures and applies them to each terrain mesh on its local instance of the terrain. Standard rendering shaders use the information contained in the result textures to render a semi-transparent overlay on the terrain that indicates areas that are visible to red units, blue units, or both. The final effect is shown in Figure 6 - Intervisibility Overlay.

To support ITDG (a 2D web-based application), the SandTable Server performs the same calculations, but also combines the results into one final 2D image. The SandTable Server then sends this 2D image to ITDG for display.

Note that the intervisibility implementation employs both distributed computing (offloading computationally intensive calculations to a PC that is better suited for carrying out such calculations) and compute shaders (harnessing the power of GPUs), which enables reasonable performance on the target device.

Electromagnetic Overlays

The process used to generate EM overlays is very similar to that used to generate intervisibility overlays. The main difference lies in the calculations performed during the LOS observer scans, and those done in the post-processing shaders.

During the observer scans, instead of merely determining which points on the terrain to which there is a direct line of sight from the observer, the precise distance between the observer and each terrain point is calculated. These distances are stored in the result texture to be used in subsequent processing.

As with the intervisibility implementation, the result textures are sent to the client MR devices via REST service invocations. However, instead of directly rendering colors specified by the result textures on the MR device, the client-side shaders actually perform radio propagation model calculations to arrive at the final value for each and every pixel. The outputs of these models depend on the distance to the transmitter and transmitter-specific EM parameters, such as power and frequency. Running this post-processing on the client side enables the user to dynamically change EM parameters on the SandTable and immediately see the results reflected without having to request additional information from the SandTable Server. This post-processing is done by the GPU, so it can be performed without overtaxing the MR device's CPU.

The color scale is then sampled and blended with the terrain color at each point.

The final result of these calculations for the Signal-to-Noise Ratio overlay is shown in Figure 7 – Signal-to-Noise Ratio Overlay.

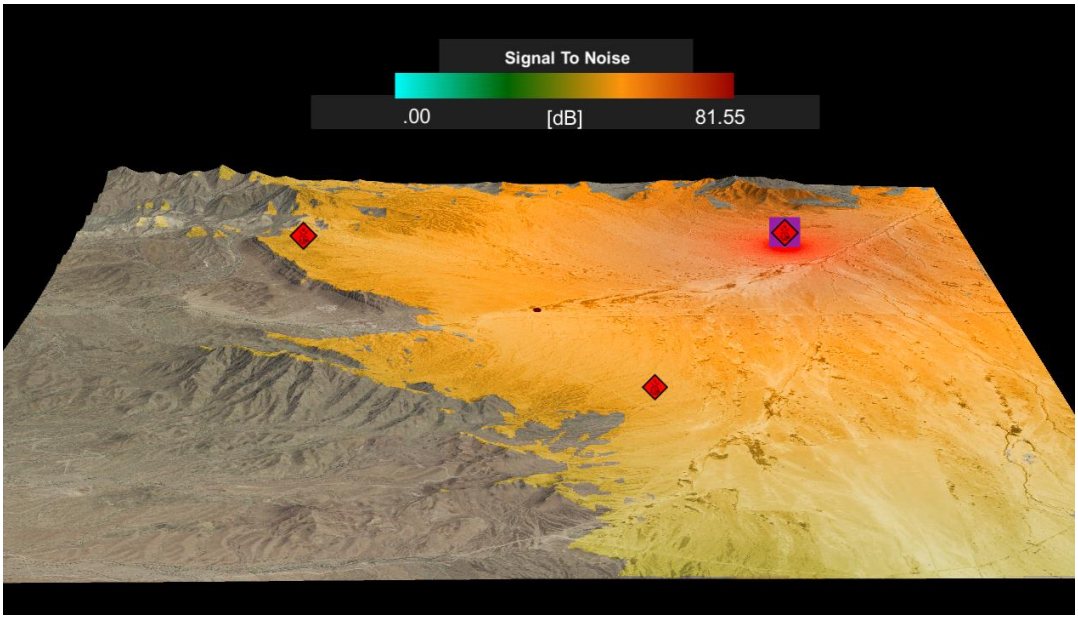

Figure 7 – Signal-to-Noise Ratio Overlay

Note that the resulting overlay also reflects line-of-sight occlusion. This is because the aforementioned distance texture doubles as a line-of-sight mask. Pixels that do not have a line of sight from the transmitter are encoded as zeros in the texture and are skipped when calculating the overlays.

User Scalability

To support multiple different device types and platforms, the ITDG Server is acts as the primary controller of all requests sent to the SandTable Server. This allows the SandTable Server to behave like a model computation program, and not have to spend resources dealing with state management. It also allows the SandTable Server computation results to be cached for quick retrieval by any client (SandTable or ITDG). This cache allows duplicate requests to have their results immediately returned from the cache, making it available to process other calculation requests. In order to be device-agnostic, the results from SandTable Server are all represented in JSON format.

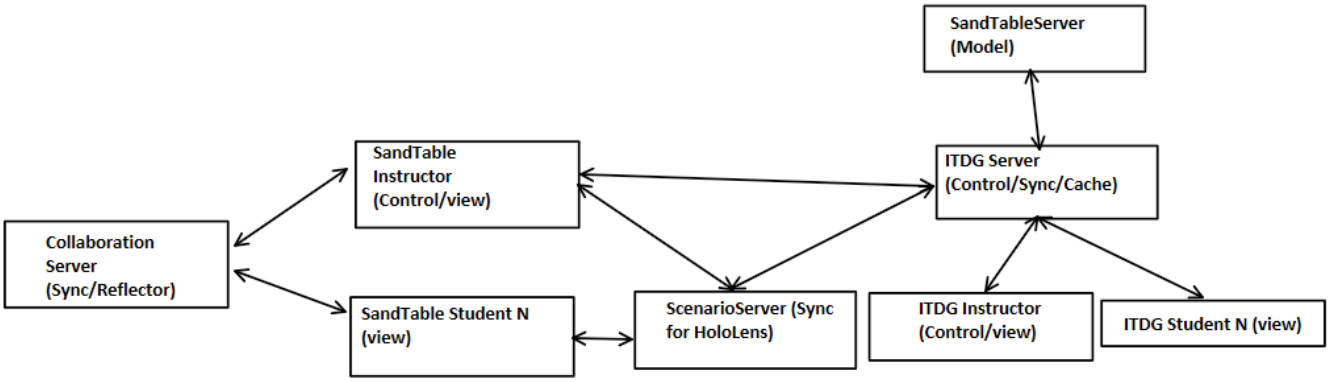

Figure 8 - ITDG System Diagram

FUTURE WORK

The EM overlay capability continues to be enhanced and refined. Plans for future work include

- Allowing point queries (displaying the value of an overlay at a user-specified point).
- Supporting terrain models with built-in Earth curvature for very large areas.
- Expressing transmitter locations as paths in addition to points to model moving transmitters.
- Adding support for additional EM propagation models.

CONCLUSIONS

In 2017, after realizing its potential, the Assistant Commandant of the Marine Corps, General Glenn Walters, directed the newly-formed Rapid Capabilities Office to distribute the Tactical Decision Kit across all 24 Marine Corps infantry battalions. Today, the distribution is complete, and the Marines are using the TDK on a daily basis for mission planning, mission rehearsal, after-action review, and signature management.

In addition, some units are using the signature management capabilities to visualize their EM footprints, as well as those of the enemy in mock scenarios, providing valuable insights into the consequences of "loud" EM footprints.

One valuable lesson learned while adding the EM visualization capability to the SandTable is that when designing software, developers must be cognizant of the capabilities of target devices. If they aren't, user experiences will suffer and limit the adoption of the applications.

ACKNOWLEDGEMENTS

The Sand Table application was developed as part of Office of Naval Research's Accelerating Development of Small Unit Decision Making (ADSUDM) program under contract N00014-16-C-1029. The Marine Corps Rapid Capability Office executed the rapid fielding of the Tactical Decision Kits. We would like to acknowledge Lt. Col. Marcus Mainz, 26th MEU Battalion Commander, and the Marines of V26, for embracing and promoting the TDK, as well as providing valuable feedback and guidance.

REFERENCES

Christopher Young, Richard Schaffer, Michael Longtin, Brian Stensrud, PhD, and LtCol Marcus J. Mainz, (2019). *Tactical Decision Kits for Infantry Training.* The 2019 Interservice/Industry Training, Simulation, and Education Conference, Orlando, FL, USA.

Eric Johnson, (2016). *What are the differences among virtual, augmented and mixed reality?* Retrieved April 16, 2019, from https://www.recode.net/2015/7/27/11615046/whats-the-difference-between-virtual-augmented-and-mixed-reality

James Ashley, (2016). *Understanding HoloLens Spatial Mapping and Hologram Ranges* Retrieved April 20, 2019, from http://www.imaginativeuniversal.com/blog/2016/03/23/understanding-hololens-spatial-mapping-and-hologram-ranges/

Milgram, Paul; H. Takemura; A. Utsumi; F. Kishino (1994). "Augmented Reality: A class of displays on the reality-virtuality continuum" (pdf). Proceedings of Telemanipulator and Telepresence Technologies. pp. 2351–34. Retrieved 2007-03-15.

Matthäus G. Chajdas, (2018). *Introduction to Compute Shaders* Retrieved April 23, 2019, from https://anteru.net/blog/2018/intro-to-compute-shaders/