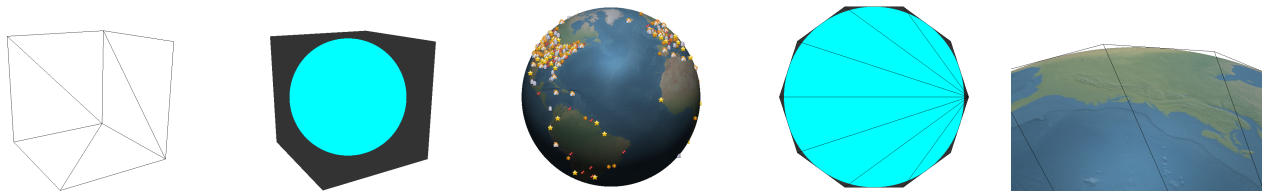
GPU Ray Casting of Virtual Globes

Patrick Cozzi* Frank Stoner†

Analytical Graphics, Inc.



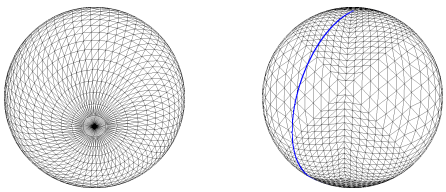
(a) (b) (c) (d) (e)

Figure 1: *(a) Front face culled bounding box (wireframe). (b) Ray/ellipsoid intersections in cyan. (c) Shaded, ray casted globe depth tested against rasterized billboards. (d) Viewport-aligned ellipsoid bounding polygon reduces ray misses. (e) Wireframe bounding polygon overlay.*

1 Introduction

Accurately rendering an ellipsoid is a fundamental problem for virtual globes in GIS and aerospace applications where the Earth's standard reference surface is non-spherical. The traditional approach of tessellating an ellipsoid into triangles and rendering via rasterization has several drawbacks [Miller and Gaskins 2009]. Geodetic grid tessellations oversample at the poles (2a), which leads to shading artifacts and ineffective culling. Tessellations based on subdividing an inscribed platonic solid lead to problematic triangles crossing the International Date Line and poles (2b).

We present a new approach to globe rendering based on GPU ray casting. Instead of tessellating the ellipsoid, we treat it naturally as an implicit surface. Simple proxy geometry bounding the ellipsoid from the viewer's perspective is rendered in order to invoke a fragment shader that casts a ray to find the ellipsoid's visible surface and shade accordingly. Our approach has the traditional advantages of ray casting implicit surfaces: infinite level of detail, trivial memory requirements, and simplicity. Furthermore, our approach reduces ray misses, runs at real-time frame rates on commodity GPUs, and easily integrates into existing rasterization-based engines.



(a) (b)

Figure 2: *(a) Over tessellation at the poles. (b) Triangles crossing the International Date Line (in blue).*

2 Our Approach

We start by rendering the ellipsoid's bounding box with front face culling (figure 1a). This invokes a fragment shader that casts a ray from the viewer to the fragment's world space position, checking for intersection with the ellipsoid. Figure 1b shows intersections in cyan and misses in gray. Fragments with ray misses are discarded. Fragments with intersections are shaded using the geodetic surface normal, yielding accurate lighting and texturing. Finally, the point of intersection is transformed into window coordinates to compute the correct depth value. Figure 1c shows a globe rendered using our approach combined with rasterized billboards.

Next, we introduce two new optimizations based on transforming the ellipsoid into a coordinate space where its representation is a sphere. First, this simplifies the ray/ellipsoid test. Second, ray misses can be reduced. Since the proxy geometry used to invoke the fragment shader only needs to bound the ellipsoid from the viewer's perspective, a viewport-aligned convex polygon is sufficient. We compute such a bounding polygon on the CPU in the transformed space (figure 1d). The bounding polygon is then rendered as a triangle fan in the original coordinate space. The number of tangent points allows a trade-off between CPU/vertex processing and fragment processing.

Using rasterization, rendering a globe resulted in rates of 112 fps (65,024 triangles) and 134 fps (960 triangles) on a NVIDIA GeForce 8400 GS at 1440x900 resolution. Our bounding box ray casting approach resulted in rates of 94 fps for full view and 78 fps for a horizon view (figure 1e). Our ray/ellipsoid intersection and bounding polygon optimizations improve the rate to 111 and 94 fps, making GPU ray casting competitive with rasterization.

Given the increasing memory bandwidth bottleneck, we believe that ray casting concise model representations, such as implicit surfaces, will have widespread use. In future work, we plan to compute an adaptive bounding polygon on the GPU and handle terrain by ray casting height fields [Dick et al. 2009].

We thank Kevin Ring, Deron Ohlarik, Vince Coppola, Joe Kider, Norm Badler, and Eric Haines for their input. We acknowledge Natural Earth (www.naturalearthdata.com) for raster data, and Yusuke Kamiyamane (www.pinvoke.com) for icons.

References

DICK, C., KRÜGER, J., AND WESTERMANN, R. 2009. GPU ray-casting for scalable terrain rendering. In *Proceedings of Eurographics 2009 - Areas Papers*, 43–50.

MILLER, J. R., AND GASKINS, T. 2009. Computations on an Ellipsoid for GIS. *Computer-Aided Design and Applications 6*, 4, 575–583.

*e-mail: pjcozzi@siggraph.org

†e-mail: fstoner@agi.com

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.

ISBN 978-1-4503-0210-4/10/0007