# 5 MAJOR CHALLENGES IN REAL-TIME RENDERING

Johan Andersson, DICE

# Overview

- What are the major challenges for us in the next 5-10 years?
  - Real-time rendering for games as well as other areas
- Which problems do we want to solve?
- What do we want to achieve & focus on?
- Based on own thoughts & feedback from people in the industry

# 25~~5~~ MAJOR CHALLENGES IN REAL-TIME RENDERING

Johan Andersson, DICE

# Challenges 2012

Same as 2010!

1. Cinematic Image Quality
2. Illumination
3. Programmability
4. Production costs
5. Scaling

Challenge #1

# CINEMATIC IMAGE QUALITY

# Cinematic Image Quality

- Goal is to achieve Cinematic Image Quality
  - Same smooth & rich pictures that CG movies have
- Need significant improvements to GPU primary visibility
  - Antialiasing
  - Transparency
  - Defocus blur
  - Motion blur
- A future solution needs to include all *together*

# Antialiasing

- Single most visible issue to improve on
  - Aliasing breaks the illusion
  - Less aliasing = more pleasing & easier to see visuals

- Sources of aliasing:
  - Geometric aliasing
  - Proxy geometry aliasing (alpha test)
  - Shader aliasing
  - Mixed resolution rendering

# Post-process antialiasing

- Lots of developments in post-AA techniques
  - MLAA, FXAA, SMAA and more
  - Good quality / performance ratio

  - See SIGGRAPH'11 course: *"Filtering Approaches for Real-Time Anti-Aliasing"* http://iryoku.com/aacourse/

- But need solutions for the full problem

# Geometric aliasing

- Current solutions: MSAA, SSAA, temporal
  - Fixed quality techniques, not adaptive
  - Problematic to scale up to very high quality
- 16x MSAA is really good quality but expensive
  - Need high rate if using coverage masks
- MSAA + deferred
  - Massive memory usage & bandwidth
  - Want access to MSAA compression surface to avoid computing ourselves
  - See Andrew's talk in the course: *"Intersecting Lights with Pixels"*

# Geometric aliasing

- Other alternatives?
  - Analytical antialiasing
    - Would be interesting to see more research
  - Pre-filtered Sparse Voxel Octrees
    - Requires *very* high resolution / large storage
    - See Cyril's talk in the course: *"Dynamic Sparse Voxel Octrees for Next-Gen Real-time Rendering"*

# Shader aliasing

- Shader aliasing becoming more of a problem
  - High-frequency specular highlights
  - High-frequency shadows
  - Amplified by HDR Bloom & Bokeh

- What is needed to make sure shaders do not output aliased values?
  - Careful handling of derivatives when texture mapping
  - LEAN mapping, EVSM shadows
  - Wednesday: *“Rock-Solid Shading: Image Stability without Sacrificing Detail”*

# Proxy geometry aliasing

- Alpha-testing for proxy geometry results in major aliasing
  - MSAA per-sample evaluation is costly & requires many samples
- Real transparency can directly solve the aliasing
  - But we need to sort, need *order-independent transparency*

*Assets from Valve*

# Transparency

- Order-dependent transparency has always been a big limitation for content creators & developers
  - Restrictive art pipeline:
    - No glass houses
    - Even windows on cars & buildings can be painful
  - Restrictive interaction between objects & effects
    - Meshes vs particles vs volumetrics
    - Lack of sorting prevents usage of other transparent techniques
- Order-independent transparency is must going forward
  - With good performance & determinism

# Order-independent Transparency

- *Adaptive Transparency* [Salvi11] is promising
  - Currently requires multi-pass & unbounded memory
  - Need render target read/modify/write to get single-pass & bounded memory
  - Composite as you go (forward lighting)

# Motion blur


- Important for sense of speed & direction
- Velocity vectors + post-process holds up quite well

# Defocus blur

- Key visual cue to perceive depth & focus
  - Guide & emotional storytelling tool

- Sprite splatting [Igarashi08] is popular
  - Works great for out of focus background
  - Very sensitive to aliasing
  - Sharp edges on strong foreground blur

# Defocus blur – visibility issue

Incorrect visibility = hard edge

Correct visibility

# Defocus & motion blur – beyond post

- Raytrace geometry
- Stochastic rasterization
  - Lots of samples required for foreground, too little = noisy
    - Critical to have fast & temporally stable image reconstruction
  - Is defocus or motion blur the most important?
    - Leaning toward defocus due to the visibility issue
    - Though MB should not be applied after DOF, need a solution that handles both properly
- Pre-filtered Sparse Voxel Octrees


Challenge #2
ILLUMINATION

# Illumination challenges

Dynamic Global Illumination

Shadows

Reflections

# Dynamic Global Illumination

- Key visual component
- Multiple dynamic alternatives now:
  - Light Propagation Volume
  - Voxel cone tracing
  - Reflective Shadow Maps + VPLs
  - Geometry pre-compute based: Enlighten
- Major trade-offs depending on perf/memory/quality

# Dynamic GI wanted characteristics

- Static & dynamic geometry
- No pre-computation required
  - Major tradeoff with performance
- Handle large scales: indoor to large outdoor
  - With minimal light leakage
- Multiple indirect bounces
  - Single is not enough for in-door
- Indirect specular reflections

# Shadows

- General shadows continue to be a major challenge
  - *"Efficient Real-time Shadows"* course
- As a game developer, I'm tired of shadows ☹
  - Not what the graphics pipeline is built for
  - Time consuming tweaking, optimizations, compromises
  - Current techniques don't scale up
  - Are there Graphics/Compute extensions that could help?

# Shadows - wanted characteristics

**Robust:**

- Stable under object, light & camera motion
- No light leakage
- No flickering
- No magic constants

**High-quality:**

- Variable penumbra
- No aliasing
- Motion blurred

**General:**

- Works with all light types
- Supports dynamic geometry
- Supports alpha-test
- Supports transparent receivers & casters
- Scalable from small to large light sources

**Fast!**

- Sparse sampling
- Good culling

# The Many Shadow problem

- Want shadows on all lights
  - Easier to author
    - No light leaking through walls
  - Doesn't limit content creators
  - Higher quality & more interactive

- Current issues
  - Amount of geometry
  - Culling
  - Draw calls
  - Non-sparse rendering

Super simple scene, 10 spotlights

# Many Shadow – potential solutions

- Efficient rasterization
  - Smart logarithmic triangle culling with spatial data structure
  - Lazy on-demand scene culling, geometry culling & graphics dispatch
    - with CPU or GPU Compute
- Raytrace geometry
  - Render gbuffer
  - For each pixel affected by a light, cast ray to light
  - Evaluate & composite directly in pixel shader, or output to light visibility masks
- Cone trace into SVO
  - Once SVO data structure is built, easy to cone trace to query light visibility
  - Soft shadows = fast! ☺

# Reflections – categories

Glossy reflections on arbitrary surfaces

Perfect reflections on mostly-planar surfaces

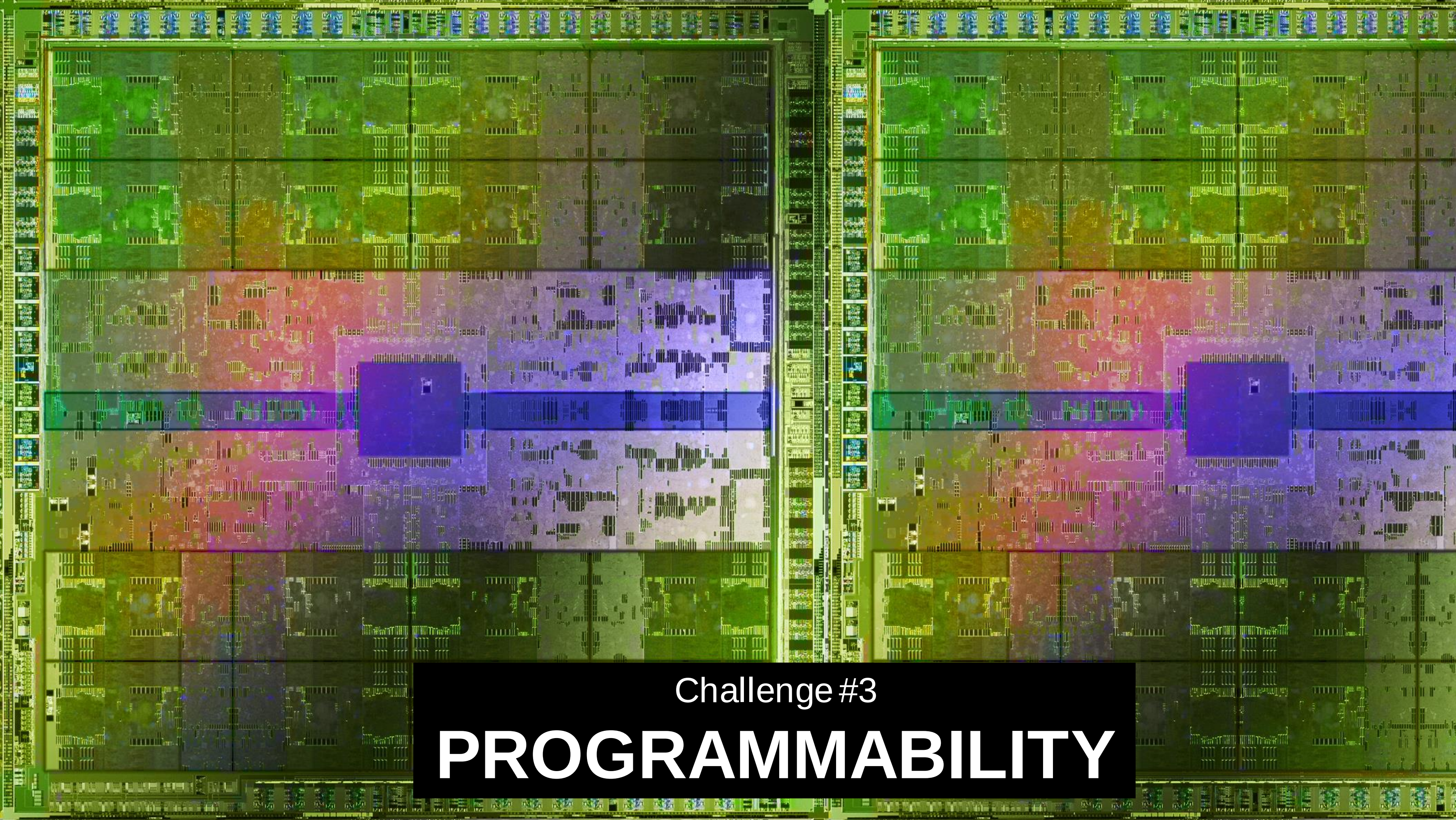
# Reflections – use cases

- Glossy reflections
  - Most surfaces, rough metal

  - Screen-space reflection
    - Fully dynamic
    - Simple & cheap
    - Visibility problems
  - Voxel Cone Tracing
    - Can be a good fit
    - Expensive to build SVO of scene

# Reflections – use cases

- Perfect reflections
  - Mostly planar surfaces: windows, water
  - Render reflected view(s)
    - Prohibitive to render scenes upfront with multiple planes
    - Want more sparse rendering
  - Raytracing is elegant, but impractical
    - Performance
    - Have to switch entire game graphics pipeline
  - Voxel Cone Tracing
    - Requires massive resolution
    - Not practical until we can use it for primary visibility



Challenge #3

# PROGRAMMABILITY

# Programmability

- Need major innovations to solve the other challenges

# Programmability - Pipelines

- Graphics pipeline is fast but fixed
  - No conservative rasterization
  - No programmable blending
  - No flexible texture filtering (min/max/derivative)

- GPU Compute can't efficiently simulate a full graphics pipeline
  - Use the graphics pipeline when possible
  - Need to enable building your own efficient GPU Compute pipelines
    - When going beyond graphics pipeline capabilities
    - Esp. important with long HW & OS lead times

# Programmability - Areas

- Need a virtual data-parallel ISA
  - Separate front-end from back-end
    - Can use same front-end (language) on all platforms (back-ends)
    - Run on all platforms and all (modern) architectures
      - Both CPU and GPU!
    - HSA with HSAIL is one promising solution
  - Most developers are multi-platform
    - Won't write serious code in platform- or vendor-specific languages

# Programmability - Areas

- Strip down the GPU SW stack & hardware abstractions
  - Only bindless access to resources
    - Need standard non-opaque texture layouts
  - Virtual memory
  - Both CPU or GPU Compute can generate GPU work
  - Layer existing fat APIs on top of it (DirectX & OpenGL)

# Programmability - Areas

- GPU Compute spawning fine-grained tasks for itself
  - Build your own pipelines, independently of CPU
  - Kepler GK110 'Dynamic Parallelism' on all chips, platforms and compute languages
- Need mechanisms to build SIMD coherency
  - Not ideal to write out giant sample lists to memory and sort
  - *Queues* as a language & HW abstraction primitive

# Programmability - Areas

- Low-latency CPU/GPU collaboration
  - Balance work, run where most efficient
    - Frostbite uses it on consoles [Coffin11] [Brisebois11]
  - GPU spawning work for CPU
  - CPU inserting more work for GPU *within* the frame
  - Simple use case for Sample Distribution Shadow Maps:
    1. Render z+gbuffer
    2. Analyze zbuffer to determine ideal shadowmap distribution
    3. Kick of CPU to cull & create shadowmap graphics display list
    4. GPU renders something else while waiting for CPU

# Programmability - Areas

- Render target read/modify/write
  - A must have base operation
  - OIT & programmable blending


Challenge #4

# PRODUCTION COSTS

# Production costs

- Games are getting bigger & more complex
  - More content
  - More variation
  - Higher quality/detail
  - More complex content production process

- What are the next big step forward for content production?
  - Quick iteration times = quality

# Production costs

- If we had the ultimate real-time renderer that solves primary visibility and illumination, how much artist time would we save?
  - Probably not that much overall unfortunately (but increase quality)
    - Having shadows everywhere and dynamic GI saves some artist time
    - Will save engineering time & support
  - Content creation is the biggest time sink
- What can save significant amount of time?
  - Scalable geometry representation
  - Procedural texturing
  - Procedural geometry
  - Content acquisition

Challenge #5

# SCALING

# Scaling

- Games & rendering use cases are needing more and more scaling. Both up and down!
  - Detail: mm to km
  - Resolution: 320x480 (IPhone3) to 5760x1200 (Eyefinity). 45x
  - Power: 1W to 300W. 300x
- Requires significant scaling in performance
- Which techniques, algorithms & pipelines are scalable?

# Scaling: Detail

- How can we increase detail while building even larger interactive worlds?
  - Scalable geometry is difficult, discrete LODs suck
    - Want an inherently scalable & filterable primary data representation. Dynamic SVOs?
  - Can’t author everything
    - Use procedural detail up close

# Scaling: Resolution

- Some of the lowest powered devices have the highest resolution screens
  - Consumers ☺
  - Developers ☹
- Graphics pipeline need a more flexible decoupling of *shading rate* vs *visibility rate*!
  - MSAA and fixed upsampling is not enough

# Scaling: Power

- Marketplace is shifting from 100+ W to 1-45 W
  - Phones (1 W), Tablets (3 W), Ultrabooks (17 W), Laptops (45 W)
- Developers typically don't care about power usage
  - But hardware/device manufacturers and consumers do
  - With cloud rendering, power is a direct cost for developer
- Need power efficient algorithms, techniques & pipelines
  - Grand challenge for the next 10 years: photo-realistic rendering at 1W

# Questions?

email: johan.andersson@dice.se

blog: http://repi.se

twitter: @repi

course page: http://bps12.idav.ucdavis.edu

# Thanks for the feedback!

Christina Coffin
Cyril Crassin
Aaron Lefohn
Marco Salvi
Andrew Lauritzen
Colin Barré-Brisebois
Ivan van Assen
Matt Collins
Charles de Rousiers
Stephen Hill
Nathan Reed
Timothy Lottes

Sebastien Lagarde
Sébastien Hillaire
Jim Vaughn
Steven Tovey
Aras Pranckevičius
Julien Guertault
Sam Martin
Niklas Lundberg
Michael Nicolella
Aaron Miller
,

Pål-Kristian Engstad
Jonathan Ragan-Kelly
Matthijs De Smedt
Doug Binks
Morgan McGuire
Andrew Richards
Daniel Wexler
Eric Smolikowski
David Möllerstedt

# References


- [Igarashi08]. Real-Time Depth-of-Field Rendering Using Point Splatting on Per-Pixel Layers. http://dxp.korea.ac.kr/homewiki/images/f/f5/PG-PointDof-submit.pdf
- [Coffin11]. SPU Based Deferred Shading in Battlefield 3 for Playstation 3. Christina Coffin. http://publications.dice.se/attachments/Christina_Coffin_Programming_SPU_Based_Deferred.pptx
- [Brisebois11] More Performance Five Rendering Ideas from Battlefield 3 and Need For Speed The Run. Barre-Brisebois et al. http://publications.dice.se/attachments/BF3_NFS_WhiteBarreBrisebois_Siggraph2011.pptx
- [Salvi11] Adaptive Transparency. Salvi et al. http://software.intel.com/en-us/articles/adaptive-transparency-hpg-2011/