Computer Graphics and GPU

Daniel Connolly & Qingmu Deng
Olin College of Engineering | Fall 2018 | Computer Architecture

ABSTRACT

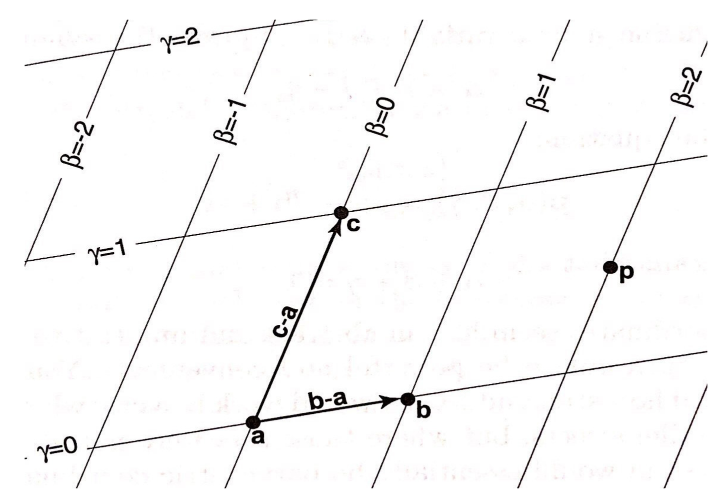
Inspired by our affinity for computer games and appreciation of realistic computer graphics, we sought to learn about how those graphics are created as well as the hardware that enables them, specifically focusing on ray tracing. We hope that our work, including our website and poster sessions during the final and Olin Expo, will provide helpful educational content regarding computer graphics, parallel programming, and parallel hardware to community members.

I. Graphics Pipeline

Triangles

Triangles are one of the most important primitives used for 3D modeling. They can be easily stored by the locations of their three vertices and can be combined to construct a full surface or object by specifying combinations of their vertices.

$$\mathbf{p}(\gamma, \beta) = \gamma(\mathbf{c} - \mathbf{a}) + \beta(\mathbf{b} - \mathbf{a}) + \mathbf{a}$$



Lighting

$$\vec{l} \cdot \hat{n}$$

Projection

$$x' = (((x/4) + 1)/2) * screen\_width$$
$$y' = (((-y/4) + 1)/2) * screen\_height$$
$$z' = (((z/4) + 1)/2) * screen\_depth$$

Rasterization

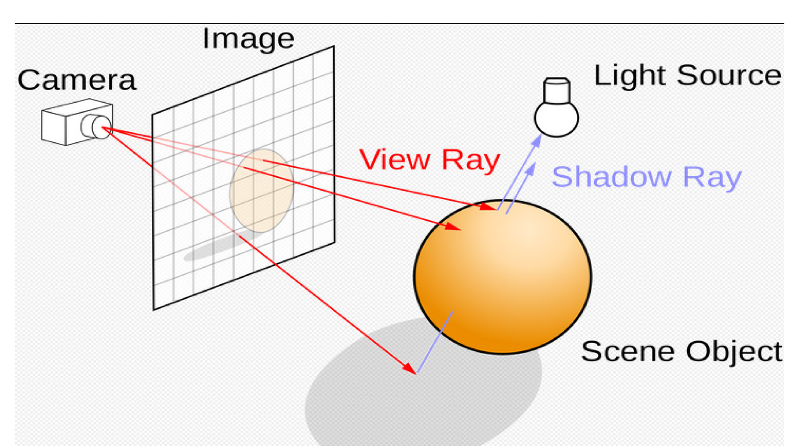
We display the 3D object on a screen by coloring the pixels representing the geometric primitives that are closest to the viewer at each pixel location.

The classic Utah teapot is a 3D test model for computer graphics.

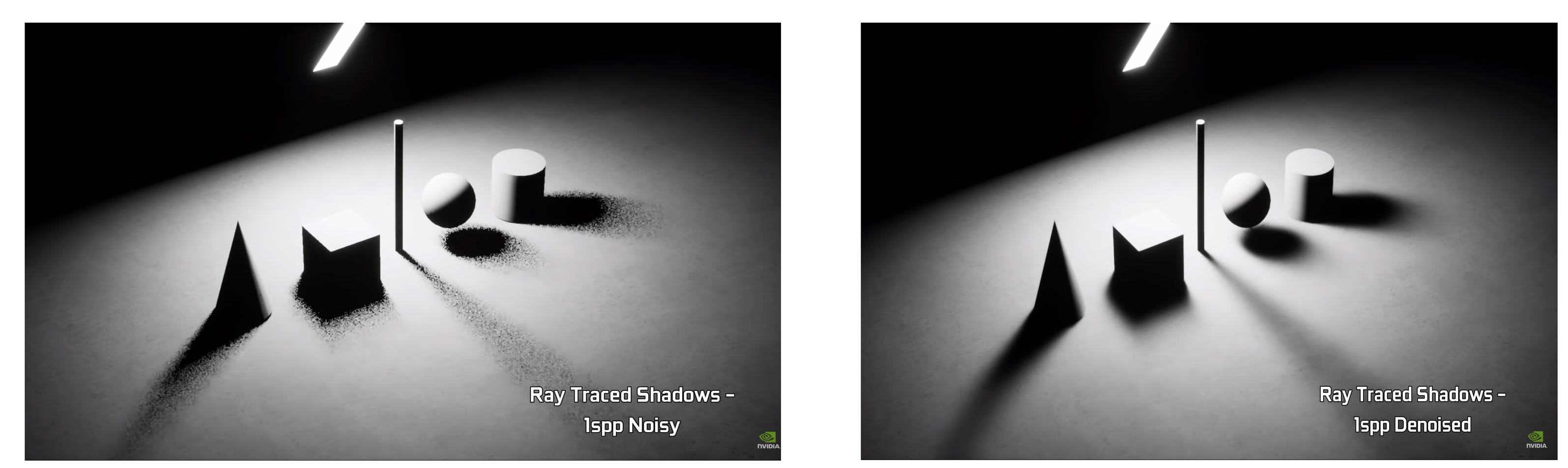
III. Ray Tracing

Ray-Triangle Intersection Test

$$\mathrm{x}_e + \mathrm{x}_d t = \mathrm{x}_a + \beta(\mathrm{x}_b - \mathrm{x}_a) + \gamma(\mathrm{x}_c - \mathrm{x}_a)$$



A Visual Explanation of the ray tracing process



At left, a 1 spp ray traced image that is not denoised; at right, a 1 spp ray traced image with denoising.

Bounding Volume Hierarchy

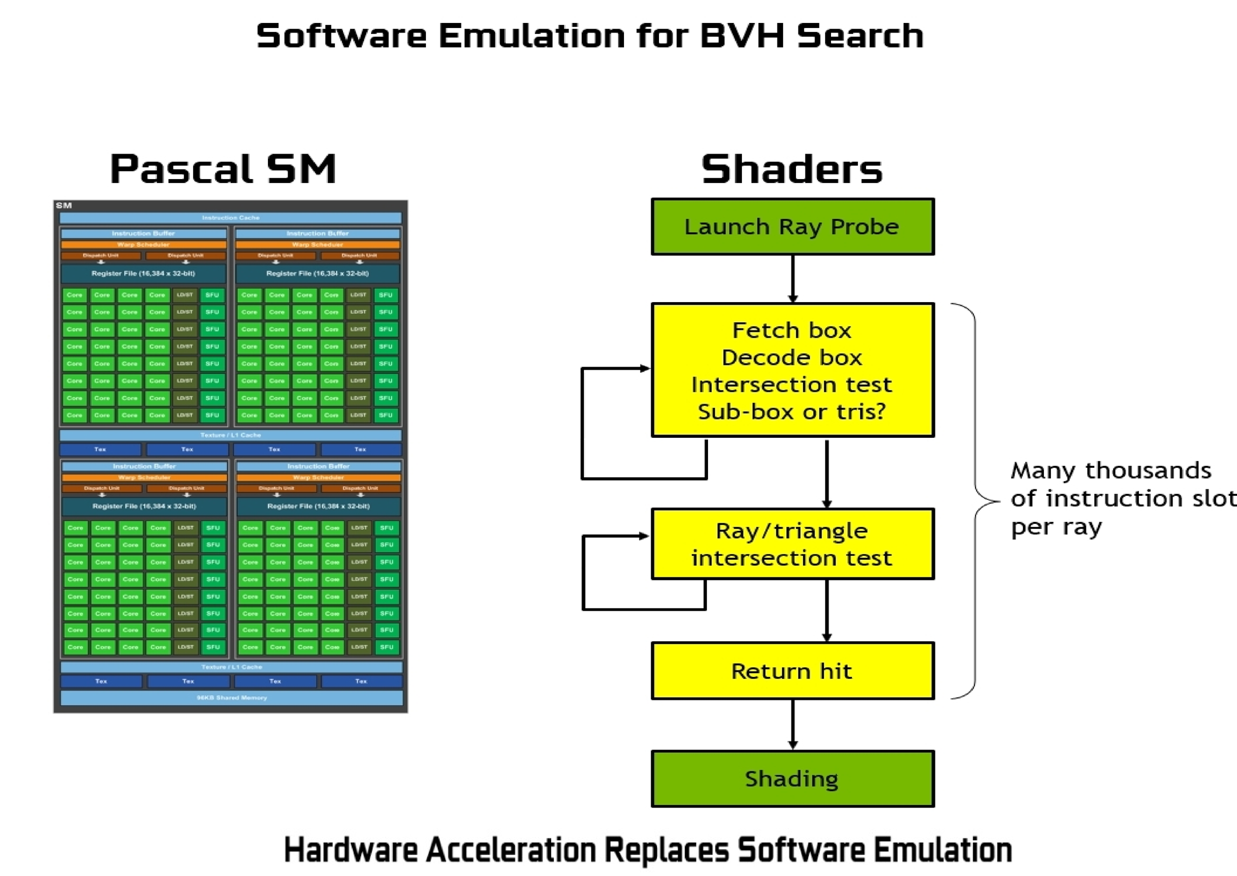
Organizing the scene into an object hierarchy of bounding volumes allows us to minimize expensive intersection tests. We store the BVH nodes with references to the leftmost child, the right sibling, and the parent node, adding in skip nodes where necessary in order to allow for efficient depth-first traversal. This structure requires more information be stored, but allows better memory usage as nodes are stored in the order in which they are accessed.

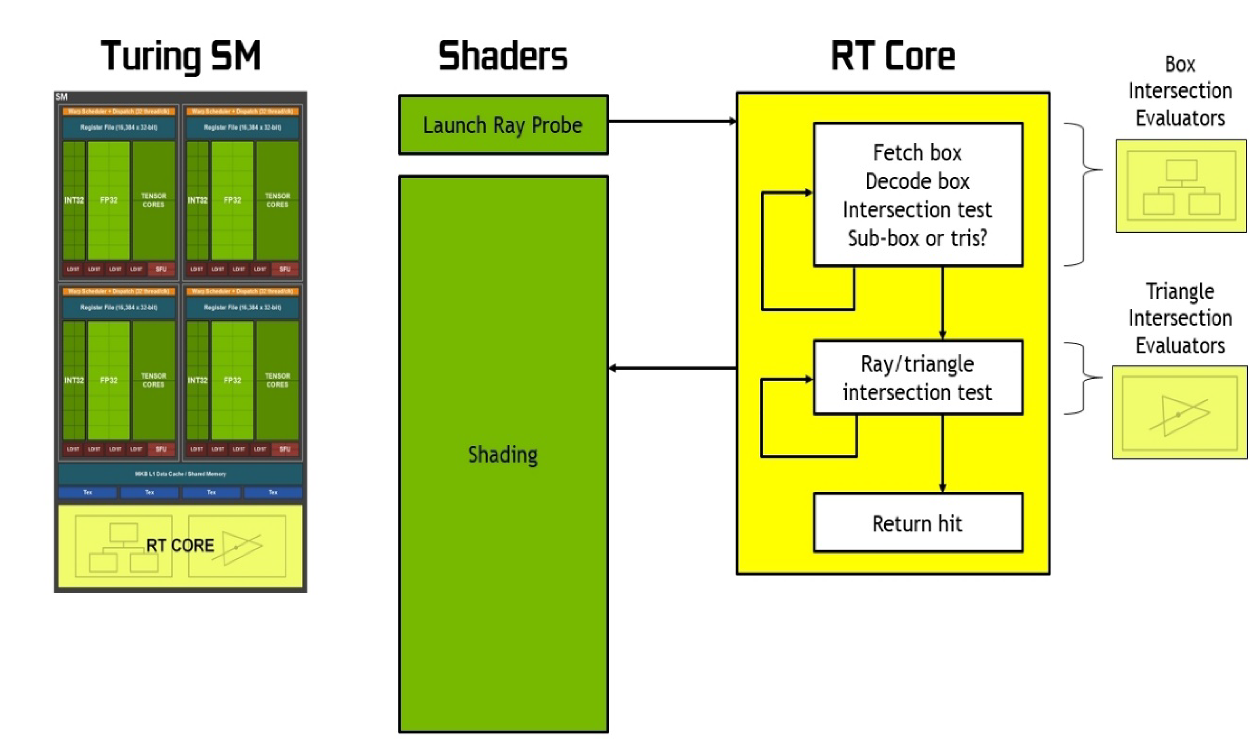
Denoising & Filtering

To provide real time ray tracing, sample rates must remain extremely low. Thus, we perform extensive denoising through complex algorithms such as Monte Carlo methods that take advantage of the parallelism present in GPUs. Ray tracing denoisers then may apply bilateral filters in conjunction with bidirectional reflectance functions, Monte Carlo integrators, and a series of approximations to achieve photorealistic images.

V. GPU Hardware

Modern GPUs support all types of parallelism, ranging from SIMD to MIMD to multithreading. They employ a hybrid rendering pipeline in order to most efficiently utilize all types of graphics rendering, including raster graphics, ray tracing, and denoising. Combined with Moore's Law, this allowed the incorporation of hardware accelerated ray tracing in the most recent NVIDIA Turing Architecture.





Reference

1. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination
2. Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies by Wald, Boulos, and Shirley
3. NVIDIA Turing GPU Architecutre Whitepaper
4. Computer Architecutre: A Quantitative Approach by Hennessy, John L.; Patterson, David A.

Please refer to our website for more references:
https://djconnolly27.github.io/GPU/resources.html