ARHeadsetKit: Bringing Affordable AR Headset Technology to the Masses

Philip Turner

Ocean Lakes High School

January 19, 2022

Author's Note:

Throughout my junior and senior school years, I took on a series of projects that ended with creating a mobile app and a software framework. Although my products did not reach my target audience, I gained valuable experience that I continue using to contribute to the open-source software community. Special thanks to Dr. Thomas Chan of CSU Northridge and Jennifer Kim for assisting me throughout the research process.

The source code and documentation for this research are published at github.com/philipturner/ARHeadsetKit. I can be contacted through email at philipturner.AR@gmail.com.

Abstract

Using Google Cardboard, I made the first AR headset experience that does not rely on expensive specialized hardware. By reconstructing and rendering the user's surroundings in VR, the AR MultiPendulum app replicated Microsoft Hololens. The process of creating the app involved heavy optimization for power efficiency and fixing bugs caused by using a GPU for general-purpose tasks. The final product was ARHeadsetKit, a high-level open-source framework for creating AR headset experiences.

Table of Contents

Introduction

Augmented reality has the potential to bring drastic change to everyday life in the near future. Smart glasses could bring numerous opportunities still in people's imagination, with paradigm shifts in every industry (Fig. 1). They bring an entirely new form of interaction with digital content - one more intuitive and responsive than touchscreens or keyboards. Up to now, this possibility has not been realized because AR headsets are incredibly expensive. Microsoft Hololens costs $3,500 and Magic Leap One costs $2,300 (Constine, 2020).

Furthermore, this technology's potential has been hampered by the general public's lack of knowledge about it. The average person does not know what the term augmented reality means, although they have used the technology in either Pokemon Go or Snapchat filters. That lack of knowledge will change once AR headset technology becomes more affordable, similar to how the first iPhone made smartphones mainstream. For several years, Apple has anticipated the future of AR headset technology and secretly worked on affordable smart glasses (Gurman, 2021).

Rumors about Apple releasing smart glasses originated in 2017 after Apple released ARKit. The smart glasses revolution was predicted to happen in 2020 (Buntz, 2011), which would have given Apple an advantage as an early adopter. Since that point, the expected deadline for Apple to release smart glasses has been continuously pushed back and is currently at 2025. After Apple failed to announce smart glasses at the iPhone 12 event, I decided to end the cycle of waiting for Apple to mainstream AR headset technology. I created ARHeadsetKit, which repurposes the Google Cardboard VR viewer as an AR headset by reconstructing the real world in VR.

Field	New Opportunities Brought by AR
Healthcare	Hololens allows surgeons to run through the procedure of surgery before performing it (Liebmann et al., 2020). This lets them simulate unexpected events during the surgery and reduces the error rate. When affordable smart glasses are released, many more surgeons could participate in this kind of training.
Education	ARHeadsetKit's tutorials introduced software developers to iOS by being engaging and hands-on. This was not the primary goal of the tutorials, but AR indirectly served as an incredibly effective education tool. When smart glasses become as ubiquitous as smartphones, they could be used in public schools.
Art	AR removes all boundaries to the size or quantity of virtual objects. Painters could draw on an infinitely large canvas without consuming material resources. In addition, AR lets someone change an object's size to visualize it more effectively. Architects could zoom in to add microscopic details to a building or shrink it down onto a table to visualize all of it at once (Noghabaei et al., 2020).
Virtual Reality	AR headsets also serve as VR headsets. When virtual objects obstruct all of the real world, the user is immersed in something entirely computer-generated. Smart glasses could allow users to experience VR games anywhere without having to carry a bulky headset such as Meta Quest (Robertson, 2019).
Workplace Productivity	An AR headset with high-enough display resolution could emulate a desktop monitor, letting someone have a multi-display desktop computer without having to pay for monitors (Qian et al., 2017). Furthermore, they could have a multi-display setup while away from home, just by pairing their headset with a laptop. This increased access to more productive workflows could overhaul the workplace just as working from home did during the COVID-19 pandemic.

Figure 1. Impacts affordable AR headset technology could have on multiple industries and aspects of daily life.

Google built support for Google Cardboard into the Android operating system (Amadeo, 2021). In 2021, they gave up on the Google Cardboard project to focus on AR instead of VR, removing first-class support for Cardboard from Android. The last Google Cardboard headsets made by Google recently sold out, with the only option now being third-party headsets.

Although Google Cardboard is tied to Android, ARHeadsetKit only runs on iOS. ARHeadsetKit must have exact measurements of each device to align with the user's eyes.

Access to exact measurements can only be feasible if ARHeadsetKit runs on a limited number of devices, and with a reliable method to identify which device an application is running on. I found that method with DeviceKit[1], combined with examining Apple's detailed schematics of every iOS device's measurements[2]. No such solution could be found on Android, which has numerous manufacturers and is well-known for inconsistencies.

Meta recently released smart glasses that take pictures, but these do not have AR capabilities (Rodriguez, 2021). They do not allow the user to manipulate virtual objects with their hands in an AR experience. In addition, they cost orders of magnitude more than my solution. Google Cardboard is essentially free, as its greatest barrier to acceptance is people's reluctance to order it from Amazon. My research also differs from a previous attempt to integrate Google Cardboard with AR (Perla & Hebbalaguppe, 2017), as only mine succeeded in providing a fluid user experience.

My research's purpose is to bring affordable AR headset technology to the masses, using already existing hardware. It runs on every iPhone starting with the iPhone 6S, with major enhancements to the experience on LiDAR-enabled devices.

[1]DeviceKit is an open-source framework published on GitHub.

[2]The Apple accessory design guidelines are published on developer.apple.com.

Methods

My research created two products: AR MultiPendulum and ARHeadsetKit. AR MultiPendulum was a free iOS app released in September 2021, which placed a physics simulation into an AR headset experience. In October, the source code from AR MultiPendulum was refactored into ARHeadsetKit to let developers besides me create AR headset experiences.

I received no external financial support while pursuing this research. The hardware used was either already owned or bought with personal money. This includes a Google Cardboard, iPhone 8 Plus, Mac mini, and an iPhone 12 Pro Max. I used additional iOS devices during beta testing for AR MultiPendulum. ARHeadsetKit was created using the Xcode IDE and the Swift programming language. Most development time was spent solving bugs and optimizing.

Optimization

An AR headset without a see-through display must maintain the highest possible frame rate at all costs. People can experience nausea when using VR (Arshad et al., 2021), and ARHeadsetKit provides usage sessions that can last as long as ten minutes. If an app's frame rate drops below 60 Hz, it will drastically increase the chance of the user experiencing nausea.

Prioritizing frame rate meant aggressively optimizing ARHeadsetKit for power efficiency. iOS throttles the display's refresh rate to 30 Hz when a device consumes energy quickly enough to lose control of its internal temperature. Running hand detection or rearranging the scene mesh 60 times per second caused overheating and dropped the refresh rate five minutes into an AR session.

The process of optimizing for power efficiency involved profiling an app prototype and gathering data from multiple sources. To detect when an iPhone became warm enough for the refresh rate to drop, the app was activated while connected to Xcode's debugger. Each test involved waiting as the iPhone progressed from the "nominal" to "fair" to "serious" thermal states[3]. A test ended when either the "serious" thermal state was reached, causing the refresh rate to drop, or 20 minutes passed. After modifying the app with a potential optimization, the test was repeated to detect whether the phone spent more time in the "fair" thermal state. To ensure thermal energy from previous tests had sufficiently dissipated, the phone was left inactive for at least 20 minutes between experiments.

Metal Frame Capture was used for precise measurements of the app's efficiency. This tool took a snapshot of all the work the phone's GPU executes over 1/60 of a second, measuring how long each operation took with microsecond precision. As a result, one could determine whether each minor change to GPU code increased or decreased execution time. To use Metal Frame Capture, a developer first had to capture a frame while the app was running and save it to a file. Due to bugs with the tool, performance data often failed to appear in Xcode. I had to open a file approximately 10 times to get data to appear once. Even when the data did appear, it only did for some of the work the GPU executed. Despite this usability limitation, Metal Frame Capture proved effective at providing performance data about ARHeadsetKit.

Debugging

Low-level optimization introduces major bugs in every programming context. Foregoing memory safety for performance allows for accidental corruption of the app's memory. For example, the GPU-accelerated mesh construction in AR MultiPendulum still has unsolved bugs. Occasionally, a small segment of the MultiPendulum simulation fails to render for one frame. This bug is caused by a fault in the algorithm for constructing the pendulum mesh.

[3] Documentation on iOS thermal states is located on developer.apple.com.

When making AR MultiPendulum, I spent a week investigating this bug, carefully analyzing screen recordings for the exact moment a graphical bug occurred. After reproducing the conditions that caused the bug, I narrowed down the exact time at which it happened. The app was modified to freeze the simulation at that time, and several screenshots of the graphical bug were taken. Although mistakes in source code were found, the bug had multiple independent causes. Some causes could not be found within the timeframe for releasing AR MultiPendulum.

In addition to capturing graphical bugs, screenshots were used to gauge the graphical quality of the app. When implementing scene color reconstruction, I compared screenshots before and after a potential enhancement to color quality. When implementing the interface render, which renders 2D text in AR, screenshots of rendered text were used. By zooming in on a screenshot, I observed whether the outlines of letters looked either smooth or pixelated.

Numerous bugs occurred during the creation of the scene color reconstruction algorithm. For example, one bug caused the reconstructed scene to turn dark when the user scanned a large volume. This bug had two causes. The first was an integer overflow, where GPU code tried to store an integer requiring 17 bits into a 16-bit register. As a result, the indices for triangles were corrupted, causing undefined behavior in the later stages of the algorithm. Finding this bug required scanning all of my GPU code, eventually arriving at the location of the bug. To ensure it never happened again, I replaced numerous instances of 16-bit integers with 32-bit integers.

The second cause was a data race, where the CPU and GPU accessed the same resource without proper synchronization. Scene color reconstruction sorts the world into a multi-level bounding volume hierarchy. The first level uses sectors spanning two meters in each dimension. The second level uses four-meter sectors and each subsequent level doubles sector width. To sort the scene, one must start with the largest level and subdivide sectors into smaller ones. While the GPU subdivides one level, the CPU prepares for the subdivision of the next.

During the data race, the CPU changed a buffer's data to reflect a future subdivision pass. The GPU was still executing the prior pass, and the buffer should not have been modified yet. Scene color reconstruction was most often tested in a room that required only one level in the bounding volume hierarchy. Since the room did not require more than one sorting stage, resource contention between the CPU and GPU never occurred. Months after publishing research on scene color reconstruction, the bug was spotted when attempting to scan a larger room.

Data Collection

Finding the second cause of the scene color reconstruction bug required gathering a massive amount of data about how the bounding volume hierarchy was constructed. I created a procedure that transformed the hierarchy's structure into text, allowing for validation of its integrity. First, I scanned a small room that encompassed the two-meter hierarchy level. Then, I moved outside of that room, expanding the scanned volume into the four-meter level. A textual representation of the hierarchy was logged several times each second, allowing me to confirm that it was corrupted when the bounding volume hierarchy gained a second level.

In addition to debugging, gathering significant amounts of data was required for artificial intelligence. On iPhones with a LiDAR scanner, I augmented Apple's 2D hand detection algorithm from Vision[4] with a 3D depth image. The result was a 3D hand reconstruction algorithm for facilitating interaction with virtual objects. Instead of using machine learning to create a neural network, a comparatively simple heuristic was created using trial and error. One component of that heuristic measured the length of each segment of the user's hand. This component then filtered out faulty length measurements caused by inaccuracy in Apple's hand detection algorithm and noise in the LiDAR depth image. Creating this component required gathering extensive data about the distribution of hand sizes and fine-tuning correction logic to limit the impact of faulty measurements on moving averages.

Originality of Research

When creating AR MultiPendulum, I set a deadline to create the app in a few months. Several ideas from my previous research were used to save time. The app's main content was a physics simulation created previously, called MultiPendulum[5]. In addition, I referenced a concept from an AR 3D modeling app prototype made at the start of my research into AR. This concept was a "central renderer" that rendered basic 3D shapes with only a few lines of code. It coalesced requests from numerous locations in source code into one centralized location, which abstracted the process of rendering. This rendering scheme was extremely productive and vital to producing AR MultiPendulum within the given deadline.

[4]Documentation for hand reconstruction in Vision is located on developer.apple.com.

[5]MultiPendulum is an online simulation based on the double pendulum.

Results

My research into AR resulted in three published products–a research paper on scene color reconstruction[6], AR MultiPendulum, and ARHeadsetKit. Overall, the research accomplished its initial objective: allow me to have an AR headset experience without waiting for the smart glasses revolution.

Before publishing to the App Store, I verified that AR MultiPendulum satisfied multiple criteria. It needed a holographic interface controlled by a hand in front of the device's camera, instead of using the touchscreen. On AR MultiPendulum and other ARHeadsetKit-based apps, the touchscreen is not the primary input mechanism. When it is used, it is treated like a massive button where the location of touch does not matter. It differs from other iOS apps that use the touchscreen as the primary input mechanism, where the location of a touch matters.

I had to thoroughly explain how to interact with the control interface in AR MultiPendulum's tutorial. If AR MultiPendulum was published a decade in the future, users would have experience with interacting with holographic interfaces for smart glasses and the explanation might have been unnecessary. Furthermore, it had to be obvious that when the user's hand moved in front of an interface element, that element was selected (Fig. 2).

The Apple App Store had strict rules for app quality. My submission was rejected multiple times, with instructions on how to fix it after each rejection. One rejection was caused by an irrelevant keyword in the app's subtitle, and another rejection was caused by a crash. As the last criterion, AR MultiPendulum had to let the user interact with MultiPendulum and not have any bugs that impact the user experience.

[6]Since research on scene color reconstruction was done well before ARHeadsetKit, it is not officially part of this research.

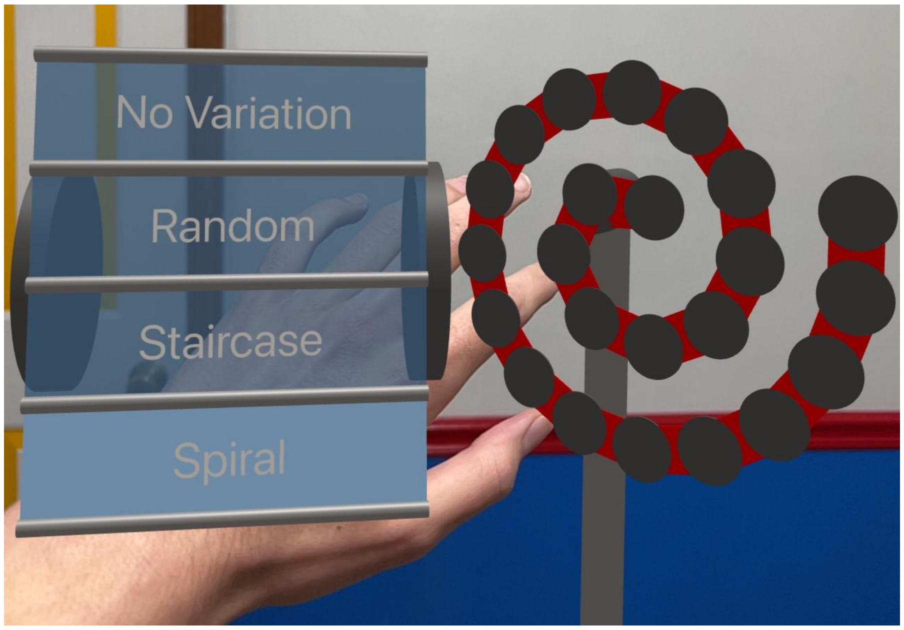

Figure 2. Left: control interface, shown in blue. The user's hand covers the "spiral" panel, causing it to highlight in opaque blue. Right: the MultiPendulum simulation changes its starting conditions to reflect the control interface.

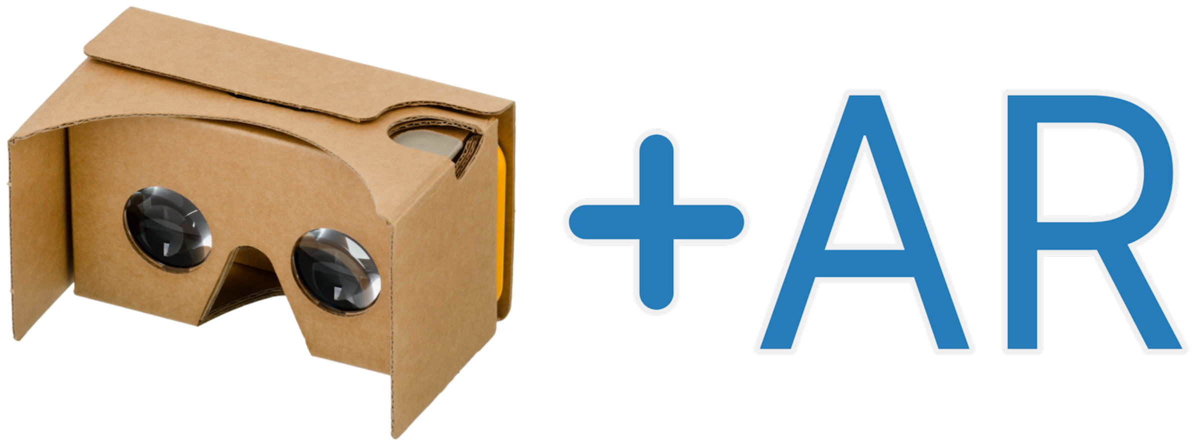

Figure 3. The ARHeadsetKit logo, one of several images that appear in the framework's documentation.

ARHeadsetKit also had to meet certain criteria before being published on GitHub. It needed enough documentation for any developer to learn how to use it. Its tutorial series had to teach the developer to design an AR headset experience, even if they could only test an app prototype in handheld mode. The tutorials needed several images to keep the reader engaged through five hours of content (Fig. 3).

I had to ensure that an app made with ARHeadsetKit could successfully compile. I tested it by refactoring AR MultiPendulum to depend on ARHeadsetKit. I did not publish the rebuilt AR MultiPendulum to the App Store because that might introduce new bugs without adding any content. However, I did publish the new source code on GitHub.

Discussion

Although AR MultiPendulum worked, getting it in the hands of its target audience–the general public–proved challenging. For this reason, I transformed it into a software framework (ARHeadsetKit) and changed my target audience to software developers. The tutorials included with its documentation could be followed by any developer, even one without prior iOS experience. Two people went through the entire tutorial series, neither of whom were familiar with creating iOS apps.

Learning ARHeadsetKit does not require owning a headset. This distinguishes it from Unity, a high-level game engine that requires owning Hololens to design AR headset experiences. Developers can design a handheld AR app, which ARHeadsetKit automatically translates into an AR headset experience. Neither of the people who took ARHeadsetKit's tutorials owned Google Cardboard at the time of doing so.

Several utility functions were created while building AR MultiPendulum and ARHeadsetKit. These allow me to more easily debug Metal commands and access instruction-level parallelism on the CPU. These were transferred into an isolated framework, called ARHeadsetUtil. ARHeadsetUtil is now used for applications not related to AR, such as digital signal processing[7].

Although the creation of ARHeadsetKit was a scientific endeavor, it was also a learning process. It began with multiple attempts to find a suitable graphics API. I initially explored Unity and Unreal Engine but needed to use Metal for its flexibility. Without any prior experience with low-level 3D graphics, I made mistakes that postponed the release of ARHeadsetKit and harmed the quality of data during research on scene color reconstruction.

[7]MetalFFT uses ARHeadsetUtil to make its code easier to read and debug.

For example, I incorrectly assumed that a GPU runs similarly to a multicore CPU. As a result, code for scene color reconstruction was needlessly complex and inefficient. I could have saved dozens of hours by understanding a GPU's SIMD architecture from the beginning.

I also spent most of the time developing ARHeadsetKit with major detriments to the quality of color data. Initially, I assumed each iOS device's back camera had a fixed horizontal frame of view of 58.903 degrees. In actuality, that number varies between devices and changes throughout an AR session. After using the correct frame of view provided by ARKit, I observed a massive improvement in the quality of scene color reconstruction. This improvement happened well after research on scene color reconstruction was published (Fig. 4 & 5).

Before Correcting FOV After Correcting FOV

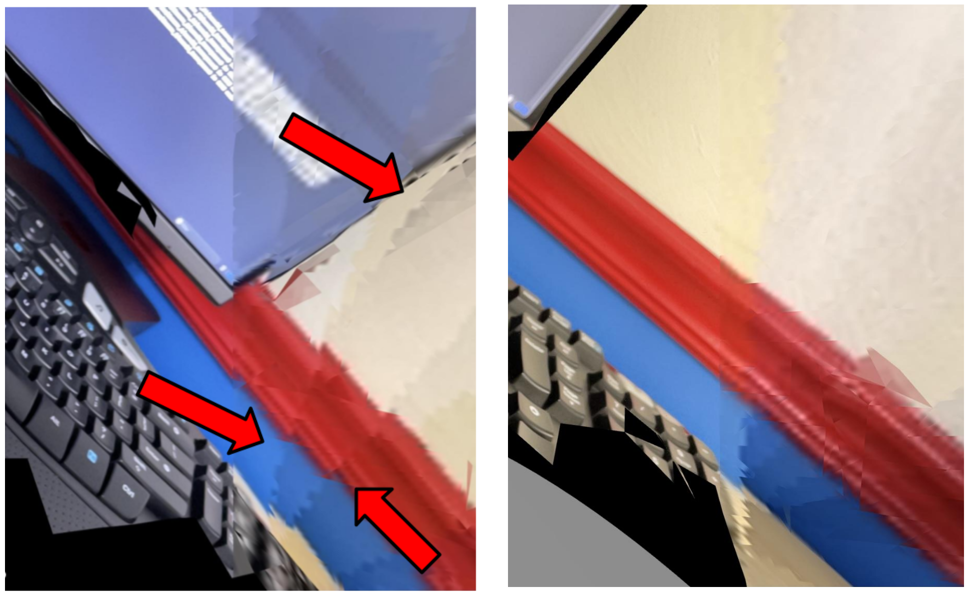

Figure 4. Left: several jagged lines appear, created on the border of the camera's FOV each time color is re-assigned to 3D geometry. Right: with the frame of view corrected, discontinuities between color updates are almost invisible. In each image, the left half is a real-time video stream and the right half is reconstructed color.

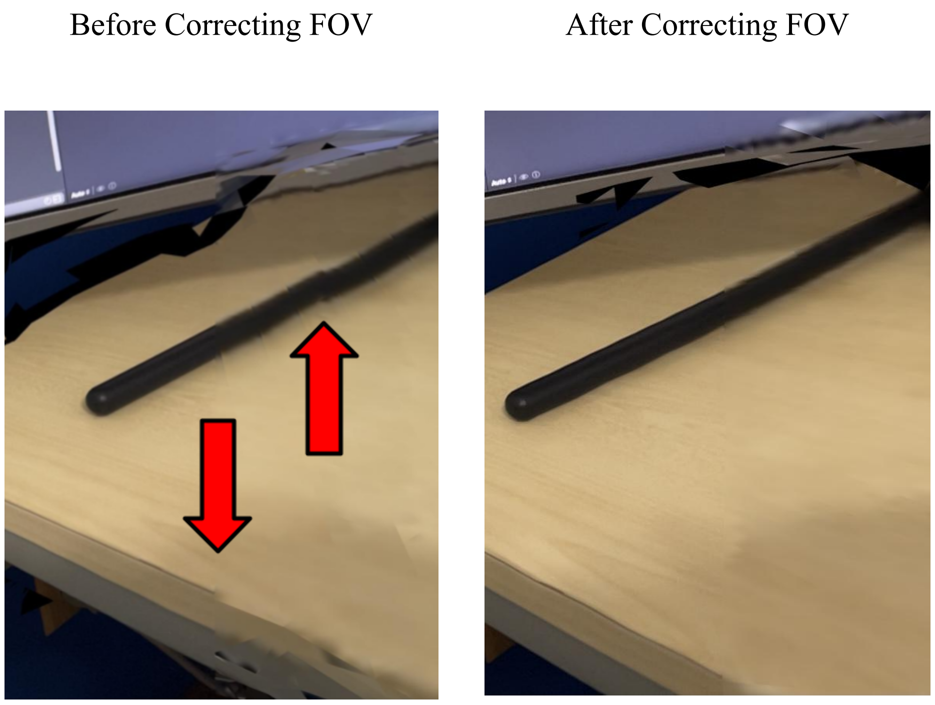

Figure 5. Left: discontinuities in the table's edge and the black monitor stand reveal the boundary between real and reconstructed color. Right: the frame of view correctly aligns with the camera, making the boundary indistinguishable. In each image, the left half is a real-time video stream and the right half is reconstructed color.

Several improvements could have been made to ARHeadsetKit. It integrates with SwiftUI in a strict way that limits what content a software developer can add to an app. ARHeadsetKit could have provided a way for experienced developers to manually assemble the SwiftUI views required for rendering AR headset experiences.

Third-party Google Cardboard manufacturers vary in the size and spacing of plastic lenses. This inconsistency causes variation in the behavior of lens distortion, which is not accounted for in ARHeadsetKit. I planned to create an AR headset experience for fine-tuning lens distortion correction to account for such variation. More than a week was spent designing a highly optimized algorithm for modifying the behavior of lens distortion correction in real-time, without dropping frame rate. Due to time constraints, the interface for fine-tuning lens distortion correction was never implemented.

Conclusion

While ARHeadsetKit does not fully replicate Microsoft Hololens, it allows people to interact with virtual objects in a simulated representation of their surroundings. As a result, they can experience what AR headset technology is capable of first-hand, without waiting for the smart glasses revolution to happen.

In late 2022, Apple is expected to release its first AR/VR headset. Although very expensive, many more developers will create software for Apple's headset than ARHeadsetKit. ARHeadsetKit will remain the cheapest and most accessible way to experience an AR headset, even though few people use it.

Works Cited

Amadeo, R. (2021, March 03). Google's VR dreams are dead: Google Cardboard is no longer for sale. Retrieved from https://arstechnica.com/gadgets/2021/03/googles-vr-dreams-are-dead-google-cardboard-is-no-longer-for-sale/

Arshad, I., De Mello, P., Ender, M., McEwen, J. D., & Ferré, E. R. (2021, November 17). Reducing cybersickness in 360-degree virtual reality. Retrieved from https://arxiv.org/abs/2103.03898

Buntz, B. (2011, November 11). Theoretical physicist Michio Kaku predicts the future of healthcare. Retrieved from https://www.mddionline.com/news/theoretical-physicist-michio-kaku-predicts-future-healthcare?scrlybrkr=fdbd9e4f

Constine, J. (2020, March 11). Desperate to exit, a $10B price tag for Magic Leap is crazy. Retrieved from https://techcrunch.com/2020/03/11/magic-steep

Gurman, M. (2021, January 21). Apple's first headset to be niche precursor to eventual AR glasses. Retrieved from https://www.bloomberg.com/news/articles/2021-01-21/apple-s-first-vr-headset-to-be-niche-precursor-to-eventual-ar-glasses?sref=ctSjKj2N

Liebmann, F., Roner, S., Von Atzigen, M., Wanivenhaus, F., Neuhaus, C., Spirig, J., Scaramuzza, D., Sutter, R., Snedeker, J., Farshad, M., Fürnstahl, P. (2020, January 17). Registration made easy -- standalone orthopedic navigation with HoloLens. Retrieved from https://arxiv.org/abs/2001.06209

Noghabaei, M., Heydarian, A., Balali, V., & Han, K. (2020, May 6). A survey study to understand industry vision for virtual and augmented reality applications in design and construction. Retrieved from https://arxiv.org/abs/2005.02795

Perla, R., & Hebbalaguppe, R. (2017, June 5). Google Cardboard dates augmented reality: Issues, challenges and future opportunities. Retrieved from https://arxiv.org/abs/1706.03851

Qian, L., Unberath, M., Yu, K., Fuerst, B., Johnson, A., Navab, N., & Osgood, G. (2017, October 2). Technical Note: Towards virtual monitors for image guided interventions - Real-time streaming to optical see-through head-mounted displays. Retrieved from https://arxiv.org/abs/1710.00808

Robertson, A. (2019, April 30). Oculus Quest review: A great system with a frustrating compromise. Retrieved from https://www.theverge.com/2019/4/30/18523000/oculus-quest-review-vr-headset-price-specs-features

Rodriguez, S. (2021, September 9). Facebook just announced its new Ray-Ban glasses - I've been using them for a couple of days, here's what they're like. Retrieved from https://www.cnbc.com/2021/09/09/facebook-ray-ban-stories-smart-glasses-review.html