Bringing Google Earth to Virtual Reality

Dominik Käser*[1], Evan Parker[1], and Matthias Bühlmann[2]
[1]Google, Inc. [2]Mabulous Software & Graphics GmbH

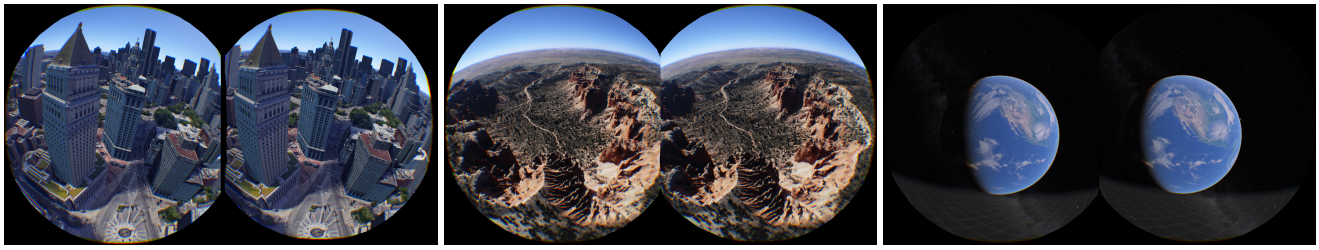

Figure 1: *Downtown Manhattan, Bryce Canyon, and the Earth, as displayed on a VR headset.*

Abstract

Since its inception, Google Earth has been brought to a variety of platforms: from desktop to mobile devices, from native to web. This talk discusses bringing Google Earth to virtual reality, a platform that poses unique challenges in user interaction and rendering. We present solutions to help users navigate planet-sized worlds in VR without losing context or becoming nauseous, and talk about techniques used to render such large worlds at the steady high frame rates that are required for VR.

Keywords: geospatial visualization, virtual reality, real-time rendering

Concepts: •**Computing methodologies** → **Virtual reality;**

1 Navigation

Moving through large worlds in virtual reality is hard. In most VR systems, users are given input controllers whose signals can be mapped to navigation operations such as teleporting or continuous flying forward and backward. In addition, with the advent of positional tracking systems, users are able to move up to a few meters within a tracked volume. Our planet has a radius of thousands of kilometers, though, so we have the challenge of mapping real world to virtual world distances. In addition, we want to preserve context for users across navigation operations while preventing them from becoming nauseous.

Our talk will cover new techniques that allow users to move around different places on the Earth and simultaneously scale the planet up and down in an intuitive manner. These techniques give users fast yet precise controls for navigating the environment. We present results from user studies and recommendations for future development based on what worked and what didn't work.

*{dpkay,eparker}@google.com, mbuhlmann@mabulous.com.
Speakers are presenting the work of a large number of people.

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4282-7/16/07
DOI: http://dx.doi.org/10.1145/2897839.2927441

2 Performance

Our VR application renders a huge data set consisting of trillions of triangles. In [Kontkanen and Parker 2014] we have shown how this data is preprocessed into levels of detail, segmented into cells and stored in the nodes of a data structure similar to an oct-tree. This allows the client application to fetch higher geometric detail from the server close to the camera and lower detail far away by traversing the tree and selecting for each region cells at different depths and only rendering that data.

While this method enables us to display the data set in real-time, in VR we face higher performance requirements than ever before. Optimizing frame-rate remains a top priority given the need to render two views at 60-120fps without dropping frames. Selecting the cells to render in each frame results in a few thousand individual draw calls per frame, which poses a performance problem using current high level graphics APIs such as OpenGL and Direct3D.

Our talk will describe how we decrease this draw call overhead and improve the performance considerably by batching the geometry and texture of individual draw calls dynamically on the CPU and render the whole scene with a single draw call. The system exploits the fact that in between two frames most of the geometry remains the same and it only sends the delta of the batched geometry and texture to the GPU. While selecting the right cells and updating the batch is computationally expensive, the talk describes how we perform this work asynchronously off the rendering thread to achieve a high rendering rate.

Acknowledgements

The above work was designed and implemented by John Anderson, Matthias Bühlmann, Adam Glazier, Dominik Käser, Per Karlsson, Aleksandr Palatnik, Evan Parker, John Rohlf, Matt Seegmiller, and Chun-Po Wang (in alphabetical order), based on prototyping by James Darpinian and years of lessons learned in the Google Earth project. The algorithms are intimately tied to the serving infrastructure built by the backend teams. In short, every triangle the client renders is the result of many teams working on data acquisition, computer vision, system architecture, and pipeline maintenance.

References

KONTKANEN, J., AND PARKER, E. 2014. Earth in google maps: rendering trillions of triangles in javascript. In *ACM SIGGRAPH 2014 Talks*, ACM, 44.