

**Proposal for UROP Determining the Benefits and Uses
of a Dedicated Physics Processing Unit**

By Sam Bradley

October 3, 2006

Introduction

Purpose: To determine if a dedicated physics coprocessor, specifically the AGEIA PhysX physics processing unit (PPU), offers benefits over an additional processor core in applications other than computer graphics.

Claim: The PhysX PPU will offer significant processing benefits for physics computations over merely adding another processor core.

Background: In general, a computer is run by a central processing unit (CPU). This unit is capable of doing all computations needed by a program to run. Most modern computers incorporate a graphics processing unit or GPU. The GPU is engineered to calculate large matrices faster than the general CPU. These matrix calculations are used heavily in generating computer graphics. The GPU takes a large part of the CPU'S computational allowing tasks to be completed much faster overall. GPU's are increasingly being used to solve other non graphics-related problems much faster than the CPU can solve these problems. Examples of this include solving large matrices, general purpose stream computing, cloud dynamics simulations, and fluid flow simulations [1][2][3][4]. The AGEIA physics processing unit, PhysX, takes a similar approach with physics calculations. Just as the GPU is designed to free the general CPU of complicated matrix calculations, the PhysX PPU is designed to relieve the CPU of complicated tasks involving physics calculations. In a demo of the game CellFactor, frame rates dropped as low as 2 frames per second without the PhysX hardware, down from an 30 to 40 [5]. The question is if these computations are general enough that the PPU could be used for more than it's intended use as the GPU is sometimes used for more than calculations of computer graphics. For example, would the PhysX coprocessor offer a performance increases in determining the aerodynamics of an airplane?

Scope: This project is meant to discover the performance and uses of the AGEIA PhysX coprocessor. Specifically it will determine (1) if the PhysX coprocessor gives a significant performance increase over an added processor core and (2) if the PhysX application programming interface (API) is general enough to be used for other tasks not related to generating computer graphics.

Statement of Work

Learning the PhysX API (Secondary Research): The PhysX PPU is accessed by the CPU through its own API. This API is specific to the PPU so I will need to read the API specification provided by AGEIA and then test some sample code.

Writing the Code (Primary Research): The primary research for this project will be various programs written by myself that test the performance of the PhysX PPU compared to a second processor core. These programs will perform some kind of general physics computations. Each program will have three different versions. (1) A version that only utilizes a single processor core (the control), (2) a version that utilizes the PhysX PPU to make the physics computations, and (3) a version that utilizes a second processor core to make the physics computation.

Analyzing the Results: The results of the primary research will provide evidence that the PhysX PPU is capable of calculating physics calculations faster than a second processor would do the same work. The results of each test case will be compared to one another to determine if there was a significant difference in how fast the overall program ran.

Writing the Report: The report will summarize all of my findings and support the claim that a PPU makes physics calculations faster than a second processor core. It will discuss the outcome of each result and specifically how it supports the claim.

Revising the Report: The report will be revised after it is completed. Dr. Willemsen and I will evaluate it.

Costs

Fiscal: The project will require the stipend for the time spent on research, approximately 120 hours (\$1400). No equipment will be required to be purchased.

Time: The following timeline will be followed to complete the tasks necessary to complete this project. The approximate time spent on each task will be as follows: (1) 20 hours learning the API, (2) 80 hours writing the code, (3) 5 hours analyzing the results, (4) 10 hours writing the paper, and (5) 5 hours revising the paper. The timeline is shown below in figure 1.

Figure 1: Timeline

	Duration of Tasks					
Task	January	February	March	April	May	June
Learning the API						
Writing the Code						
Analyzing the Results						
Writing the Report						
Revising the Report						

Educational Outlook:

Faculty Sponsor: Dr. Willemsen will be my faculty sponsor. He will assist me in designing the programs to test the performance of each case. He will use the results of this project in future projects that may involve physics computations.

Objectives: In completing this project, I will have become a great deal more familiar with concepts of computer graphics and gain an understanding of different approaches to solving the same problem. I am currently studying general concepts of computer graphics in class with Dr. Willemsen and learning OpenGL, a graphics API to the GPU, which will assist me in this project. The knowledge of OpenGL also gives me a familiarity of how the PPU might be programmed. The completion of the project will also be beneficial to me when looking for a job when I finish my degree in the spring.

Conclusion

Summary: The proposed project will determine whether a dedicated PPU is beneficial over an addition CPU. The results of the project will be used by Dr. Willemsen who is interested in using the hardware and researching it's uses further and the experience will allow me to develop and advanced understanding of computer graphics. By following the sated timeline, the project will be finished by the June 15th deadline.

Contact: To discuss this proposal, please contact Sam Bradley at (218) 349-6182, or by email at brad0250@d.umn.edu.

- [1] Bolz, J., Farmer, I., Grinspun, E., & Schröder, P. (2003). Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. *ACM Computer Graphics (SIGGRAPH 2003)*, 917-924.
- [2] Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., et al. (2004). Brook for gpus: stream computing on graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23 (4), 777-786.
- [3] Harris, M. J., Baxter, W. V. I., Scheuermann, T., & Lastra, A. (2003). Simulation of cloud dynamics on graphics hardware. In *Proceedings of the acm siggraph/eurographics conference on graphics hardware*, 92-101.
- [4] Scheidegger, C., Comba, J., & Cunha, R. (2005). Practical cfd simulations on the gpu using smac. *Computer Graphics Forum*, 24 (4), 715-728.
- [5] <http://en.wikipedia.org/wiki/PhysX>