

Real-Time Graphics Architecture

Lecture 2 The Graphics Pipeline

Kurt Akeley

Pat Hanrahan

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

This lecture

Overview today

Details in subsequent lectures

Today's topics

- Defining graphics architectures
- The graphics pipeline
- Computation and bandwidth requirements

Declarative vs. imperative

Declarative (what, not how)

- Descriptive: specify input and desired result
 - E.g., `OmitHiddenSurfaces()`;
- Example systems:
 - RenderMan scene description
 - Inventor and Performer scene graphs

Imperative (how, not what)

- Procedural: specify actions to create the result
 - E.g., `EnableDepthBuffer()`;
- Example systems
 - PostScript and Xlib
 - OpenGL and Direct3D

CS448 Lecture 2

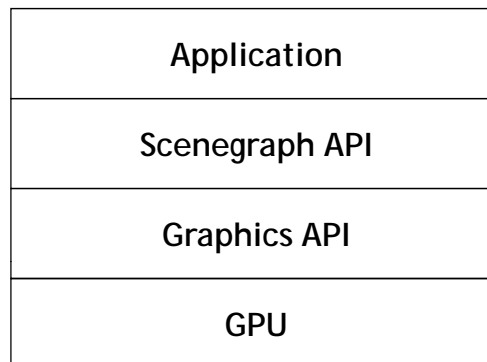
Kurt Akeley, Pat Hanrahan, Spring 2007

Graphics system stack

Declarative



Imperative



CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

A procedural API defines an architecture

We mean architecture in the Brooks/Blaauw sense:

- Separate from and above implementation

Requires a thorough specification

- Good examples: OpenGL, Direct3D 10
- Not so good examples: early Direct3D versions

(unusual) can separate API interface and architecture

- Direct3D 10 (interface)
- WGF 2.0 (architecture)

OpenGL drawing commands

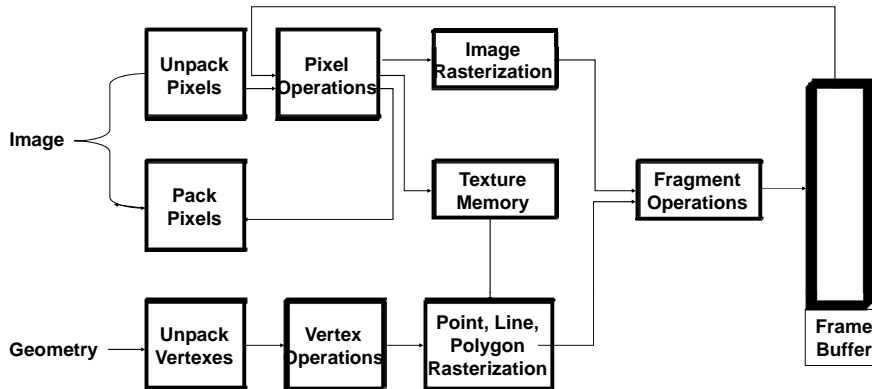


```
glBegin(GL_POLYGON);  
  glColor(RED);  
  glVertex3i(0,0,0);  
  glVertex3i(1,0,0);  
  glVertex3i(0,1,0);  
glEnd();
```



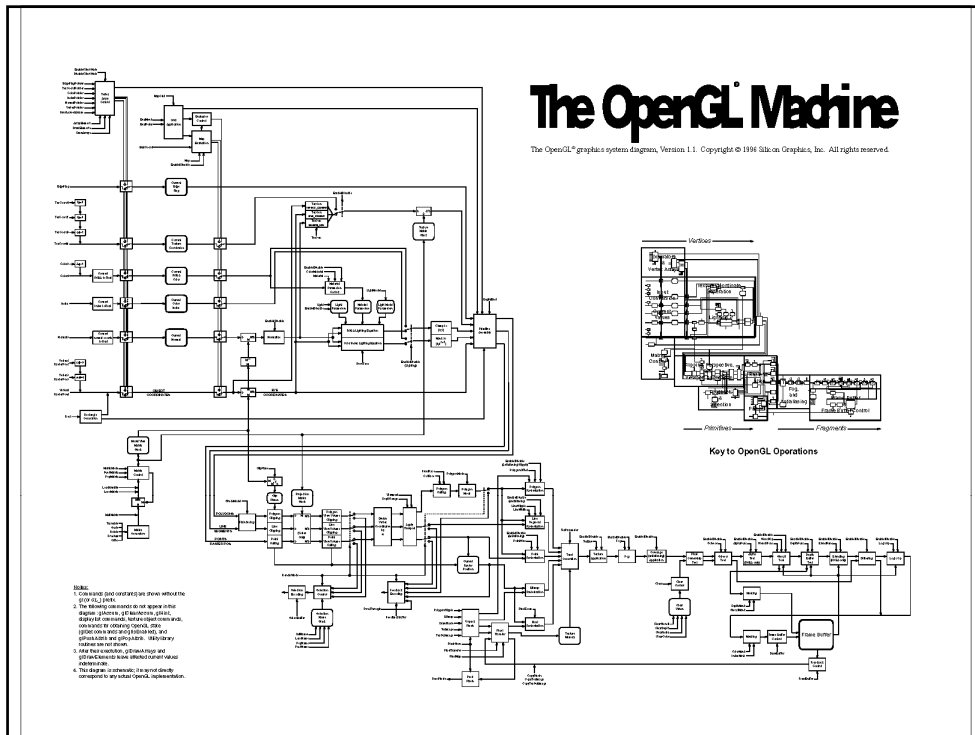
```
glBegin(GL_POLYGON);  
  glColor(RED);  
  glVertex3i(0,0,0);  
  glColor(BLUE);  
  glVertex3i(1,0,0);  
  glColor(BLUE);  
  glVertex3i(0,1,0);  
glEnd();
```

OpenGL architecture



CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007



ISA vs. API

CPU architecture

- Specified as ISA (instruction set architecture)
 - But could be API (e.g., Java machine)
- ISA is extremely low level
 - Constraining for implementors
 - Powerful for programmers

GPU architecture

- Specified as an API (but could be ISA ... shaders)
- API is somewhat higher level
 - Less constraining for implementors
 - Less powerful for programmers (but still imperative!)

ISA vs. API

Can think of graphics API as VLIW-like instructions:

- Specify operations
 - Multiple functional units
 - Orthogonal operations
- Specify data paths
 - Composition of operations

Popular for 3rd generation GPUs

Useful for understanding equivariance

OpenGL (ISA-based) equivariance example

Let

R = render a scene, and

T = translate by an integral number of pixels

Then

$$R(T(\text{primitive})) = T(R(\text{primitive}))$$

OpenGL is not a pixel-exact specification

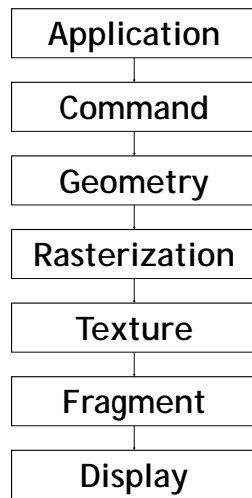
- Equivariance guarantees allow powerful usage
- With less implementation constraint

EQUIVariance is not the same as INVariance

- OpenGL specifications missed this distinction

The Graphics Pipeline

Canonical graphics pipeline



Forward-Algorithm

A trip down the graphics pipeline

Application

- Simulation
- Input event handlers
- Modify data structures
- Database traversal
- Primitive generation
- Utility functions



Command

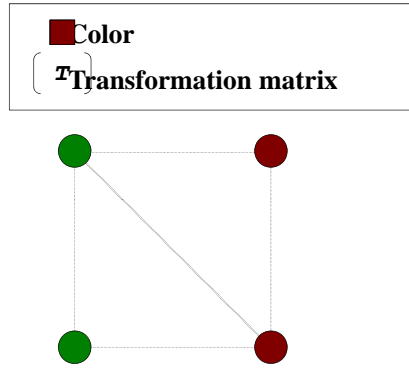
Command buffering

Command interpretation

Unpack and perform format conversion

Maintain graphics state

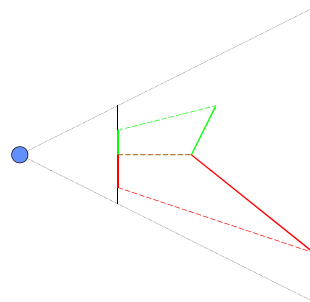
```
glLoadIdentity( );  
glMultMatrix( T );  
glBegin( GL_TRIANGLE_STRIP );  
glColor3f ( 0.0, 0.5, 0.0 );  
glVertex3f( 0.0, 0.0, 0.0 );  
glColor3f ( 0.5, 0.0, 0.0 );  
glVertex3f( 1.0, 0.0, 0.0 );  
glColor3f ( 0.0, 0.5, 0.0 );  
glVertex3f( 0.0, 1.0, 0.0 );  
glColor3f ( 0.5, 0.0, 0.0 );  
glVertex3f( 1.0, 1.0, 0.0 );  
...  
glEnd( );
```



Geometry

Evaluation of polynomials for curved surfaces

Transform and projection



Geometry

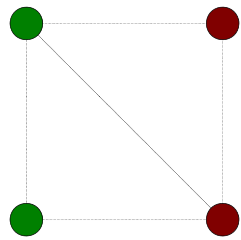
Evaluation of polynomials for curved surfaces

Transform and projection (object \rightarrow image space)

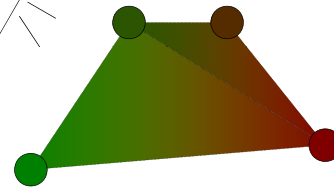
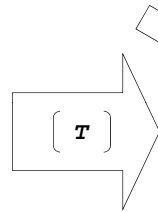
Clipping, culling and primitive assembly

Lighting (light sources and surface reflection)

Texture coordinate generation



Object-space triangles



Screen-space lit triangles

CS448 Lecture 2

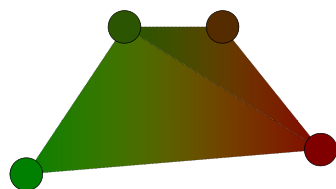
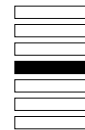
Kurt Akeley, Pat Hanrahan, Spring 2007

Rasterization

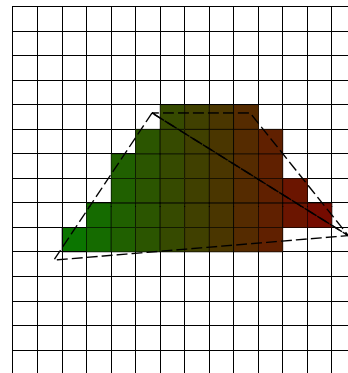
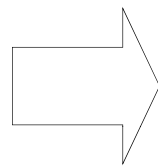
Setup (per-triangle)

Sampling (triangle = {fragments})

Interpolation (interpolate colors and coordinates)



Screen-space triangles



Fragments

CS448 Lecture 2

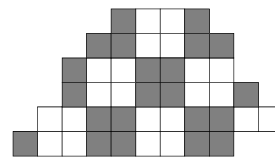
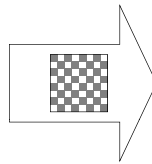
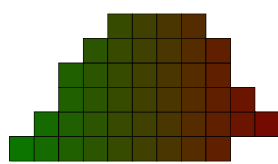
Kurt Akeley, Pat Hanrahan, Spring 2007

Texture

Texture transformation and projection

Texture address calculation

Texture filtering



Fragments

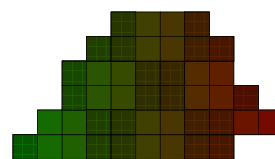
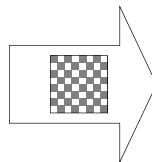
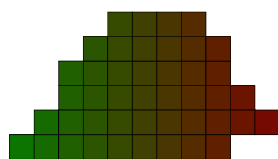
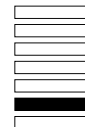
Texture Fragments

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Fragment

Texture combiners



Fragments

Textured Fragments

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

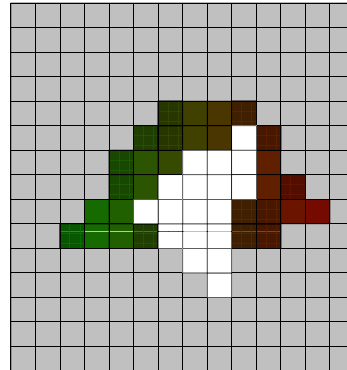
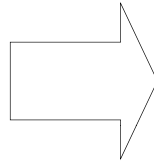
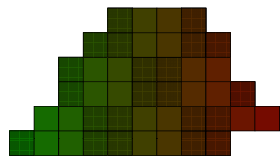
Fragment

Texture combiners and fog

Owner, scissor, depth, alpha and stencil tests

Blending or compositing

Dithering and logical operations



Textured Fragments

Frame buffer Pixels

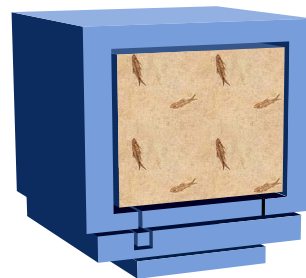
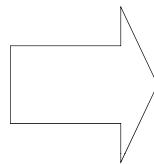
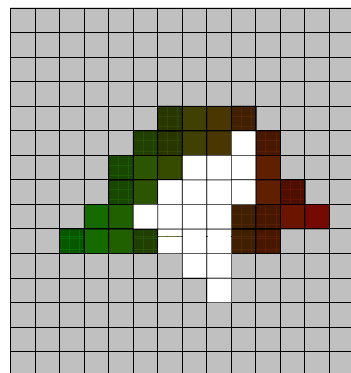
CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Display

Gamma correction

Analog to digital conversion



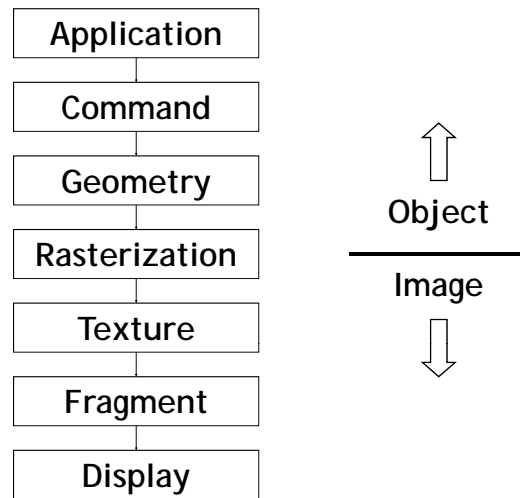
Frame buffer Pixels

Light

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Canonical graphics pipeline

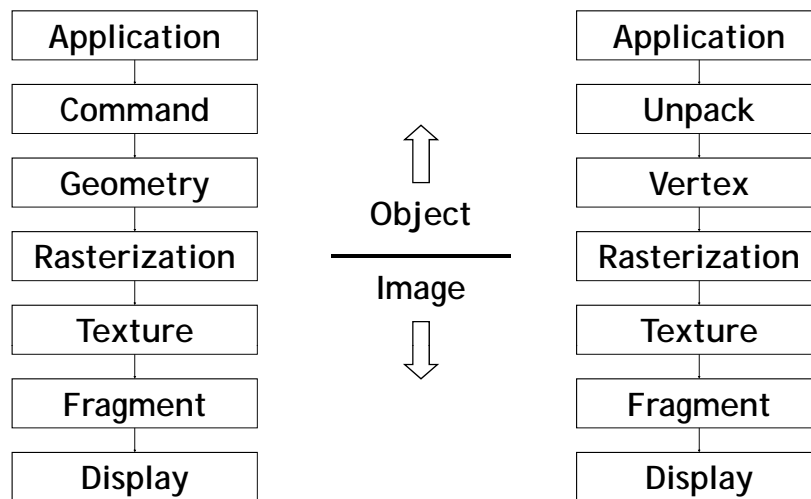


CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

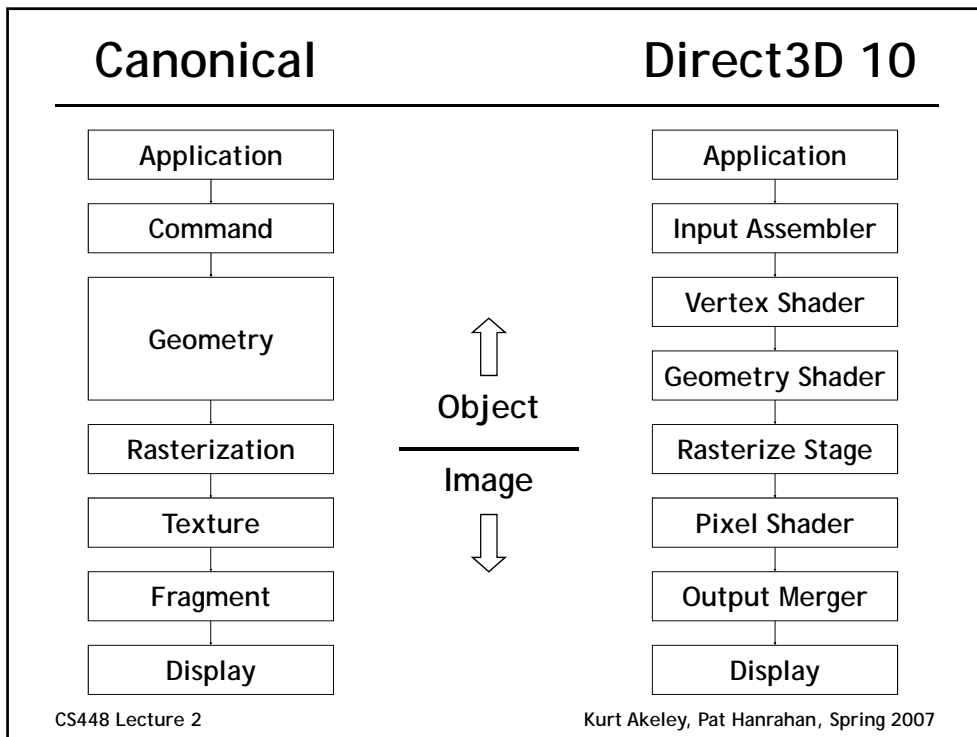
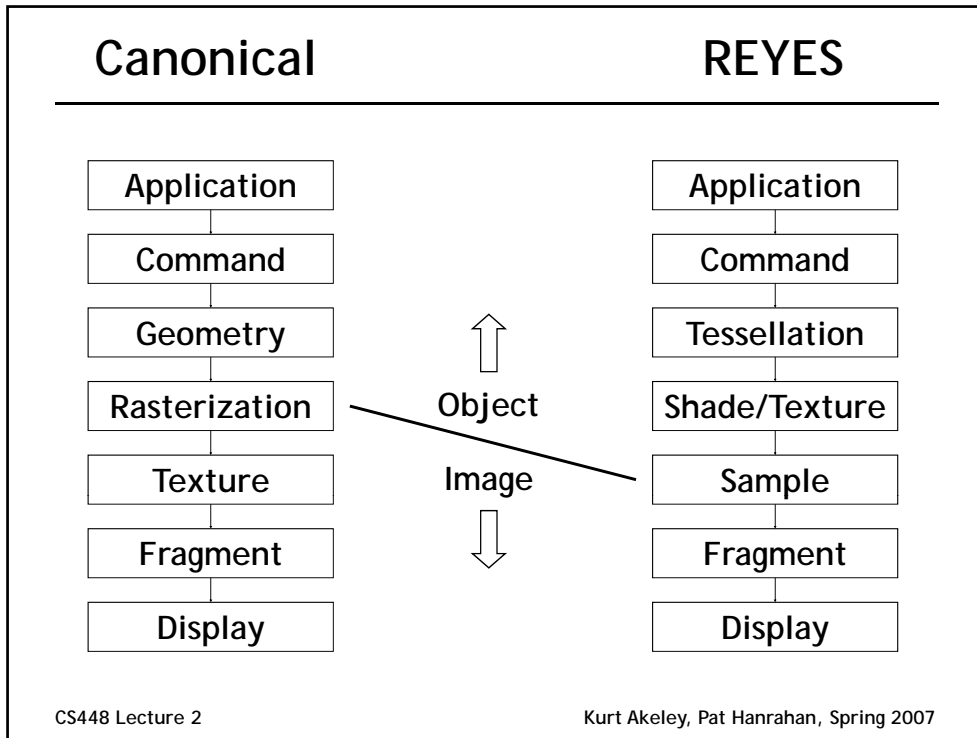
Canonical

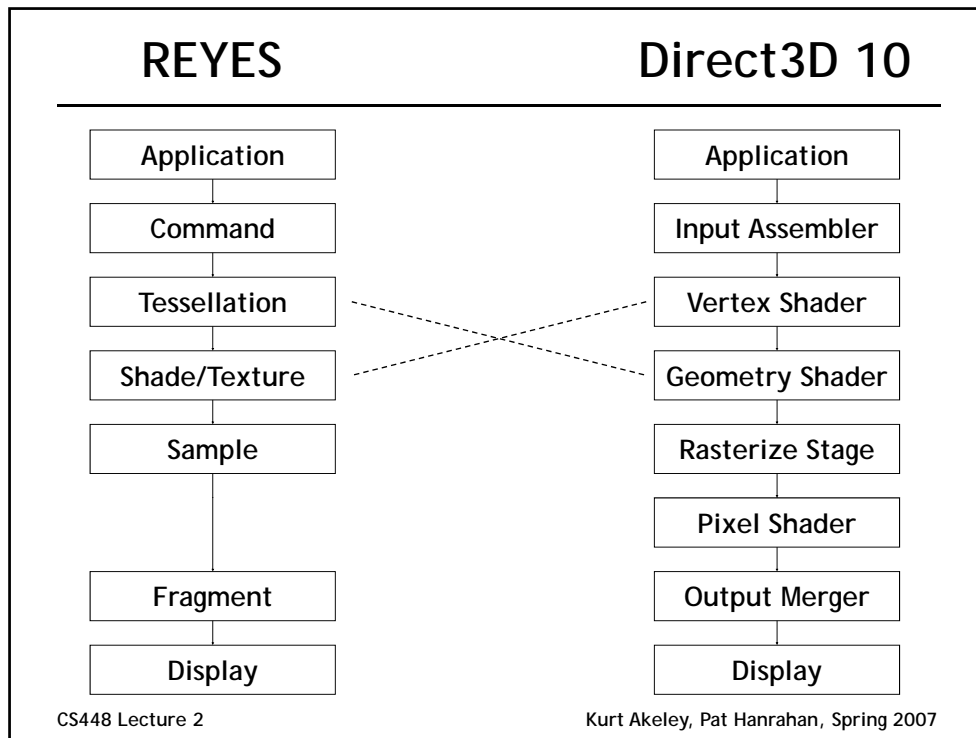
OpenGL 1.2



CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007





Sequence differences

Semantic

- New operations, e.g.,
 - Full-scene antialiasing
- Different pipelines, e.g.,
 - REYES shading and texture mapping in object space
 - Direct3D 10 tessellation *after* vertex shading

Implementation (no semantic difference)

- Early Z-cull (as in Direct3D 10)
- Tiled rendering
- Abstraction
 - abstraction distance

Graphics state (aka context)

Required to minimize data transmission

Resources (shared or global, persistent)

- Fonts
- Texture
- Display lists

Attributes

- Appearance: Lights, Materials, Colors, ...
- Transformation: camera, model, texture, ...
- Options: fb formats, constant per-frame

Graphics State

Ideally small and bounded

e.g., maximum number of lights

OpenGL: ~12kb

Distributed throughout the pipeline

Difficult to manage (forces major design decisions)

Must often be broadcast; must be consistent

Difficult to implement context switching

OpenGL has a single context; X has multiple contexts

One active context → windows are difficult

Expensive to query state

The Wheel of Reincarnation

General-purpose processor + display processor

Display processor evolution:

1. Refresh display from vector- or frame-buffer
2. Augment with drawing processor
3. Augment with structured display list processor
4. General drawing processor + display processor

Myers and Sutherland: Break cycle and KISS

J. Clark: Terminals should be workstations and workstations should support immediate-mode graphics efficiently

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Computational Requirements

Functionality vs. frequency

Geometry processing = per-vertex

Transformation and Lighting (T & L)

Floating point; complex operations

100s of millions of vertices

Fragment processing = per-fragment

Blending and texture combination

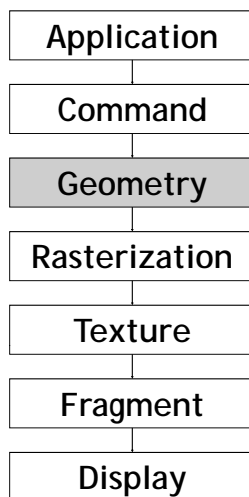
Fixed point; limited operations

10s of billions of fragments

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Geometry computations



Geometry (per-vertex)

Assumptions:

-1 infinite light

-Texture coordinates

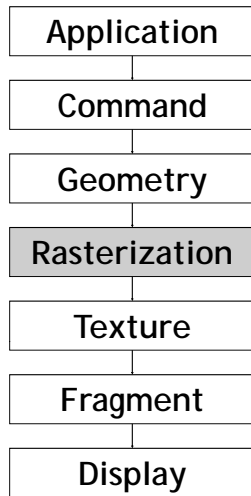
ADD	CMP	MUL	DIV	SPE
40	8	53	1	1

Rough estimate: 100 ops

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Rasterization computations (vertex)



Rasterization: per-vertex
Assumptions:
- 7 interpolants (z,r,g,b,s,t,q)

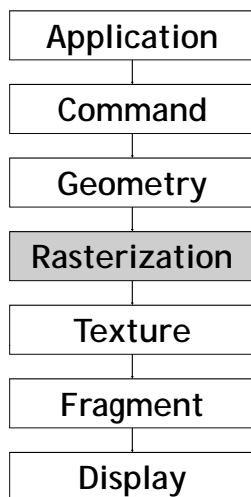
ADD	CMP	MUL	DIV	SPE
62	22	55	4	0

Rough estimate: 150 ops

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Rasterization computations (fragment)



Rasterization: per-fragment
Assumptions:
- 7 interpolants (z,r,g,b,s,t,q)

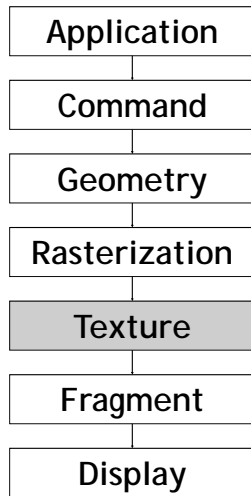
ADD	CMP	MUL	DIV	SPE
16	3	6	0	0

Rough estimate: 25 ops

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Texture computations



Texture: per-fragment
Assumptions:
-Projective texture mapping
-Level of detail calculation
-Trilinear interpolation

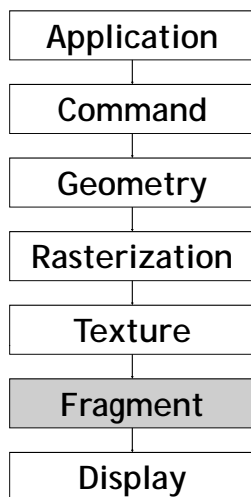
ADD	CMP	MUL	DIV	SPE
42	5	48	1	3

Rough estimate: 100 ops

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Fragment computations



Fragment: per-fragment
Assumptions:
-Texture blending
-Color blending
-Depth buffering

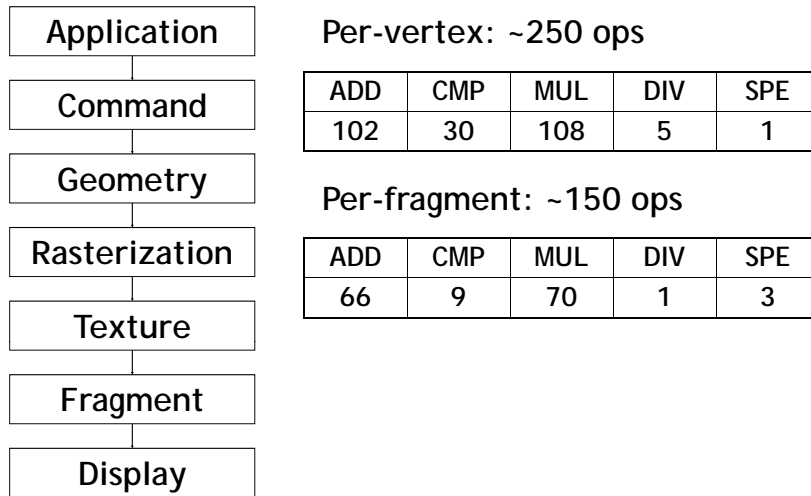
ADD	CMP	MUL	DIV	SPE
8	1	16	0	0

Rough estimate: 25 ops

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

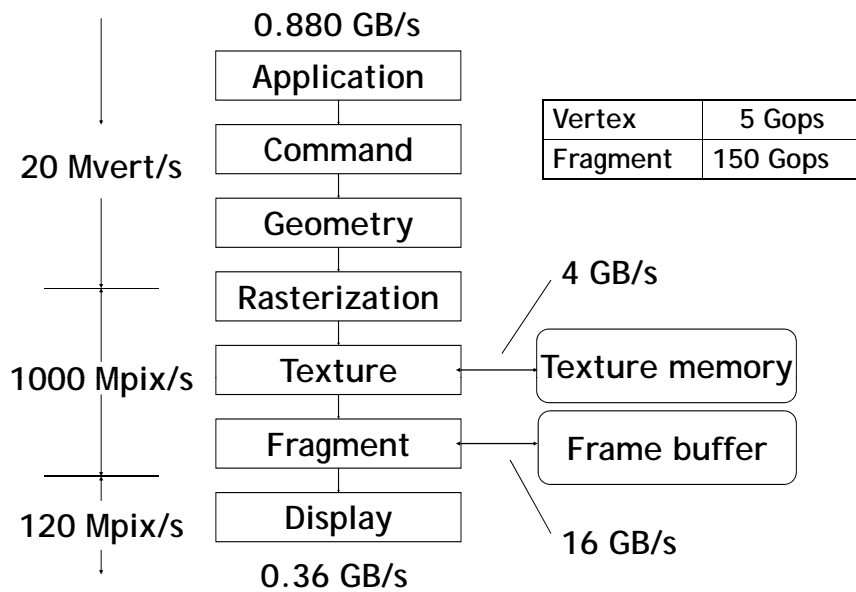
Total computations (rough estimate)



CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Bandwidths (rough estimate)



CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Comparison to current GPU

	Gops/sec	Gbytes/sec
Our 2001 estimation	155	21
GeForce 7900 GTX (early 2006)	2100	51

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Summary

An imperative API defines an architecture

- Much as an ISA defines one

An imperative API defines an abstract machine

- Allows variations in implementation
- But large “abstraction distances” are problematic

The canonical OpenGL-like pipeline persists

- Many features added over the years
- OpenGL has slowed the Wheel of Reincarnation

We have some sense of GPU speeds and feeds

CS448 Lecture 2

Kurt Akeley, Pat Hanrahan, Spring 2007

Real-Time Graphics Architecture

Lecture 2 The Graphics Pipeline

Kurt Akeley

Pat Hanrahan

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