Industrial Augmented Reality: Mobile or Web?

Jacob Gibson

11/10/2020

Table of Contents

Abstract

As the world moves to Industry 4.0, Augmented Reality (AR) applications will move to the forefront of the way people work, shop, and play. In accordance with that, applications like Siemens Teamcenter Visualization that are currently used to review product designs and model factory floors will need to be updated to provide that AR functionality. First, I create an Android AR app that can stream data from a Teamcenter Visualization instance running on a desktop to an Android phone. Next, I investigate the other plausible route for an AR upgrade to Teamcenter Visualization - a web app utilizing Siemens PLM VisWeb. Finally, I compare the two based on usability, accessibility, ease of development, and performance metrics. Ultimately, I provide guidance as to whether the mobile or web platform is currently the most suited to AR development.

Introduction

Siemens is a global company focused on industry, energy, and healthcare. It is the largest manufacturer in Europe, but it also produces a large array of software in its Digital Industries Software division. The CAD software they create is used by engineers, architects, and other professionals all around the world. One such software is Siemens Teamcenter Visualization (TcVis). Used by 24 of the top 25 automotive manufacturers and 9 of the top 10 aerospace manufacturers, TcVis is an application that provides digital markup and visualization features for the product lifecycle. Users can access 2D and 3D design data, collaborate with other users, and even view that data in Virtual Reality (VR). These powerful tools allow users to create and analyze digital prototypes that surpass physical prototypes in cost, time, and quality. Users can also visualize digital twins of existing objects, which can be combined with VR and collaboration features to do things like maintenance training in a safe learning environment.

Moving forward, Siemens would like to add AR capabilities to Teamcenter Visualization to enable even more use cases for the application. As technologies improve and manufacturers adopt things like AI, machine learning, and IoT, quality becomes an even more complex goal to attain. Businesses aiming for quality will need to have greater speed, transparency, and accuracy that is hard to attain with the technologies they are using today (Radziwill, 6). AR can solve many of these problems by providing an accurate, fast, and easy to use way to visualize products in any stage of the product lifecycle. Hence, adding AR capabilities to TcVis would add an incredible amount of value to the app.

Given that Teamcenter Visualization can have very high graphical requirements, it was decided that the best AR solution would be a web or standalone app that can connect to an instance of Teamcenter Visualization that acts as a server. A distributed system like this is common in AR apps due to the relative weakness of mobile devices compared to desktop applications (Lima, 2).

A working prototype for the web app already exists in the form of Siemens PLM VisWeb's AR features. This app could be used as the basis for web app metrics, but a standalone app needed to be created. The app needed to be able to connect to an instance of Teamcenter Visualization over a TCP connection, display the streamed data in AR, and react to changes in the Teamcenter Visualization instance in real time. I detail the goals and results of this step of research in its corresponding sections under Goals and Accomplishments and Results.

In the next phase of the research, the two apps needed to be analyzed on the bases of usability, accessibility, ease of development, and performance. Usability and Accessibility were chosen so that the app can be used by as many people and in as many ways as possible. Given that the app will be primarily controlled through the Teamcenter Visualization instance running on a desktop, this metric considers things like available platforms and possible input methods. Ease of Development is the most minor of the metrics chosen, but valuable

information nonetheless. This metric considers things like the language the app is written in, how easily it interfaces with existing libraries, and available development environments. Finally, the Performance metric aims to guide Siemens to the app that will provide the most perceived value to their customers. The two apps will be measured on things like frames per second, latency from the desktop to the phone, etc.

Background

Industry 4.0

The rise of Internet of Things (IoT) in the world is leading to what many experts are calling Industry 4.0. As the world embraces IoT, the entire product lifecycle will be augmented. Producers will have access to powerful new design and manufacturing options enabled by more efficient factories and big data. Consumers, on the other hand, will have access to total customization in the products they consume. In an IoT factory, every stage of production will have access to knowledge of every other stage. Every designer, assembler, and warehouse worker will be able to see where each product came from and where each is headed at any time. This tight integration will allow factories to shift from the mass production of generalized goods to the mass production of specialized goods (Becker, 2). For example, a consumer looking for a new pair of shoes could log on to the company's website, design a shoe with the exact colors and look that they desire, and purchase that shoe for no extra cost over a "standard" model. Better yet, a consumer in the market for a new car could select the exact parts they wanted in their car before the car is even made.

These changes may seem like a sci-fi future today, but these changes are coming sooner than one would expect. In fact, many shoe manufacturers are already offering totally customizable shoes and many factories are already accepting IoT devices for monitoring and maintenance. It will be important, then, that software stays on pace with these changes and

provides the new solutions that producers and consumers will need in Industry 4.0. Central to those new needs is VR and AR (Becker, 5). Producers need these immersive tools to better monitor their factories and analyze the products that their users are creating, while consumers will need these tools to visualize the product they're buying in the environment around them. For example, AR technology enables a consumer to try on a shoe right in their living room or see a new car in their driveway. Siemens Teamcenter Visualization already has powerful VR functionality but will certainly benefit from getting in on the ground floor of Industrial AR tools.

Industrial Augmented Reality

Industrial Augmented Reality (IAR) is a subset of AR defined as Augmented Reality for product design, manufacturing, training, maintenance, and logistics (Büttner, 1). Pokémon Go is a common example of commercial or recreational AR, and while impressive in its own right, it lacks many of the features that IAR provides. IAR places the focus on precision, accuracy, and performance above all else. There has already been a wealth of research done into IAR applications, but most of it tends to focus on AR projections and AR head-mounted-displays (Büttner, 3). This research aims to show the viability of mobile IAR, but the web app and Android app both allow for head-mounted-display IAR. Additionally, most prior IAR research has been into modeling factories and increasing worker performance (Büttner, 4). There is a noted void of IAR for product lifecycle management in both academia and industry, which is something this research aims to fill.

Goals

Android App

The prototype Android App created in this research must meet the following criteria:

- **Connect to a Teamcenter Visualization instance over TCP.** The Teamcenter Visualization instance will act as a server and send a bytestream of the currently rendered model to the Android app. The Android app will render that streamed model in the Augmented Reality scene.
- **Include Augmented Reality functionality.** The app will utilize the Android ARCore API to render an AR scene. Direct Model libraries will then be integrated with the ARCore rendering to allow for the easy addition of streamed data from Teamcenter Visualization.
- **Allow for basic tracking of the streamed model through the placement of anchors.** These anchors will be placed on a flat surface in the AR scene by tapping the screen. Once an anchor has been placed, the streamed model will stick to that point until a new anchor is placed.
- **Ensure that all Teamcenter Visualization functionality will be properly streamed to the Android app.** Teamcenter Visualization's powerful functionalities should be sent to the Android app "for free." This means that an AR user could easily add measurements, markups, cross sections, etc. to their model and see the altered model in real time on the app. This goal is here to ensure all of the changes these functions create on the model are captured in the UPCS stream and sent along to the app. Hence, changes made here will actually be made to Teamcenter Visualization and not the Android app.

Comparison and Metrics

The following comparisons will be made, and metrics will be collected between the apps:

- **Usability and Accessibility.** The two apps will be compared on basic usability and ease of set-up. Both will initially have very minimal user interfaces, so this comparison will also include a survey of the different devices each app would be available on.
- **Ease of Development.** The two apps will be compared from a coding perspective. This will include things like the programming language used and the development

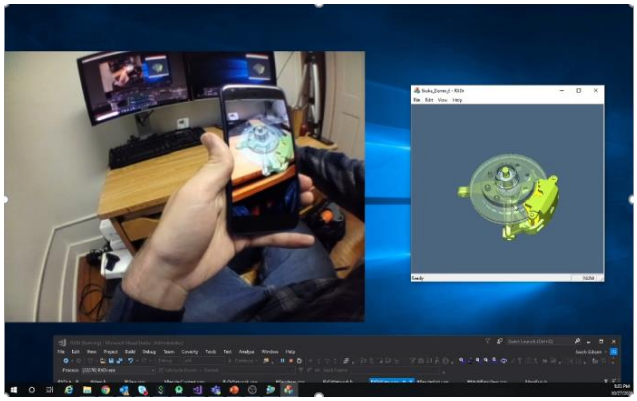
environment required. Furthermore, the apps will be compared on how easy or hard it would be to integrate existing Siemens rendering libraries.

- **Performance.** Finally, the apps will be compared on a few performance metrics. Particularly, the framerate of the apps will be measured with a variety of models to see which performs better with different kinds of data on a real phone. Additionally, the latency between the phone and desktop will be measured for each solution with a variety of models.

Accomplishments and Results

Android App

The app created has met all but the last of its goals. The app started off as a Siemens's Direct Model renderer that connected to a simplified Teamcenter Visualization viewer called RxDr. This simplified viewer has the same rendering capabilities and code as Teamcenter Visualization, but without all the extra features on top like measurement, cross section, etc. RxDr was used to provide a more lightweight development environment, as the developer did not need to run a full Teamcenter Visualization instance. The app creates a TCP connection with the RxDr server and then receives data over a UPCS stream. This same connection can relatively easily be created with a Teamcenter Visualization instance by moving the RxDr code into a new module in the Teamcenter Visualization code. Hence, the first goal has been met.



Next, the Direct Model rendering used in the initial app would need to be integrated with the OpenGL ES rendering of an Android ARCore app so that the app could leverage all of the ARCore functionality. Luckily, Direct Model is also built on OpenGL ES rendering and so the integration of the two went very smoothly. The app uses ARCore rendering as the base, and then adds the Direct Model rendering on top, completing the second goal.

After that, the app would need to have some basic tracking. Eventually, the app will utilize industrial-strength tracking through the VisionLib library. Using VisionLib, the app will be able to map the AR model onto a physical version of that same model in the real world, allowing users to digitally edit the virtual version and see their changes overlaid on the physical version in AR. As a proof of concept and as a form of tracking for users without a physical version of their model, the app here will implement ARCore's anchor-based tracking. ARCore will pick up on flat surfaces around the user, after which the user can click any point on the surface to anchor the model. The model will snap to that location and remain there until the user taps again to move the anchor somewhere else. Even if the anchor moves off of the screen, ARCore will remember where the anchor was and find it again when that point returns on screen. This basic tracking satisfies the third goal of this research.

The final goal, streaming all Teamcenter Visualization functionality to the app, has not yet been reached. Originally, this goal seemed relatively straightforward. The code in RxDr that

handled the TCP connection and UPCS stream needed to be ported into TcVis in a new module. The new module, VP Augmented Reality, was created but it turned out that porting the code over was a more involved process than originally thought. In RxDr, the streaming code could rather easily be added to the basic rendering the app provides. Unfortunately, TcVis's complexity means that there's a lot more to be done than just porting the code. A new ribbon would need to be made in the UI, the new module would need to get many more interfaces than RxDr needed, and some of the code would need to be updated for the latest version of Direct Model. In total, this process would probably take a full-time developer a little less than a month.

Comparison of the Apps

Of course, this research is most relevant to Siemens so that the management can determine the best course of action moving forward. Thus, it's important to compare the Android AR and VisWeb AR solutions. Below, the apps are compared on the criteria of Usability/Accessibility, Ease of Development, and Performance.

Usability/Accessibility Metrics

One of the most important differences between the two apps is the platforms that they can be run on. The app created in this research runs on Android, so it can be used on any Android device such as mobile phones, tablets, and even certain AR Head Mounted Displays. VisWeb AR, on the other hand, uses WebXR. This means that it can be used by any device that has access to a browser with WebXR support. Many tablets and mobile phones have access to such browsers, and it's possible that some Head Mounted Displays could as well. Plus, WebXR is available on both iOS and Android, widening the potential userbase further.

The apps also have some user interaction differences. Currently, both apps can detect flat surfaces and place the model on top of the surface. The models can also be anchored at any point on the surface. In the Android app, the user can then rotate and resize the model on

their desktop server and the changes will be sent to the app. In VisWeb, on the other hand, the user can use two fingers in a pinching motion to resize the model the same way a user could zoom on a webpage or photo. These interactions highlight a key difference between the two apps – the Android app tends to rely on the server for user interaction while the VisWeb user interacts with the app directly. While this seems like a clear win for VisWeb, it's a bit of a double-edged sword. In VisWeb, every bit of functionality needs to be added to the app directly, while the Android app gets all the TcVis functionality for free through the desktop server. The user could perform any bit of TcVis functionality on the server – for example a cross section – and the user will see that cross section in AR automatically.

Overall, the VisWeb solution currently wins out in terms of Usability and Accessibility. WebXR makes it accessible on more platforms than just Android devices, and it currently has more user interaction in the actual AR app instead of on the desktop server. Despite this, the Android app still has potential, and a possible UI solution is explored in the Future Work section.

Ease of Development

As mentioned in the Usability/Accessibility section, Android has a development advantage in that it will get access to TcVis functionality automatically. Android also has a few other benefits over VisWeb in terms of ease of development. Since the Android app runs on C++, existing Siemens libraries can relatively easily be integrated into the app. Contrast this with PLM VisWeb, where existing libraries need to be ported over into JavaScript calls that the app can use.

Another helpful advantage for Android is that both ARCore and Siemens's Direct Model render using OpenGL ES. This made the integration of the two very smooth, but unfortunately VisWeb doesn't have the same benefit. WebXR is also built on top of OpenGL ES, but VisWeb itself doesn't use Direct Model for its rendering at this time. Eventually, Direct Model may make it into VisWeb, but until it does the Android app will have that advantage.

While VisWeb won out in terms of Usability and Accessibility, the Android app is the clear winner in terms of Ease of Development. Given the same investment, I believe the Android app could be developed more quickly than the VisWeb.

Performance Metrics

Unfortunately, this research was concluded before Performance Metrics could be collected. The necessary changes to collect metrics like framerate and server latency haven't yet been added to either solution, but it's also being worked toward for both solutions. Additionally, the COVID pandemic created challenges for the developers to test both solutions on the same device.

While there aren't any performance comparisons between the two apps yet, it will be interesting to see if there are differences between the two. WebXR uses WebGL to render its graphics, which is itself based on OpenGL ES. WebGL is the preferred method for mobile real-time processing due to its similarities to OpenGL ES, so we can expect similar performance in terms of raw rendering capabilities (Ioniță, 4). WebXR also uses ARCore behind the scenes when used on an Android device, so both apps will use the same AR API when tested on a single device. The difference will lie in the extra efficiency that WebXR or Direct Model can provide, if any.

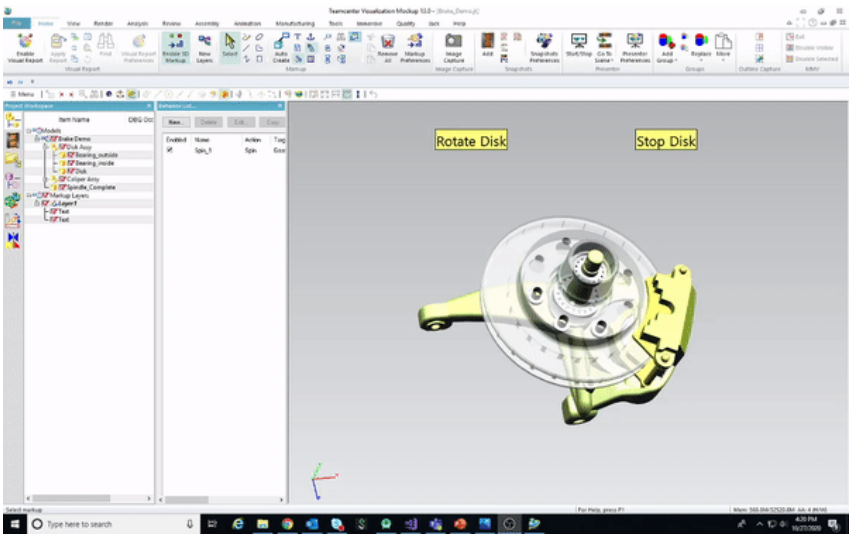
Conclusions and Further Work

Overall, this research was very successful. The prototype created provides a great potential starting point for Siemens AR that relies on desktop Teamcenter Visualization. It has superior ease of development over the VisWeb solution, worse usability and accessibility, and unknown but most likely similar performance. Of course, there is still plenty of room for the app to grow and for this research to continue in the coming months.

The first area of growth would be to carry out the performance comparison between the apps. This work is already underway and both teams are working together to get access to a tablet to test both solutions on. In total, it would take a full-time developer to complete the performance comparison in around a week, making this next step attainable relatively soon.

Next up would be to finish up the final goal of this research. Connecting the Android app to Teamcenter Visualization would be a huge step forward for the prototype, but as said in the Accomplishments and Results section, would take a full-time developer around a month to complete. Hopefully, project management will see the progress made so far and the performance comparison and put more resources on the project. As of now, this project is only worked on in free time at the end of each Agile iteration, so being accepted by the project management would be a great boon to the project's speed.

After doing the performance comparison and connecting the app to TcVis, we would like to add scene authoring to the app. With scene authoring, users could add buttons and other controls to TcVis on the desktop, and then see those controls in AR on the app. This functionality would add a wide variety of UI options for the user and truly transition the app from a prototype to a realized product. Below is a mockup of the scene authoring using simple buttons that make the disc in the brake assembly spin or stop.



References

Becker, Christian, et. al. “A Survey on Human Machine Interaction in Industry 4.0.” Feb. 2020. 45 pages. https://arxiv.org/pdf/2002.01025.pdf

Büttner, Sebastian, et. al. “The Design Space of Augmented and Virtual Reality Applications for Assistive Environments in Manufacturing: A Visual Approach.” June 2017. 8 pages. https://thomaskosch.com/wp-content/papercite-data/pdf/buettner2017design.pdf

Ioniță, Cristian and Bărbulescu, Alexandru. “Real-time Video Processing in Web Applications.” Sept. 2015. 4 pages. https://arxiv.org/ftp/arxiv/papers/1712/1712.02438.pdf

Lima, João Paulo, Teichrieb, Veronica, and Kelner, Judith. “A Standalone Markerless 3D Tracker for Handheld Augmented Reality.” Feb. 2009. 15 pages. https://arxiv.org/ftp/arxiv/papers/0902/0902.2187.pdf

Radziwill, Nicole. “Let’s Get Digital: The many ways the fourth industrial revolution is reshaping the way we think about quality.” Oct. 2018. 10 pages. https://arxiv.org/ftp/arxiv/papers/1810/1810.07829.pdf

ARCore (Version 1.20.0) [Software Development Kit]. Retrieved from https://developers.google.com/ar/develop

WebXR (Version 20200724) [Application Programming Interface] Retrieved from https://www.w3.org/TR/webxr/