Request for REU Supplement to

CPATH EAE: τέχνη- Evaluation, Adoption, Extension
NSF Award #0722313

James Westall, PI
Robert Geist, co-PI

School of Computing
Clemson University
Clemson, SC 29634-0974

1 Background.

The goal of the τέχνη project is a complete re-design of the undergraduate curriculum in computing. The new design uses problem-based instruction as a vehicle but differs from others in the depth and scope of the problems addressed. As described in [2, 3], fundamental tenets of the approach are:

- A visual problem domain will most quickly capture the attention and interest of students who have grown up in a society that is increasingly visually oriented.
- A connection between scientific and artistic components will stimulate both deductive thought and creativity. (τέχνη, is the Greek word for *art*. It shares its root with τεχνολογία, the Greek word for *technology*.)
- Toy problems, typically discussed and assigned in computing curricula, are of little value in effecting the principal desired accommodation, the ability to solve new, full-scale, real-world problems.

Thus the problems are large-scale, with one problem per semester, and they are selected from problems that have arisen naturally in the research investigations of the graphics faculty. As an example, in a recently completed CS3 (Data Structures) course at Clemson University, students were assigned the task of writing (in C++) a ray-tracing system capable of generating images such as that shown in Figure 1. This image contains 528,979 texture-mapped polygons, and so a linear implementation of ray-polygon intersection testing with texture lookup would require hours of runtime on a high-end CPU. The students were allowed 3 minutes. Thus the only feasible approach required mastering sophisticated data structures and their associated search routines.

An alternative, orthogonal approach to solving such a problem would simply invoke a large number of parallel processing cores to take advantage of the inherently parallel structure of the task. Nevertheless, explicit instruction in parallel programming is often given short shrift in the typical undergraduate computing curriculum. There are two reasons. First, unlike the use of sophisticated acceleration structures and their associated algorithms, parallel computing per se will not reduce the computational complexity of the task. Second, until recently, the cost of making large numbers of processing cores available to undergraduates has been prohibitive.

The fully-programmable, SIMD architectures now available at low cost in graphics processing units (GPUs) are effecting an important change. As of this writing, an NVIDIA GT 240 card with 96 processing cores can be purchased for less than $100. Although the objection regarding computational complexity remains, the prospect of turning hours of computation time into minutes with minimal cost remains a strong attraction. A large community of researchers now investigates general purpose computation on GPUs. (See http://gpgpu.org/.) We believe that incorporating GPGPU instruction into the undergraduate curriculum is overdue.

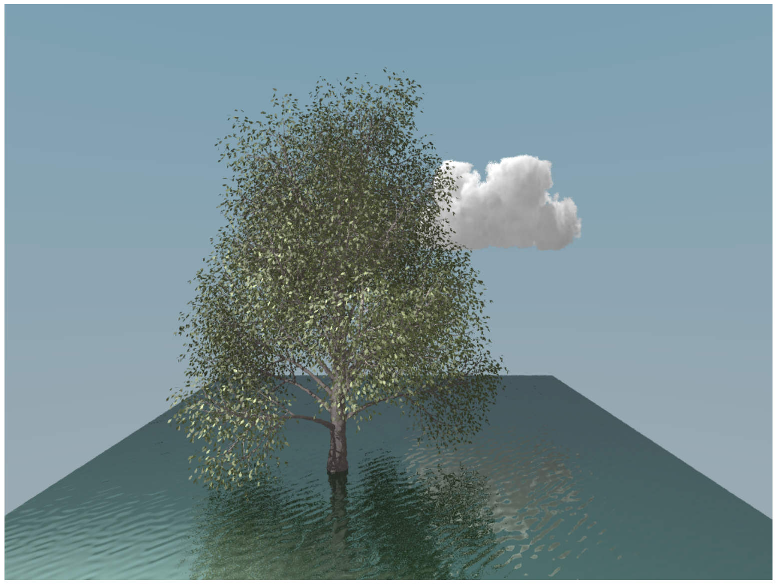

Figure 1: Target scene for CS3 ray-tracing project.

2 The Proposed Research Experiences.

We request support for two REU students to assist in designing a new τέχνη-based course in parallel processing, focusing in particular on GPGPU and the OpenCL programming language. OpenCL [5] is a new, industry standard language proposed by Apple and supported by NVIDIA, Intel, Sony, and AMD, which facilities access to a variety of high-performance parallel architectures. It is somewhat similar in interface structure to NVIDIA's proprietary Compute Unified Device Architecture (CUDA) [6], but it allows development of platform-independent code in much the same way that OpenGLSL extended NVIDIA's earlier Cg. OpenCL provides library extensions to C/C++ that allow close-to-the-metal software design on heterogeneous platforms, e.g., mixtures of CPU and GPU cores. The OpenCL Platform Model comprises a *host* connected to one or more OpenCL *devices*, each of contains one or more *compute units*, each of which contains one or more *processing elements*. A *host program* manages the parallel execution of functions, called *kernels* that execute on the *devices*.

Consistent with τέχνη design, the new course will require an attendant, semester-long project that serves as a conduit to deliver key course concepts, in this case, host-kernel communication, synchronization, and management of the available memory hierarchies, which are often numerous.

Each student will be charged with designing and implementing one such project. One will focus on extending the CS3 ray-tracing project to include both parallel execution and the diffuse illumination and caustics available through photon mapping [4]. Standard ray-tracers are particularly weak at handling diffuse illumination. Further, although the acceleration structures used in designing the CS3-level ray-tracer can deliver impressive performance for some complicated scenes, rendering objects with significant internal reflection and refraction, e.g., an automobile taillight, can bring such performance to a quick halt. The design goals include a scene model and a visually impressive, OpenCL-based rendering thereof which can be obtained on a single NVIDIA GT200 (240 cores) in less than one minute. Rendering the same scene sequentially on a high-end CPU, even with sophisticated acceleration structures, should require at least an hour to achieve comparable visual quality.

The second candidate project will be a particle system. Particle systems are particularly effective for visualization of fluid flows including water or airborne smoke, dust, or clouds. The computational effort required to update particle properties (position, velocity, acceleration) and render millions of particles has relegated most of the techniques that offer the best visual results, e.g. [7], to offline, non-interactive processing and opened the door to several approximation techniques, e.g. [8]. Here the key to an effective problem design will be securing compelling visual results in real-time on the GT200 or like platform, when sequential rendering on a CPU is decidedly unimpressive. It has been our experience with flow modeling (using NVIDIA'S CUDA) that a 60× speedup when moving from CPU to GPU is often available, and so a reasonable window of opportunity exists. For either project, the use of persistent threads [1], i.e., kernel functions which do not exit but rather pull work from global work queues, is likely to be important in achieving the performance goals.

3 PI/co-PI Experience with Undergraduate Research.

We supported three REU students under NSF Award 0113139, "ITR/SI: Design and Implementation of a Graphics Supercomputer from Commodity Components," PI Geist, co-PI Westall. Two students were supported for twelve weeks during the summer of 2002. Shelly Sawyer was a junior at Anderson College majoring in both mathematics and art. Michael Baker was a junior at Wofford College majoring in computer science. The students developed a graphical monitoring system that could show, in a single window on any remote display, the complete status of all nodes in our distributed rendering system: power, Linux, Xserver, and NIS.

An additional student was supported during the summer of 2005 under this same award. Robert Dixon was a rising junior at Clemson University majoring in computer science and a National Scholar. At this time, GPGPU research was just beginning, and the only available access was through shader languages. Robert developed a user-friendly API for solving a broad range of matrix computation problems using *GLSL* (the *OpenGL Shading Language*).

4 The Selection Process.

The opportunity will be advertised through the CPATH web site, http://www.cpath-community.msu.edu/. Although specific students have not been selected for this opportunity, those who have completed CS3 (Data Structures) and taken one or more τέχνη-based courses, either at Clemson or CPATH partner institutions, would have a decided advantage in assisting us in the design of new courses.

References

[1] Timo Aila and Samuli Laine. Understanding the efficiency of ray traversal on gpus. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 145–149, New York, NY, USA, 2009. ACM.

[2] T. Davis, R. Geist, S. Matzko, and J. Westall. τέχνη: A first step. *ACM SIGCSE Bulletin*, 36(1):125 – 129, 2004.

[3] T. Davis, R. Geist, S. Matzko, and J. Westall. τέχνη: Trial phase for the new curriculum. *ACM SIGCSE Bulletin*, 39(1):415 – 419, 2007.

[4] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A.K. Peters, Natick, MA, 2001.

[5] Khronos Group. The opencl specification, version 1.0. http://www.khronos.org/registry/cl/specs/opencl-1.0.48.pdf, October 2009.

[6] NVIDIA Corp. Nvidia cuda programming guide, version beta 2.0. www.nvidia.com/object/cuda_get.html% linebreak[0]#cuda2.0beta, April 2008.

[7] Simon Premože, Tolga Tasdizen, James Bigler, Aaron E. Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Comput. Graph. Forum*, 22(3):401–410, 2003.

[8] Cem Yuksel, Donald H. House, and John Keyser. Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):99, 2007.