The Graphics Pipeline

Topics

1. Defining graphics architectures
2. The graphics pipeline
3. Computation and bandwidth requirements

Papers

Required

1. The design of the OpenGL graphics interface, M. Segal and K. Akeley

Optional

1. The OpenGL Specification, M. Segal and K. Akeley

Both papers will be available from course web site:

http://www.graphics.stanford.edu/courses/cs448a-01-fall

Lecture slides will also be available online

Graphics Systems and Libraries

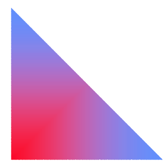
Declarative (What, not How)

- Describe the scene
- For example: virtual camera
- Systems
  - RenderMan scene description
  - Inventor and Performer scene graphs

Imperative (How, not What)

- Emit a sequence of drawing commands
- For example: load model-view matrix
- Systems
  - OpenGL
  - PostScript and Xlib

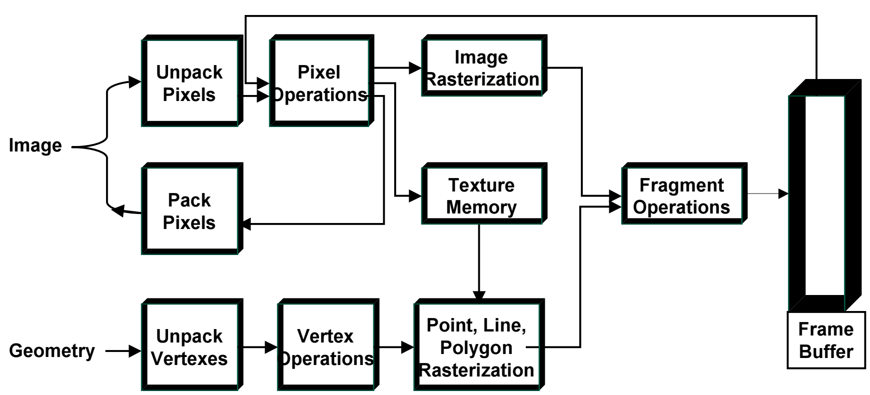
Drawing Commands

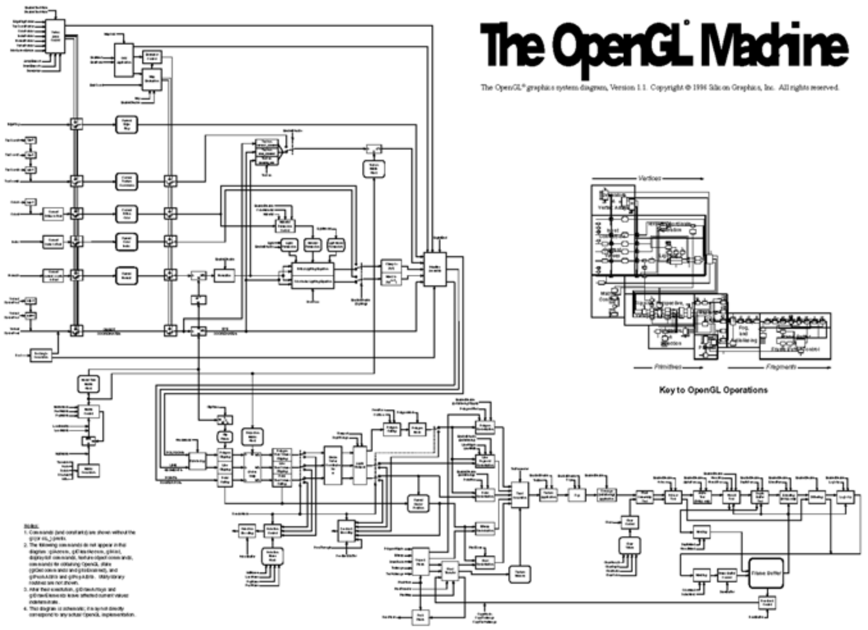


```
glBegin(GL_POLYGON);
  glColor(RED);
  glVertex3i(0,0,0);
  glVertex3i(1,0,0);
  glVertex3i(0,1,0);
glEnd()
```

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


OpenGL Architecture





API = ISA

VLIW-like instructions

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

ISA Specification

Invariance

- Does NOT precisely define drawing commands

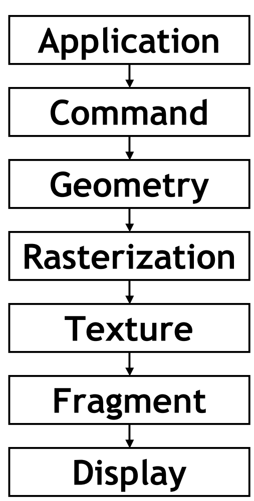
  For example:

  Does not specify what pixels are inside a triangle

  Does not specify the precision of intermediate calculations

  - Image drawn by two systems may differ
- Does require invariance across modes
  - Image drawn in two modes must be the "same"

Modern 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

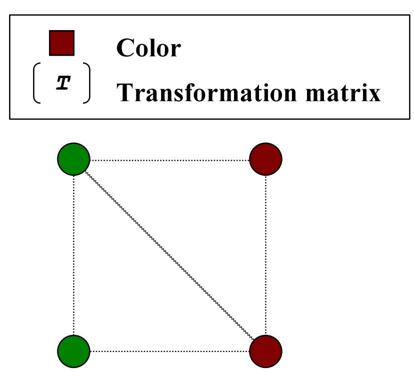
Command buffering

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( );
```

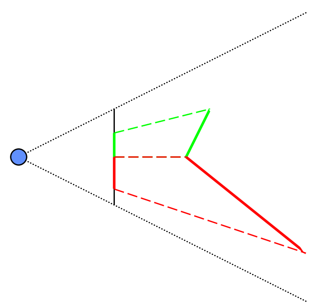


Geometry

Evaluation of polynomials for curved surfaces

Transform and projection

Clipping, culling and primitive assembly



CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Geometry

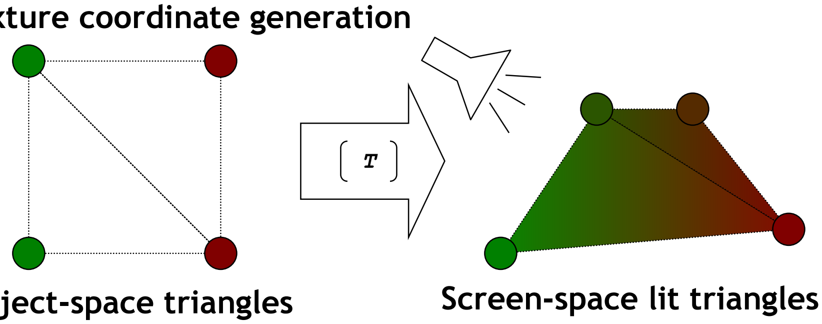
Evaluation of polynomials for curved surfaces

Transform and projection (object -> image space)

Clipping, culling and primitive assembly

Lighting (light sources and surface reflection)

Texture coordinate generation



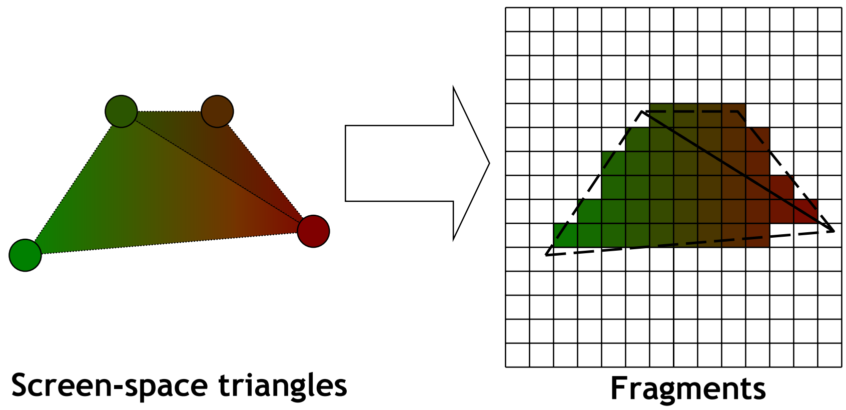
CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Rasterization

Setup (per-triangle)

Sampling (triangle = {fragments})

Interpolation (interpolate colors and coordinates)



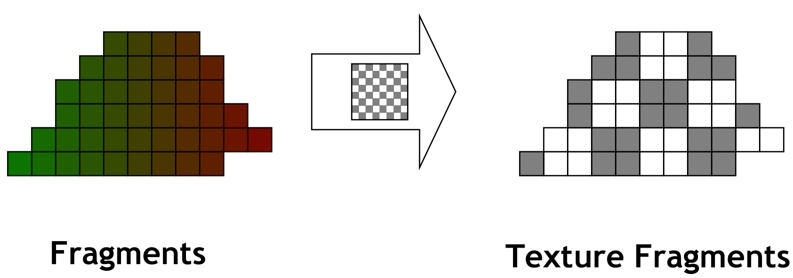
CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Texture

Texture transformation and projection

Texture address calculation

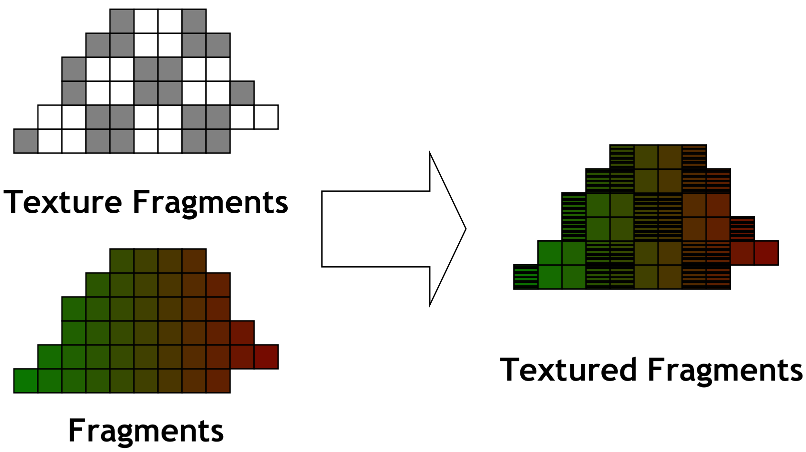
Texture filtering



CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Fragment

Texture combiners



CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

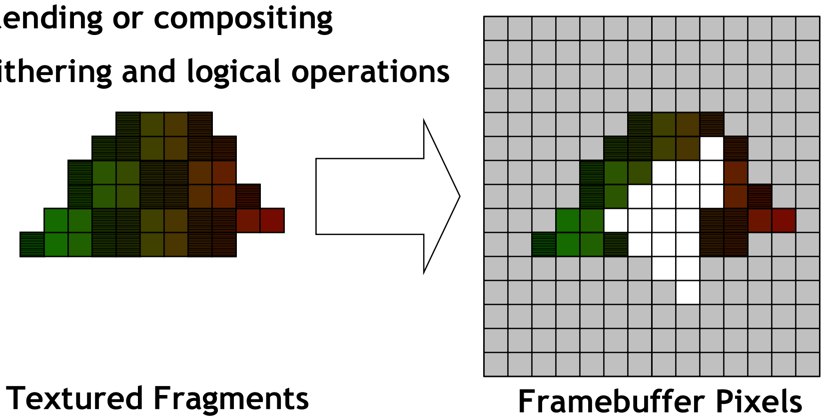
Fragment

Texture combiners and fog

Owner, scissor, depth, alpha and stencil tests

Blending or compositing

Dithering and logical operations

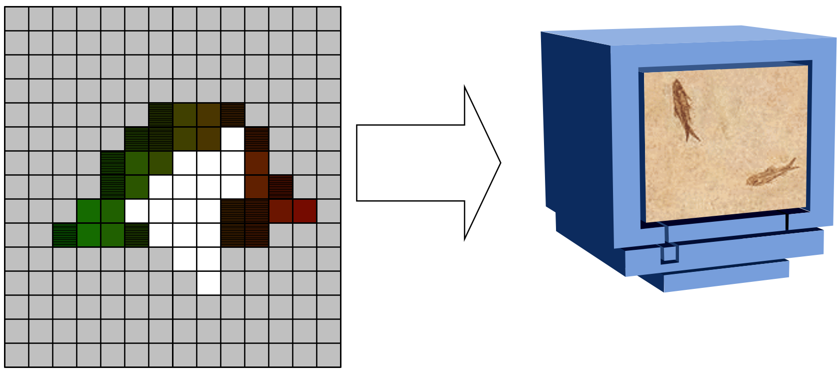


CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Display

Gamma correction

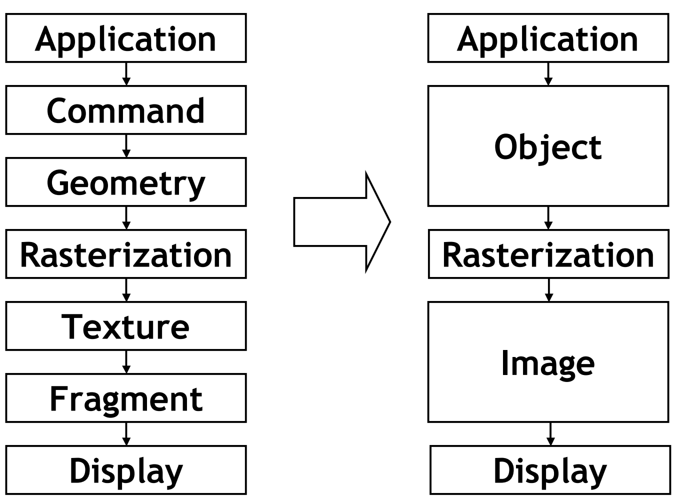
Analog to digital conversion



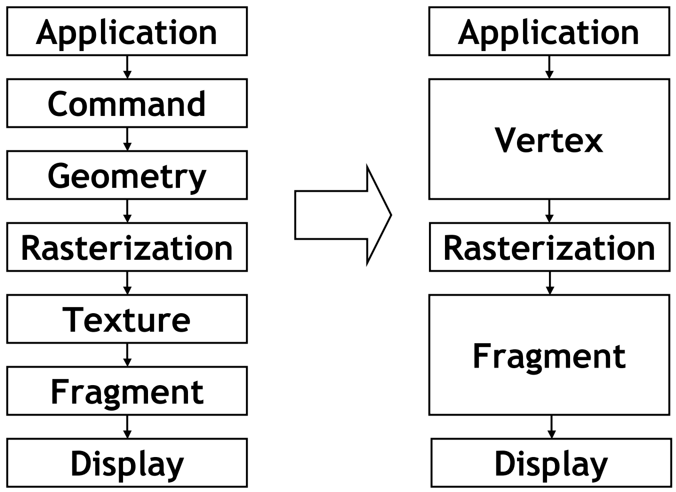
Framebuffer Pixels

Light

Graphics Pipeline



Graphics Pipeline



CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Functionality vs. Frequency

Geometry processing = per-vertex
- Transformation and Lighting (T & L)
- Floating point; complex operations
- 10 million vertices

Fragment processing = per-fragment
- Blending and texture combination
- Fixed point; limited operations
- 1 billion fragments

CS448 Lecture 2 Kurt Akeley, Pat Hanrahan, Fall 2001

Evolution of the Graphics Pipeline

History

- Framebuffers: display
- Geometry processing: transformation and lighting
- Rasterization: hidden surface and simple shading
- Texturing: perspective correct texture lookup
- Antialiasing: multisampling
- Shading: multiple textures and texture combiners

Where and how to insert new functionality?

Inserting Functionality: Order

Hidden-surface elimination

- painters algorithm = hide-first
- z-buffer = hide-last

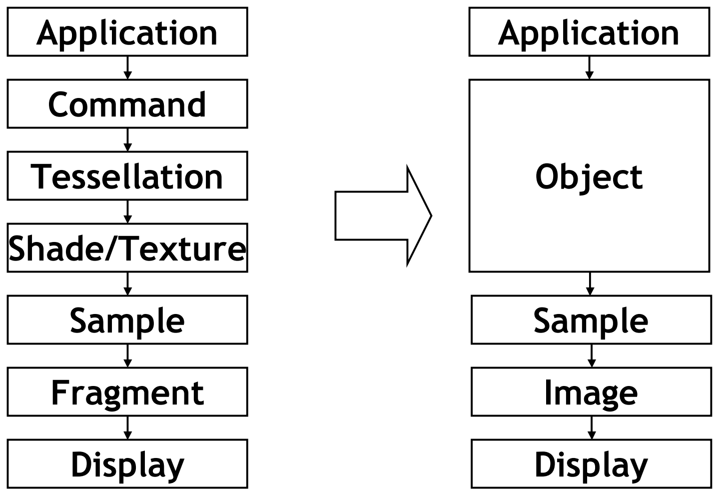
Texturing

- Fragment textures = texture-last
- Vertex textures = texture-first

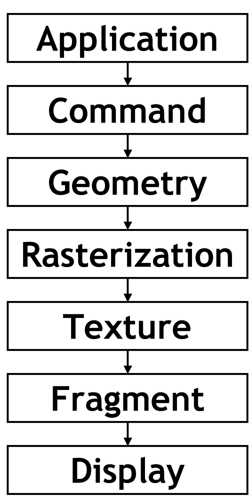
Shading

- Vertex shading = shade-first or shade-first-vertex
- Fragment shading = shade-last-fragment
- Deferred shading = shade-last-pixel

Reyes Architecture



Computational Requirements

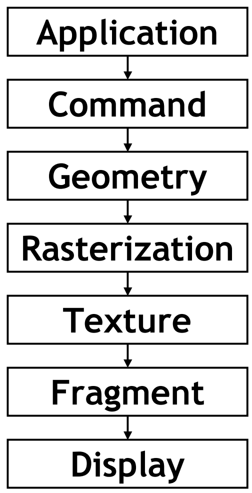


Geometry (per-vertex)
Assumptions:
-1 infinite light
-Texture coordinates

ADD	CMP	MUL	DIV	SPE
40	8	53	1	1

Rough estimate: 100 ops

Computational Requirements

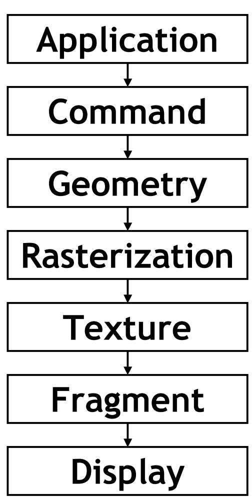


Rasterization: per-vertex
Assumptions:
- 7 interpolants (z,r,g,b,s,t,q)

ADD	CMP	MUL	DIV	SPE
62	22	55	4	0

Rough estimate

Computational Requirements

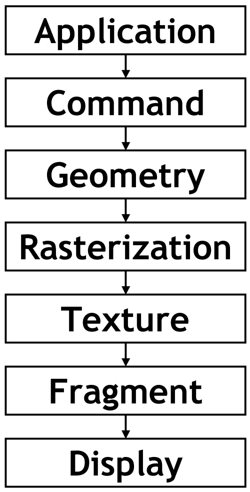


Rasterization: per-fragment
Assumptions:
- 7 interpolants (z,r,g,b,s,t,q)

ADD	CMP	MUL	DIV	SPE
16	3	6	0	0

Rough estimate

Computational Requirements

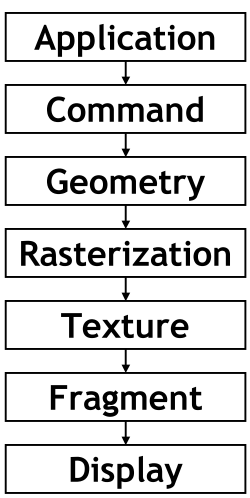


Texture: per-fragment
Assumptions:
-Projective texture mapping
-Level of detail calculation
-Trilinear interpolation

ADD	CMP	MUL	DIV	SPE
42	5	48	1	3

Rough estimate

Computational Requirements

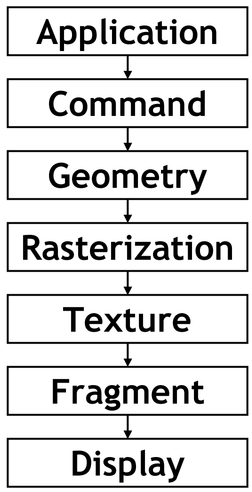


Fragment: per-fragment
Assumptions:
-Texture blending
-Color blending
-Depth buffering

ADD	CMP	MUL	DIV	SPE
8	1	16	0	0

Rough estimate

Computational Requirements



Per-Vertex

ADD	CMP	MUL	DIV	SPE
102	30	108	5	1

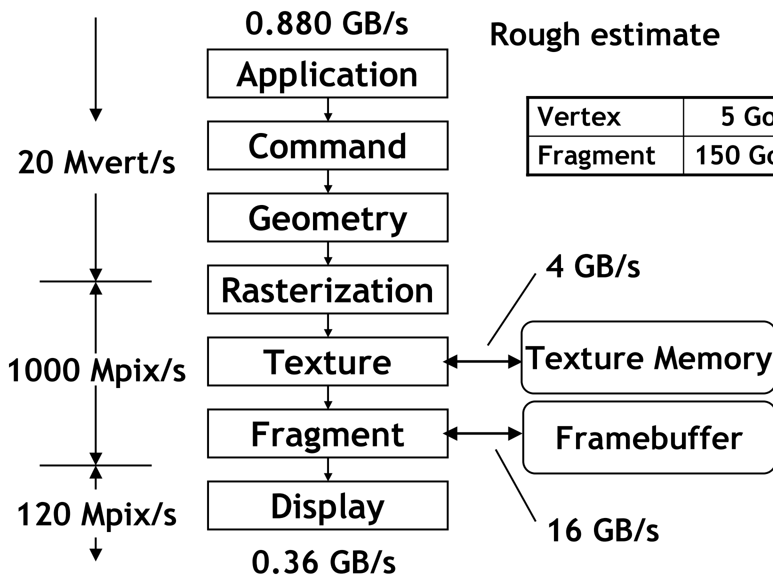
Per-Fragment

ADD	CMP	MUL	DIV	SPE
66	9	70	1	3

Rough estimate

Communication Requirements

Rough estimate



Vertex	5 Gops
Fragment	150 Gops

Graphics State or 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

Hard to switch contexts

OpenGL has a single context; X has multiple contexts

One drawing process; windows are difficult

Difficult to query state