Advances in Real-Time Rendering in Games

SIGGRAPH 2011
Rendering in Cars 2

Chris Hall
Christopher.Hall@disney.com

Rob Hall
Robert.W.Hall@disney.com

Dave Edwards
David.W.Edwards@disney.com

Advances in Real-Time Rendering in Games
Cars 2 Motivation

Different gameplay demands different technology
## Differences

<table>
<thead>
<tr>
<th>Toy Story 3</th>
<th>Cars 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platforming</td>
<td>Racing</td>
</tr>
<tr>
<td>2 player split screen</td>
<td>4 player split screen</td>
</tr>
<tr>
<td>Average 30fps</td>
<td>Essential to maintain 30fps</td>
</tr>
</tbody>
</table>

© Disney/Pixar

Advances in Real-Time Rendering in Games
<table>
<thead>
<tr>
<th>Toy Story 3</th>
<th>Cars 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>All dynamic lighting</td>
<td>Lightmaps, Light probes, limited dynamic</td>
</tr>
<tr>
<td>Dynamic cascaded shadow mapping</td>
<td>Simplified shadowmaps for dynamic only</td>
</tr>
<tr>
<td>SSAO, Depth of Field, Glow, God Rays, Sparkle, Bloom, Deferred Ambient</td>
<td>HDR, Bloom, Motion Blur, Color Correction</td>
</tr>
</tbody>
</table>

© Disney/Pixar
Advances in Real-Time Rendering in Games
Outline

- Light Probes
- HDR color precision
- Early stencil shadow culling
- PS3 Post Processing
Light Probes

Chris Hall

Advances in Real-Time Rendering in Games
Motivation

- 4 Players
- Lightmaps for all world geometry
- Real-time lighting didn’t match
Light probes

- Capture light from a point in space
- Bounce lighting
- Environment Mapping
Global probe

- Used for bounce lighting for outdoors
- Either artist defined light rig or captured probe

- Environment Map
- Irradiance Map (SH)
Bounce lighting data

- Store as spherical harmonics
- Order 3 SH = 108 bytes per probe
- Can pack in direct lighting for free
Probe Capture

- Render cubemaps on GPU
- Save as 16F for HDR
- Atlas for speed
- Bounced lighting
  - Cubemap to SH projection [Sloan], DirectX SDK
Irradiance Volume

- Volume with a bunch of light probes
- Allows for varied bounced light throughout the world
- Very popular to use with lightmaps
  - [Greger 1998]
  - [Tatarchuk 2005]
  - [Mitchell, McTaggart and Green 2006]
Volume choice

- Racing game
- 2-5 miles of track per world
- Mostly outside
- Lots of thin, curvy areas
- Coverage isn’t essential
Uniform grid volume

- Box Volume
- Can be rotated and scaled to fit anywhere
- Box split with variable amount of slices (density x/y/z)
- Probes placed along the slices of the volume
Grid Analysis

- Simple structure and easy to implement
- Entire data is saved into a continuous array
- Sample with box intersection tests
  - Can access each probe by an offset
  - $O(1)$ to sample inside the grid
  - Cost is only spent inside volumes
- Wastes space
Fading Regions

- Blend between global probe and volume lighting
- Outer Fading Volume
- Inner Fading Volume
- Fading amount
Invalid points

- Probes outside world have incorrect lighting
- How to detect
- Replace with correct lighting
- [Kontkanen and Laine]
Volume Lookup

- We need some way to light our objects with them
- CPU Based
  - Assign/blend closest SH per mesh
  - Pass SH data through to GPU
- GPU Based
  - Per pixel or per vertex
  - Sample probes on the GPU
CPU Assignment

- For each mesh, sample the lighting at the mesh center
- Intersection tests
  - Create an OBB for each volume
  - Check if center point of mesh is inside
- Trilinear Interpolation
Shader constants

- Per instance shader constants
- Calculate color in shader
- Breaks down for large objects
  - Can break mesh apart with vertex color blending
- Same problem for world lighting
Overlapping Volumes

- Blending is challenging
Time Averaging

- Blend % of the last frame’s SH into the current SH
  - Artist adjustable per world
- Trilinear filtering substitute
- Avoid for first frame
- [Mitchell, McTaggart and Green]
Environment Maps

- Probe without a volume
- Worked really well for road reflections
- Assign environment map probes to volumes

© Disney/Pixar
Assigning Environment Maps

- Use volume’s environment map if inside a volume
- Otherwise, use the global probe
- Switch based on the fading region
- Switch was a pop
- Overlapping volumes avoid pops by sharing cubemaps

© Disney/Pixar
Direct Lighting

- Pack direct lighting into probes
  - Can evaluate lighting in SH and add to bounce
- No extra performance cost
- Dependent on grid density
Lighting Overrides

- Directional and ambient lights added if inside volume
- Allows artists to control lighting
- 2d volume
- 1d volume
- Area lights
Artist tricks

- 2d volume
Artist Tricks

- Single point volume
Artist Tricks

- Area lights
Ending Thoughts

- Uniform grid is easy and fast
- Used little memory and scaled well for 4 players
- Lots of flexibility with artists
- Future ideas
  - GPU lookup of light probes

© Disney/Pixar

Advances in Real-Time Rendering in Games
Trimming the GPU Pipeline

Rob Hall

Advances in Real-Time Rendering in Games
Overview

- Reduce cost for HDR rendering
- Reduce shadow cost
  - Scale shadow maps for 4 player split screen
  - Use multi-resolution rendering for deferred shadow mask

© Disney/Pixar
HDR requirements

- If possible, use a 32 bits per pixel target format
- Support all hardware alpha blend states
- Limited range from [0,32] is acceptable
- Reduce banding as much as possible
## Format Chart

<table>
<thead>
<tr>
<th>Format</th>
<th>Range</th>
<th>Alpha Blend</th>
<th>Bilinear Filtering</th>
<th>Perf cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>sRGB</td>
<td>[0,1]</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>LogLuv [Larson]</td>
<td>[10^{19}, 10^{19}]</td>
<td>Limited</td>
<td>No</td>
<td>ALU</td>
</tr>
<tr>
<td>RGBM [Karis]</td>
<td>[0,6]</td>
<td>Limited</td>
<td>No, but works OK</td>
<td>ALU</td>
</tr>
<tr>
<td>7e3</td>
<td>[0, 31 7/8]</td>
<td>Yes</td>
<td>Yes</td>
<td>Alpha states</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>double fill rate</td>
</tr>
<tr>
<td>R11G11B10</td>
<td>[0, 2^{16}]</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>16F</td>
<td>[-2^{16}, 2^{16}]</td>
<td>Yes*</td>
<td>Yes*</td>
<td>Double fill rate</td>
</tr>
</tbody>
</table>

* Except Xbox 360
Sample Image 16F linear

© Disney/Pixar

Advances in Real-Time Rendering in Games
And other 16F linear images
<table>
<thead>
<tr>
<th>Format</th>
<th>Range</th>
<th>Alpha Blend</th>
<th>Bilinear Filtering</th>
<th>Perf cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>sRGB</td>
<td>[0,1]</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>LogLuv [Larson]</td>
<td>[10^19, 10^19]</td>
<td>Limited</td>
<td>No</td>
<td>ALU</td>
</tr>
<tr>
<td>RGBM [Karis]</td>
<td>[0,6]</td>
<td>Limited</td>
<td>No, but works OK</td>
<td>ALU</td>
</tr>
<tr>
<td><strong>Xbox 360</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7e3</td>
<td>[0, 31 7/8]</td>
<td>Yes</td>
<td>Yes</td>
<td>Alpha states double fill rate</td>
</tr>
<tr>
<td><strong>PS3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R11G11B10</td>
<td>[0, 2^16]</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>16F</td>
<td>[-2^16, 2^16]</td>
<td>Yes*</td>
<td>Yes*</td>
<td>Double fill rate</td>
</tr>
</tbody>
</table>

* Except Xbox 360
PS3 Luv

- Similar to LogLuv - compresses 64 bpp to 32bpp
- Encode luminance with a sqrt instead of a log to avoid a costly exp2 operation on the SPU
- Store luminance in 16 bit fixed point, 3 int 13 frac format
- Range is $[0, \sim 64]$ \[ \left[ 0, \left( 7 + \frac{8191}{8192} \right)^2 \right] \]
- Code sample in Appendix A
7e3 Banding

© Disney/Pixar
What’s causing this?

Scene Render
- 32 bit float internal

Render Target
- 10 bit float format

Post Process
- Tone Mapping
- 32 bit float internal

Frame Buffer
- 8 bit fixed point

© Disney/Pixar

Advances in Real-Time Rendering in Games
Tone Mapping

Scene Render: 32 bit float internal

Render Target: 10 bit float format

Post Process: Tone Mapping

Frame Buffer: 8 bit fixed point

Precision Loss!
Tone Mapping Components

Exposure

\[
\text{Target\_Color} = \text{Scene\_Color} \times (\text{Expected\_Exposure} / \text{Prev\_Frame\_Avg\_Luminance})
\]

Simplified Operator [Hable] [Reinhard]

\[
\text{Tone\_Mapped\_Color} = \frac{\text{Target\_Color}}{1.0f + \text{Target\_Color}}
\]
Pre-exposed color

Scene Render

32 bit float internal

Exposure

Render Target

10 bit float format

Post Process

Tone Mapping Operator

32 bit float internal

Frame Buffer

8 bit fixed point
No pre-exposed color

© Disney/Pixar
With pre-exposed color
No pre-exposed color
With pre-exposed color
Artifacts

- Overall range is clamped

Clamped Image

Red = Error in Clamped Image
HDR Results – Xbox 360

- Used 7e3 with pre-exposed color
- No tiling needed on a Non-MSAA 720p target
HDR Results – PS3

- Used higher bandwidth 16F format
- Cheaper than LogLuv or RGBM due to GPU being ALU bound

- SPU benefitted from the PS3Luv encoding due to lower bandwidth and ALU costs
Shadows

Reduce shadow rendering time in half
Scaling shadows for multiple viewports

- Four viewports = 4x number of draw calls and geometry
- Cutting resolution only reduces fill rate

+ + + = 4x

© Disney/Pixar
Scaling shadows for multiple viewports

- Four viewports = 4x number of draw calls and geometry
- Cutting resolution only reduces fill rate

Solution: Render less in the shadow maps + + + = 4x
Two types of objects

Static Objects – Things that never change position

Dynamic Objects – Those that can
Light Map

- Precalculates all the shadows for static objects

© Disney/Pixar
Dynamic objects

- Need to also receive shadows from light map
- Only need coarse transitions when going in and out of shadow
Low Resolution Shadow Map

- Use a 256x256 shadow map
- Super cheap ~0.1ms
- Use simple proxy geometry
LightMap Only
With Low Resolution Shadow Map

© Disney/Pixar
Dynamic Shadows

- Only draw the dynamic objects in two cascades
- Reduced shadow distance
- Reprojection artifacts OK since tracks are on a 2d plane

© Disney/Pixar
Lightmap and Low Res Shadow Map
With Dynamic shadows
Reducing deferred shadow mask

- Reduce number of pixels processed

\(\frac{1}{4}\) Size Render Target → Bilateral Upsample

© Disney/Pixar

Advances in Real-Time Rendering in Games
Unavoidable artifacts

- Edge Artifacts
- Lower resolution
- Too visible to ignore

© Disney/Pixar
MLAA and Early Stencil Culling [Jimenez]

© Disney/Pixar

Advances in Real-Time Rendering in Games
Early Stencil Culling

- Culls fragments before hitting the pixel shader
- Supported on PS3, 360, and modern PC graphics cards
- PC is automatic, PS3 and 360 manually controlled
- Latency between writing and testing

Early stencil value

© Disney/Pixar
Deferred shadows with early stencil

- Render shadows at 1/16 resolution [Hutchinson]
- Fill full resolution early stencil with 1/16 shadow mask
- Re-render shadow edges at full resolution using early stencil test
What’s good enough at low resolution?

- Shadow values that are 0 or 1
- Cascade selection
- Most pixels in cross bilateral filter
Render at 1/16 size

1/16 Deferred shadow mask → Dilate edges
Render at Full Resolution

- Point sample 1/16 target
- Turn on early stencil writes
- If it is inside the dilate region, then texkill

© Disney/Pixar

Advances in Real-Time Rendering in Games
Render high res shadow mask [Ownby]

- Turn on early stencil test
- Early stencil culls pixels filled in previous pass
- Only renders ~30% of the pixels
Two pass bilateral blur

- Keep early stencil test on
- Only blurs ~30% of the pixels
Edge Artifacts

Early Stencil Culling  Without
Edge Artifacts

- Early stencil is just a mask
- Dilate does not cover the blur regions
- Only happens at extreme closeups with a wide dilate

© Disney/Pixar
Conclusion

- Pre-exposed color is very effective when rendering to limited precision targets
- Low res shadow map for dynamic objects is cheap
- Deferred shadow mask rendering time effectively cut in half
SPU Post Processing

David Edwards

Advances in Real-Time Rendering in Games
Motivation and Background

- In Toy Story 3 PS3 GPU performance typically lagged behind the Xbox 360, some effects had to be simplified or dropped on PS3 version.
- Nearly half of GPU time in Toy Story 3 on PS3 was related to post processing.
- PS3 has a lot of power that wasn’t being fully utilized.
Cars 2 SPU Post Process Pipeline

- SPU post process executes concurrently with the GPU
- GPU rendering reduced by ~10 ms
Cell Broadband Engine

- 6 SPUs available
- 3.2 GHz
- 128 vector registers
- 256KB Local Store

© Disney/Pixar
**SPU vs. PS3 GPU Pixel Shading**

<table>
<thead>
<tr>
<th></th>
<th>SPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution Speed</strong></td>
<td>4 SPUs * 3.2 GHz = 12.8 GHz</td>
<td>24 pixels * 500 MHz = 12.0 GHz</td>
</tr>
<tr>
<td><strong>Instructions</strong></td>
<td>SIMD, General Purpose</td>
<td>SIMD, graphics centric (special math functions, texturing)</td>
</tr>
<tr>
<td><strong>Data Model</strong></td>
<td>Must prefetch into local store before use, no filtering</td>
<td>Texture cache, efficient threading model to hide latency, filtering</td>
</tr>
</tbody>
</table>

- For most effects SPU near same performance as GPU
- But runs concurrently, so, theoretically, we could nearly double performance of GPU

© Disney/Pixar
GPU Code

FLOAT3 colorGradedColor = tex3D( samColorGradeLut, ( OUT.Color1.rgb * lutScale ) + lutOffset ).rgb;
SPU vs. PS3 GPU Pixel Shading

SPU Code

Advance in Real-Time Rendering in Games

© Disney/Pixar
SPU Post Processing Implementation

GPU
- Main Scene
- Post Processing
- Hud

SPU Offloading

GPU
- Frame 0
- Frame 1
- Frame 2
- Frame 3

SPU
- Frame 0
- Frame 1
- Frame 2

© Disney/Pixar

Advances in Real-Time Rendering in Games
SPU Post Processing Implementation

- SPU post processing adds some overhead
  - Source textures must reside in main memory
    - SPU can’t (realistically) read from VRAM
    - Adds nearly 10 MB to main memory
    - Adds about 1.5 ms of GPU overhead
  - Either SPU or GPU can copy back to VRAM so use whatever is not bottlenecked
SPU Post Processing Effects

- Scene average log luminance
- Tonemapping
- Morphological antialiasing (This saves main scene GPU time too!)
- Motion blur
- Downsamples / Upsamples
- Highpass
- Gaussian Blurs
- Color correction
- Stereo 3D
- Screen space ambient occlusion (not used in Cars 2)
SPU Post Process Performance

12.25 ms Total
SPU Post Process Performance

1.15 ms – three picture in pictures

© Disney/Pixar

Advances in Real-Time Rendering in Games

SIGGRAPH 2011
SPU Post Process Performance

1.4 ms – Average Log Luminance/Tonemaps
SPU Post Process Performance

4.5 ms – Morphological Anti-aliasing
3.2 ms – Downsampling/Motion Blur/Composite
SPU Post Process Performance

2.0 ms – HDR Bloom/Composite

© Disney/Pixar

Advances in Real-Time Rendering in Games
Other SPU Jobs mixed in with Post Processing
Stereo 3D

Advances in Real-Time Rendering in Games
Cars 2 Stereo 3D

- Implemented as part of SPU post processing pipeline
- And hence was free!!
- No reduction in performance, graphics content, effects, and/or resolution

© Disney/Pixar
Eye Separation \( \approx 6.4 \) cm

Screen Width

(Calculated from TV size)
Stereo 3D

Zero Parallax Plane

Eye Separation ~ 6.4 cm

Screen Width
(Calculated from TV size)

Screen Distance

Advances in Real-Time Rendering in Games

© Disney/Pixar
Zero Parallax Plane

Screen Distance

Screen Width
(Calculated from TV size)

Eye Separation ~ 6.4 cm
Stereo 3D

Zero Parallax Plane

Eye Separation ~ 6.4 cm

Screen Width
(Calculated from TV size)

Screen Distance

Advances in Real-Time Rendering in Games

© Disney/Pixar
Traditional Stereo 3D

- Render both stereo pairs fully including post processing effect.
- Performance Cost is 2x
- Can be optimized
  - Scene cull once, share results with both eyes
  - Reuse shadow map
  - Use lower resolution target, PS3 has hardware upscaling modes for 3D
  - Reduce content or effects
Traditional Stereo 3D

- Traditional Stereo 3D in 4 player split screen game
  - Very high geometry cost (8x)
  - Reduced resolution in 4 player split screen would not be acceptable
Stereo 3D as Post Processing

Depth + Center

Left

Right

© Disney/Pixar

Advances in Real-Time Rendering in Games
Stereo 3D as Post Processing

Occlusions and Disocclusions

© Disney/Pixar
Stereo 3D as Post Processing

Occlusions and Disocclusions
Stereo 3D as Post Processing

Occlusions and Disocclusions

Now Occluded

© Disney/Pixar

Advances in Real-Time Rendering in Games
Stereo 3D as Post Processing

Occlusions and Disocclusions

No Longer Occluded
Stereo 3D as Post Processing

Occlusions and Disocclusions

© Disney/Pixar
Stereo 3D as Post Processing

- Other Issues
  - View dependent lighting, and reflections
  - Translucent objects (no depth value written)
Stereo 3D as SPU Post Effect

- SPU has some advantages over GPU for stereo 3D

<table>
<thead>
<tr>
<th></th>
<th>GPU</th>
<th>SPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gathered Reads</td>
<td>Really Good</td>
<td>Good</td>
</tr>
<tr>
<td>Scattered Writes</td>
<td>Complex/Inefficient</td>
<td>Good</td>
</tr>
</tbody>
</table>

- Scattered Writes is inverse of Gathered Reads
  - Gathered Reads: What texel should I sample for the current pixel?
  - Scattered Writes: What pixel should I write to for the current texel?
Stereo 3D as SPU Post Effect

- Given a depth value, we can reproject the location for each left and right eye, allowing us to write the color value to the new location
- This is Scattered Writes, which can be done efficiently on the SPU
- However, the GPU cannot do this efficiently
  - A Gathered Reads approach ends up using approximated reprojections

© Disney/Pixar
Advances in Real-Time Rendering in Games

Stereo 3D as Post Processing

Source

© Disney/Pixar
Stereo 3D Implementation

- Human eyes are horizontal
- Stereo reprojection only shifts to the left or right
- We can process each scan line independently
Stereo 3D Implementation

Item Buffer

New X  Old X  Depth  Unused

N-Items where N = width

© Disney/Pixar

Advances in Real-Time Rendering in Games
Stereo 3D Implementation

Step 1: Clear depth of each item in item buffer
Step 2: Iterate over depth buffer and fill item buffer

Depth Buffer:

Item Buffer:

© Disney/Pixar
Step 3: Hole Filling

Item Buffer: Two kinds of holes
Stereo 3D Implementation

Step 3: Hole Filling

Item Buffer:

Rounding Holes

Two kinds of holes
Stereo 3D Implementation

Step 3: Hole Filling

Item Buffer:

Disocclusion Holes

Two kinds of holes
Stereo 3D Implementation

Rounding Holes

Item Buffer: 

These are not real holes in the scene
Just fill in by interpolating the before and after items

© Disney/Pixar
Stereo 3D Implementation

Disocclusion Holes

Item Buffer:
Stereo 3D Implementation

Disocclusion Holes

Item Buffer: ROADHOLELIGHTNING

Item Buffer: ROADROADLIGHTNING
Stereo 3D Implementation

Left Eye Showing Disocclusions

© Disney/Pixar
Stereo 3D Implementation

Step 4: Raster

Item Buffer:

- For each output pixel
  - Interpolate item at each pixel center using the New X value
  - This gives us an interpolated Old X value
  - Use the interpolated Old X value to linear filter into the color buffer and output
Advances in Real-Time Rendering in Games

Stereo 3D as Post Processing

© Disney/Pixar
Stereo 3D as Post Processing
Stereo 3D Implementation

Left Eye With Screen Disocclusion

© Disney/Pixar
Source with tweaked aspect ratio to widen view

© Disney/Pixar
Left Eye Final

© Disney/Pixar
Conclusion

- Moving post processing onto SPU was a big win
- It also gave us Stereo 3D for free
- SPU is a good fit for Stereo 3D post processing
Questions?

Advances in Real-Time Rendering in Games
References

- 3D Stereo Design Guide, PlayStation®3 Library Documentation
- 3D Stereo Programming Guide, PlayStation®3 Library Documentation
- Hutchinson, N., Knight, B., Ritchie, M. Parrish, G., and Moore, J. Screen space classification for efficient deferred shading. In SIGGRAPH 2010
References

Advances in Real-Time Rendering in Games
Stereo 3D Post Processing Gotchas

- Pixel separation can get high when coming out of screen, hence more disocclusion artifacts
  - So clamped min Z
- Small TVs = Higher Pixel separation
  - Clamped distortion to ensure max pixel separation stayed within decent limit
SPU Tips and Tricks

- Use SPA (SPU Pipelining Assembler)
  - Optimizes loops by pipelining, to achieve maximum instruction throughput
  - In some cases this nearly speeds it up 2x
- Double buffer SPU input and output buffers, most SPU effects can run at near 100% CPU utilization, without any memory stalls

© Disney/Pixar
SPU Post Process Performance

Execution vs. DMA stalls

© Disney/Pixar

Advances in Real-Time Rendering in Games
Stereo 3D issues

- Out of screen 3D is difficult
  - Windows violation
- Uncomfortable eye positions/movements
  - 2D focuses only on screen (comfortable)
  - 3D focuses in front and behind
    - Can’t focus on the whole screen at once
    - Quick depth focus change is hard
    - Going cross-eyed is uncomfortable
Appendix A - PS3 Luv sample code

GPU Encode:
Modified function from LogLuv [Karis]:

```c
const static float fx16Scale = 8192.0 / 65535.0;
vResult.zw = unpack_4ubyte( pack_2ushort( sqrt(Xp_Y_XYZp.y) * fx16Scale )).xy;
```

SPU Decode:
```
cuflt = Convert Unsigned Integer To Float
fm = Floating Point Multiply

cuflt lum, lum, 13
fm lum, lum, lum
```