Toward Stream Filtered Ray Tracing on a DSP

Kevin Bensema, Jesse Porch, Jared Heinly, Shawn Recker, & Christiaan Gribble
Department of Computer Science
Grove City College

Introduction
This work-in-progress explores the implementation of stream filtered ray tracing using the Storm-1 stream processing architecture developed by Stream Processors, Inc. This poster presents our efforts to map the stream filtering approach to the Storm-1 processor and highlights the current progress toward a full-featured ray-based rendering system for the architecture.

Current status
To support ray tracing on the integer-only data-parallel unit of Storm-1 processors, we have implemented a full-featured integer ray tracing pipeline targeting environments that lack hardware support for the IEEE 754 floating-point standard [Heinly et al. 2010]. Images generated with the integer ray tracer are shown above.

In addition to the integer-only ray tracing pipeline we have:

• developed a library of 32- and 64-bit integer operations for the data-parallel unit, including complex functions such as multword square root and 64-bit/32-bit division;

• created an object-oriented framework for creating and managing computational tasks and data movement across the architecture’s RISC processor and the data-parallel unit; and

• implemented a prototype rendering system based on stream filtering to leverage the compute capabilities of the Storm-1 processor.

The system, though still under active development, demonstrates the feasibility of high-performance rendering with stream filtering. Upon completion, we hope to have a full-featured ray-based rendering system for use with the Storm-1 series processors.

Stream filtered ray tracing
Stream filtering [Gribble & Ramani 2008] recasts the basic ray tracing algorithm as a series of filter operations that exploit coherence by partitioning arbitrarily sized groups of arbitrary rays, or ray streams, into active and inactive subsets throughout the rendering process. In particular, ray streams are subjected to sets of conditional statements, or stream filters, that isolate those rays exhibiting some property of interest:

```c
out_stream filter<property>(in_stream)
{
    foreach ray in in_stream
    if (ray.test(property) == true)
        out_stream.add(ray)
    return out_stream
}
```

Gribble and Ramani [2008] have shown that the core operations in ray tracing, including traversal, intersection, and shading, can be written as a sequence of stream filters.

Target architecture: SPI Storm-1 series processors
The stream filtering approach opens a new design space that offers many interesting implementation alternatives and provides a compelling design for future ray-based graphics hardware [Ramani et al. 2009].

More immediately, however, our work targets a full implementation of stream filtered ray tracing on the Storm-1 series processors developed by Stream Processors, Inc. [SPI 2008]. The Storm-1 architecture is based on a standard RISC processor that offloads performance-critical tasks to an integer-only, data-parallel processing unit designed to exploit parallelism using conditional streams [Kapasi et al. 2000].

Acknowledgements
This work was funded by the Sweeney Scientific Research and Instrumentation Fund at Grove City College. The authors gratefully acknowledge the assistance of Dr. Ujval Kapasi, formerly Founder & Director of Software Applications at Stream Processors, Inc.

References