

Group Concept Proposal

September 20, 2004

Aly Merchant Cynthia Dahl

1 Introduction

Modern graphics processing units (GPUs) are highly programmable, and typically accept low-level assembly instructions. Shading languages like Sh and BrookGPU have been developed to program these units in high-level, C-style code. Not only do such languages allow for portability over various graphics systems, but they provide a simpler programming environment, since low-level, hardware-specific information need not be taken into account. As high-level shading programs become more complex, however, one must consider a serious problem: a program may exceed hardware limitations, such that it cannot be compiled in a single pass. These hardware constraints include the number of instructions, textures, and storage registers.

2 Benefits

Today's GPUs may be adequate for the average user, but they remain insufficient for some important applications. Areas such as film special effects require more complex shading programs to create realistic computer graphics. An increased resource space will also serve useful in general purpose computing. The fast processing speeds of GPUs makes them ideal co-processors in a variety of fields, including robotics, computer vision, and linear algebra.

3 Current technology

To overcome the problem of resource constraints, the hardware must be virtualized; that is, a mechanism must exist to allow high-level programs to use an indefinitely large number of resources. A paper from Stanford University (<http://graphics.stanford.edu/papers/rds/>) presents an algorithm *the Recursive Dominator Split (RDS)* for this virtualization by partitioning shading programs into multiple, hardware-appropriate passes. RDS aims to minimize the number of partitions, while operating in polynomial time.

4 Rationale

Our project will involve the implementation the RDS algorithm and the subsequent integration with the Sh compiler (<http://libsh.org>), which currently breaks for complex input. The Sh compiler, in development at the University of Waterloo, converts high-level code from Sh, a real-time shading language extending C++, into low-level code for programmable GPUs. Building on the Sh compiler will allow us to work on a solution to the virtualization problem without having to design an entire shading language. Additionally, if our project succeeds, our code will be released as part of the Sh distribution. We will also consider extensions of the RDS algorithm: either expanding our code to handle multiple program types or implementing newer virtualization algorithms which may extend or replace RDS.

5 Objectives

- Implement the RDS virtualization algorithm
- Integrate our code into the Sh OpenGL back end for stream programs
- The algorithm should find nearly optimal partitions
- Implement an extension to the algorithm to: allow for flexibility in usage or improve performance

6 Resource requirements

Since this project consists entirely of software design, there are no requirements for special facilities. The hardware required is a modern graphics card: an ATI card more recent than the Radeon 9500 or a NVidia card more recent than the GeForce FX 5200 model. One of the group members already has the necessary hardware, so that will also be unnecessary. All of the software being used for development is freely available.

7 Task list

<i>Tasks</i>	Aly	Cynthia
<i>Setting up the test environment</i>		
Compiling Sh's SM (hardware emulation) back end		R
Compiling Sh's OpenGL ARB back end	R	
Installing other shading languages (Ashli, RTSL)		R
Testing the Sh installation using the Shrike demo	R	
<i>Writing</i>		
Finding details on previous RDS implementations	R	
Group technical proposal	A	R
<i>Implementing the RDS algorithm</i>		
Dominator tree module		R
Scheduling module	R	
Partitioning module	A	R
Writing unit tests for all modules		R
Executing unit tests on the OpenGL back-end	R	
Finding good parameters for the cost function	A	R
Write a full search algorithm to find optimal partitions		R
Compare RDS with the full search (optimal) algorithm	R	
Marking up documentation for the Doxygen system	R	
<i>Integrating with Sh</i>		
Adding interface code	R	
Communicating with Waterloo Sh developers	R	
Updating stream programs to execute RDS when necessary		R
Updating ShTransformer to allow for extensions beyond OpenGL specific code	A	R
Executing regression tests in Sh	R	
Comparing the Sh implementation with Ashli (ATI) and RTSL (Stanford)	R	A
<i>Extensions</i>		
Perform feasibility study for multiple output partitioning algorithms	R	
Perform feasibility study for vertex and fragment program targets		R
Selecting and implementing one of the extensions	R	A
Merging the extension with the existing code	A	R
Testing performance of the new extension	R	
Marking up documentation for the Doxygen system		R
<i>Finishing up</i>		
Finding or writing code to demo the project	R	
Preparing for the oral presentation	A	R
Preparing the poster for the design fair		R
Writing the final report	R	A