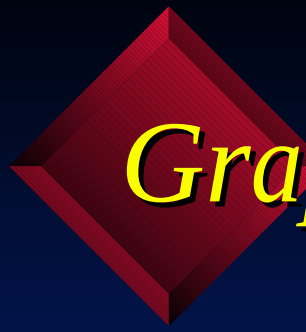
Software Engineer

SCEA

TM



Graphics Processing Unit

Hardware Overview

Maximize pixel flow (60 FPS)

Discussion of API

Hardware Overview

- Command port
  - 64 byte Command FIFO (First command In is First Out)
    - 16 words
    - command sizes in libgpu.h
    - stores a max of about 3 commands, but normally 1 command

Hardware Overview

- 3 data ports
  - 1 DMA channel shares 2 DMA types
    - Slice
    - Source Chain
  - 1 special DMA channel
    - Ordering table clear
  - 1 32 bit data port
    - CPU copy

Hardware Overview

2k Texture Cache

- 64x64 4bit
- 64x32 8bit
- 32x32 16bit
- no alignment necessary
- if it is not aligned, it rolls it over to other side of cache

Hardware Overview

2k Texture Cache

- can't turn it off
- loads cache misses every 8 bytes

Reads are costly!

Hardware Overview

Slice mode DMA

1) Multiple blocks of data
2) Size of blocks is constant
3) Data in contiguous addresses
4) Memory to GPU (LoadImage) or GPU to Memory (StoreImage)
5) Shares channel with Source Chain

Hardware Overview

Slice mode DMA

6) Gives up control of bus between chunks

7) Function is LoadImage

8) 64 byte blocks

9) Rest is i/o copied

make sure LoadImage sizes are multiple of 64 bytes

Hardware Overview

Source Chain DMA

1) Multiple blocks of data
2) Blocks have different sizes
3) Blocks are allocated at different addresses
4) Memory to GPU memory ONLY
5) Shares channel with Slice

PlayStation™

Hardware Overview

Source Chain DMA

6) Automatically loads the order table into command FIFO

7) Data Req controls loading of data

Note: Slice DMA CAN NOT happen when Source Chain is going.

LoadImage can not be added to DrawOTag

Hardware Overview

- Source Chain DMA
  - Order table size: 1 word by XXX array
    - Top 8 bits of an element contains the number of words to DMA (DC)
    - Bottom 24 bits contains pointer to next element in linked list (NPTR)
  - DC
    - For primitives, DC is in libgpu.h
    - For array elements, DC is 0.

Hardware Overview

Source Chain DMA

- If DC is 0, DMAC will set up next address and finish DMA without transferring anything.
- Consolidating size of order table saves processing.
- ClearOTag walks array list to set DC to 0.
- ClearOTagR is in hardware.

Hardware Overview

- Source Chain DMA
  - Bottom 24 bits of order table element is pointer of the next element (NPTR).
  - *addPrim* ALWAYS adds to the front of the linked list
    - *addPrim* is inline version of *AddPrim*
  - DC and NPTR combined is called the CHAIN DESCRIPTOR.

Hardware Overview

Ordering Table Clear DMA

1. No data transfer
2. Initializes array elements of order table for source chain dma
3. Puts 0 in top nibble
4. Points other 28 bits to the next array element

Hardware Overview

Data Request

1. DREQ occurs when FIFO is empty
2. DREQ asks DMAC for next primitive
3. DREQ used with source chain
4. DMA occurs in background, can kill program if bus error

Hardware Overview

Interrupt

1. Occurs when Load / Store / ClearImage / DrawOTag is finished
2. Interrupt can occur in other parts of code

   When debugging, be aware.

Hardware Overview

Special Hardware for Sprites and Flat Polys

1. Allows performance of writing 2 pixels/cycle
2. Sprites have no scaling or rotation so cheats are used.
3. Flat poly has no texture so hardware cheats are used.

PlayStation™

Hardware Overview

- Life of the GPU?
  - 2 States of GPU
    - DrawOTag state
      - obey Command FIFO
    - Normal command state
      - obey command register

Hardware Overview

DrawOTag state

0. GPU is set to the source chain DMA
1. GPU draws what is in 64 byte FIFO.
2. When FIFO is empty and GPU may still be drawing, generate DREQ.
3. DMA Controller starts DMA and refills FIFO.

Hardware Overview

DrawOTag state

4) When Drawing is done, and FIFO is filled, goto 1.

5) DMA Controller automatically turns off source chain DMA

6) Call DrawSync(0) which waits for GPU idle, instead of FIFO empty.

Hardware Overview

- GPU Drawing
  - Writing GT4abr
    - Copy POLY_GT4 to FIFO with DrawOTag or DrawPrim
    - GPU gets command from FIFO
    - GPU splits square, top right to bottom left
    - GPU starts with first pixel, top,left,top,down

Hardware Overview

- GPU Drawing
  - Writing GT4abr
    - Check texture cache for hit
    - If miss, bring 8 byte chunk to cache
    - Perform gouraud shading for each split triangle
    - Do a RMW right for abr

Hardware Overview

- Shading Bugs
  - Adjusting unshared vertices brightness causes uneven shading
    - Avoid adjusting unshared vertices
  - Textures use 5 bit multiplier for shading
    - Color is reduced to 5 bit for multiplier
    - Textures shading is banded

Hardware Overview

- Writing 8 bytes
  - Description of 1 RW cycle of 8 bytes
    1. 4 reads of 2 bytes from texture cache (4 cycles)
    2. Page Break (3 cycles)
    3. Write to VRAM (variable number of writes)
    4. If end of cacheline, page break (3 cycles) else goto 1

Hardware Overview

GPU Reading

- 8 bytes was chosen because less than 8 bytes causes more PBs
- More than 8 bytes means copying 1 byte has a large overhead

Gratuitous Demo

60 FPS Ridge Racer

PlayStation™

Maximize Pixel Flow

- Maximizing Pixel Flow
  - Know what your limits are
    - Chose appropriate types of polygons
  - Run CPU in dcache and icache
    - Increase the DMA bandwidth
  - Decrease the number of written pixels while making it look good
    - Work creatively within those limits
    - MIP Mapping, art layout

Maximize Pixel Flow

- Ideal Raw Numbers
  - Pixel Writing
    - 33.8688 MegaPixels / Second
      - 67.7376 MegaPixels / Second Sprite & Flat Polys
  - Data Loading
    - 33MHz x 4bytes = 132 MB / s

Maximize Pixel Flow

Theoretical Drawing Speed

- F3, F4, Sprites
  - = 2 pixels / cycle
  - = 66 Megapixels/second
- G3,GT3 on cache
  - = 1 pixel / cycle
  - = 33 Megapixels/second
- G3abr/GT3abr
  - = 0.5 pixel / cycle
  - = 16 Megapixels

Maximize Pixel Flow

LoadImage 33MHz*4bytes=132MB/sec.

- Loads from 16->132 MB / s
- Speed decreases when:
  1) CPU accesses bus (LoadImage is STOPPED;CPU controls bus)
     a) CPU and GPU share the main bus
     b) Interleaved DMA cycles
     c) Up to 1/2 as slow
     d) Solution is avoid CPU bus hits
        use only data cache and icache when loading

Maximize Pixel Flow

Speed decreases when:

2) Run from a development station

a) LoadImage generates interrupt when finished

b) Debugger intercepts interrupt, communicates with PC.

Faster PC means faster speed.

Maximize Pixel Flow

Speed Increases when:

1) Increase sizes of LoadImages

   Decreases the number of interrupts

2) Use 64 bytes alignment

   I/O copy is avoided

Discussion of API

- API Suggestions
  - Please call SetGraphDebug(x)
    - x=1 Check parameters
    - x=2 Dumps primitives et al
  - Please install Mess1.com in autoexec.bat
    - Look for messages in Msg window

Discussion of API

```
u_long aClearOTag( u_long *otp, int n)
    {
    while (--n)
        {
        *otp = (u_long)(otp) + 4;  / / Point to next array element
        *otp = (u_long)*otp&0xFFFFFF; / / Clear Top Nybble
        otp++;  / / Increment order table pointer
        }
        otp[0]=0x00FFFFFF; / / Industrial Strength DMA stopper
     }
```

Discussion of API

- API
  - What does the ResetGraph printf mean?
    - ResetGraph: jtb=80016560, env=800165a8
      - jtb is the jump table of all gpu hardware commands
      - env is the global data structure
    - Can't remove it because SCEI needs it for compatibility testing

Discussion of API

What does the stat printf mean?

- GPU timeouts
  - a) Something stomping over a primitive
  - b) Forgetting to double buffer the primitive.

Discussion of API

API

- Max Texture Page is 256x256
- Texture mapping is linear.
- Drawing is done on triangles, though quads are supported

Discussion of API

- API
  - 1 64 function command queue
    - Immediate functions: PutDispEnv, ClearOTag,ClearOTagR,DrawPrim, ResetGraph
  - Order Table
    - Linked list of commands copied over by DMA
    - Order table is added to Software queue

Discussion of API

- API
  - DrawSync(0)
    - waits for an empty software queue
    - then waits for an empty command FIFO
    - then waits for a GPU idle
  - DrawSync(1)
    - asks for number of commands in the queue
    - returns the number

Discussion of API

ResetGraph(1)

- a command reset
- interrupt reset
- software queue reset

Discussion of API

- Half pixel problem
  - GPU was designed for 3D
    - did not test 2D
    - Center of vertex is put in the middle of a texel
- Solutions:
  - Scale up the sprite
  - Make art 1 pixel bigger

Discussion of API

Linear Texture Mapping

- Linear Texture Mapping means textures are skewed when the polygon skews shape.

Solutions:

- Subdivide polygons for speed up to 4x4
- Perspective texture mapping is possible on a constant Z by software only
  - Faster than DivideFT4, 8x8 subdivisions by 1.0x-1.5x

Fortuitous Demo

Perspective Texture Mapping on Constant Z

PlayStation™

Discussion of API

- Linear Sampling
  - When scaling, the sampling is applied linearly.
  - GPU always reads in 8 bytes chunks.
  - A 16x reduction in a 64 byte wide texture results in a lot of reads.

Discussion of API

MargePrim concatenates primitives to fit in 16 word command FIFO.

Best / worst example:

DR_MODE+SPRT+DR_MODE

Fills FIFO with 16 words

But DR_MODE is slow.

Discussion of API

- Drawing mode (DR_?) primitives are slow.
  - Avoid them.
  - To change texture page for sprites, use POLY_FT3

Discussion of API

- LoadImage
  - Library 3.5 adds LoadImage primitive
  - You CAN add LoadImage to order table
  - Uses source chain dma instead of slice dma
  - Block size is a little less than 64 bytes

Questions