PlayStation 2 Clipping

Colin Hughes
Sony Computer Entertainment Europe
Standard graphics pipeline

• Vertices passed to T&L unit
• Transformed to clip space
• Hardware clips to viewport

• Programmer has no hassles.
  – Not the case with PS2
Why no clipping

- Clipping in hardware is very expensive.
  - At least one FP divider and vector multiplier required
  - Up to 3 for full pipelined hardware
    - relying on symmetry to clip +&- together
  - This hardware lies idle for a great deal of the time
    - PS2 designed to maximize use of hardware
      - Have software handle infrequent cases
PS2 Hardware

make better games
PS2 Overview

• MI Ps CPU core
  – 128 bit register set, multimedia extensions

• Vector Units
  – SI MD FP processors, operate on 4xSP
  – Individual data & code memory
    • Can run independently of core cpu
  – VU0 closely tied to core
  – VU1 closely tied to GS
PS2 Overview

• Graphics Synthesisor
  - Rasteriser with 4MB embedded memory
  - Optimised for high fill rate
  - Very simple.
  - No processing of vertex data
    • Only 2D scissoring supported

• This means work for the programmer!!
Internal datapaths

EE Core

VU0

VU1

GIF

I$

D$

SP

VIF0

VIF1

Timers

DMA controller

Rambus interface

IPU

SIF
For a simple renderer all we are interested in are the following components:

- **Rambus interface**
- **VIF1**
- **VU1**
- **GIF**
PS2 simple graphics pipeline

- Geometry sent to VU1 via DMA, along with all state information.
- VU1 applies all transform, lighting, and clipping.
  - Back to old days..
- Polys sent directly from VU1 to GS
  - Special internal link from VU1 memory to G1 F.
Simple transform

- Without worrying about clipping, transform is vector matrix operation followed by homogeneous divide.

\[
\begin{align*}
\text{Mul } ACC,vf31,vf1w \\
\text{Madd } ACC,vf30,vf1z \\
\text{Madd } ACC,vf29,vf1y \\
\text{Madd } vf2,vf28,vf1x \\
\text{div } Q,vf0w,vf2w \quad (vf0w = 1.0) \\
\text{Mul } vf2,vf2,Q
\end{align*}
\]
How does this compare?

• 6 instructions
  - 5 multiples and one divide
• Compare to 4 for Vertex shader
  - no divide as DX transforms to clip space

• VU ops contain upper and lower instruction!!!
  - Mul is upper
  - Div ( and integer / loadstore ) are lower
Ground rules

• Only integer arithmetic and accumulator have 1/1 throughput/latency

• FP ops have 1/4
• Load/Store is 1/4
• Divide is 7/7 - This is the killer
  – Scalar divide unit only
Real VU code

NOP
NOP
NOP
MULAw ACC,vf31,vf0
MADDAz ACC,vf30,vf1
MADDAy ACC,vf29,vf1
MADDx vf2,vf28,vf1
NOP x 3
NOP
NOP x 6
MULQ vf2,vf2,Q
NOP x 3
FTOI4 vf2,vf2
NOP x 3
NOP
LQI vf1,(vi1++)
NOP
NOP
NOP
NOP
NOP
NOP x 3
DIV Q,vf0w,vf1w
NOP x 6
NOP
NOP x 3
NOP
NOP x 3
SQI vf2,(vi2++)
Real VU code

NOP
NOP
NOP
MULAw ACC,vf31,vf0
MADDAz ACC,vf30,vf1
MADDAy ACC,vf29,vf1
MADDx vf2,vf28,vf1
NOP x 3
NOP
NOP x 6
MULQ vf2,vf2,Q
NOP x 3
FTOI4 vf2,vf2
NOP x 3
NOP

LQI vf1,(vi1++)
NOP
NOP
NOP
NOP
NOP
NOP x 3
DIV Q,vf0w,vf1w
NOP x 6
NOP
NOP x 3
NOP
NOP x 3
SQI vf2,(vi2++)

26 Cycles!! Very Bad

make better games
Better code....

• Only 6 upper instructions and 3 lower instructions are doing anything
• Theoretical performance is determined by the DIV throughput

• Even with texture correction, 7 cycles can be achieved.
  – ST needs to be projected to STQ for GS
  – 7 upper, 5 lower
Optimised transform

ftoi4.xyz vf12,vf8
mulq.xyz vf14,vf13,Q
mulq.xyz vf9,vf9,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf8
madday.xyzw ACC,vf29,vf8
maddx.xyzw vf8,vf28,vf8
ftoi4.xyz vf12,vf9
mulq.xyz vf14,vf13,Q
mulq.xyz vf10,vf10,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf9
madday.xyzw ACC,vf29,vf9
maddx.xyzw vf9,vf28,vf9
ftoi4.xyz vf12,vf10
mulq.xyz vf14,vf13,Q
mulq.xyz vf11,vf11,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf10
madday.xyzw ACC,vf29,vf10
maddx.xyzw vf10,vf28,vf10
ftoi4.xyz vf12,vf11
mulq.xyz vf14,vf13,Q
mulq.xyz vf8,vf8,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf11
madday.xyzw ACC,vf29,vf11
maddx.xyzw vf11,vf28,vf11
lq.xyz vf8,2+12(vi1)
lq.xy vf13,0+6(vi1)
iaddi vi1,vi1,3
div Q,vf0w,vf11w
sq.xyzw vf12,2-3(vi1)
ibeq vi1,vi2,prelit_exit
sq.xyz vf14,0(vi1)
lq.xyz vf9,2+12(vi1)
lq.xy vf13,0+6(vi1)
iaddi vi1,vi1,3
div Q,vf0w,vf8w
sq.xyzw vf12,2-3(vi1)
ibeq vi1,vi2,prelit_exit
sq.xyz vf14,0(vi1)
lq.xyz vf10,2+12(vi1)
lq.xy vf13,0+6(vi1)
iaddi vi1,vi1,3
div Q,vf0w,vf9w
sq.xyzw vf12,2-3(vi1)
ibeq vi1,vi2,prelit_exit
sq.xyz vf14,0(vi1)
lq.xyz vf11,2+12(vi1)
lq.xy vf13,0+6(vi1)
iaddi vi1,vi1,3
div Q,vf0w,vf10w
sq.xyzw vf12,2-3(vi1)
ibne vi1,vi2,prelit_loop
sq.xyz vf14,0(vi1)
Optimized transform

ftoi4.xyz vf12,vf8
mulq.xyz vf14,vf13,Q
mulq.xyz vf9,vf9,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf8
madday.xyzw ACC,vf29,vf8
maddx.xyzw vf8,vf28,vf8
ftoi4.xyz vf12,vf9
mulq.xyz vf14,vf13,Q
mulq.xyz vf9,vf9,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf9
madday.xyzw ACC,vf29,vf9
maddx.xyzw vf9,vf28,vf9
ftoi4.xyz vf12,vf10
mulq.xyz vf14,vf13,Q
mulq.xyz vf11,vf11,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf10
madday.xyzw ACC,vf29,vf10
maddx.xyzw vf10,vf28,vf10
ftoi4.xyz vf12,vf11
mulq.xyz vf14,vf13,Q
mulq.xyz vf8,vf8,Q
mulaw.xyzw ACC,vf31,vf0
maddaz.xyzw ACC,vf30,vf11
madday.xyzw ACC,vf29,vf11
maddx.xyzw vf11,vf28,vf11
lq.xyz vf8,2+12(vi1)
lq.xy vf13,0+6(vi1)
 iaddi vi1,vi1,3
div Q,vf0w,vf11w
sq.xzw vf12,2-3(vi1)
 ibeq vi1,vi2,prelit_exit
 sq.xyz vf14,0(vi1)
lq.xyz vf9,2+12(vi1)
lq.xy vf13,0+6(vi1)
 iaddi vi1,vi1,3
div Q,vf0w,vf11w
sq.xzw vf12,2-3(vi1)
 ibeq vi1,vi2,prelit_exit
 sq.xyz vf14,0(vi1)
lq.xyz vf10,2+12(vi1)
lq.xy vf13,0+6(vi1)
 iaddi vi1,vi1,3
div Q,vf0w,vf9w
sq.xzw vf12,2-3(vi1)
 ibeq vi1,vi2,prelit_exit
 sq.xyz vf14,0(vi1)
lq.xyz vf11,2+12(vi1)
lq.xy vf13,0+6(vi1)
 iaddi vi1,vi1,3
div Q,vf0w,vf10w
sq.xzw vf12,2-3(vi1)
 ibeq vi1,vi2,prelit_loop
 sq.xyz vf14,0(vi1)
Clipping (Simple FOV)

• View frustum normally set up as
  \[-1 < \frac{X}{Z} < +1\]
  \[-1 < \frac{Y}{Z} < +1\]
  near < Z < far

• another form (as long as Z is valid)
  \[-Z < X < +Z\]
  \[-Z < Y < +Z\]
  near < Z < far
VU support

- VU contains special CLI P instruction
  - CLI P vfa.xyz,vfb.w
    - $-|w| < x < +|w|$
    - Same for $y$ and $z$
      - Easy in HW: Implement single $|xyz| > |w|$ with sign check
      - $|w|$ also allows cc to be correct for negative $w$
  - Places 6 result bits in shift register
    - Holds up to 4 check results (24 bits)
  - Logical condition instructions can show result for triangle or quad
Setting up clipping

• $-|W| < X$ or $Y < +|W|$ is simple
• near $< W <$ far needs thinking about

$n < w < f$

$\frac{1}{f} < \frac{1}{w} < \frac{1}{n}$

$0 < \frac{1}{w} - \frac{1}{f} < \frac{1}{n} - \frac{1}{f}$

$0 < (f-w)/wf < (f-n)/nf$
Setting up clipping

\[
0 < \frac{(f-w)}{wf} \frac{nf}{(f-n)} < 1 \\
0 < \frac{((f-w)n)}{( w*(f-n))} < 1 \\
0 < \frac{(2n*(f-w))}{( w*(f-n))} < 2 \\
-1 < \frac{(2n*(f-w))}{( w*(f-n))} - \frac{(w*(f-n))}{( w*(f-n))} <+1 \\
-1 < \frac{(2fn - 2wn -wf + wn )}{(w*(f-n))} < +1 \\
-1 < \frac{(2fn - w(n+f ))}{(w*(f-n))} < +1 \\
-|w| < \frac{( 2fn -w(n+f ) )}{( f-n )} < +|w| 
\]
Homogeneous space

• X and Y checks have valid regions behind the camera (negative eyespace Z)

• However the near / far plane check only passes the region with positive w

• This means that the erroneous X/ Y regions are not valid Z regions..
  - Z clipping must occur first!!!
Implementing clipping

• PS2 renders strips
• Each new vertex is a new triangle
• If any vertex fails the cliptest the triangle needs clipping

• Note: Clipspace isn’t screen space
  – Scale and bias need to be added
VU clipspace renderer

mulaw ACC,vf31,vf0  
maddaz ACC,vf30,vf1  
madday ACC,vf29,vf1  
maddax vf3,vf28,vf1  
clipw.xyz vf3,vf3  
mula ACC,vf3,vf27  
maddw.xy vf4,vf26,vf3  
nop  
nop  
mulq.xyz vf4,vf4,Q  
mulq.xyz vf2,vf2,Q  
ftoi4.xyz vf4,vf4  
nop  
nop  

Lqi vf1,(vi2++)  
lqi vf2,(vi2++)  
nop  
nop  
nop  
div Q,vf0w,vf3w  
fand vi1,0777777  
fcget vi5  
iaddiu vi1,vi1,0x7fff  
mfir vf4.w,vi1  
mfir vf2.w,vi5  
nop  
nop  
sqi vf2,(vi3++)  
sqi vf4,(vi3++)
Clipspace renderer

- This code is 10 upper and 10 lower instructions
- All triangles are transformed, and any that need clipping (via the FCAND) are marked as non draw for the GS
- The calculated clipcodes are stored for later use in an unused field
Guard band clipping

• If our triangles are small, we don’t really need to clip them.
• Just discarding the triangles outside the clip region is enough.
• The GS 2D scissoring can clip efficiently for small triangles.
• All we need to do is to increase the size of the clip pyramid to allow a ‘guard band’
Full clipping

- Full clipping is very slow
  - Sutherland Hodgman algorithm
  - Check against each plane
- We reuse the clipcodes generated in the previous code
- The OR of the vertex clip codes determines the planes to check against
  - Remember to clip against Z first!!
Per plane checks

- Line segment clip test.
- 4 cases for output
  - Both visible: Endpoint
  - None visible: No output
  - Leaving: Intersection
  - Entering: Intersection and endpoint

  - Intersection needs new clipcode generation

- Intersection calculation requires multiply and divide operations
Intersection for w

- Vf1 is p1, vf2 is p2, vf3 is w plane

- sub vf4,vf2,vf1 \quad \text{p2-p1}
- sub vf5,vf3,vf1 \quad \text{w-p1}
- div Q,vf0w,vf4w \quad 1/(p2w-p1w)
- mulw ACC,vf1,vf4w \quad p1.(p2w-p1w)
- maddw vf6,vf4,vf5w \quad (w-p1w)(p2-p1)
- waitq
- mulq vf6,vf6,Q \quad \text{Final result}
Reuse of code

• Difficult to reuse code due to scalar component used in plane intersection

• One solution:
  – Rotate vector between checks
  – xzyw to yzwx
  – Clipping code always checks against W

• Or just three hardwired routines!!
Interpolation

• Only ST and XYZ are projected properly
• RBGA interpolation needs to be adjusted after projection to match GS

• This is only a problem for big triangles
  - Not GS friendly anyway
Clipper output

• The output from the clipper for one triangle is a \( n \) sided polygon

• This can be rendered as a fan or a strip
  – Fan: No need to reorder vertices
  – Strip: Can be stitched in with unclipped triangle strip to ensure no state change problems
Optimizations

• Don’t clip degenerate triangles
  – They won’t draw anyway, so cull them

• At the expense of much more unreadable code, the clipping operation can be folded into the strip rendering:
  • See the code for details

• For small triangles not all 6 clipplanes will intersect: Only 3 at most
Optimizations

• Not all objects have to go through the clipper
• If the bounding volume for an object or a section of mesh is not clipped none of the triangles inside will be clipped
  – Use switchable VU code with two paths
Tearing

- Numerical inaccuracies can occur at the boundary of valid and clipped triangles
  - Sometimes this is caused by a different transform path in the renderer
    - Direct object to screen matrix for unclipped
    - Object to clip space / clip space to screen otherwise

- Only solution is to unify the maths!!
Thank you

- Code and a more detailed document will be available at:
  - www.playstation2-linux.com
  - Also at GameDeveloper site

- You can email me at:
  - colin@users.playstation2-linux.com