ARCalVR: Augmented Reality Playground on Mobile Devices

Menghe Zhang
University of California, San Diego

Karen Lucknavalai
University of California, San Diego

Weichen Liu
University of California, San Diego

Kamran Alipour
University of California, San Diego

Jürgen P. Schulze
University of California, San Diego

ABSTRACT

With the development of ARKit and ARCore, mobile Augmented Reality (AR) applications have become popular. Our ARCalVR is a lightweight, open-source software environment to develop AR applications on Android devices, and it gives the programmer full control over the phone's resources. With ARCalVR, one can do 60fps marker-less AR on Android devices, including functionalities of more complex environment understanding, physical simulation, virtual object interaction and interaction between virtual objects and real environment.

KEYWORDS

Augmented Reality, Virtual Reality, ARCore, Android, Mobile Development, Realistic Lighting, Spherical Harmonics

ACM Reference Format:
Menghe Zhang, Karen Lucknavalai, Weichen Liu, Kamran Alipour, and Jürgen P. Schulze. 2019. ARCalVR: Augmented Reality Playground on Mobile Devices. In *Proceedings of SIGGRAPH '19 Appy Hour*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3305365.3329732

1 INTRODUCTION

ARCore provides the functionality needed to use the Android phone for AR applications: camera-based 6 degree of freedom motion tracking, as well as recognition of flat surfaces. The latter can be used to place virtual objects in the physical environment. Our aim is to build an opensource, lightweight Android native framework to easily develop Android applications. Our framework integrates tracking and environment understanding features for basic AR applications. We also integrated a physics module, and implemented a lighting estimation module. For better user experience, ARCalVR has a complete menu system to enable different ways to interact with virtual objects.

2 SYSTEM STRUCTURE

Figure1 shows the overview of system structure and its dependencies. ARCalVR is an Android Native software framework to create Augmented Reality applications. We will go into details of each component in the following sections,

SIGGRAPH '19 Appy Hour, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6306-8/19/07.
https://doi.org/10.1145/3305365.3329732

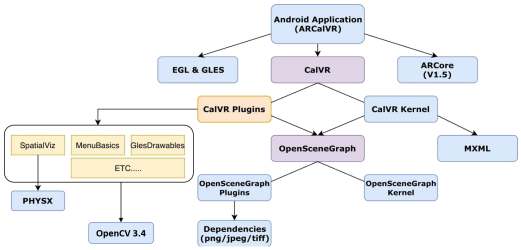

Figure 1: System structure and its dependencies

2.1 Hardware and software platform

ARCalVR is targeted on Android platform For development and testing, we use Samsung S9 with Qualcomm's Snapdragon 845 SoC and Adreno 630 GPU running at 710 MHz, API level 26. ARCalVR is an AR extension to virtual reality visualization framework CalVR.[Schulze et al. 2013] CalVR is a C++ open source framework that implements the typically used VR functionality of middleware. ARCalVR extends it with AR tracking system and ports to mobile devices.

2.2 Rendering and Display

We focused on how the graphical data moves through the system based on Android graphics architecture. The low-level component EGLSurface is provided by Android. Using EGL calls, we can create and access windows through the operating system to render our scene. GLSurfaceView provides a helper class to manage EGL contexts, inter-thread communication and interaction with the Activity lifecycle.

After creating and configuring a renderer to GLSurfaceview, we can then manipulate GL context on both Java and Native C++ sides. CalVR [Schulze et al. 2013] takes high-level graphis toolkit OpenSceneGraph (OSG), written entirely in Standard C++ and OpenGL, which enables us to write raw GL codes, shader files and adapt them as part of the scene structure.

2.3 User-Interaction Interface System

ARCalVR provides a menu module and mobile-device-adaptive interactions for users to interact with the system. We placed the menu in our 3D environment when it is called, facing towards the user. To map a linux based framework to mobile framework, we implemented a multi-finger-detection to replace mouse clicks.

To interact with virtual objects we back-project the touch position to the near plane of the camera into our 3D environment as shown on Figure 2. We manipulate objects with one finger drags, and have two different modes to handle translations and rotations.

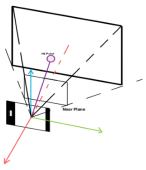

Figure 2: Hit test

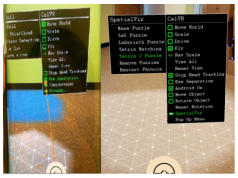

Figure 3: Menu System

2.4 Lighting

We implemented three different approaches for lighting in our Android app: ARCore Native, Single Source, and Spherical Harmonics. In our demo application, the user can select on a per object basis which of the three lighting methods to use.

2.4.1 ARCore Native. AR Core Native comes directly from ARCore. This method calculates the average pixel intensity of the captured image, and renders the objects based on that average brightness. This is our lighting baseline to compare other lighting estimation implementations.

2.4.2 Single Source. Single Source builds on the functionality available in ARCore to improve the lighting of the scene. We calculate the pixel location of the brightest point in the captured image. This is then used as a 3D point light to provide additional lighting to help create a more realistic rendering for diffuse and specular surfaces.

2.4.3 Spherical Harmonics. Spherical Harmonics lighting relies on a detected environment map, so we implemented two ways to adaptively build up this environment map. So as the user continues to use the app, we gather more images and create a more complete and accurate environment map.[Ramamoorthi and Hanrahan 2001]

- Use OpenCV to stitch input images into a panoramic image. This method is slow but accurate.
- Use projection and view matrix from ARCore to directly project the image onto a sphere.

To compute the Spherical Harmonics on this environment map we downsize the image and the following calculations are completed every 50 frames.

- We compute the integral of the environment over one dimension ϕ. This calculation is parallelized and completed in multiple threads.
- Once this is complete, one thread then computes the resulting 9 Spherical Harmonic coefficients by integrating over the other angle θ.

Every frame these Spherical Harmonic coefficients are sent to the shader to calculate the diffuse colors and render objects based on the current environment map. The flow of the Spherical Harmonics Algorithm can be seen in Figure 4.

3 DEMO APP: SPATIALVIZ

Since we were creating an AR extension to CalVR we wanted to experiment with taking a CalVR plugin, that was built for the Linux based version and see if we could then run it on the Android AR environment. The plugin we decided to use was a Spatial Visualization trainer.

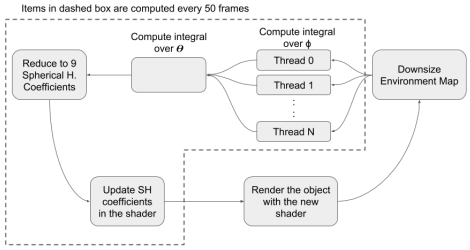

Figure 4: Showing the flow of the Spherical Harmonics Algorithm used within the Android app. Note that the Spherical Harmonics are calculated and updated every 50 frames.

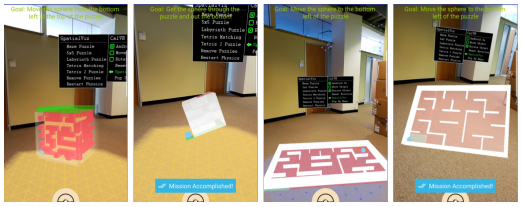

Figure 5: Showing three of the different puzzles created in the Training app. Each involve the rotation and manipulation of the different puzzles to get the ball to go where indicated.

The goal of this plugin was to provide a basic application that takes advantage of the Virtual Reality environment. It creates virtual 3D puzzles that can be rotated and moved around in a 3D space. This way the user can hopefully receive the same or comparable benefits they would as if it were a real 3D object [Cathrine 2010]. In Figure 5 you can see three of the different puzzles created in the Trainer. Each of these involve the use of the PhysX library to create the interaction and movement between the puzzles and the ball. You can also see the toast messages displayed to the user when they have completed the puzzle.

As mentioned the puzzles in our Spatial Visualization Trainer make use of the of PhysX library, and so we felt that this along with the nature of the application would be a great plugin to test the incorporation of ARCore with existing CalVR applications.

4 CONCLUSION

Based on CalVR, ARCalVR integrates ARCore as tracking system to build our Android Native framework for the development of Augmented Reality applications.

REFERENCES

Hill Cathrine. 2010. *Why So Few? :Women in Science, Technology, Engineering, and Mathematics.* Washington, D.C. : AAUW.

Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* ACM, 497–500.

Jürgen P Schulze, Andrew Prudhomme, Philip Weber, and Thomas A DeFanti. 2013. CalVR: an advanced open source virtual reality software framework. In *The Engineering Reality of Virtual Reality 2013*, Vol. 8649. International Society for Optics and Photonics, 864902.