Developing Project Proto-HEAD (Prototype Holographic Environment for Analysis of Data)

James L. Dean

ABSTRACT

With Project Proto-HEAD, or Prototype Holographic Environment for Analysis of Data, a Johns Hopkins University Applied Physics Laboratory (APL) team sought to learn more about augmented reality (AR) and whether it significantly improves simulation and data analysis over using a PC workstation. Proto-HEAD had a simple concept: given a 3-D data set, visualize that data set as a hologram within the physical world. The development process afforded the team significant insight into the challenges and potential approaches for visualizing and interacting with very large data sets. These insights are useful for new applications that visualize and interact with large data sets and complex models.

INTRODUCTION

Augmented reality (AR) offers potential for new, deeply immersive, and disruptive applications. The immersive technology ecosystem has matured enough that several industries beyond the entertainment business are seriously, and excitedly, considering new innovative and game-changing applications. Yet, while the technology is maturing at an ever-increasing pace, would-be users are still searching for that compelling benefit AR provides. That is, AR is here in a big way, but many are still trying to figure out what to do with it.

APL staff members have been working in this domain for decades, but as technology advances, they are increasingly interested in exploring how immersive technology like AR might be applied to the critical work the Lab does for its sponsors and the nation. Eager to learn more, several staff members took an APL strategic education course focusing on the Unity game engine for virtual reality (VR)/AR applications. This course inspired Project Proto-HEAD, or Prototype Holographic Environment for Analysis of Data—the team's introductory foray into the world of AR.

THE CONCEPT OF PROJECT PROTO-HEAD

In addition to having a general interest in this emerging technology, the team sought to explore a difficult question: Do AR concepts provide a significant and compelling benefit within the realm of simulation and data analysis over using the ubiquitous PC workstation? With a little imagination, an individual could see how AR grants a user full appreciation of the scale of their simulations. Imagine standing underneath the wing of a full-scale commercial aircraft and witnessing airflow transition from laminar to turbulent, and then being able to identify, at a glance, where flow separation occurs along the control surfaces. Alternatively, perhaps the

intrepid researcher wants to see how the current physical configuration of a room will affect airflow as a virtual fire spreads throughout the room, and then physically rearrange the room and immediately see a different evolution in airflow. In this notional case, would being inside the physical room, with holographic data projected onto physical surfaces, provide a compelling benefit compared with simply running simulations on a computer?

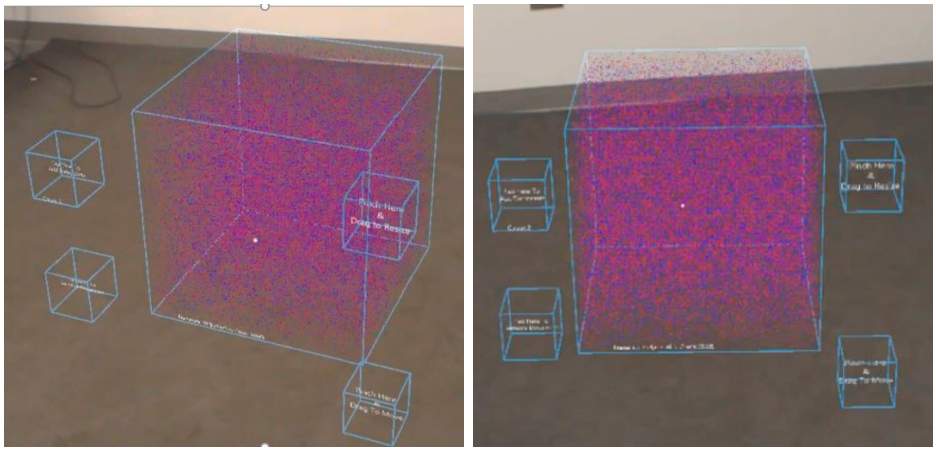

Figure 1. Project Proto-HEAD. External data are rendered within a preset holographic spatial grid, allowing the user to manipulate and move around the data to visualize them at any angle.

The concept for this project was simple: given a 3-D data set, visualize that data set as a hologram within the physical world (see Figure 1), focusing on software design concepts, capabilities, and limitations during development and testing. The intention was not to develop the one "killer application" but rather to improve understanding of key aspects in loading and rendering large data sets given currently available hardware for AR.

DEVELOPING PROJECT PROTO-HEAD

The Microsoft HoloLens was a natural choice for a hardware platform for several reasons, freedom of unhindered movement being one. Others include the ability to collaborate with multiple HoloLens users in the same virtual environment. A particularly critical reason is the software support that Microsoft provides for the Unity game engine. Unity's interface is simple to learn and easy to scale into more complex applications—a feature that is extremely helpful for new users unfamiliar with video game development. Microsoft's Mixed Reality Toolkit provided the essential software components to use the HoloLens's capabilities.

The data set used for development and testing was a simple point cloud providing data for position in Cartesian space plus an additional value representing a measurable entity, such as temperature or speed for example. These data loaded and rendered at the start of the application and could then be manipulated by the user via HoloLens hand gestures to reposition, rescale, and reorient.

A handful of challenges were revealed during the development of Proto-HEAD. Some of those challenges were expected—for example, ensuring that the hologram centered at a desirable default location. A few challenges were quite surprising, however. Among the more significant challenges was the limited processing and rendering power of the HoloLens combined with the team's limited expertise in computer graphics and rendering.

The first implementation of the application generated and rendered a series of holographic spheres, following the assumption that the data set represented a simple point cloud. Each sphere was assigned a color based on the value of the data set associated with that sphere. Performance degradation was immediately apparent once the HoloLens worked to render each sphere. In fact, only 200 spheres rendered at any one time before the HoloLens's performance suffered.

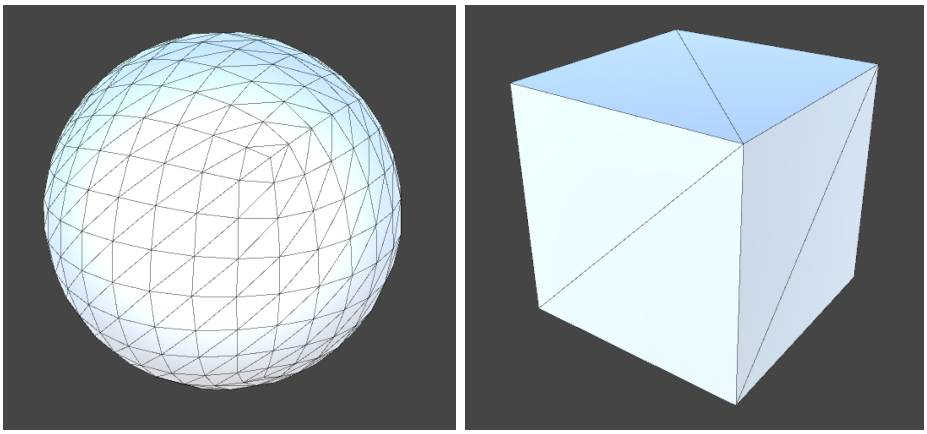

Figure 2. Fundamental 3-D shapes rendered within Unity. By default, Unity renders spheres using 515 vertices and 768 faces. Unity renders cubes using 8 vertices and 12 faces by default.

This was a naive approach; the performance drop was due to the mesh fidelity necessary to render each sphere properly. By default, the Unity game engine renders a single sphere using 515 vertices and 768 faces (see Figure 2), all of which contain information to compute color, lighting and shadows, reflections, etc. within the graphics-rendering pipeline.

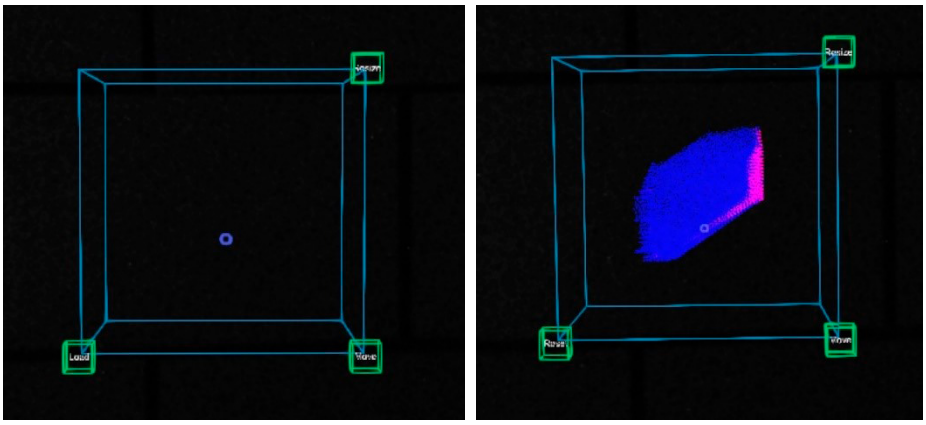

Figure 3. Early versions of the Proto-HEAD hologram display. The user saw the holographic container without any data loaded (left). After the user focused on the "load" box and then made a HoloLens air-tap gesture, the data were loaded and rendered inside the holographic box (right). The reticle in purple followed the user's forward gaze.

The goal for the second implementation was to reduce the amount of vertices and faces that the HoloLens would render, or eliminate them entirely if possible. For example, replacing each sphere with the more fundamental cube reduces the count of vertices to 8 and faces to 12 (see Figure 3). However, this solution would only alleviate the problem, as those shapes still required a mesh in order to render properly.

The solution was instead to use a so-called particle system in Unity. Particle systems are constructs that act as finishing touches to visual effects. Visualizing sparks flying from a virtual fire is a simple way to think about how particle systems are used. After replacing the rendered spheres with such a particle system, the HoloLens was able to render 20,000 individual data points without suffering in performance, an increase of two orders of magnitude compared with using spheres.

The next significant challenge was enabling interaction between the user and the hologram itself. This challenge was a combination of three problems. The first was preventing the user from pushing holograms through physical objects. This was easy to resolve thanks to the HoloLens's ability to detect and understand the physical space around it.

The second problem was detecting when the user wanted to manipulate the rendered data set. Without a handheld controller connected to the device, hand gestures in the air were the next immediate solution. The HoloLens provides a way to detect various hand gestures. The simplest gesture that worked consistently was the air-tap gesture, in which the user brings a hand in view as if to pinch something in the air. A surprising amount of finesse is needed for the HoloLens to recognize this gesture. The user must complete the gesture in several steps. The user must first bring the hand into view of the device's cameras and then open the hand with the index finger and thumb far apart, in a "tap-ready" position. The HoloLens developers implemented visual cues notifying the user that the device recognizes this position. After displaying these cues, the system waits for the user to complete the actual air-tap motion before signaling that the gesture has occurred.

The last problem was using the air-tap gesture as a signal to manipulate the data set. This challenge had less to do with how to technically implement this functionality and more to do with how to implement it with a proper understanding of the context of the action. For example, the user might not be looking at the holographic data set when performing the air-tap gesture. Alternatively, perhaps instead of moving the hologram to a different position in space, the user might want to resize and rotate the hologram. The solution was rudimentary: create two separate holograms to isolate scaling and translating. In other words, when translating the rendered data set, the user had to first look at the hologram specifically associated with translation before performing the air-tap gesture. Only then was the user able to move the hologram in space.

PROTO-HEAD'S INITIAL DEBUT AND LAST UPDATES

Proto-HEAD debuted successfully at a small XR symposium at APL. Feedback on the project (and the ideas behind it) was almost universally positive, with critical criticisms focused entirely on the air-tap gesture. Several users encountered contextual misunderstandings. When manipulating the hologram, they performed the gesture and then moved their hand out of the HoloLens's view before performing the desired manipulation. In hindsight, the team recognized the obvious potential for user confusion in this action. A better approach would have been to use the HoloLens's clicker, a device that performs the equivalent of the air-tap gesture, to minimize the potential for misunderstanding.

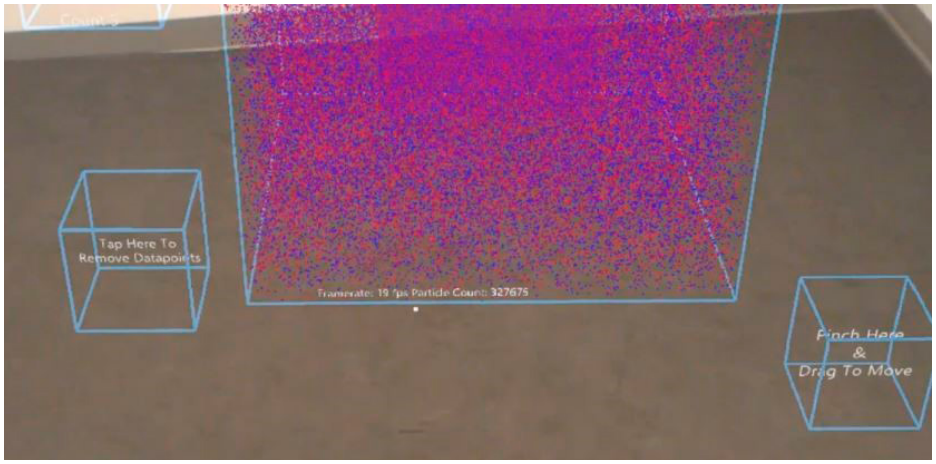

Figure 4. The last implementation of Proto-HEAD. This implementation rendered 327,675 data points, randomly placed within the central sphere, while maintaining a frame rate of approximately 19 frames per second.

Even though the initial showcase was successful, the application's limitations for rendering points were unsatisfactory. The particle system was quite simple to implement and use, but the default rendering software built for and used by Unity's particle system software limited the maximum number of data points that could be visualized.

The particle system code and other fundamental rendering code (known as graphics shaders) perform the required conversions and computations to understand exactly how to visualize information onto the screen. These shaders live and work within the computer graphics hardware, rather than the central processing unit. As such, the graphics processing hardware is specially designed to perform relatively simple computations in an extremely parallelized environment. Therefore, utilizing the graphics hardware as efficiently as possible required replacing the particle system and its standard shaders with custom shaders that create and render geometry directly.

The custom-built shaders, originally developed by APL's Optimized 3D Data Visualization Team, obviated the requirement for particle systems to create and visualize data points and, as an added bonus, simplified the project's source code. The last iteration focused on generating randomized data points within the space enclosed by the central cube to quickly test and realize new performance limitations (see Figure 4). Using the custom shader, Proto-HEAD was capable of rendering over a quarter-million data points before suffering any loss in performance, an order-of-magnitude increase over the default Unity shader.

CONCLUSION

Proto-HEAD's objective was to explore whether AR concepts provide a compelling benefit for data analysis and simulation over using a PC and, if so, how. While the PC has the advantage of being the mainstay of data analysis and simulation, AR provides a compelling benefit by visualizing 3-D models and data in a more natural 3-D environment. However, further exploration and understanding of such critical components as user interfacing and rendering is necessary to gain deeper insight into how AR concepts can become a compelling alternative to the PC.

Proto-HEAD gave the development team an opportunity to take their first deep dive into AR and its potential to provide new, immersive tools for research and development. Even though the application was simple by design, the experience provided significant insight into the challenges and potential approaches for visualizing and interacting with very large data sets. These insights will be useful for new applications dedicated to visualizing and interacting with large data sets and complex models because performance will be paramount to developing a highly successful AR application.

James L. Dean, Information Technology Services Department, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

James L. Dean is a computational engineer in the Information Technology Services Department at APL. He holds a BS in applied physics and mathematics from the University of Northern Iowa and an MS in aerospace engineering from Old Dominion University. He has a broad range of expertise including numerical modeling and simulation, data science and visualization, and software development for applications involving mixed reality, graphics rendering, and the Unity game engine. James has contributed to numerous mixed reality projects throughout APL with applications in missile defense, fluid dynamics analysis, construction, and systems engineering. His email address is james.dean@jhuapl.edu.