AN ACCELERATION OF FPGA-BASED RAY TRACER

Raisa Malcheva, PhD
Mohammad Yunis, MA
Donetsk National Technical University, Ukraine

Abstract

The Hardware implementations of the Ray Tracing algorithm are analyzed. A possibility of not all pixels tracing is discussed. The structure of Modified FPGA-based system is proposed. A productivity of Modified Ray Tracing algorithm is researched.

Keywords: Ray Tracing, FPGA, pixel, interpolation, productivity

Introduction

Ray tracing is one of the numerous techniques that exist to render images with computers. Ray tracing has been used in production environment for off-line rendering for a few decades now. That is rendering that doesn't need to have finished the whole scene in less than a few milliseconds. Real-time ray tracing is a very active field right now, as it's been seen as the next big thing that 3D accelerators need to be accelerating. Ray Tracer is really liked in areas where the quality of reflections is important. A lot of effects that seem hard to achieve with other techniques are very natural using a Ray Tracer. Reflection, refraction, depth of field, high quality shadows. Of course that doesn't necessarily mean they are fast.

Graphics card on the other hand they generate the majority of images these days but are very limited at ray tracing. The main specialty of ray tracing algorithm - an action is done for every pixel on the screen, so this could take a long time for complex scenes. Increasing features of ray tracing can be done by a dramatic increase in time spent with calculations. Not only must the program find all the intersections with the primary rays (as in ray casting), but it must also find all the intersections for each secondary and shadow ray.

Actually from 75 percent to over 95 percent of a ray tracer’s time is spent with such calculations [1]. That is decreasing of number of the tracing pixels without the changing of the final image is an actual research task.

Main Text

Analyzes of ray tracing algorithm shows that an increasing features of ray tracing can be done by a dramatic increase in time spent with calculations. Not only must the program find all the intersections with the primary rays, but it must also find all the intersections for each secondary and shadow. Quality, in this case, is not cheap, and it only gets more expensive, as you will see in the following section. In nowadays the new technology from CUDA NVIDIA and FPGA acceleration engines can be applied for ray tracing algorithm implementation.

NVIDIA acceleration engines

NVIDIA, the leader in GPU computing, today introduced the NVIDIA® OptiX™ ray tracing engine, part of a suite of application acceleration engines for software developers. NVIDIA acceleration engines make it easy to incorporate valuable, high-performance capabilities into applications, while simultaneously reducing development time.

NVIDIA application acceleration engines [2] include:

- NVIDIA® OptiX™ engine for real-time ray tracing;
- NVIDIA® SceniX™ engine for managing 3D data and scenes;
- NVIDIA® CompleX™ engine for scaling performance across multiple GPUs;
- NVIDIA® PhysX® 64-bit engine for real-time, hyper-realistic physical and environmental effects.

As the world's first interactive ray tracing engine to leverage the GPU, the NVIDIA OptiX engine is a programmable ray tracing pipeline enabling software developers to easily bring new levels of realism to their applications using traditional C programming. By tapping into the massively parallel computing power of NVIDIA® Quadro® Processors, the OptiX engine greatly accelerates the ray tracing used across a spectrum of disciplines, including: photorealistic rendering, automotive styling, acoustical design, optics simulation, volume calculations and radiation research. Application developers are utilizing the OptiX engine to redefine what's possible for designers, engineers and researchers.

FPGA acceleration engines

Performance of existing software implementations is still severely limited by modern processors that require many processors to achieve real-time performance. Therefore, in Saarland University (Germany) a structure of ray tracing hardware implementation on FPGA is developed [3]. Operating at a frequency of 90 MHz, the hardware implementation of ray tracing algorithm achieves real-time rate of 20 to 60 frames per second at a wide range of 3D scenes to support texturing, multiple light sources and multiple levels of reflection or transparency. A feature of this system is reuse of a transformation unit for tasks decision that include effective search of an intersection of rays with triangles. Despite the additional support for dynamic scenes this approach reduces the overall cost of the hardware part at 68%. Also to accelerate ray tracing algorithm the mathematical core on FPGA can be used. It shows more speed operations, the operands are represented by 64-bit numbers. These cores include Xilinx Floating-Point Operator Core. This core provides developers with tools to perform hardware for data processing in floating point format on FPGAs. This kernel supports multiplication, addition, subtraction, division, square root, comparison operations. Performance kernel provided by the fact that transactions are conducted on FPGA DSP cores, whose frequency is 300 MHz. This system allows to generate imagers with 320 × 240 and 512 × 480 resolutions.

Interpixel interpolation

To increase the size of images a modifined algorithm with an interpixel interpolation ia proposed. It based on the assumption that adjacent pixels of traced images have roughly the same (or very similar) color options. The main idea of algorithm is to trace pixels in any steps (horizontal and vertical) and than to apply an interpolation to determine the color components for untraced pixels. To realize this actions the Interpixel interpolation block is aded to the FPGA-based system (fig.1).

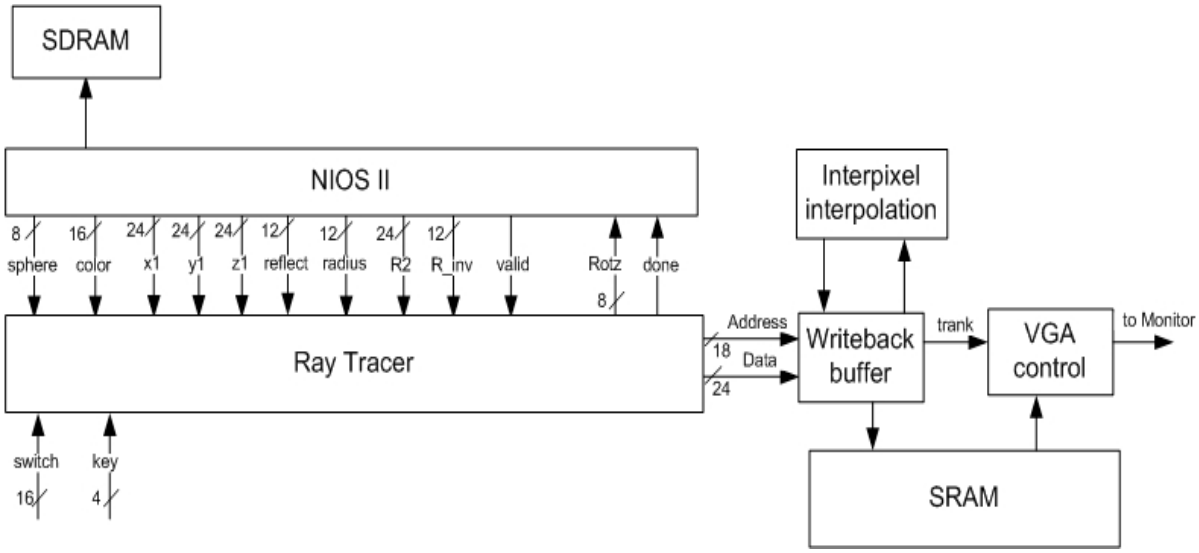

Fig.1. Structure of the modified FPGA-based system.

The ray tracing algorithm with interpixel interpolation is developed and simulated using Message Passing Interface for .NET. technology (cluster NeClus, DonNTU, fig.2).

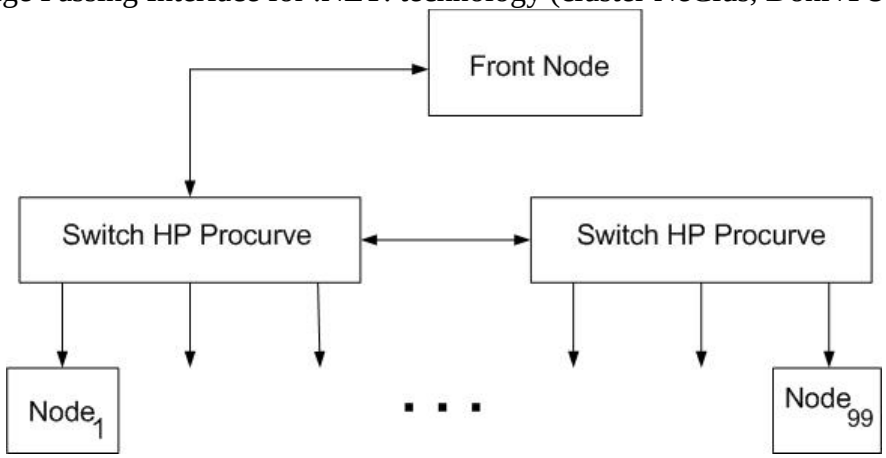

Fig.2. Configuration of NeClus.

NeClus uses an operative system ScientificLinux 5.4; packets Torque PBS and Maui to control the resources and tools for parallel programming MPI: openmpi-1.2.4, mpich-ch_p4-gcc-1.2.7, lam-7.1.4. A scene for the algorithm testing and reseaching is shown on fig.3. A coefficient of color differences is used to estimate an image and to perform adjustment of an interpolation step [4]. Results of researches are shown on fig.4.

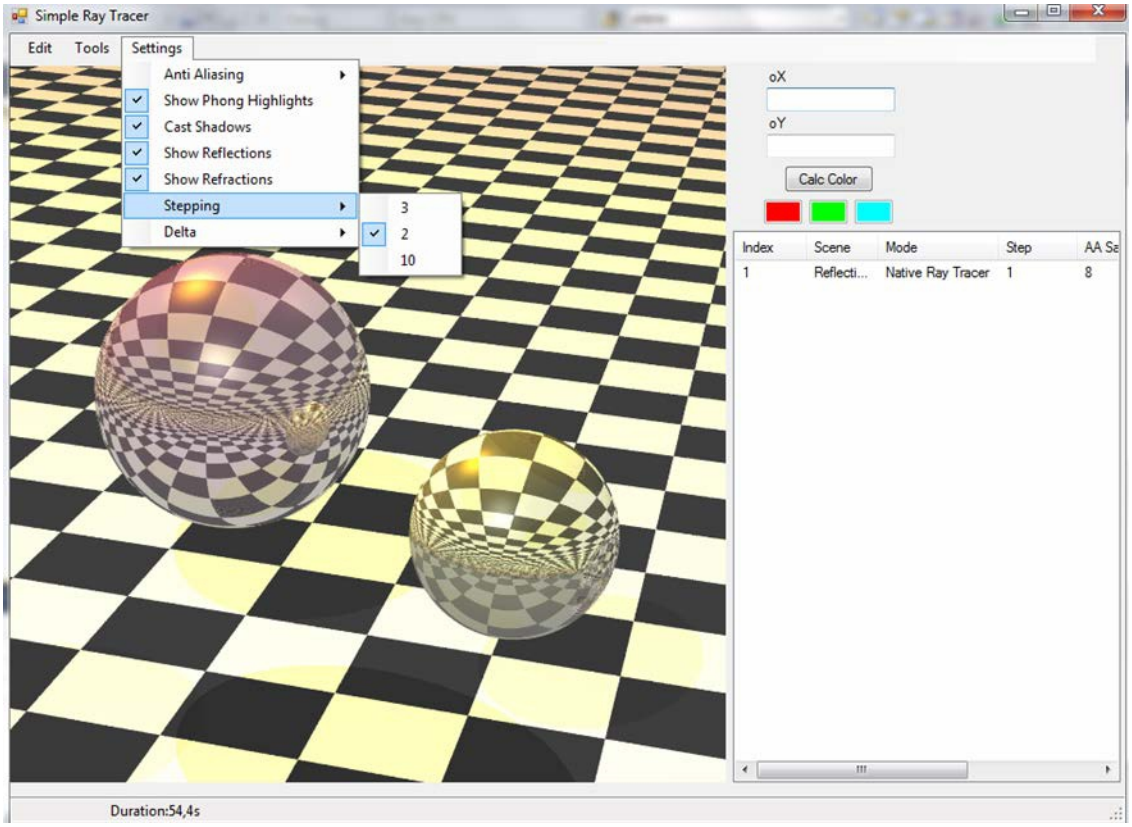

Fig.3. Scene for the algorithms testing.

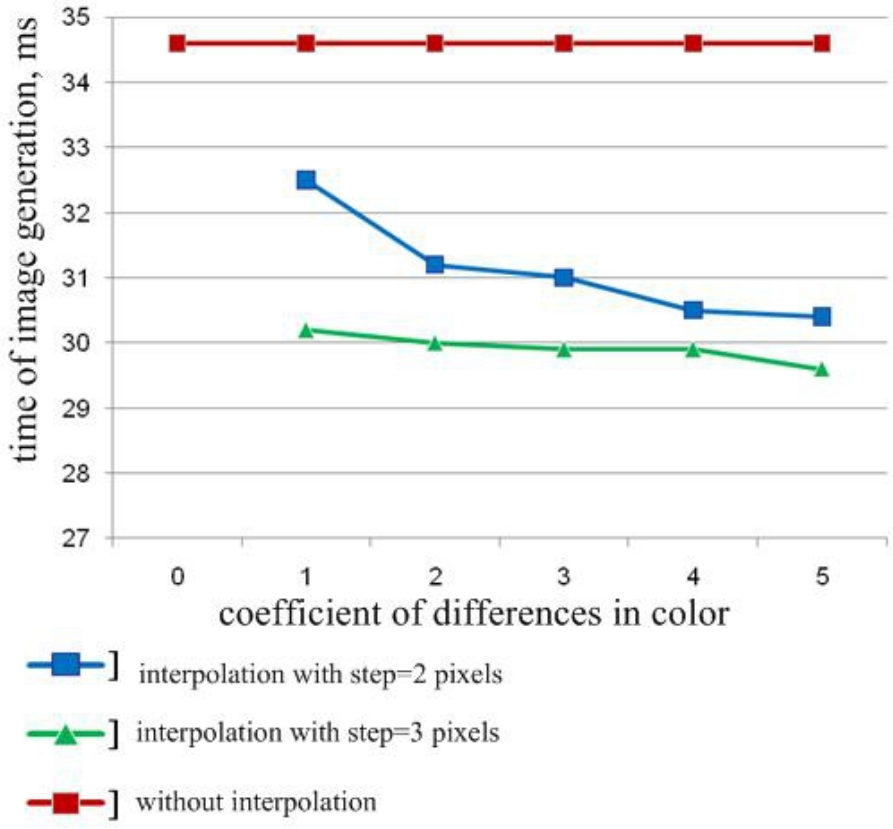

Fig.4. Results of researches.

Conclusion

Analysis of results shows that the modifided ray tracing algorithms with the interpixel interpolation reduces the time of an image generation on 12,5 - 15%. To selected a step of tracing and interpolation a coefficient of differences in color is used. Future directions are to check a block interpolation and to extend the number of processor units.

References:

Foley, James D. Computer Graphics: Principles and Practice. Reading, Mass.: Addison-Wesley, 1990.

Danny Shapiro. NVIDIA OptiX Engine Joins Groundbreaking Suite of Application Acceleration Engines, SIGGRAPH 2009, NEW ORLEANS—Aug. 4, 2009.

Schmittler Jörg. Realtime Ray Tracing of Dynamic Scenes on an FPGA Chip, Computer Science, Saarland University, Germany, 2004, pp.8.

R.V. Malcheva, M. Younis. Research of effect of step of ray tracing and coeefficient of difference in color components on the time of an image generation, Proceedings of Donetsk National technical university, vol. 14(188), Donetsk, 2011, pp. 195-201.