

ECE/CS 412

Introduction to Computer Graphics

Class 16

Pradeep Sen
Advanced Graphics Lab



Last time

- Texture mapping



Today

- Wrap up with texture mapping (mipmaps)
- Vertex/fragment programs



Mipmapping

- MIP- *multum in parvo* (many things in a small place)
- Technique in which to perform pre-computed filtering for minification



Mipmapping

- Accessing the mipmap



Problems with mipmapping

- Decimated textures all have a square footprint (isotropic)



Anisotropic filtering

- Approximate the pixel “footprint” with an ellipse
- Compute contribution in ellipse through multi-sampling

Programmable shaders

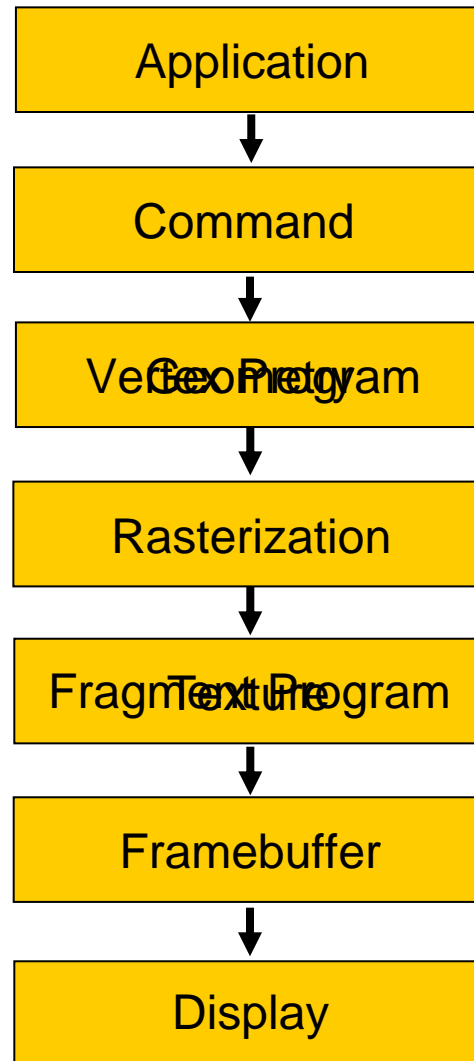


Vertex and fragment programs

- Adding programmability to the graphics pipeline



Programmable graphics pipeline



Vertex program

- Extension GL_ARB_vertex_program

Simple vertex program

```
!!ARBvp1.0
ATTRIB v                = vertex.position; # vertex position

# pass in the.mvp matrix
PARAM.mvp[4]            = { state.matrix.mvp };

# define the output
OUTPUT oPos              = result.position # output vertex

# transform the vertex v by the.mvp matrix
DP4 oPos.x,.mvp[0],v;
DP4 oPos.y,.mvp[1],v;
DP4 oPos.z,.mvp[2],v;
DP4 oPos.w,.mvp[3],v;

END
```

Simple vertex program

```
!!ARBvp1.0
ATTRIB v          = vertex.position; # vertex position

# pass in the.mvp matrix
PARAM.mvp[4]      = { state.matrix.mvp };

# define the output
OUTPUT oPos       = result.position  # output vertex

# temporaries
TEMP temp;

# transform the vertex v by the.mvp matrix
DP4 temp.x,.mvp[0],v;
DP4 temp.y,.mvp[1],v;
DP4 temp.z,.mvp[2],v;
DP4 temp.w,.mvp[3],v;

MUL oPos,temp,9999;

END
```



Per-vertex attributes (inputs)

- Some of the per-vertex inputs to the vertex program include:
 - vertex.position
 - vertex.weight
 - vertex.weight[n]
 - vertex.normal
 - vertex.color
 - vertex.fogcoord
 - vertex.texcoord, unit 0
 - vertex.texcoord[n], unit n
 - vertex.attrib[n]

Vertex outputs

- The interpolant outputs of the vertex program (and the inputs to the fragment program) include:
 - `result.position`
 - `result.color`
 - `result.fogcoord`
 - `result.texcoord`, unit 0
 - `result.texcoord[n]`, unit n

Vertex program instruction set

ABS	v	v	absolute value
ADD	v,v	v	add
ARL	s	a	address register load
DP3	v,v	ssss	3-component dot product
DP4	v,v	ssss	4-component dot product
DPH	v,v	ssss	homogeneous dot product
DST	v,v	v	distance vector
EX2	s	ssss	exponential base 2
EXP	s	v	exponential base 2 (approximate)
FLR	v	v	floor
FRC	v	v	fraction
LG2	s	ssss	logarithm base 2
LIT	v	v	compute light coefficients
LOG	s	v	logarithm base 2 (approximate)
MAD	v,v,v	v	multiply and add

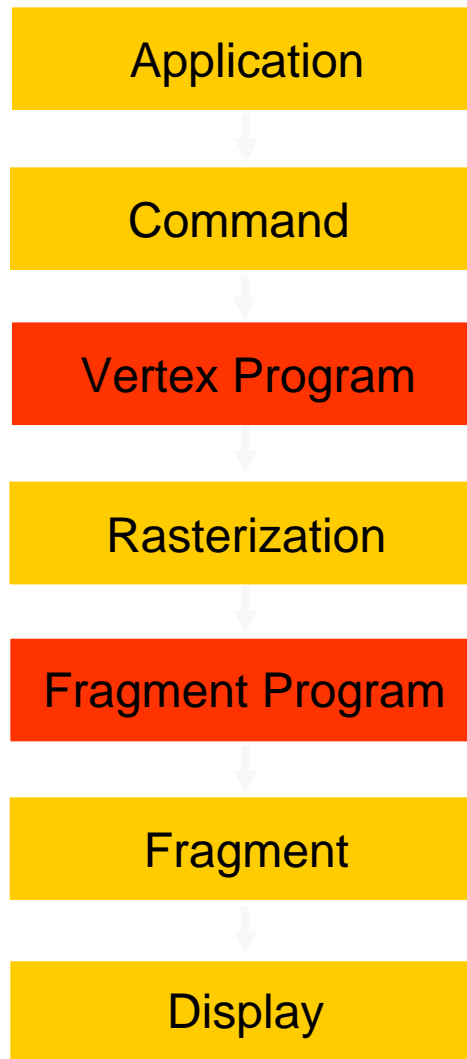
Vertex program instruction set

MAX	v, v	v	maximum
MIN	v, v	v	minimum
MOV	v	v	move
MUL	v, v	v	multiply
POW	s, s	ssss	exponentiate
RCP	s	ssss	reciprocal
RSQ	s	ssss	reciprocal square root
SGE	v, v	v	set on greater than or equal
SLT	v, v	v	set on less than
SUB	v, v	v	subtract
SWZ	v	v	extended swizzle
XPD	v, v	v	cross product

Vertex program binding and loading



Programmable graphics pipeline



Fragment program

- Often known as “pixel shaders”
- Extension `GL_ARB_fragment_program`

Simple fragment program

```
!!ARBfp1.0
```

```
MOV result.color, {1, 0, 0, 0};
```

```
END
```



Per-fragment inputs

- Also known as interpolants
- Same list as before (output of vertex program)

Fragment program outputs

- Only two outputs:
 - `result.color`
 - `result.depth`

Fragment program instruction set

ABS	v	v	absolute value
ADD	v,v	v	add
CMP	v,v,v	v	compare
COS	s	ssss	cosine with reduction to $[-\pi,\pi]$
DP3	v,v	ssss	3-component dot product
DP4	v,v	ssss	4-component dot product
DPH	v,v	ssss	homogeneous dot product
DST	v,v	v	distance vector
EX2	s	ssss	exponential base 2
FLR	v	v	floor
FRC	v	v	fraction
KIL	v	v	kill fragment
LG2	s	ssss	logarithm base 2
LIT	v	v	compute light coefficients
LRP	v,v,v	v	linear interpolation
MAD	v,v,v	v	multiply and add

Fragment program instruction set

MAX	v,v	v	maximum
MIN	v,v	v	minimum
MOV	v	v	move
MUL	v,v	v	multiply
POW	s,s	ssss	exponentiate
RCP	s	ssss	reciprocal
RSQ	s	ssss	reciprocal square root
SCS	s	ss--	sine/cosine without reduction
SGE	v,v	v	set on greater than or equal
SIN	s	ssss	sine with reduction to $[-\pi, \pi]$
SLT	v,v	v	set on less than
SUB	v,v	v	subtract
SWZ	v	v	extended swizzle
TEX	v,u,t	v	texture sample
TXB	v,u,t	v	texture sample with bias
TXP	v,u,t	v	texture sample with projection
XPD	v,v	v	cross product

Reading

- Angel thru Ch 8



Reading

- Angel thru Ch 8

