Implementing RenderMan on the Sony PS2

Ian Stephenson

National Centre for Computer Animation
Bournemouth University, UK
istephen@bournemouth.ac.uk

1 Introduction

As graphics hardware increases in power, driven by low cost consumer level products, there is an inevitable blurring between real time, and high end production techniques. Games developers are increasingly able to use techniques which where previously too slow, while producers of animation cannot ignore the powerful custom hardware available for rendering.

Though newer graphics cards provide greater flexibility, most are still highly customized, providing a very efficient, but fixed pipeline. The Sony PlayStation 2 by contrast provides the developer with an almost entirely “soft” pipeline.

We chose to port our renderer, a fairly standard Micro-Polygon (MP) based implementation of RenderMan, to the PS2 Linux platform. It was not intended that the results would be real time, rather to implement RenderMan as fully as possible, and observe to what degree the hardware could be exploited.

2 The Emotion Engine Core

The main processor of the PS2 is a 300MHz MIPS R5900. While moderately powerful, the performance of the EECore is restricted by its limited memory cache. Accessing memory is a major bottleneck. 16K of scratchpad memory is available and this was used for many key data structures, such as lighting information, and mesh data.

The EECore’s main function is to control the other units, and performs as little calculation as possible. It performs all front end parsing function, reading in RIB files, and generating data structures for each object read, which are passed immediately to the backend.

3 VU0: Geometry

The PS2 has 2 vector co-processors. These each have their own memory, and are capable of operating independently of the EECore. VU0 has 4K of Code and Data RAM which were used to evaluate parametric surfaces. VU0 is also used to measure the size of the micro-polygons generated.

The EECore loads the parameters for a surface into VU0’s data ram, and asks it to generate a grid. Depending on the size of the MP’s within that grid, the resulting points are copied to scratch ram. As grid generation, and measuring are both performed on the VU, excellent performance is achieved.

4 VU1: Shading

Shading was implemented as a simple stack-based SIMD machine. VU1 data memory is used to store the stack, and push operations transfer values to the VUmem from main memory. This allows the actual calculations to be performed by VU1, independently of the core, which can decode the next instruction while the current one is still executing. Two separate stacks for floats and vectors where used, allowing vectors to be stored in the correct format for the VU instruction set.

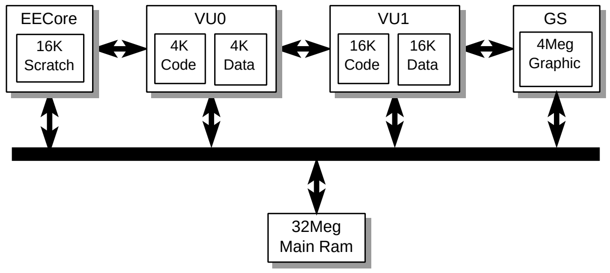

Figure 1: The PS2 Architecture

An SIMD machine would typically only perform operations upon nodes which are currently active. However the cost of testing the state of a node is usually higher then performing a calculation — particularly as the VU1 unit is operating at high speed in parallel with the main processor, and memory access is so critical. The node status flags are therefore only used to control pop operations, which write data back to main memory, changing the value of variables.

As VU1 has only 16K of data memory, the size of grids was restricted to 128 points. Larger grids significantly increase performance, but invariably cause stack overflows.

5 GS: Rasterization

VU1 perspective projects the shaded grid, and passes it directly to the Graphic Synthesizer (GS) for rasterization. During this time the EECore and VU0 are generating a grid for the next object.

The GS is designed to handle polygonal data, and is ill suited for use as the final stage of a MP renderer. It has many features such as texture mapping which we were unable to utilize. The depth resolution of the Z buffer proved not to be a significant problem, even when used in 16 bit mode, provided that clipping planes were set correctly. However the 12.4 fixed point representation of 2D coordinates generates rounding errors, resulting in missing pixels in the final image. This is made more obvious by the single precision maths, and limited grid size of the previous stages.

6 Conclusion

The renderer was capable of handling generic RenderMan shader and scene files. The major limitations resulted from the restricted size of the VUmem stack, and the poor rasterization by the GS.

DMA and threading are poorly supported by PS2 Linux. It is estimated that performance could be almost doubled were an effective implementation of these features available.