IBM Boeblingen Lab

Application Example
Terrain Rendering Engine (TRE)

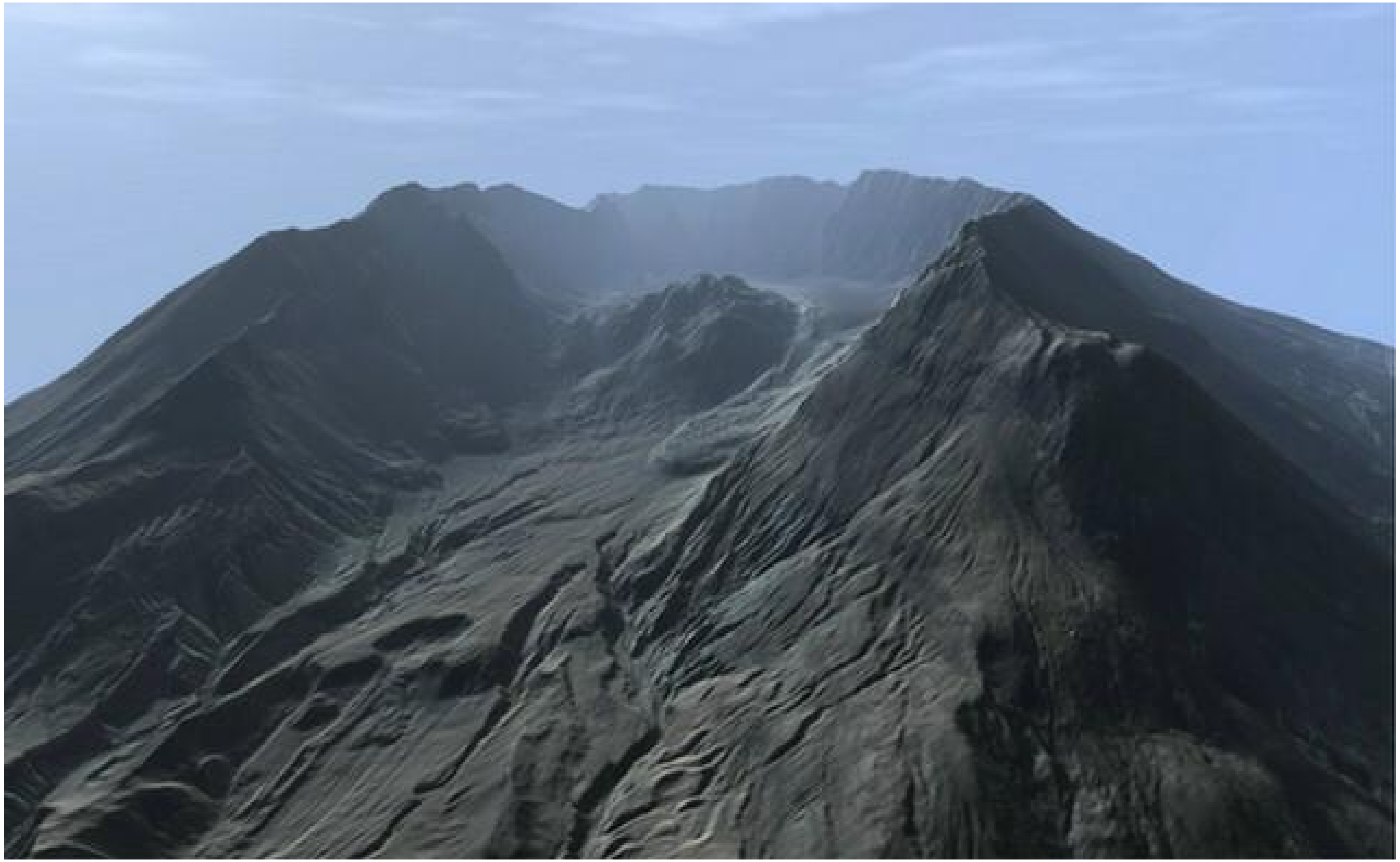
CELL Software Design Considerations

- **Two Levels of Parallelism**
  - Regular vector data that is SIMD-able
  - Independent tasks that may be executed in parallel
- **Computational**
  - SIMD engines on 8 SPEs and 1 PPE
  - Parallel sequence to be distributed over 8 SPE / 1 PPE
  - 256KB local store per SPE usage (data + code)
- **Communicational**
  - DMA and Bus bandwidth
    - DMA granularity – 128 bytes
    - DMA bandwidth among LS and System memory
  - Traffic control
    - Exploit computational complexity and data locality to lower data traffic requirement
  - Shared memory / Message passing abstraction overhead
  - Synchronization
  - DMA latency handling

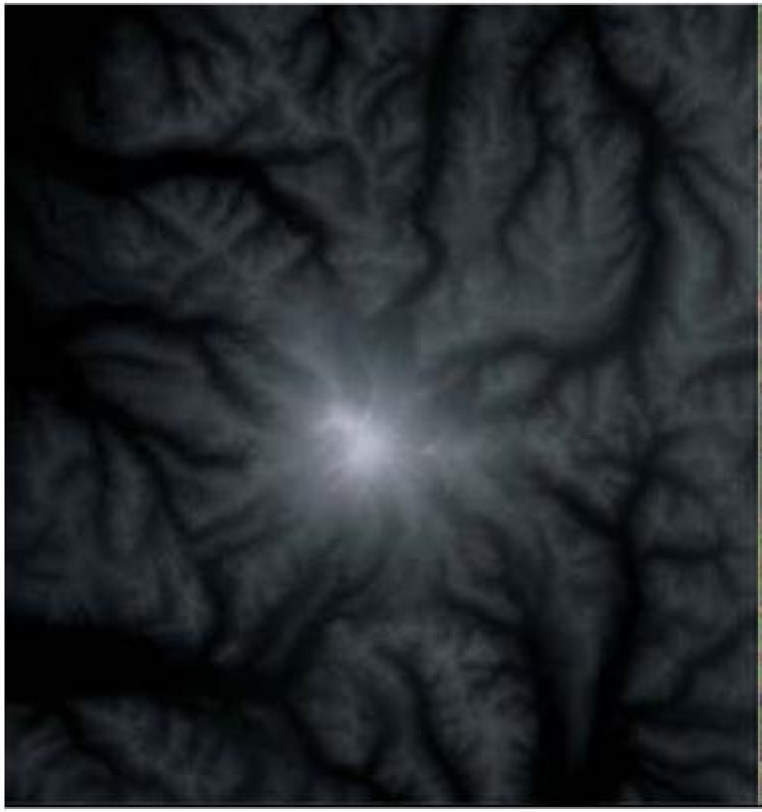
Typical CELL Software Development Flow

- **Algorithm complexity study**
- **Data traffic analysis**
- **Experimental partitioning and mapping of the algorithm and program structure to the architecture**
- **Develop PPE Control, PPE Scalar code**
- **Develop PPE Control, partitioned SPE scalar code**
  - Communication, synchronization, latency handling
- **Transform SPE scalar code to SPE SIMD code**
- **Re-balance the computation / data movement**
- **Other optimization considerations**
  - PPE SIMD, system bottle-neck, load balance

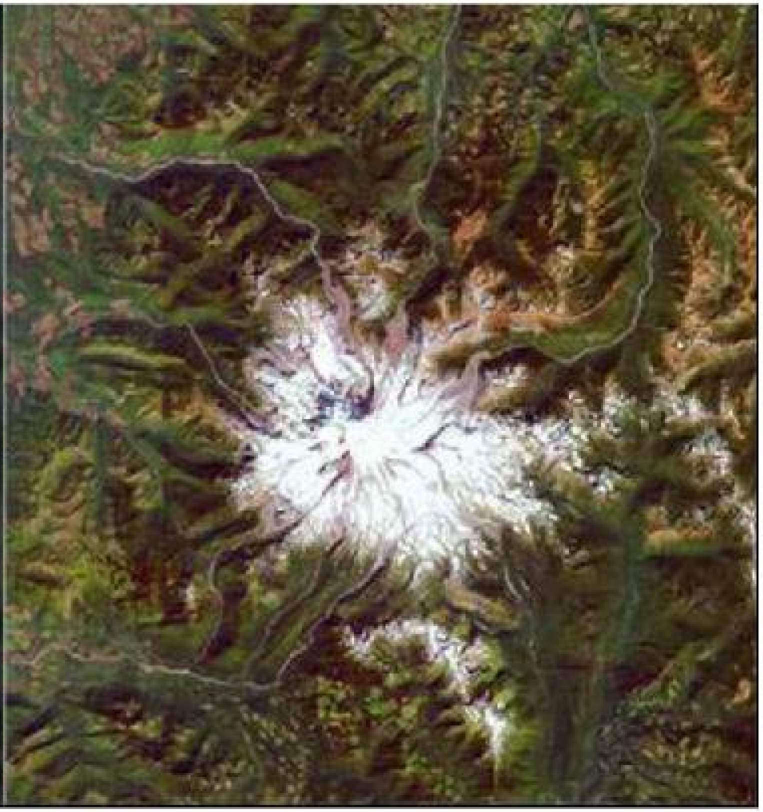
TRE Demo



Source Data

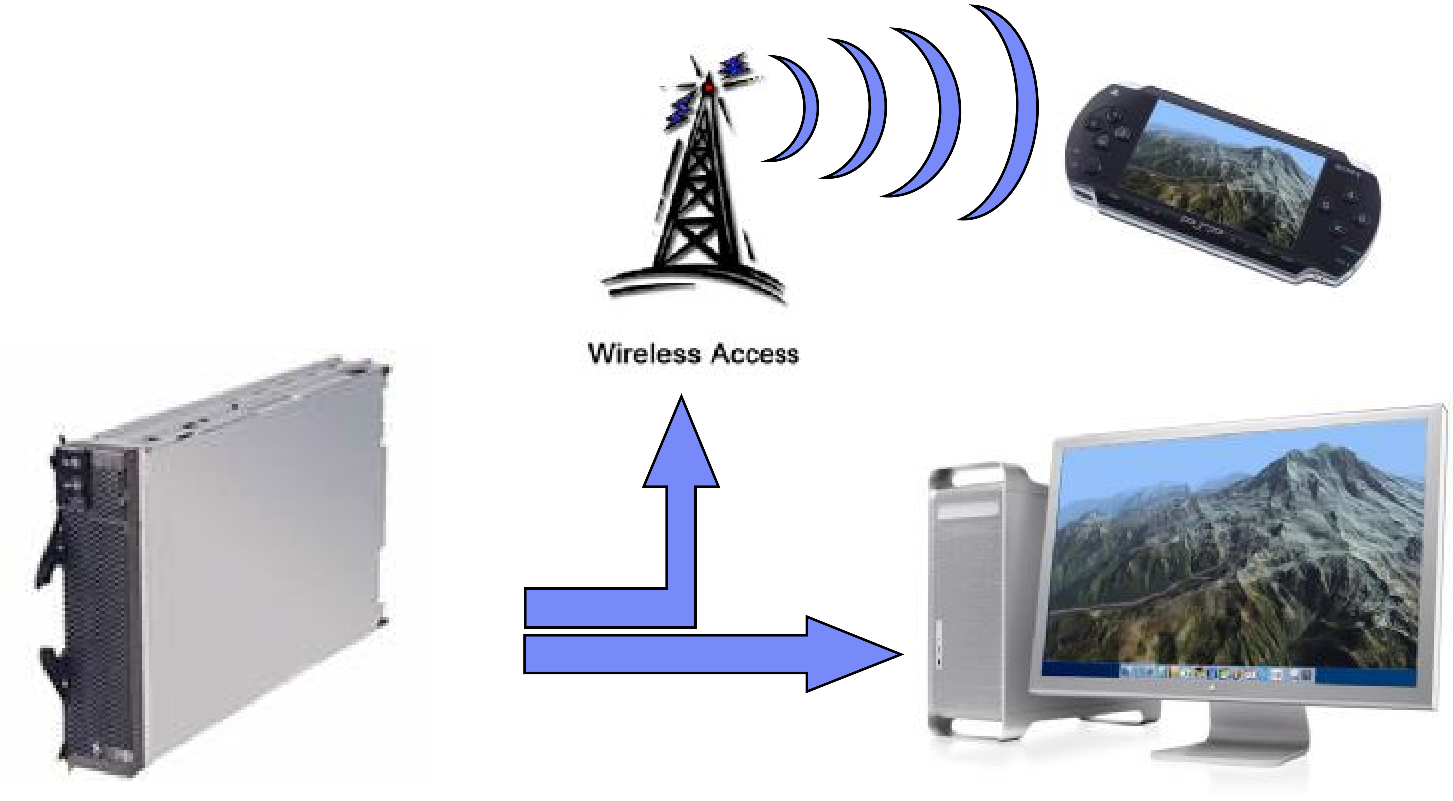


Digital elevation model



Color satellite image

Terrain Rendering Engine (TRE) System Configurations



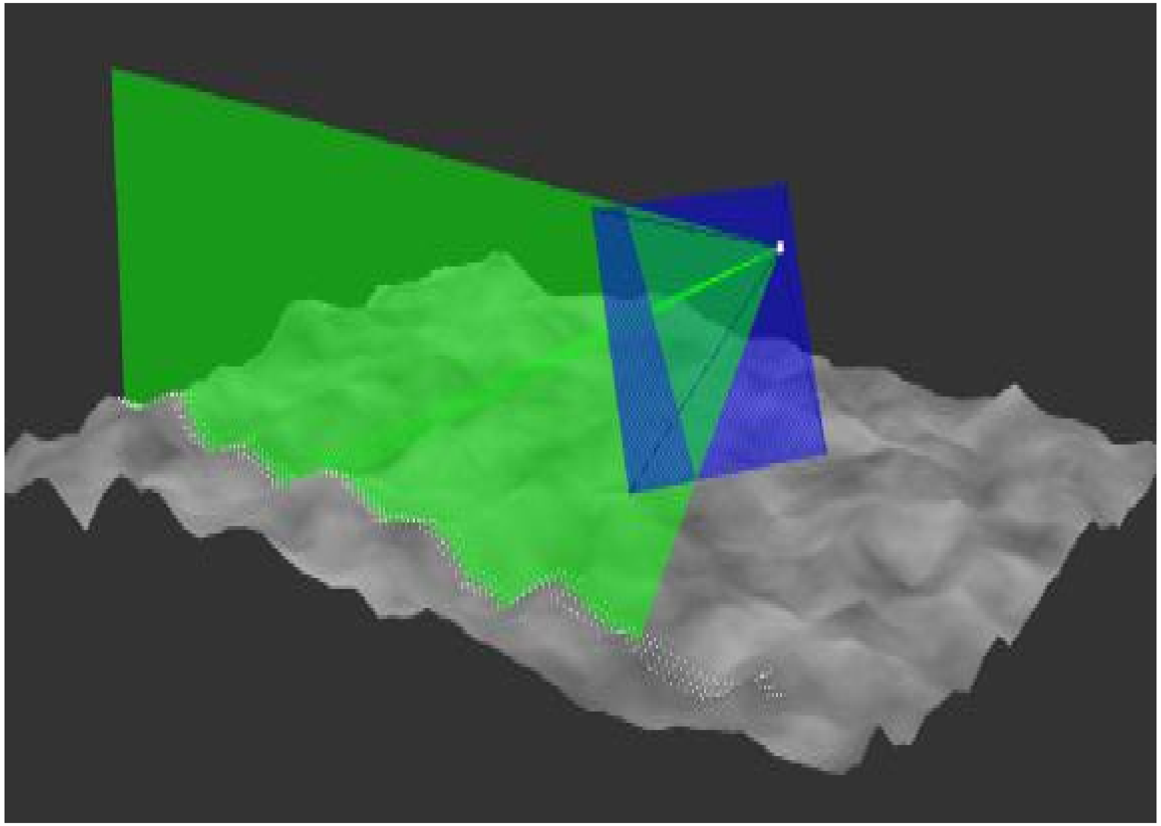
Wireless Access

Cell Blade System

Network

Clients

Ray Casting



TRE Client

- **Input data loading and delivery to the server**
- **Smooth path generation both user directed and random**
- **Joy stick directed flight simulation**
- **Rendering parameter modification**
  - Sun angle
  - Lighting parameters
  - Fog and haze control
  - Bump mapping control
  - Output image size
  - Number of SPEs to render
- **Server connection and selection**
- **Map cropping**
- **Streaming image decompression and display**

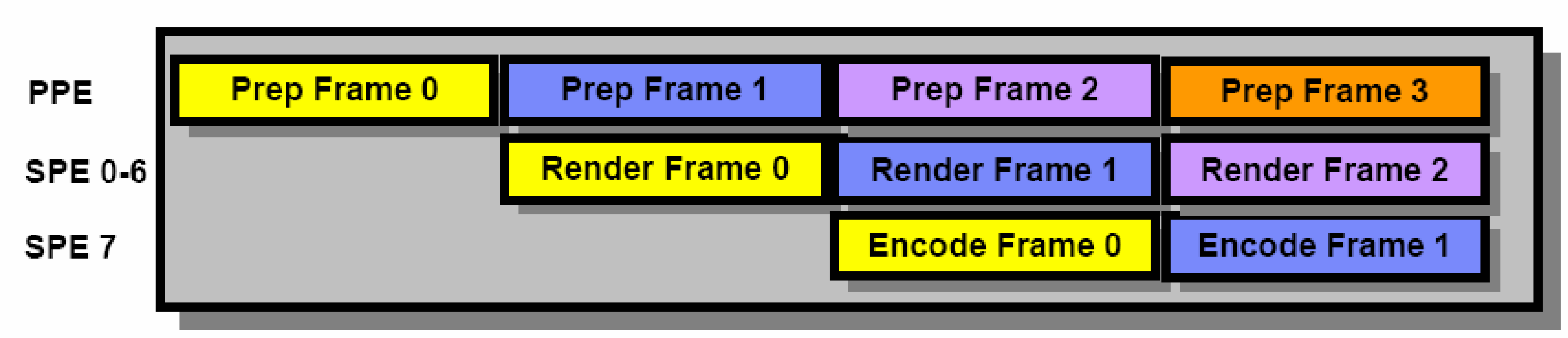
TRE Server – on CBE

PPE

1. **Frame preparation and SPE work communication.**
2. **Network tasks involving frame delivery.**
3. **Network tasks involving client communication.**

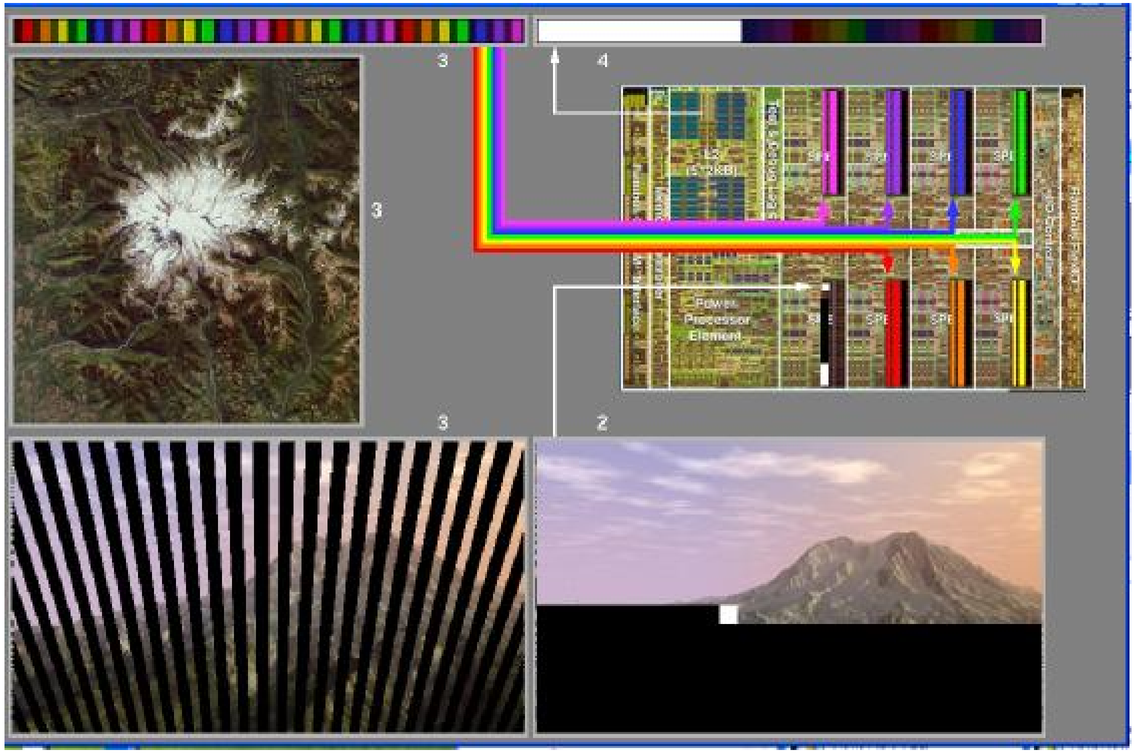
SPE

1. **Execute ray-casting kernel**
2. **Frame encoding – image compression**

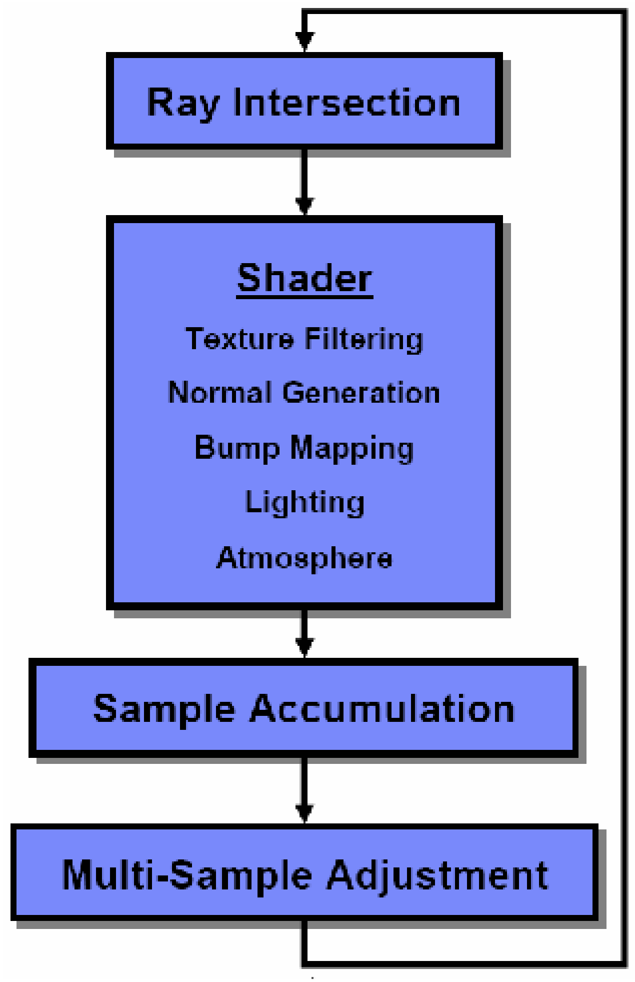


PPE	Prep Frame 0	Prep Frame 1	Prep Frame 2	Prep Frame 3
SPE 0-6		Render Frame 0	Render Frame 1	Render Frame 2
SPE 7			Encode Frame 0	Encode Frame 1

Chip


3
4
3
3
2
L2
(512KB)
Power
Processor
Element
SPE
SPE
SPE
SPE
SPE
SPE
SPE
SPE

Ray Kernel – Executes on SPE



- Decompose each vertical cut into rays
- Compute ray/terrain intersections
- Evaluate surface surface shader at each intersection
- Update accumulation buffer with new samples

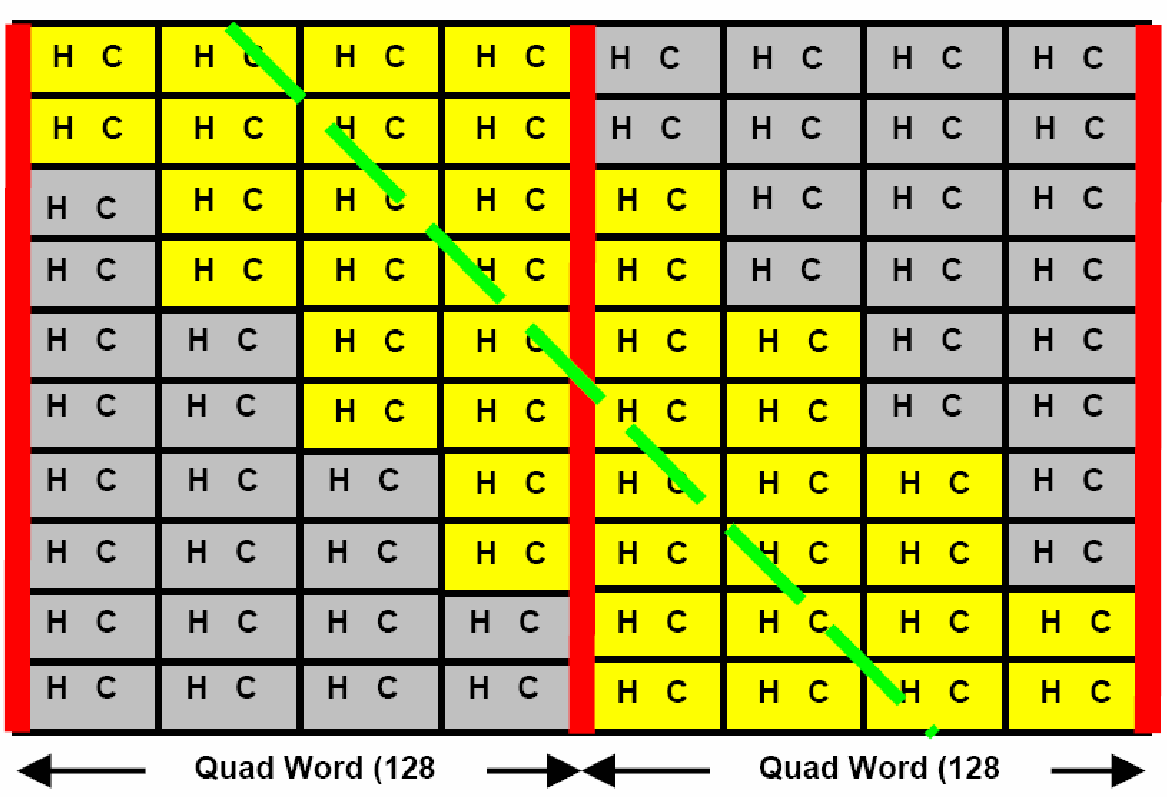
Most cycles spent in

- Ray intersection
- Shader

Major Computational Steps in Ray Kernel

- **Intersection test is broken into two phases**
  - Search for initial ray's intersection
  - Finds each intersection for rays in the vertical cut
- **Shader**
  - Texture filtering via a two by two color neighborhood.
  - Surface normal computation via four by four height neighborhood
  - Bump mapping via a normal perturbation function [3]
  - Surface illumination via diffuse reflection and ambient lighting model.
  - Visible sun plus halo effects
  - Non-linear atmospheric haze
  - Non-linear ground fog
  - Resolution independent Clouds computed via multiple octaves of Perlin noise evaluated on the fly [4].

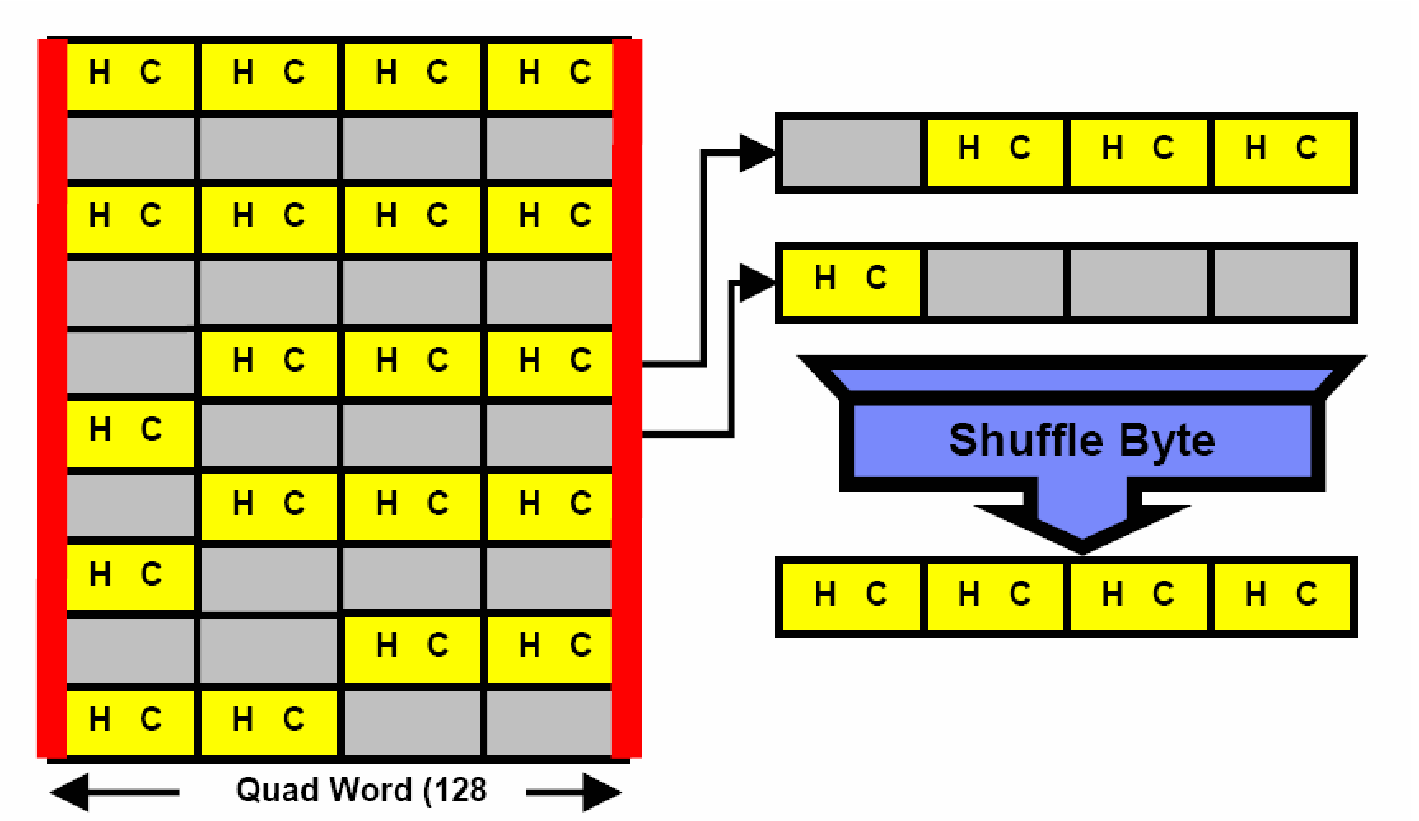
Height / Color Data Memory Layout



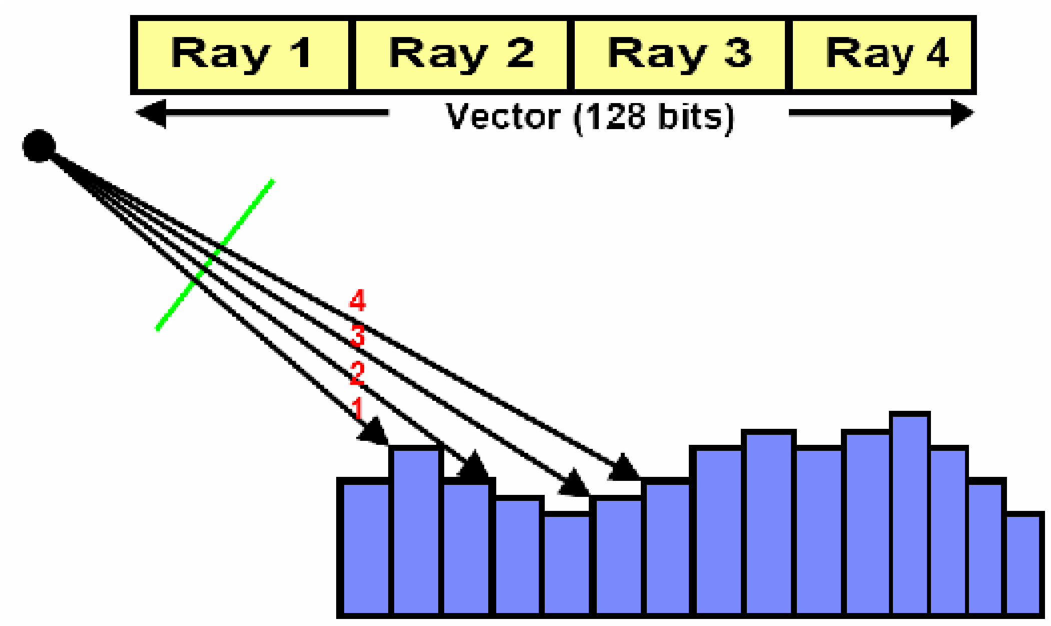
16 bits of color (5/6/5) and 16 bits of height per map point

- Compute and data fetch phases of the ray kernel execute in parallel by exploiting the asynchronous execution of the SPE's DMA engine (SMF) and SIMD core (SPU). Multiple input and output buffers are used to decouple the two phases.
- Surface shader requires four by four height neighborhood to compute the surface normal and two by two neighborhood for texture filtering.
- Use of DMA list to gather height/color data blocks along the intersection of vertical cut plane and the height/color plane

Data Alignment



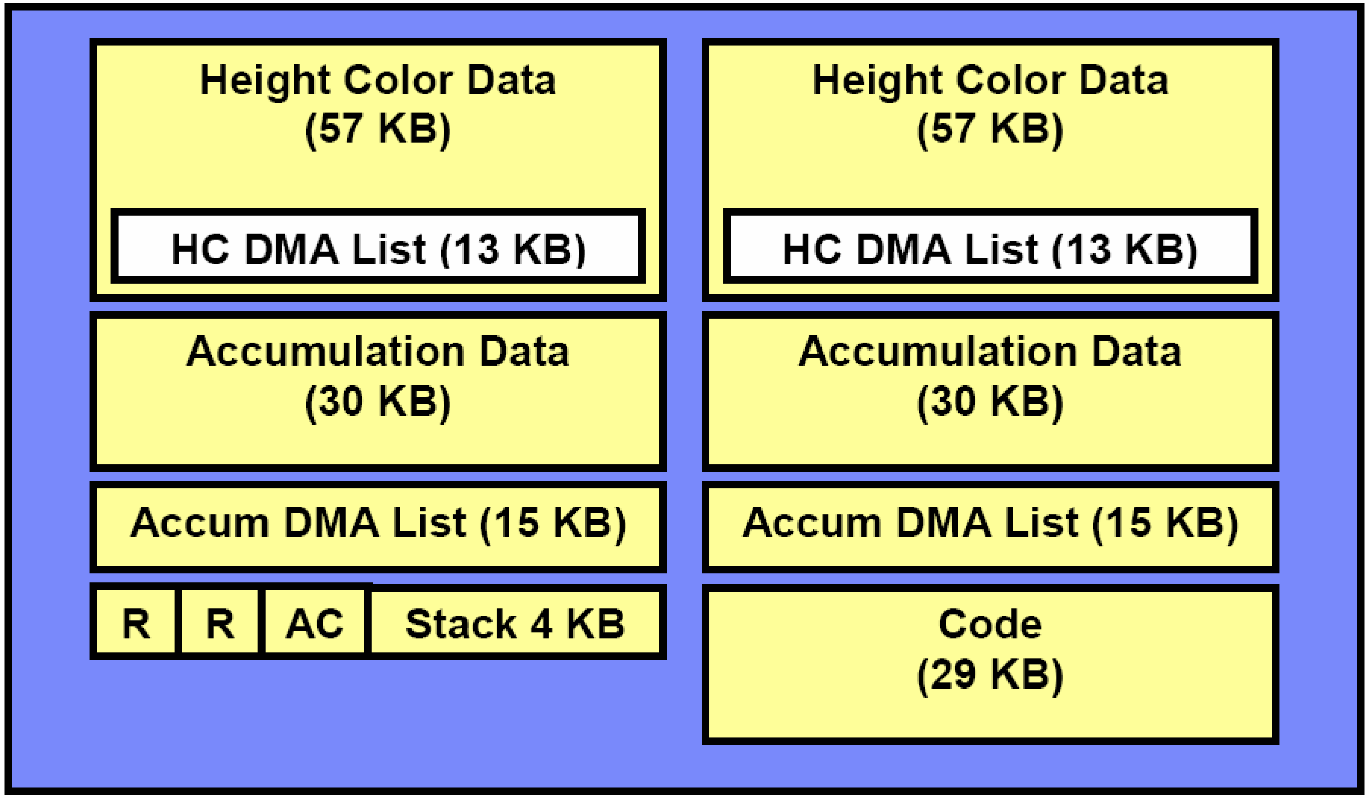
Exploiting SIMD



Samples are computed in a SIMD fashion by searching for four intersections at a time and when all four are located they are then evaluated in parallel by the surface shader.

Rays are packed four at a time into the single precision floating point channels of each vector register.

All ancillary information need for the shader is packaged in the same format using additional vector registers.



Height Color Data (57 KB)

HC DMA List (13 KB)

Height Color Data (57 KB)

HC DMA List (13 KB)

Accumulation Data (30 KB)

Accumulation Data (30 KB)

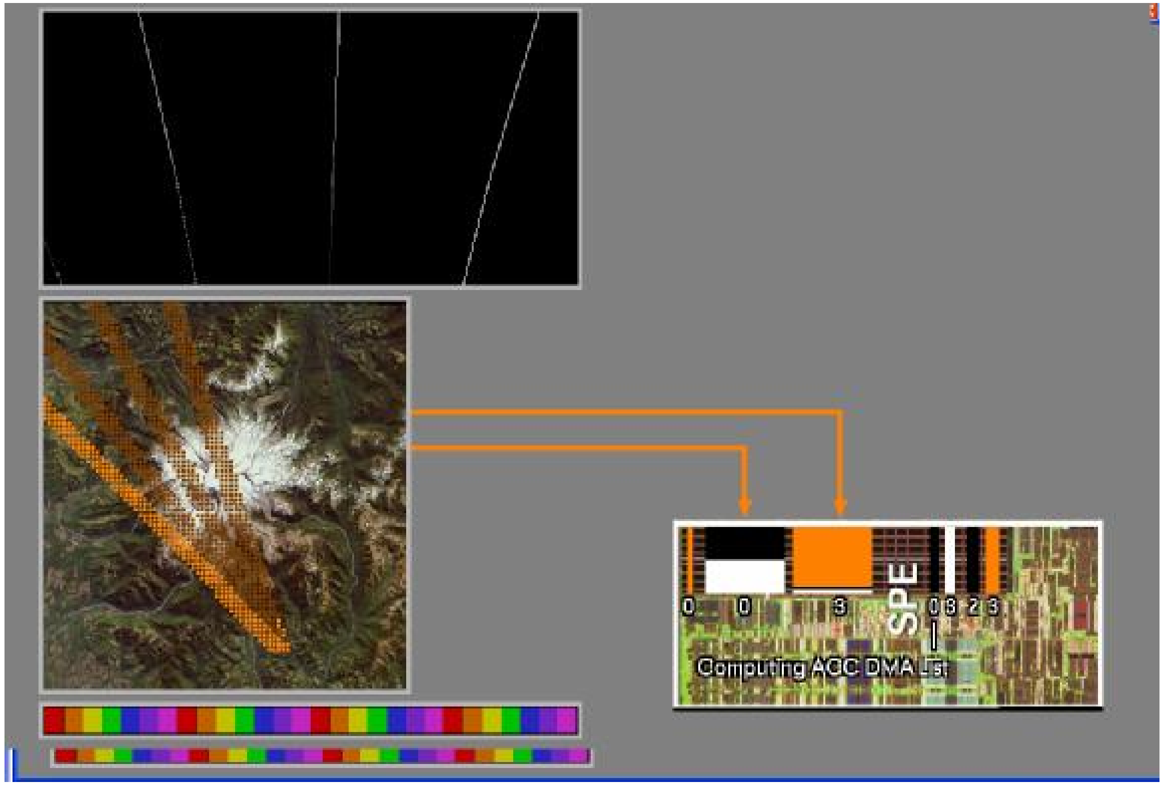
Accum DMA List (15 KB)

Accum DMA List (15 KB)

R | R | AC | Stack 4 KB

Code (29 KB)

SPE



TRE Performance

- **2.0 GHz Apple G5 0.6 frames/sec**
  - 40% of cycles spent waiting for Memory
- **3.2 GHz Cell 30.0 frames/sec**
  - 1% of cycles spent waiting for Memory
- **Cell has 50x advantage**

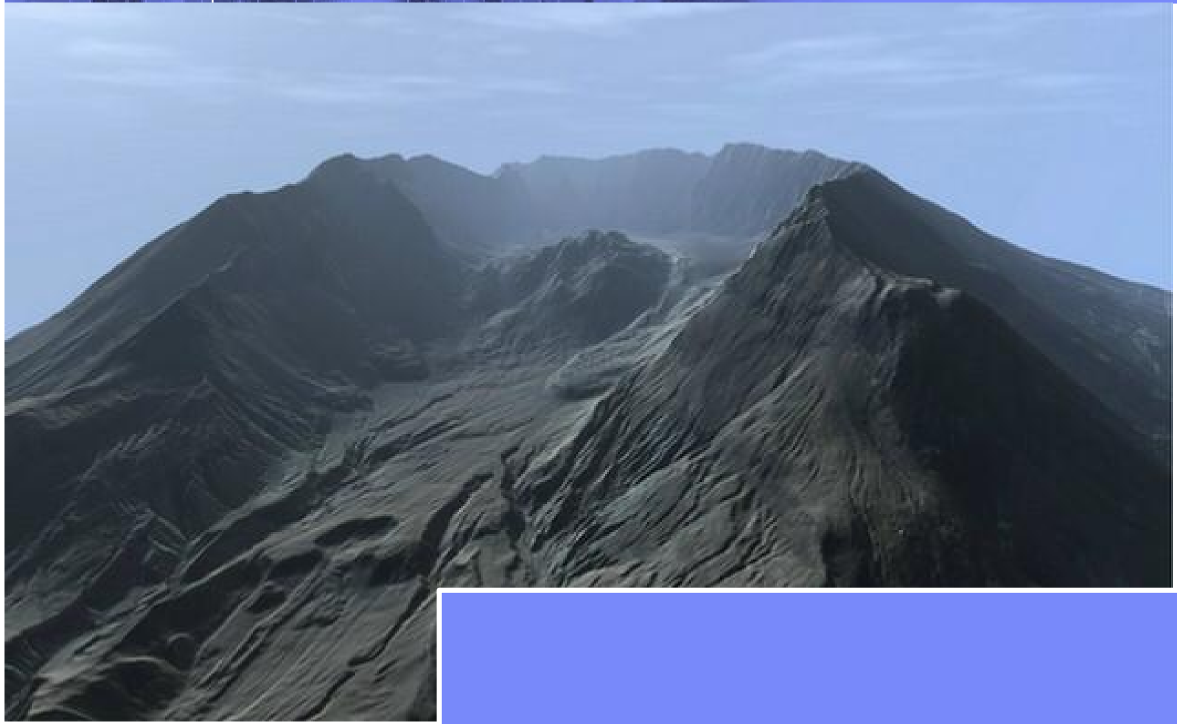
Parameter settings for benchmark:

•Output image size 1280x720 (720p)

•Map size 7455x8005

•Visibility to full fog/haze 2048 map steps to full haze

•Multi-sampling rate 1.33 x (2 – 8 Dynamic) or ~2-32 samples per pixel



Questions ?

Appendix

Publicly Available Information

- Introduction to the Cell Broadband Engine White Paper
- Cell Broadband Engine Public Registers Guide (subset of CDA version)
- Cell Broadband Engine Linux Reference Implementation Application Binary Interface Specification
- SPU C/C++ Language Extensions Software Reference Manual
- SPU Application Binary Interface Specifications
- SPU Assembly Language Specifications
- Broadband Engine Linux Application Binary Interface Specification
- Cell Broadband Engine SDK Libraries, Overview and User’s Guide
- Cell Broadband Engine Architecture
- Cell Broadband Engine Datasheet
- SPU Instruction Set Architecture Specifications
- Cell Broadband Engine Processor Full System Simulator
- XLC Alpha Edition for Cell Broadband Engine
- IBM Cell Broadband Engine Software Sample and Library Source Code
- GCC Toolchain for Cell Broadband Engine
- Cell Broadband Engine SPE Management Library
- Linux Kernel patch for Cell Broadband Engine
- SDK Installation script

- Introduction to the Cell Microprocessor, Article
- A 4.8GHz Fully Pipelined Embedded SRAM in the Streaming Processor of a Cell Processor, Article
- A Double-Precision Multiplier with Fine-Grained Clock-Gating Support for a First-Generation Cell Processor, Article
- A Streaming Processing Unit for a Cell Processor, Article
- The Design and Implementation of a First-Generation Cell Processor, Article
- Microprocessor Report - Cell Moves into the Limelight, Analyst Report
- Microprocessor Reports - 2004 Technology Awards, Analyst Report

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page is http://www.ibm.com
The IBM Microelectronics Division home page is
http://www.chips.ibm.com