Exploiting the Shader Model 4.0 Architecture Summary

Trenton Plager

Introduction & Related Work (Unified Shader Architecture)

This section of the paper discusses past methods of graphics processing, particularly regarding how the different stages in the rendering pipeline have been separated and consolidated over time. It introduces a general overview of the major changes that were made to the pipeline with the update to shader model 4.0. The introduction ends by providing a history of programmable graphics processing units, from the 1980s to 2007, and how each unit impacted the rendering pipeline. The Unified Shader Architecture section describes the differences between this new architecture and the previous architecture, as well as the advantages of the new unified architecture.

Geometry Shader

The Geometry Shader section of the paper describes one of the largest differences between the previous shader architecture and the new architecture in shader model 4.0. This difference is the addition of a new shader type that runs after the vertex shader, called the Geometry shader. It takes a primitive and attributes relating to it, and outputs a list of primitives that have attributes that don't necessarily match the attributes of the input. These shaders can also access vertices nearby a primitive, allowing the capture of the neighborhood of a primitive. One potential use case of the geometry shader that the writers introduce is the calculation of the silhouettes of 3D models. Another potential use is increasing the number of subdivisions of a 3D model by increasing the number of primitives that it outputs. As shown in the image of a teakettle, this can improve the appearance of a model because it can make the model appear rounder where necessary.

Array Textures

This section is shorter than the sections surrounding it. Primarily, it discusses the addition of a new extension to the upgraded model 4.0: the addition of an EXT_texture. It also discusses what exactly TexImages are and how their dimensions can be modified to change the number of layers that they represent. Finally, it explains how the data structures can be used and how they are beneficial rather than using the equivalent in fixed-function fragment processing. In particular, the paper suggests terrain as a good use case because its data can be stored as an array. This can be useful, for one reason, because they user has the ability to access single points in the array.

Layered Rendering

Because of the improvements made earlier in the revised rendering pipeline from shader model 4.0, it is now possible to render different scenes/views in a single pass. This involves the use of MRT's, or Multiple Render Targets. Some applications of this method are dynamic Environment mapping, two pass motion blur, and two pass multiple lights with dynamic shadows. This section of the research paper even provides guidance on how to implement these different uses of Layered Rendering using Layered Rendering.

Transform Feedback

The final section of this paper, unlike many research papers is simply discussing another new feature implemented in shader model 4.0. This feature allows the system to put data to memory directly from the pipeline so that the system doesn’t have to retrieve that data so many times by going through the entire pipeline again. In short, as the paper describes, it allows the program to change the data about a shape on the GPU without ever involving the CPU. In fact, in addition to this use case, this feature also allows the user to create simple physics simulations entirely on the GPU. This improves speed and efficiency because it removes one potential bottleneck in the pipeline. It is also much more scalable than performing physics on the CPU.