

# Texture

## Real-Time Graphics Architecture

Kurt Akeley

Pat Hanrahan

<http://graphics.stanford.edu/courses/cs448-07-spring/>

## Topics

---

1. Projective texture mapping
2. Texture filtering and mip-mapping
3. Early implementations: RealityEngine/InfiniteReality
4. Texture locality and caching
5. Texture latency and prefetching
6. Coping with latency using multi-threading
7. Trends and pitfalls

## Readings

---

### Required

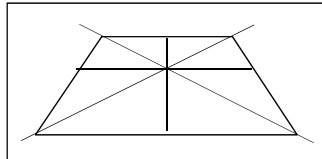
1. Z. Hakura, A. Gupta, The design and analysis of a cache architecture for texture mapping
2. H. Igehy, M. Eldridge, K. Proudfoot, Prefetching in a texture cache architecture

### Background

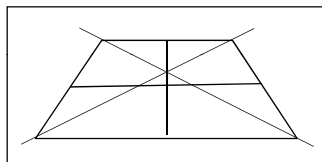
1. P. Heckbert, Texture mapping polygons in perspective
2. P. Heckbert and H. Moreton, Interpolation for polygon texture mapping and shading
3. J. Blinn, Hyperbolic interpolation
4. L Williams, Pyramidal parametrics

## Projective Texture Mapping

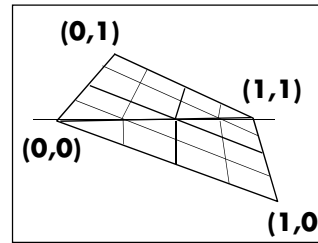
# Linear Perspective



**Correct Linear Perspective**



**Incorrect Perspective**



**Linear Interpolation, Bad**

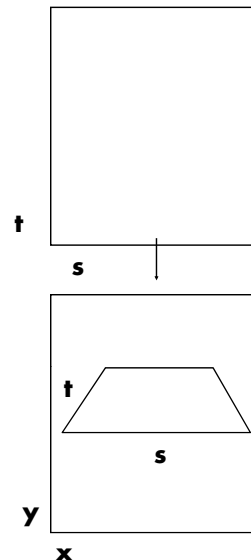
**Perspective Interpolation, Good**

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# Texture Mapping

2D (3D) Texture Space  
 | Texture Transformation  
 2D Object Parameters  
 | Parameterization  
 3D Object Space  
 | Model Transformation  
 3D World Space  
 | Viewing Transformation  
 3D Camera Space  
 | Projection  
 2D Image Space



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Texture Mapping Polygons

---

Forward transformation: linear projective map

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} s \\ t \\ r \end{bmatrix}$$

Backward transformation: linear projective map

$$\begin{bmatrix} s \\ t \\ r \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

## Perspective-Correct Interpolation

---

$$[x, y, z, 1, r, g, b, a, s, t, r, 1, \dots]$$

| **Transform & Clip**

$$[xw, yw, zw, w, r, g, b, a, sq, tq, rq, q, \dots]$$

| **Project (/w)**

$$[xw', yw', zw', 1, r, g, b, a, sw', tw', rw', qw', \dots]$$

| **Rasterize and Interpolate**

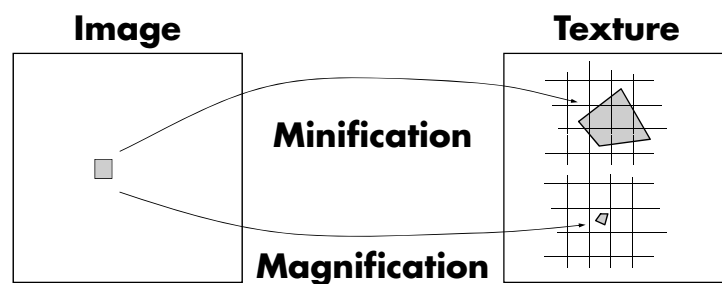
$$[xw', yw', zw', w', r, g, b, a, sw', tw', rw', qw', \dots]$$

| **Project (/qw')**

$$[xw', yw', zw', w', r, g, b, a, sw' / qw', tw' / qw', rw' / qw', 1, \dots]$$

## Texture Filtering (Mip-Mapping)

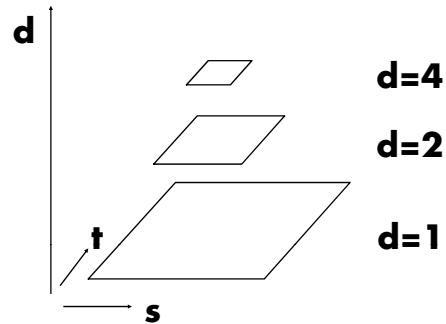
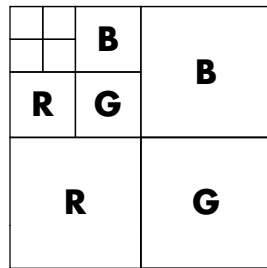
### Filtering Textures



- Texture footprint  
Footprint changes from pixel to pixel  
i.e. not shift-invariant
- Resampling theory: two cases
  1. Magnification => Interpolation
  2. Minification => Filter (averaging)

# MipMaps - L. Williams

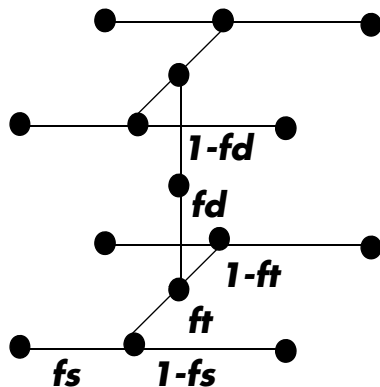
**Multum In Parvo = Many things in a small place**



**Address: 7 CMP(3 FIX + 4 RANGE), 2 MUL, 2 ADD**

# Texture Filtering

Constant time filtering



$$\text{lerp}(t, v_1, v_2) = v_1 + t(v_2 - v_1)$$

Linear (LERP)

1 MUL + 2 ADD / comp

Bilinear

3 LERPs

3 MUL + 6 ADD / comp

Trilinear

7 LERPs

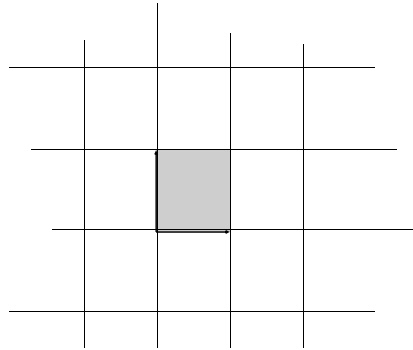
7 MUL + 14 ADD / comp

Quadrilinear

15 LERPs

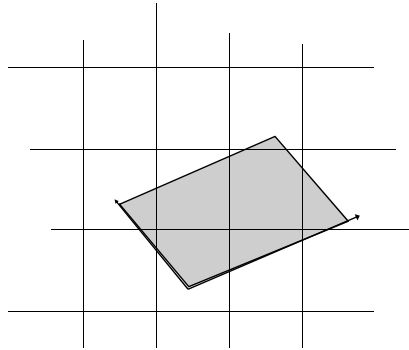
15 MUL + 30 ADD / comp

# Derivatives



$$\frac{\partial s}{\partial x} = s(x+1, y) - s(x, y)$$

$$\frac{\partial s}{\partial y} = s(x, y+1) - s(x, y)$$



$$\frac{\partial t}{\partial x} = t(x+1, y) - t(x, y)$$

$$\frac{\partial t}{\partial y} = t(x, y+1) - t(x, y)$$

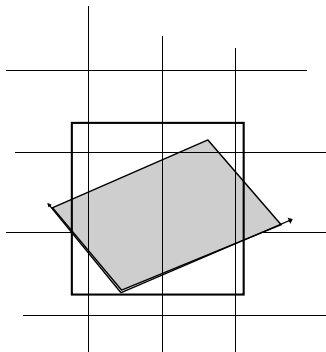
CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# mipd

Approximates quadrilateral with a square

$$A = \begin{vmatrix} \frac{\partial s}{\partial x} & \frac{\partial t}{\partial x} \\ \frac{\partial s}{\partial y} & \frac{\partial t}{\partial y} \end{vmatrix}$$



$$d = \sqrt{A}$$

$$mipd = \log_2 d$$

Common formula:  $A \approx \max \left( \sqrt{\frac{\partial s^2}{\partial x} + \frac{\partial t^2}{\partial x}}, \sqrt{\frac{\partial s^2}{\partial y} + \frac{\partial t^2}{\partial y}} \right)$   
 [Heckbert]

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Compute & Bandwidth Requirements

	ADD	MUL	CMP	DIV	SPE	READ
Project		4		1		
LOD	6	4	1		3	
Address	2	2	7			8
Filter	10	14				
Total	18	24	8	1	3	8

Texture access not regular

Requires high bandwidth

Address computation and filtering arithmetic intensive

Performance goal: billions of fragments per second

*Most demanding stage of the graphics pipeline*

## History of Hardware Texture Units

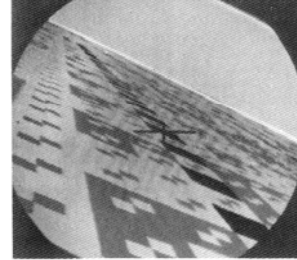


# History

## Flight simulators

GE Apollo Simulator 1963

Clever method for procedurally generating textures



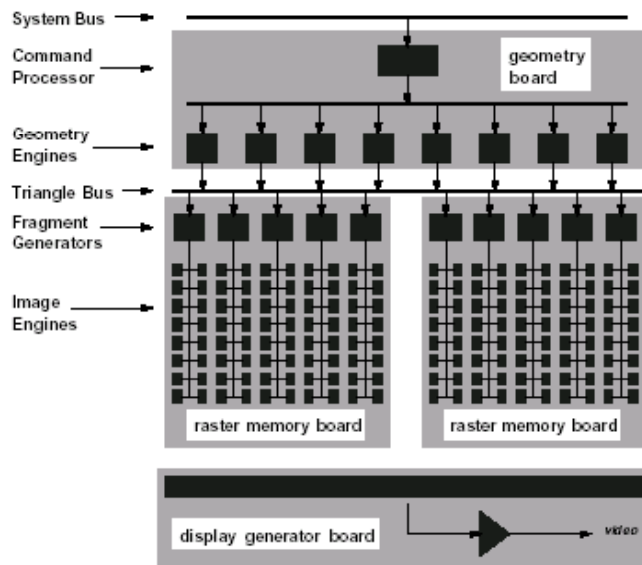
## Workstations

SGI RE and IR and others ...

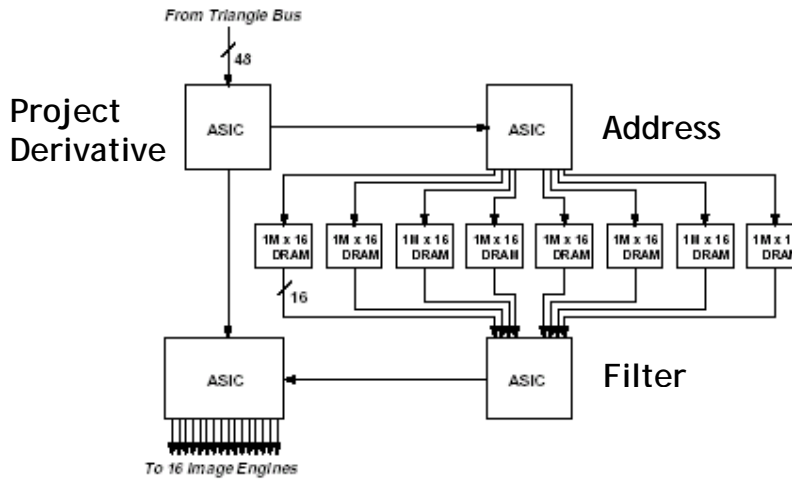
## Single-chip PC

3DFX and Nvidia and others ...

# RealityEngine (3<sup>rd</sup> Generation)



# RE Fragment Generator



Capacity: 16 MB = 8 MT (>1024<sup>2</sup> mipmap)

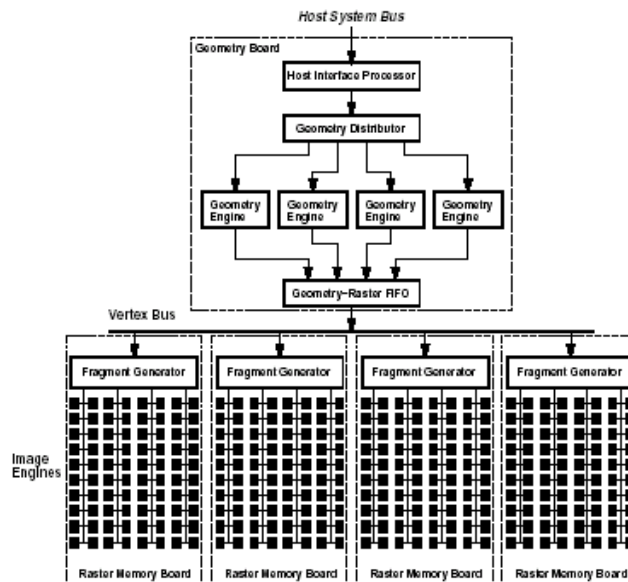
Texture replicated per fragment generator (5,10,20)x16MB

Fill Rate: 12 MT/s x (5,10,20) = (60,120,240)

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# InfiniteReality (3<sup>rd</sup> Generation)

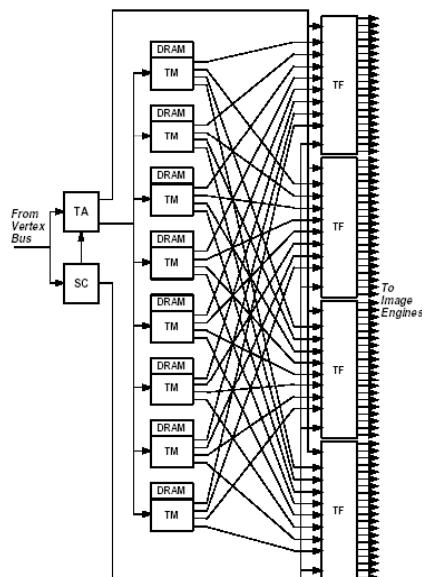


CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007



## InfiniteReality



2x2 fragment quads (TA)  
 8 texture memories (TM)  
 4 fragments/texture mem.  
 2x2 texture filterers (TF)  
 8 x 4 crossbar TM->TF

Capacity: (16 MB, 64, 256)  
 Fill: 200 MT/s x (1,2,4)

# Texture Locality and Caching

## Memory Access

---

High bandwidth required

- 1 GT/s  $\Rightarrow$  32 GB/s texture read (8 16-bit texels)

Mip map accesses

- Interleaving reduces coherence

Memories

- Large granularity
- High latency

## Solutions

---

### Caching

- Reduce the bandwidth requirement
- Match granularity of accesses to memory

### Prefetching

- Hide the high latency of memory accesses
- Handle highly variable latency

Compression (not covered)

## Unique T/F Ratio

---




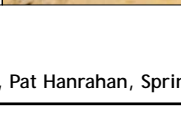
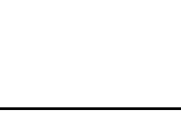
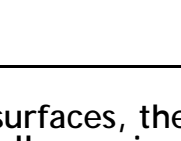
Key statistic: *unique texel to fragment ratio*

- Average memory bandwidth required

Definition:

Total texels accessed / Total fragments generated

## Texture Locality

	Percent trilinear	Unique T/F	Image
<b>quake</b>	<b>30%</b>	<b>0.033</b>	
<b>quake2x</b>	<b>47%</b>	<b>0.092</b>	
<b>flight</b>	<b>62%</b>	<b>0.706</b>	
<b>flight2x</b>	<b>87%</b>	<b>1.554</b>	
<b>qtv</b>	<b>0%</b>	<b>0.569</b>	
<b>qtv2x</b>	<b>100%</b>	<b>2.832</b>	

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Principle of Texture Thrift

Given a scene consisting of 3D textured surfaces, the amount of texture information minimally required to render an image of the scene is proportional to the resolution of the image and is independent of the number of surfaces and the size of the textures.

$$T = d t l$$

d - depth complexity

t - average number of textures per surface

D. Peachey, Texture on demand, PIXAR Technical Memo, 1990

Mipmaps

1. Constant time to filter a textured fragment
2. Output sensitive algorithm

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Textures: Spatial Locality

---

Bilinear filtering

2 x 2 neighborhood (4 texels)

Trilinear filtering

2 x 2 x 2 neighborhood (8 texels)

Samples shared by neighboring fragments

Small cache captures this spatial locality

Reduces the bandwidth by a factor of 4-8

## Textures: Temporal Locality

---

Parameters influencing texture locality

- Small repeated textures
- Average magnification of textures
- Percentage of magnified textures
- Level of detail bias
- Average minification when mipmapping

$\text{Frac}(d) \sim 0 \Rightarrow T/F \sim 1.25$

$\text{Frac}(d) \sim 1 \Rightarrow T/F \sim 5$

Empirically around 2.0

# Texture Cache Design

## Cache parameters

- Line size (blocking)
- Cache size (working set)
- Direct-mapped or associative

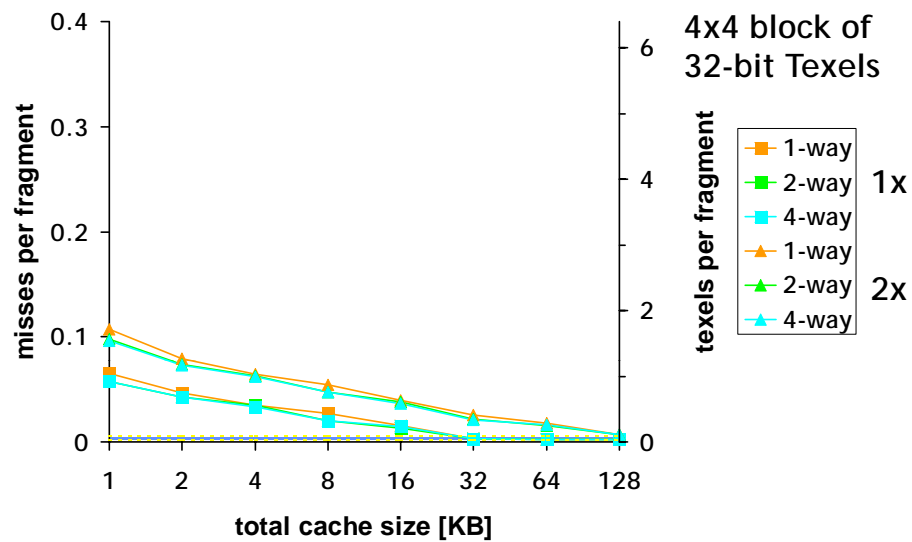
## Texture

- Representation of textures in memory
- Rasterization order

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# Quake Miss Rate

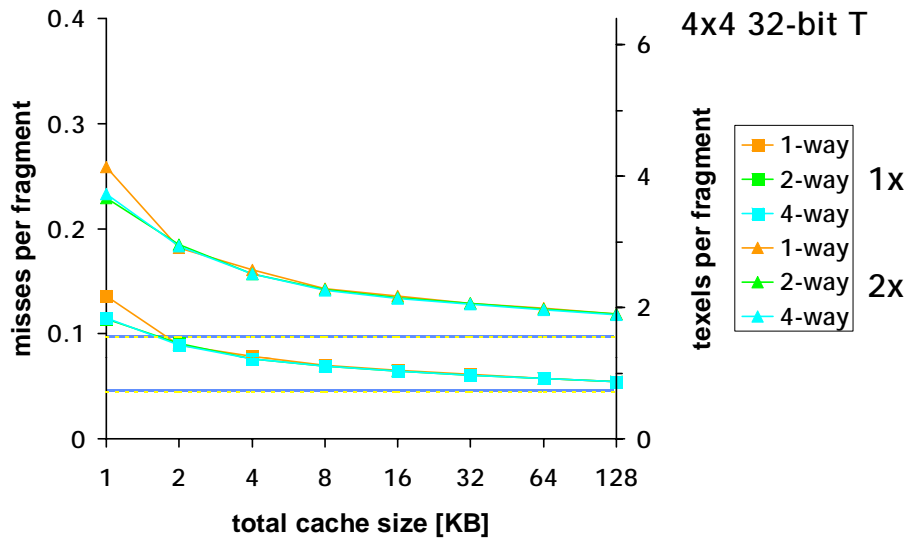


CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007



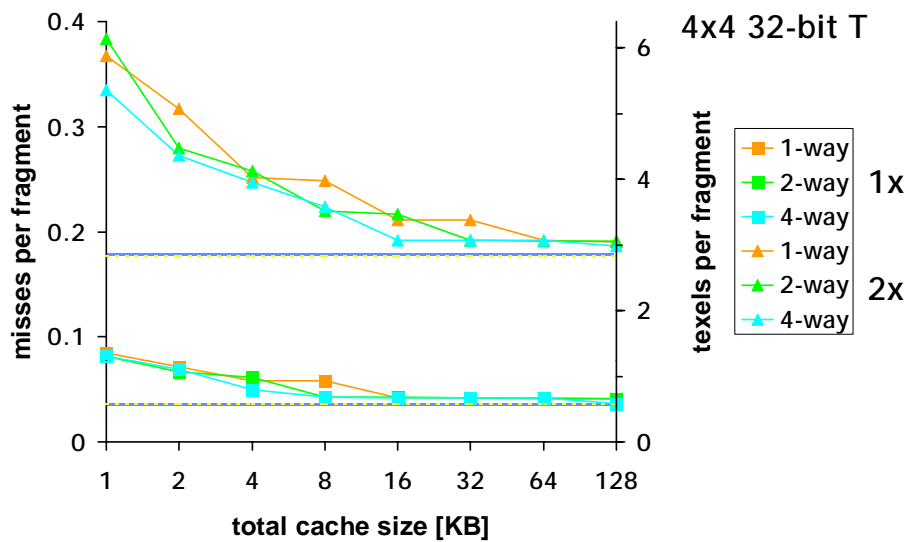
## Flight Miss Rate



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

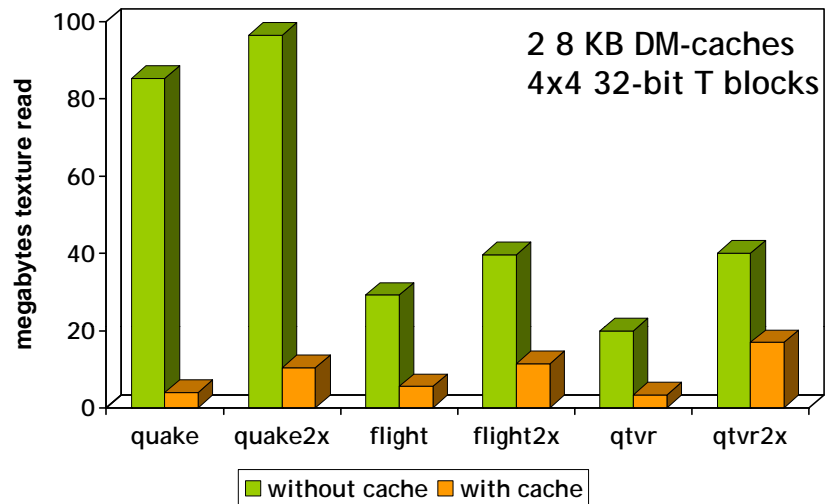
## QTVR Miss Rate



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

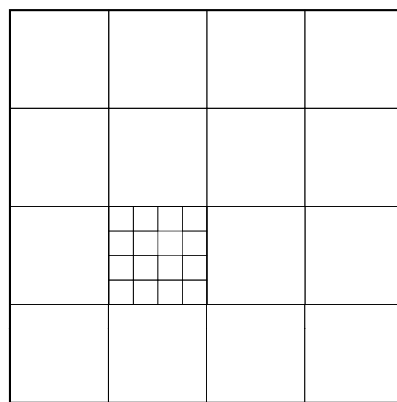
## Bandwidth Savings



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Texture Blocking



2D blocks

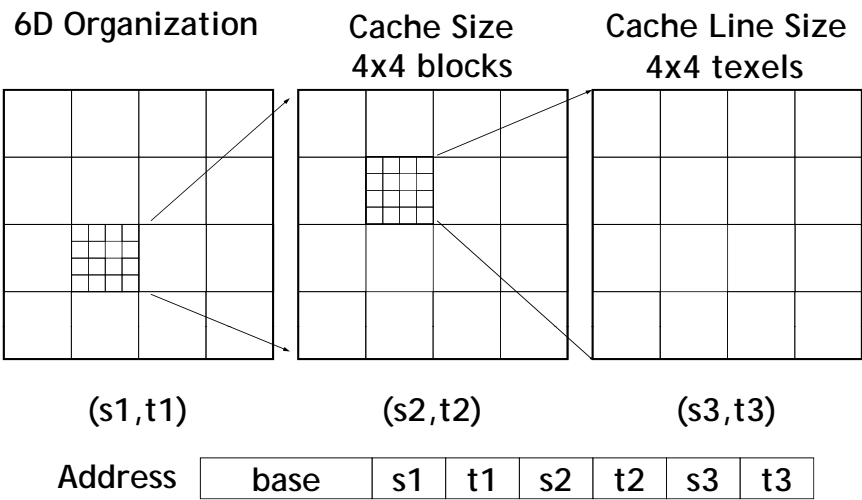
Hide orientation effects

Texture Map

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

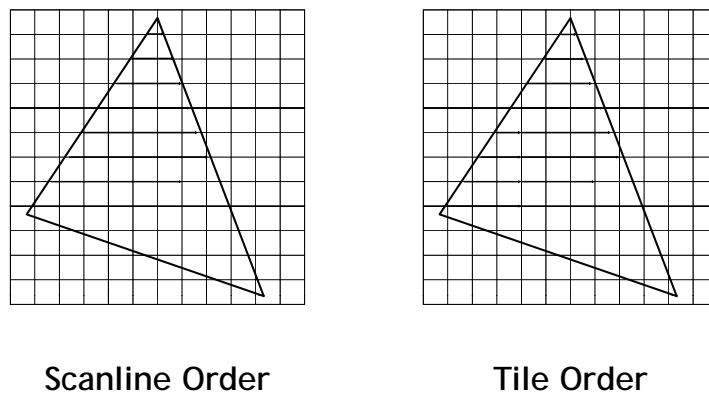
# Texture Blocking



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# Rasterization Order



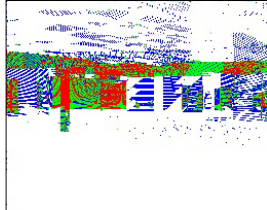
CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

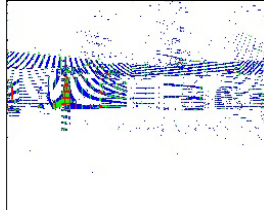
## Tiling and Blocking Results

**32KB, 2-way Associative, 128 byte lines**  
**8x8 blocks, 8x8 tiles**

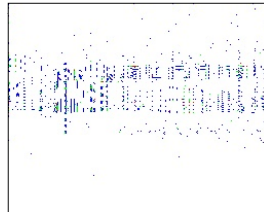
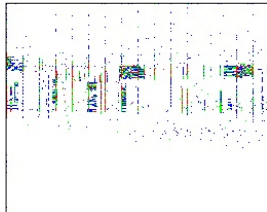
**Linear**



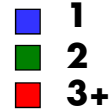
**Blocked Textures**



**Tiled Rasterization** **Blocked and Tiled**



**Misses**



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## Summary: Texture Caching

- Reasonably small working sets
  - 16-32 KB caches has 95% hit rate
- Separate caches for even and odd mip-levels to prevent conflicts
  - Alternatively 2-way associative cache
- Blocked textures further reduces miss rate
- Tiled rasterization further reduces miss rate
- 6D tiling minimizes working set

**Conclusion**

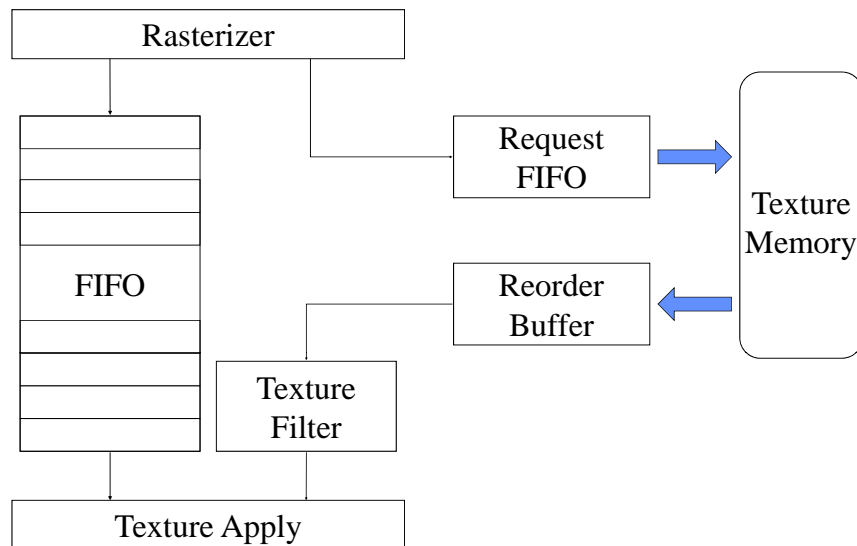
Caches highly effective for reducing texture memory bandwidth (roughly 5-10:1)

CS448 Lecture 6

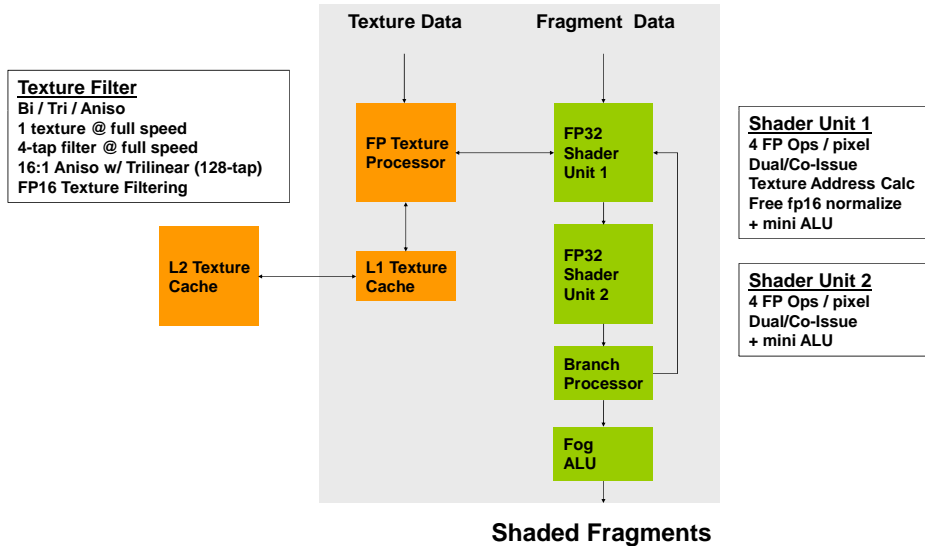
Kurt Akeley, Pat Hanrahan, Spring 2007

# Texture Latency

## Texture Prefetching Architecture



## GeForce 6800 Fragment Processor



CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

## CPU vs GPU

### Intel 3 Ghz Pentium 4

- 12 GFLOPS peak performance (via SSE2)
- 6 GB/sec peak memory bandwidth
- 44 GB/sec peak bw from 8K L1 data cache

### NVIDIA GeForce 6800

- 45 GFLOPS peak performance
- 36 GB/sec peak memory bandwidth
- 21 GB/sec peak bw from ?K L1 data cache

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007

# Graphics Texture Caches

Caches optimized for

- Exploiting spatial locality when mip-mapping

- Performing address arithmetic

  - 6D blocking

  - Projective floating point addresses

- Texture decompression

  - Block compression

  - 4 8-bit pixel components to 8 floats

- Performing filtering

  - Access to neighbors

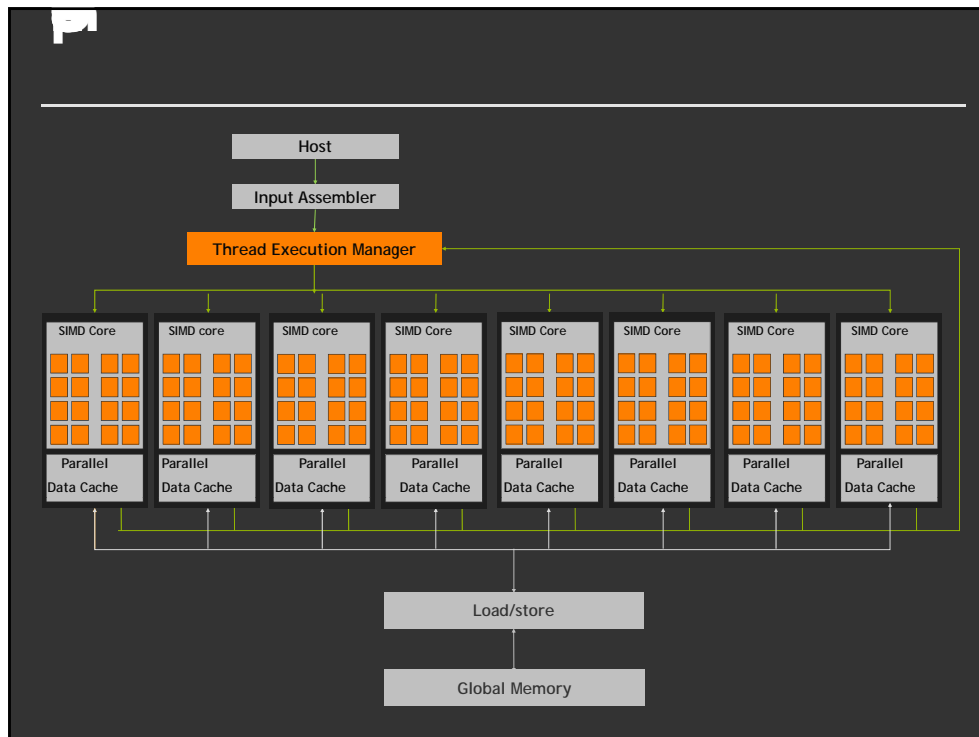
Caches not optimized for

- Exploiting temporal locality and increasing bandwidth

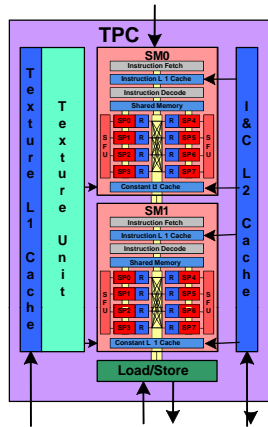
- Tolerating latency

CS448 Lecture 6

Kurt Akeley, Pat Hanrahan, Spring 2007



# Texture Processor Cluster (TPC)

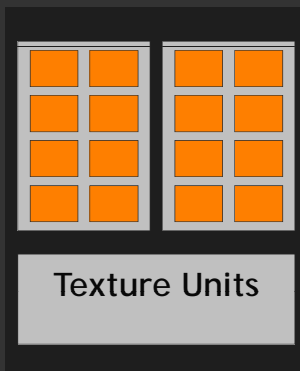


- **Texture-Processor Cluster (TPC)**
  - Texture unit, L1 texture cache
  - 2 Streaming Multiprocessors (SM)
  - 8 FP MAD / clock
  - L2 Instruction & Data Caches
- **Memory and Texture access**
  - Texture, load/store interfaces
  - Registers decouple latency

© David Kirk/NVIDIA and Wen-mei W. Hwu, 2007  
ECE 498AL, University of Illinois, Urbana-Champaign

47

## Each core



- 8 functional units
- SIMD 16/32 "warp"
- 8-10 stage pipeline
- Thread scheduler
- 128-512 threads/core
- 16 KB shared memory

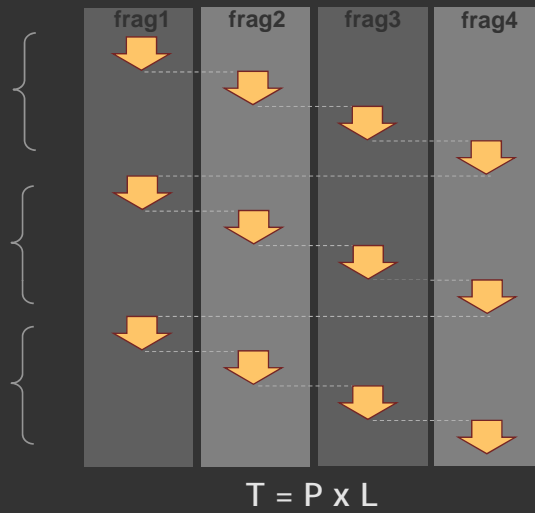
**Total #threads/chip**

$$16 * 512 = 8K$$

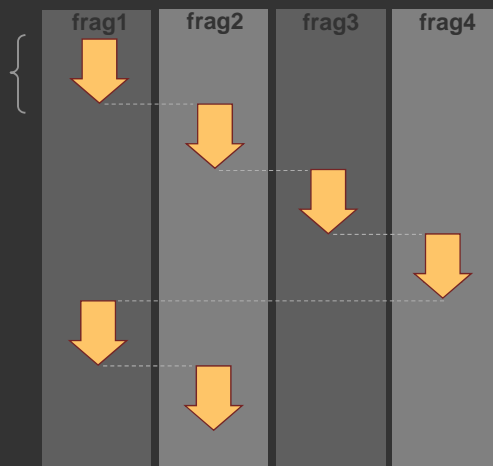




## Change threads each cycle (round robin)



## Change thread after texture fetch/stall



## Wrap-Up

### Trends and Pitfalls

---

#### Trends

- Programmable texture coordinate generation
  - Dependent textures
- Better quality filters
- Many textures per surface
- Virtual texture memory(?)
- GPUs are multi-threaded SIMD engines
  - SIMD because of parallelism
  - Multi-threaded to handle texture latency

## Trends and Pitfalls

---

### Pitfalls

- 2D texture memory allocation; use 1D!
- Texture thrashing (draw in texture order)
- Handling borders is complicated
- Precision: very large textures (swimming)

## Additional Topics

---

CATS and RATS; RIP-MAPS

Anisotropic filtering

Detail textures

Texture compression

Texture management; clip-maps

Parallel texture caching