The Art and Technology of Whiteout

Abe Wiley
Thorsten Scheuermann
Game Computing Applications Group
AMD

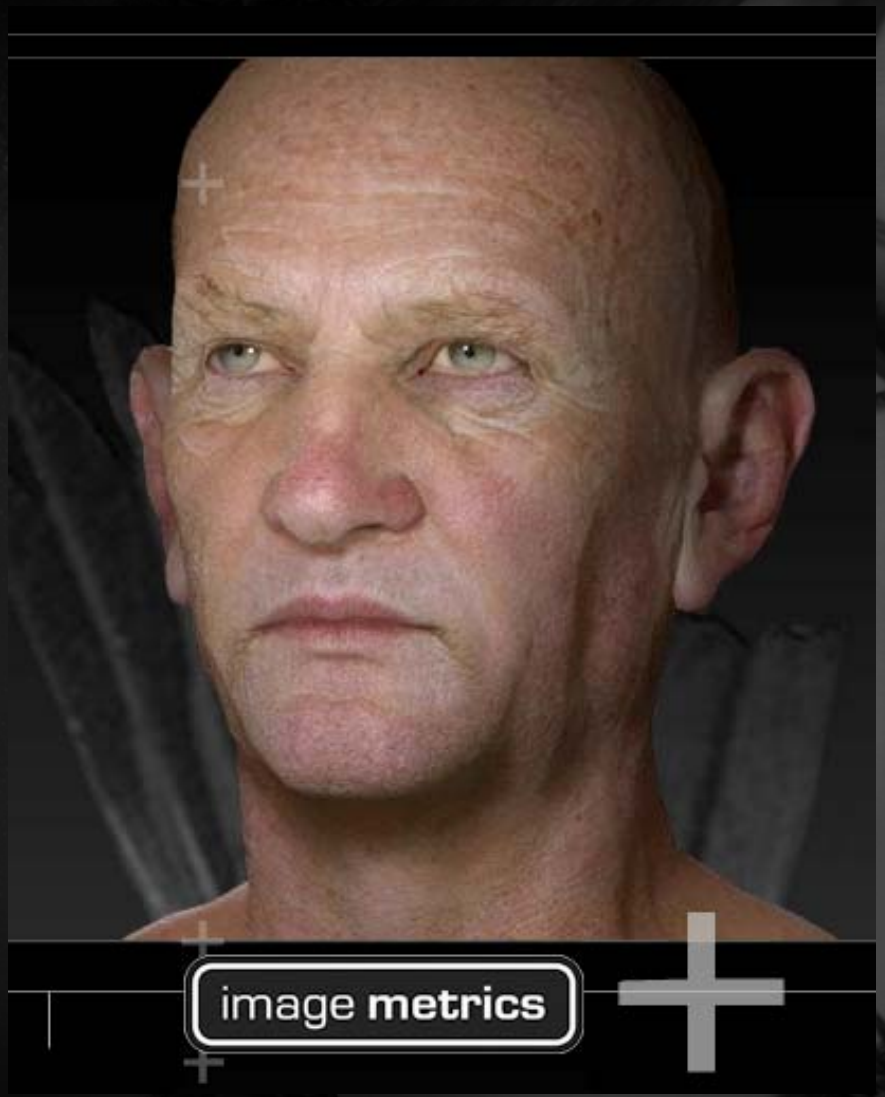
Introduction

AMD Smarter Choice

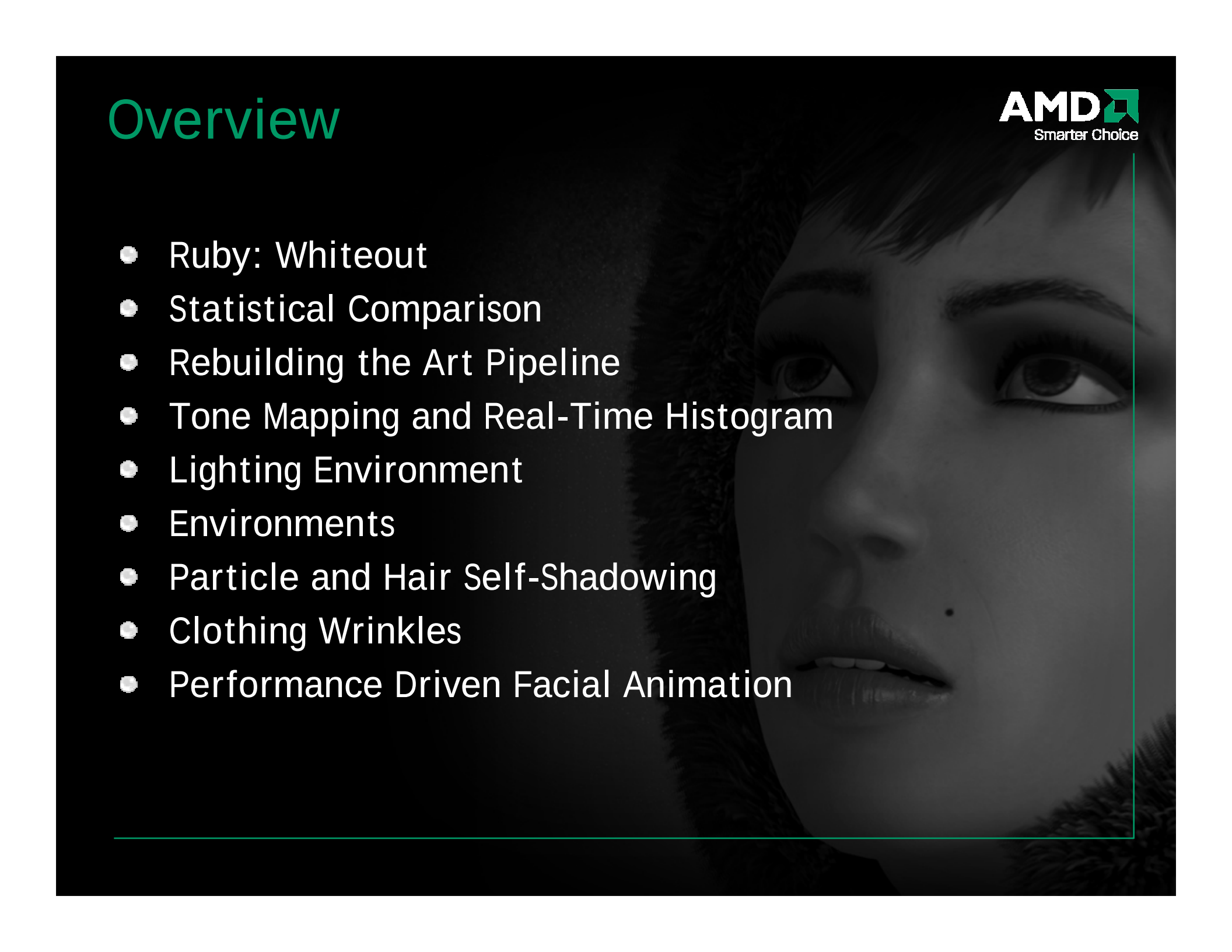
- Abe Wiley
  - Lead Artist
- Thorsten Scheuermann
  - Lead Engine Programmer
- Game Computing Applications Group
  - Demos and Research
  - Formerly 3D Applications Research Group at ATI
  - Focusing on GPU and CPU technologies for Game Development

Image Metrics

- Image Metrics
  - Performance Driven Facial Animation
- Patrick Davenport
  - Executive Producer
- Dr. Mike Rogers
  - Senior Research Engineer



Overview



- Ruby: Whiteout
- Statistical Comparison
- Rebuilding the Art Pipeline
- Tone Mapping and Real-Time Histogram
- Lighting Environment
- Environments
- Particle and Hair Self-Shadowing
- Clothing Wrinkles
- Performance Driven Facial Animation


AMD
Smarter Choice
DEMO
Ruby: Whiteout
AMD

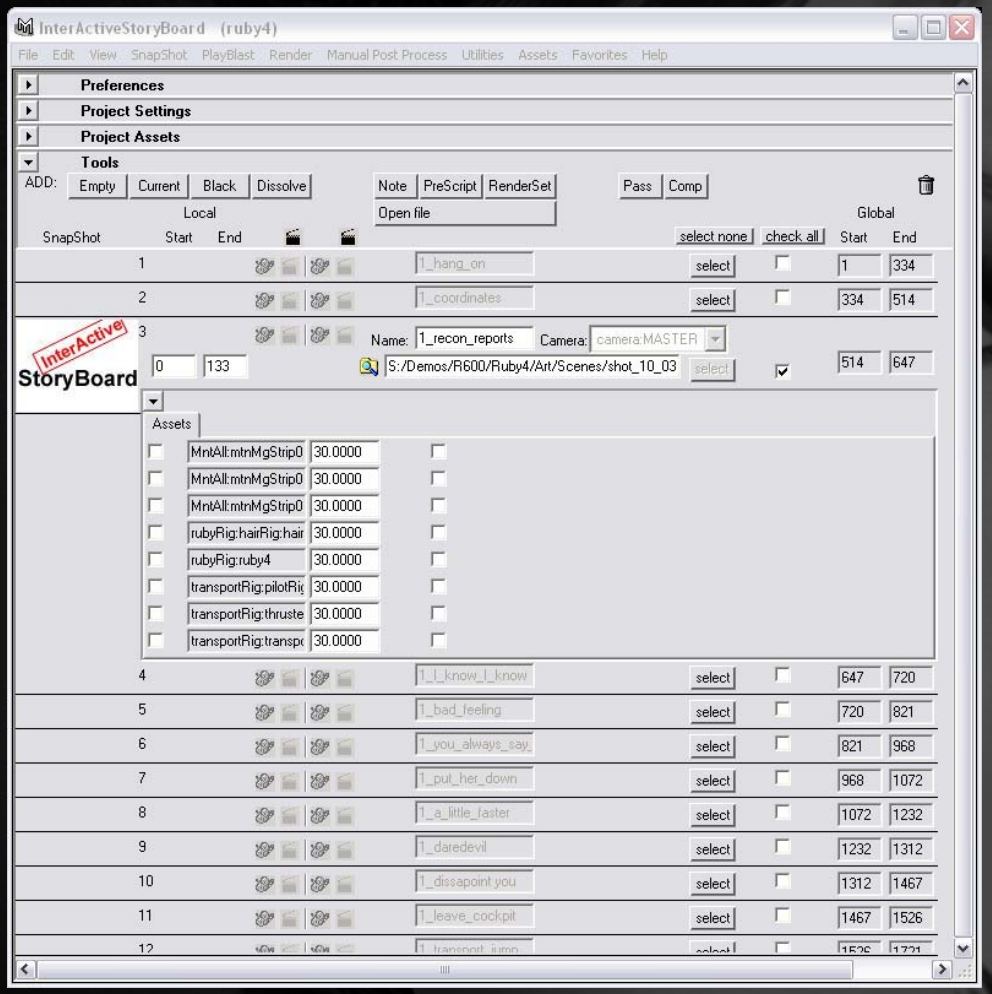
Ruby Statistics

AMD Smarter Choice

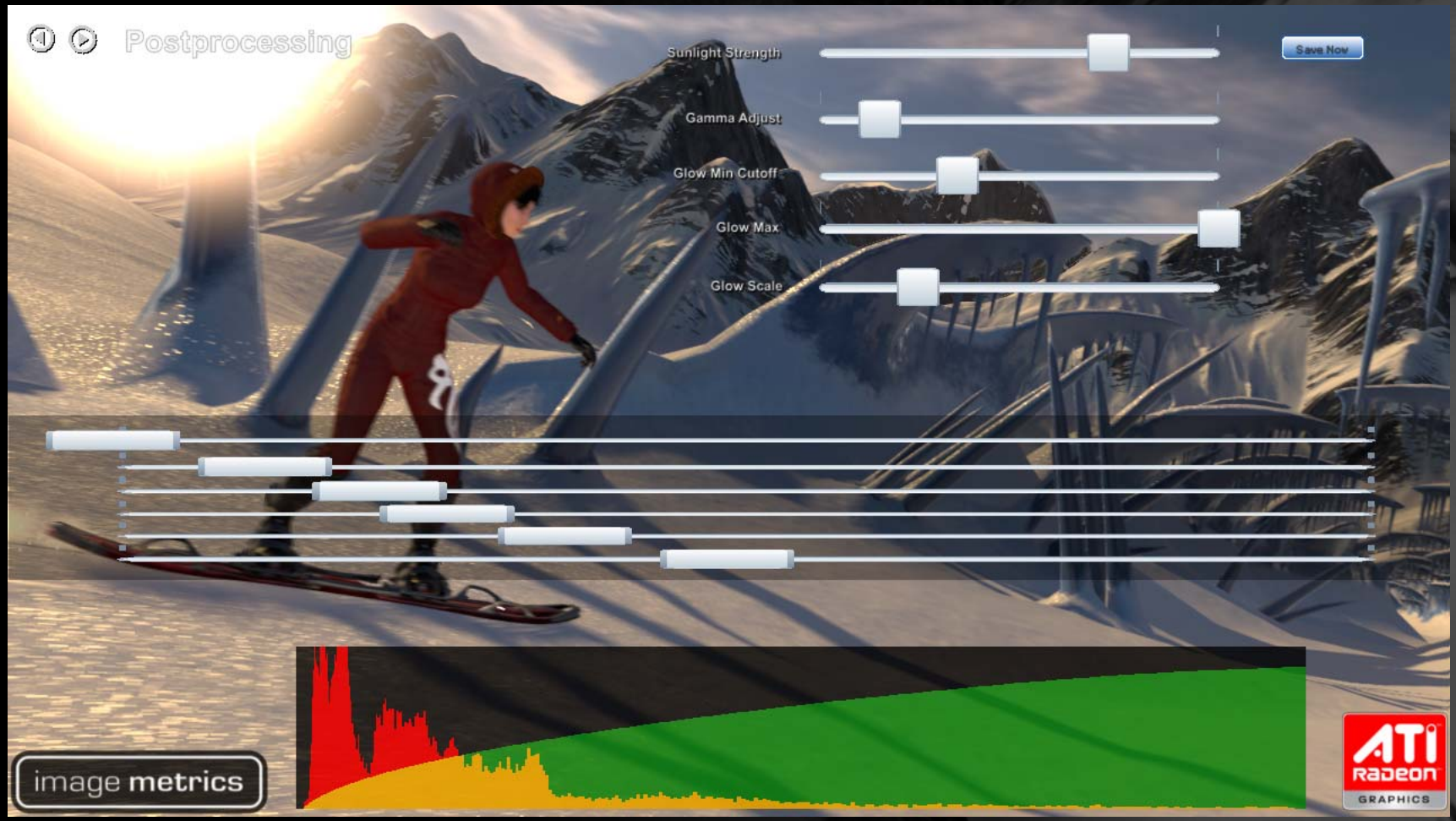
	DoubleCross	The Assassin	Whiteout
Ruby Poly Count	80,000	80,000	200,000
Avg. Triangles/Frame	227,212	546,087	1,069,503
Max Triangles/Frame	556,305	1,018,312	2,150,521
No. of Pixel Shaders	100	316	210
Avg. Pixel Shader Length	20	74	142
Simultaneous Facial Animation Morph Targets	4	4	> 128
ALU:Tex Ratio	4:1	7:1	13:1

Re-building the Art Pipeline

- Interactive Storyboard (IASB)
- Flexibility + Control = Creativity
- Shot Based Structure
- Lua Scripting
- Shot-by-Shot Post Processing, Distance Fog, and Tone Mapping



Artist Editable Post Processing



High Dynamic Range Rendering

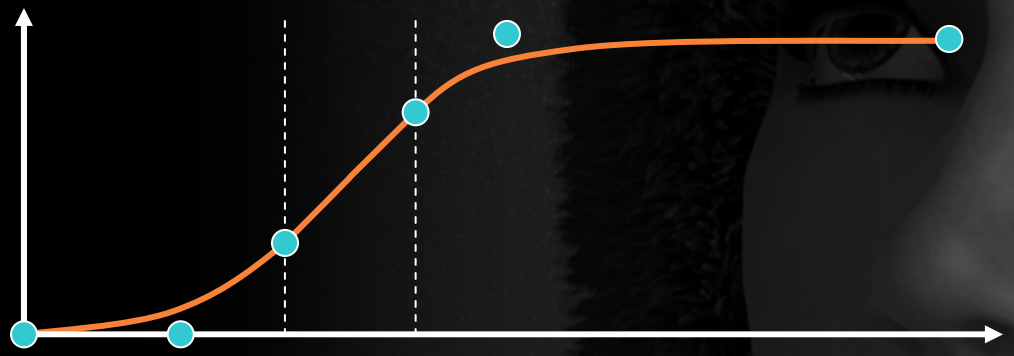
AMD Smarter Choice

- Render into multi-sampled float16 buffer
- Tone map into monitor-displayable brightness range
- Artist controlled tone-mapping curve in main demo
- Auto-exposure using real-time histogram and Ward's Histogram Adjustment TMO
- Histogram of final rendered frame useful as a visualization tool for artists

Artist-Controlled Tone Mapping

AMD Smarter Choice

- 6-point S-curve
  - 2 quadradic beziers connected with linear segment



- Evaluated for each pixel in tone mapping shader
- Could build a lookup texture each frame
  - Radeon HD 2900 fast enough to do the math

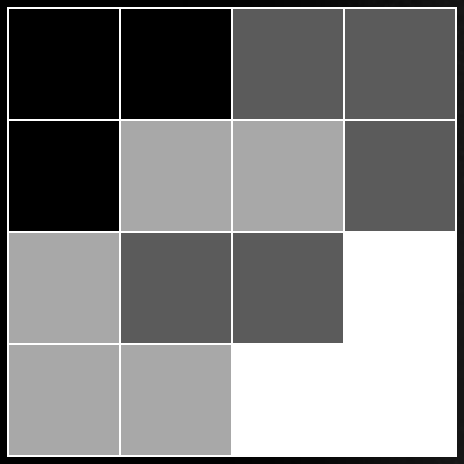
Real-time Histogram Algorithm

AMD Smarter Choice

- Store histogram in render target
  - One pixel per bin
- Scatter data through vertex shader:
  - Render one point per input pixel
- Vertex shader maps pixel color → histogram bin
  - Convert bin into output location for histogram render target
- Increment histogram bin using additive blending

Building the Histogram

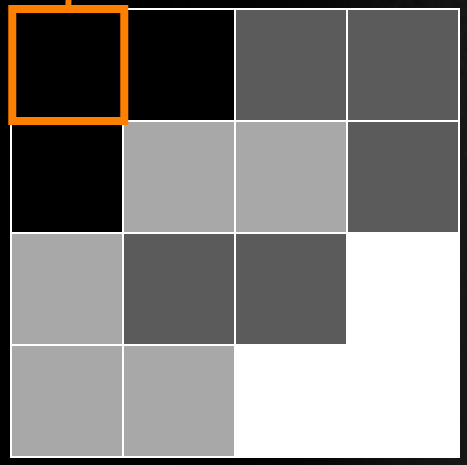
AMD Smarter Choice



Output render target

0	0	0	0

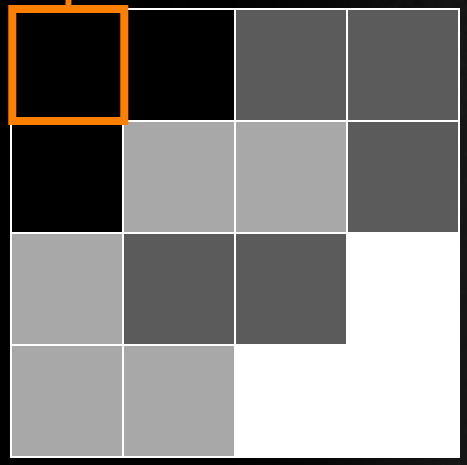
Building the Histogram

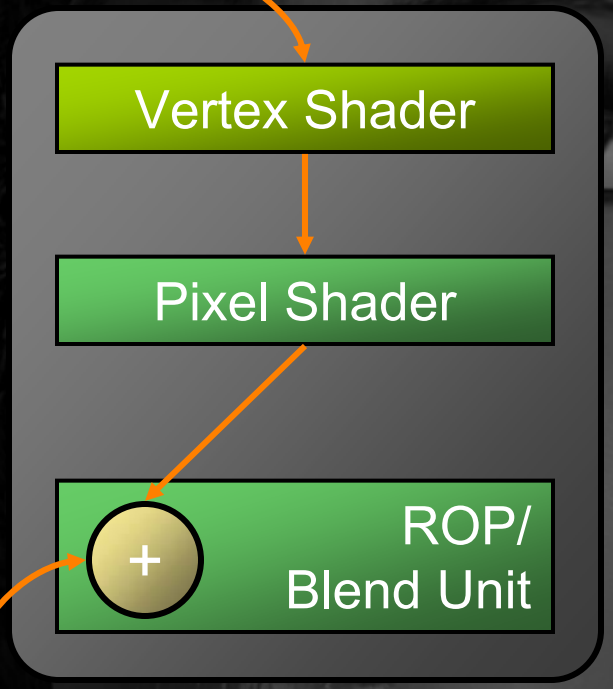


Output render target

0	0	0	0

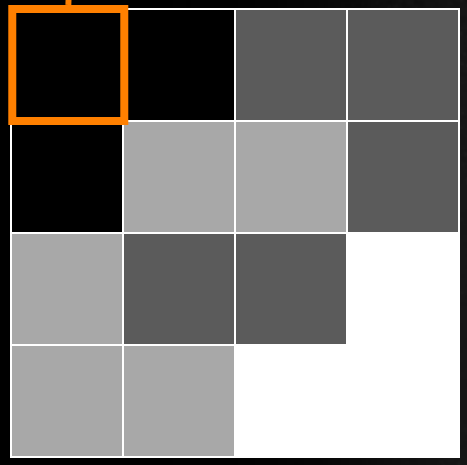
Building the Histogram

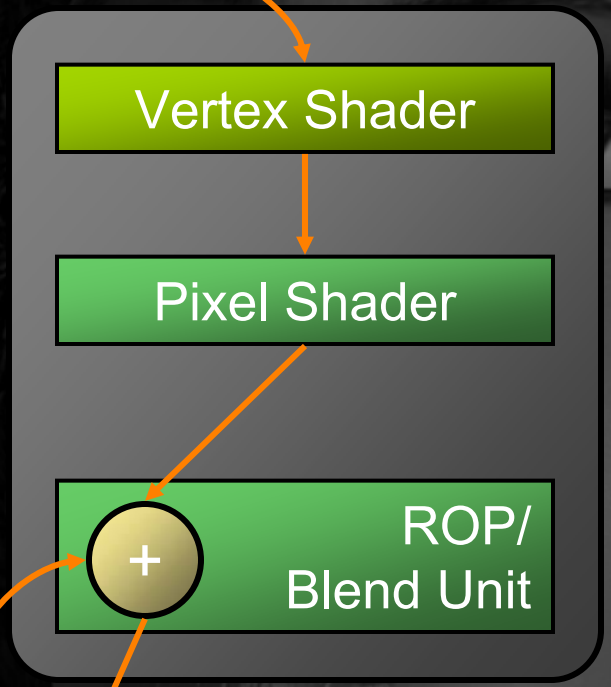




Output render target

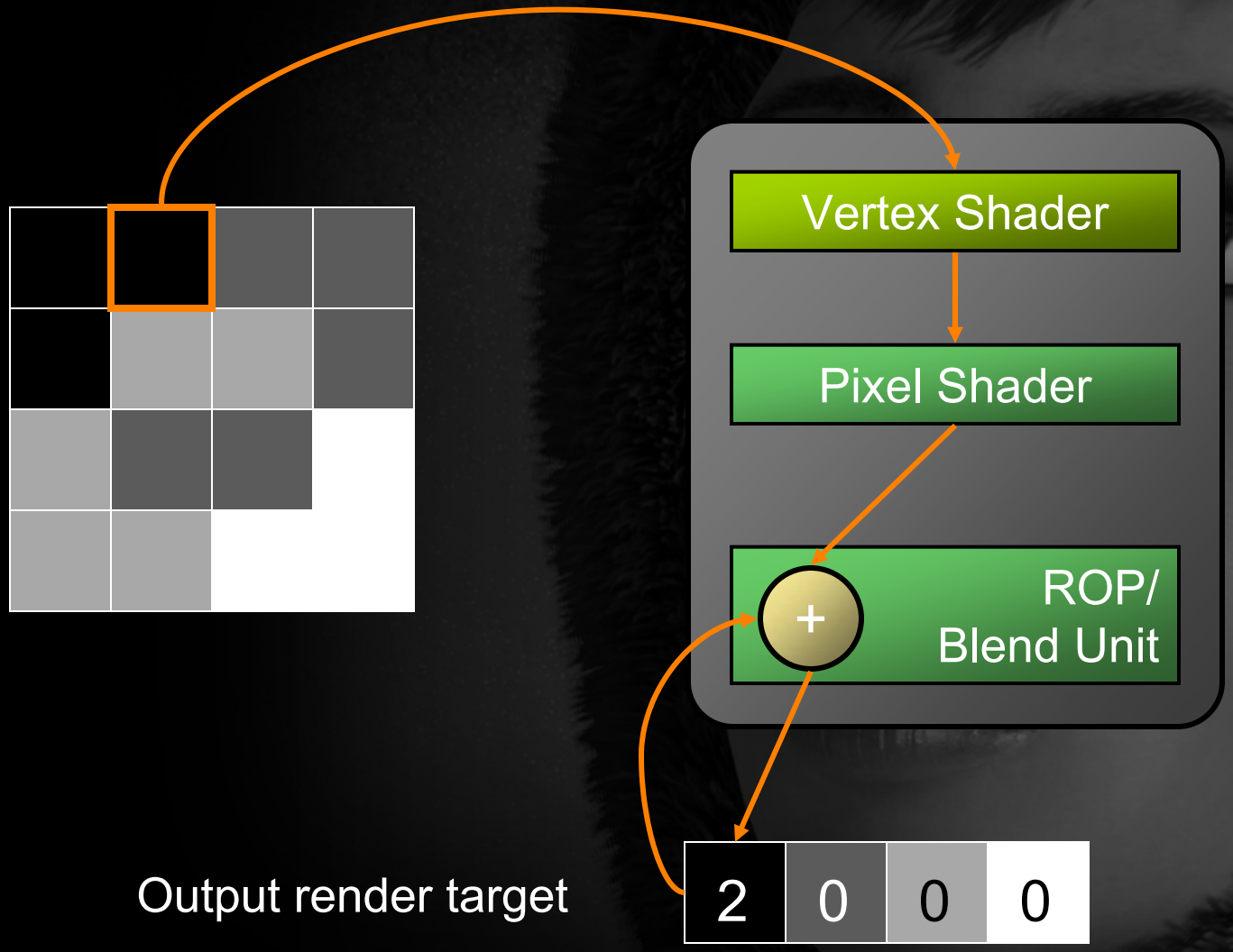
Building the Histogram



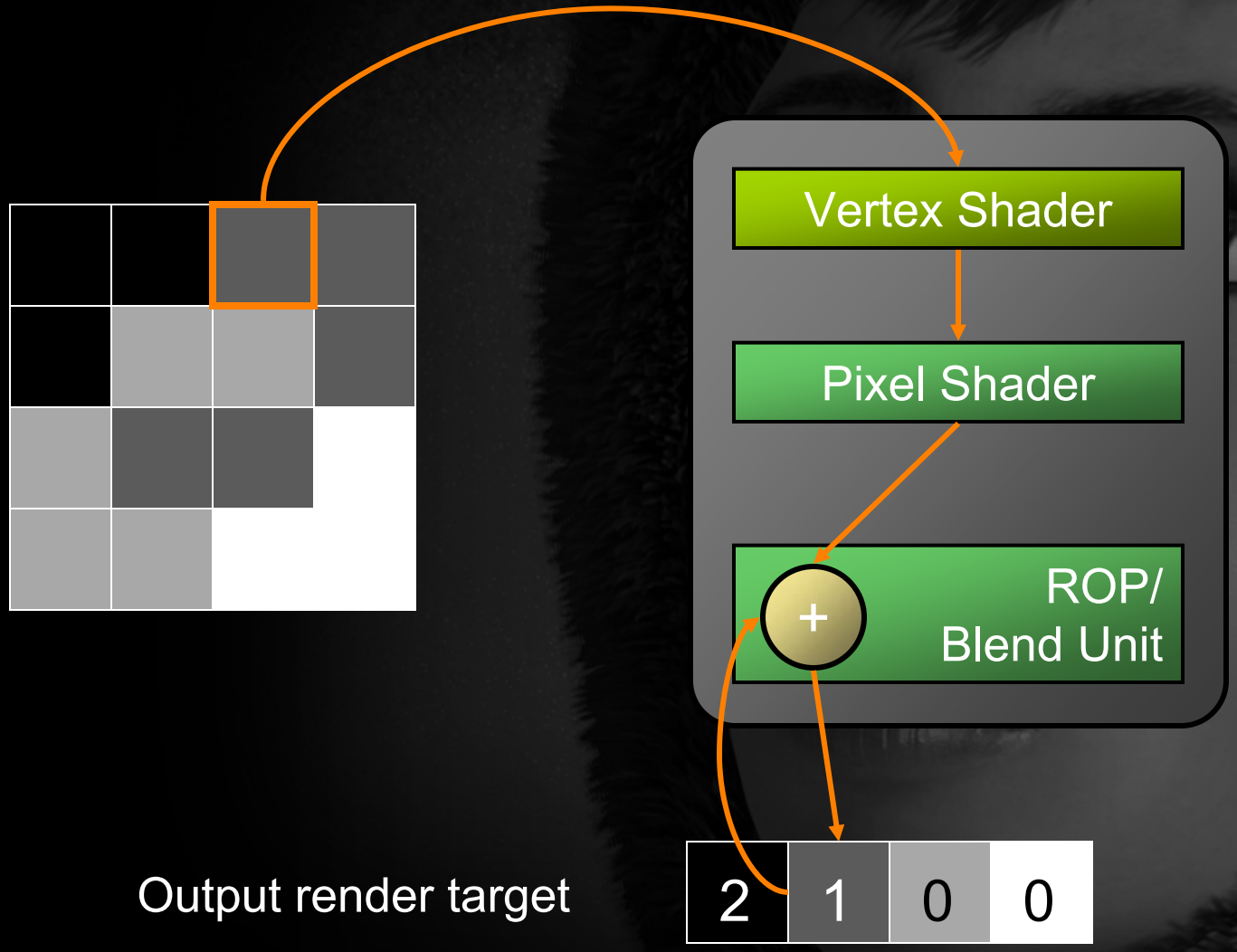


Output render target

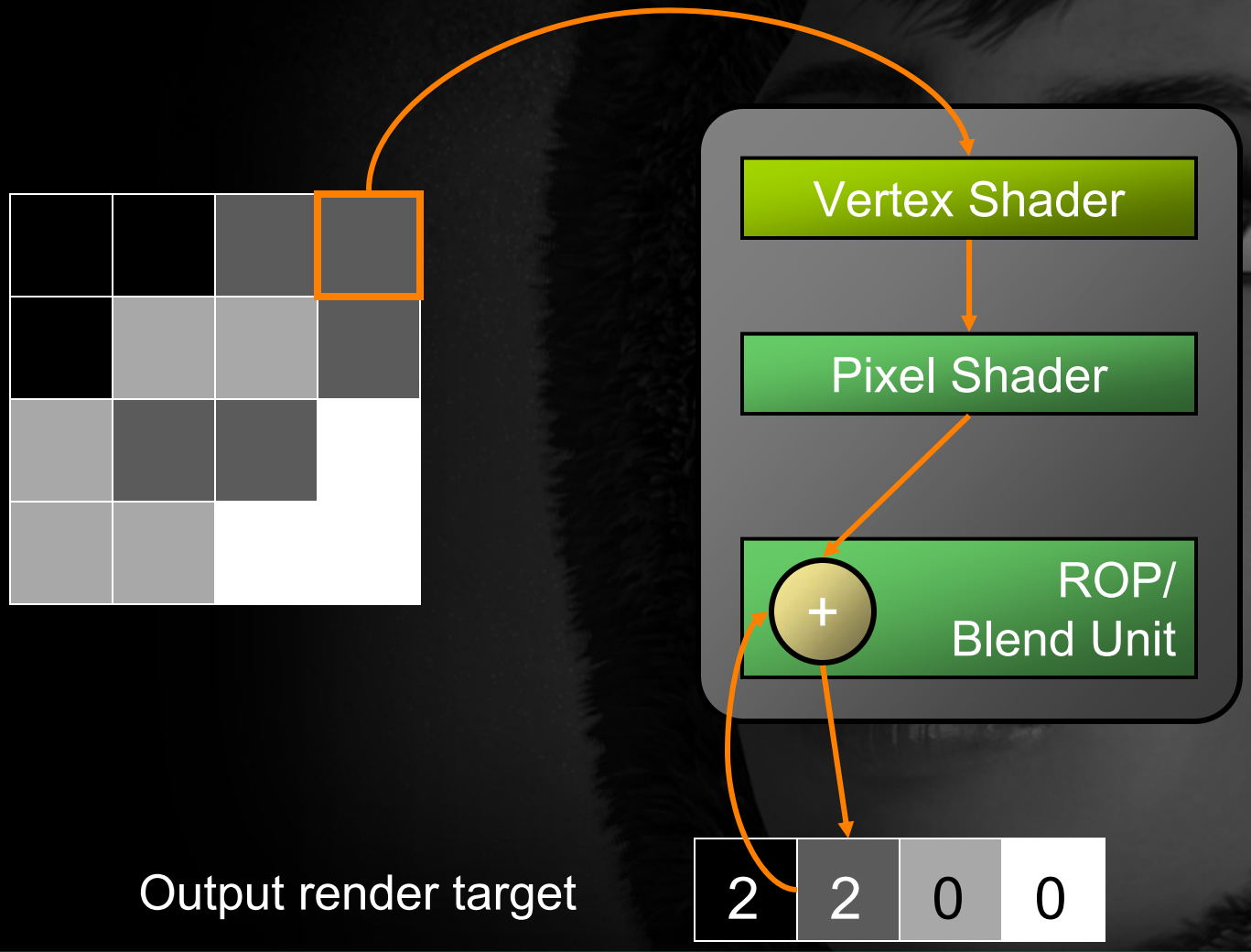
Building the Histogram



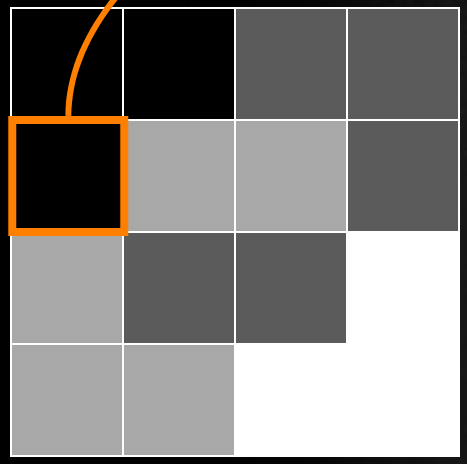
Building the Histogram

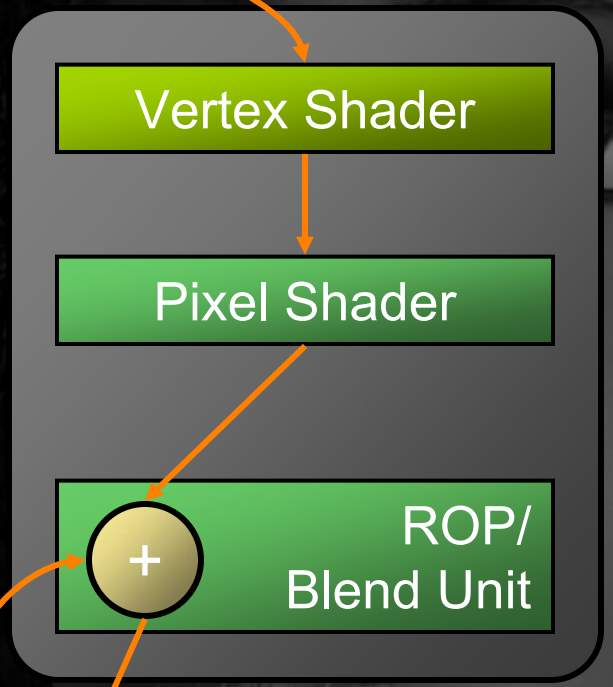


Building the Histogram



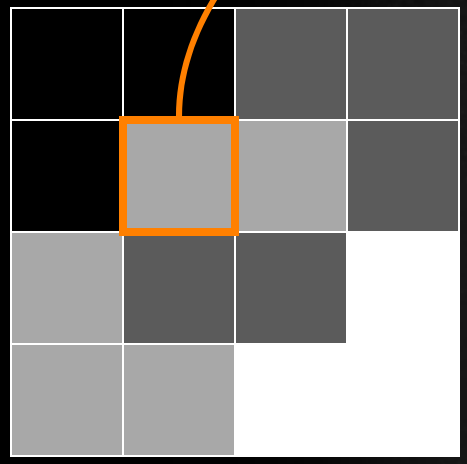
Building the Histogram

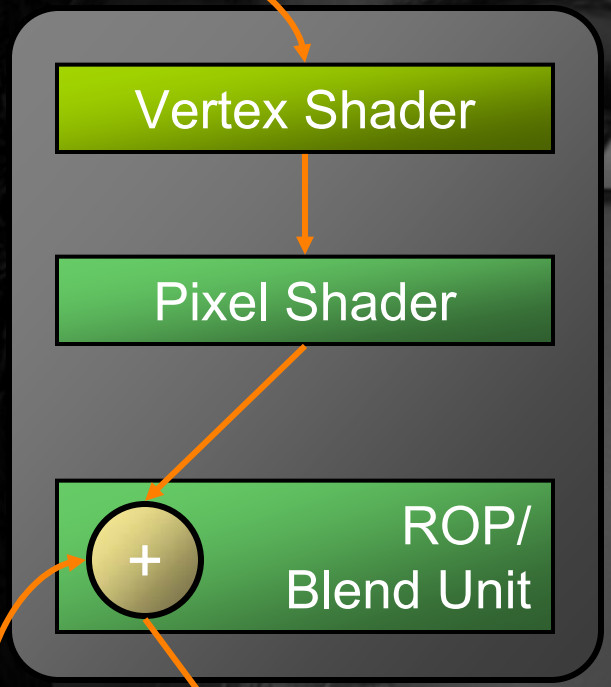




Output render target

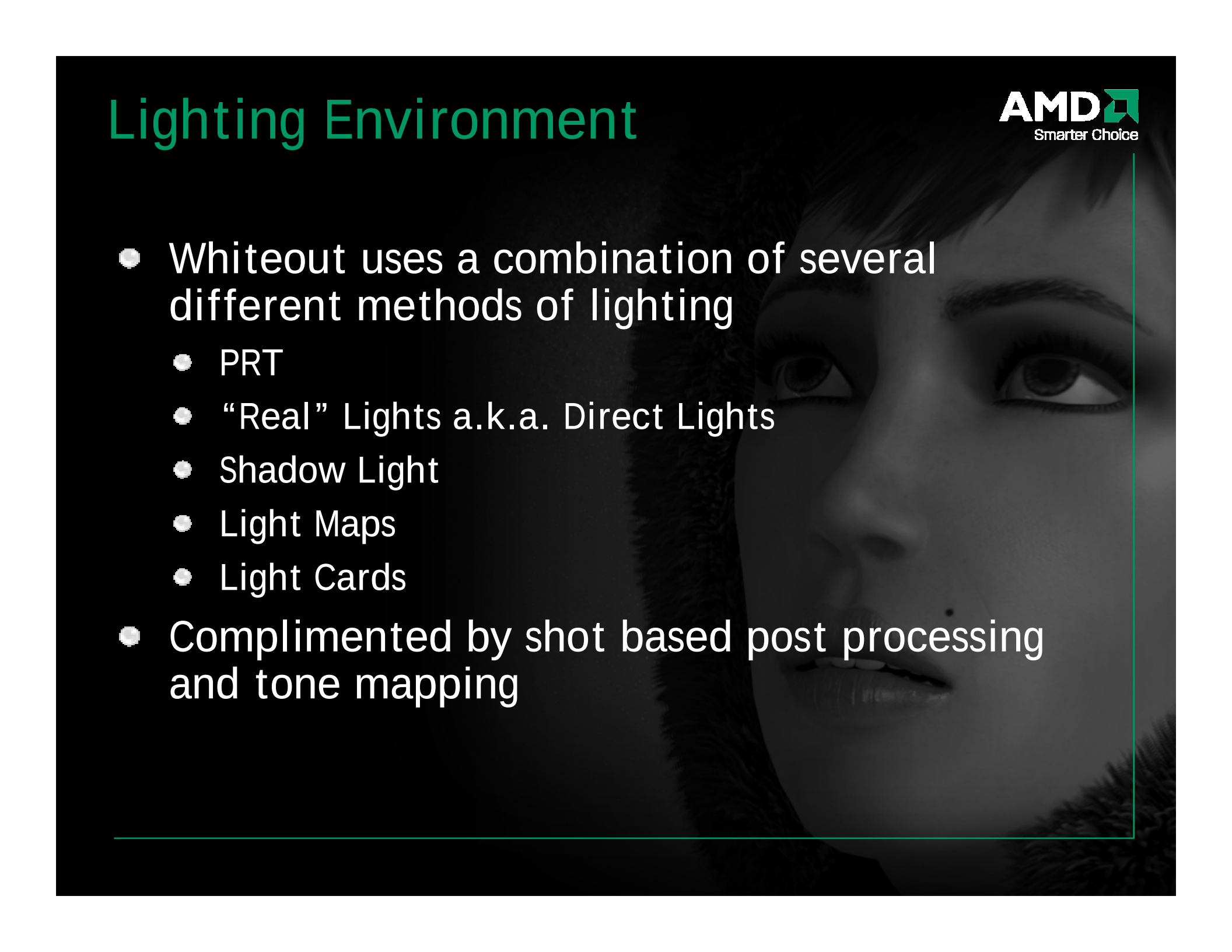
Building the Histogram





Output render target

Lighting Environment

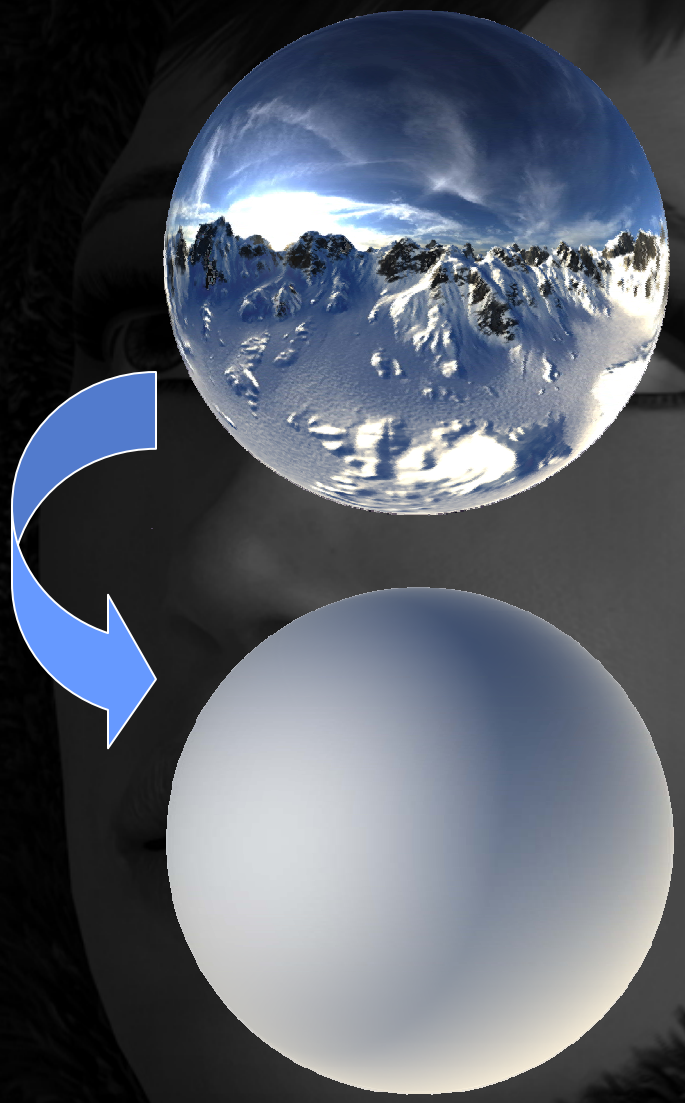


- Whiteout uses a combination of several different methods of lighting
  - PRT
  - “Real” Lights a.k.a. Direct Lights
  - Shadow Light
  - Light Maps
  - Light Cards
- Complimented by shot based post processing and tone mapping

Pre-Computed Radiance Transfer (PRT)

AMD Smarter Choice

- Per-vertex data describing reaction to light from different directions
  - Sub-surface scattering
  - Bounced lighting
- Lighting is captured as a cube map and converted to SH representation



PRT in Ruby 2 vs. Ruby 4

- Pre-computed SH irradiance volumes
  - Extra storage overhead
  - Lighting samples don't adjust to dynamic/ destructible environments
- SH coefficients generated from a dynamic cube map every frame
- Dynamic SH projection
  - Feasible if you already have a dynamic cubemap, e.g. for Hero characters

Dynamic SH Projection

AMD Smarter Choice

- Converts cube map to SH coefficients
- Lookup textures for solid angle * SH coefficients for each input cubemap texel
- For each cube map texel:
  - Multiply with corresponding lookup data
  - Sum everything up to get final SH coeffs
- Final SH coeffs stored in 2D texture
  - Vertex shaders use texture fetches to get SH data
- We use 4^{th} order SH

PRT Skinning

AMD
Smarter Choice

- PRT data only valid for mesh pose it was pre-computed for
- For skinned objects: Rotate PRT SH coeffs by skinning matrix
- Then convolve with lighting coeffs
- Not technically correct…
  - Self-occlusion/bounced reflection doesn't update based on skinned pose
- … but works well in practice

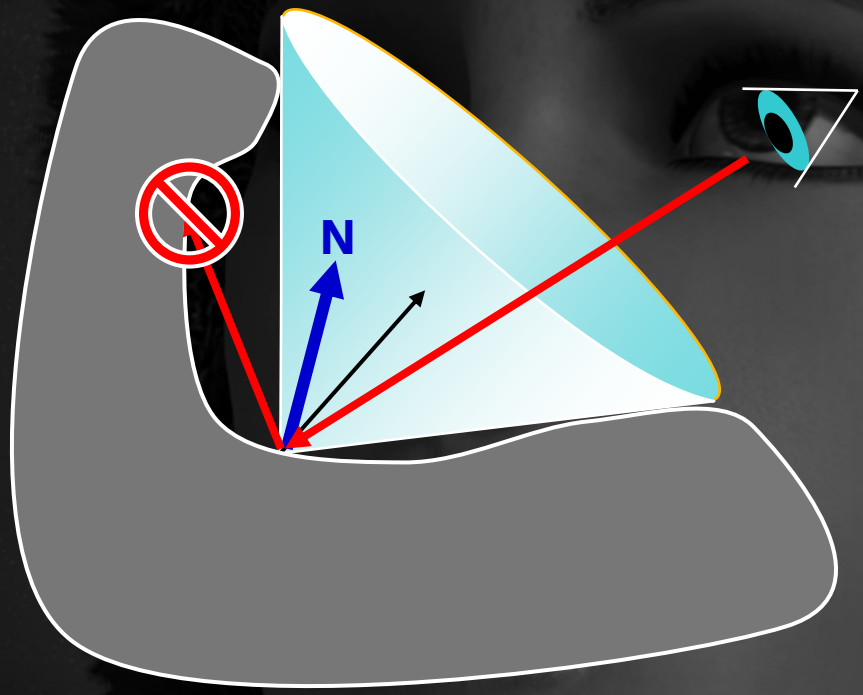
Normal Mapping and PRT

AMD Smarter Choice

- Vertex Shader
  - Rotate PRT coeffs by skinning matrix
  - Only convolve 3rd- and 4th-order PRT bands with lighting environment
  - Pass rotated 2nd-order PRT coeffs to pixel shader
- Pixel shader:
  - Determine rotation matrix so that bump-mapped normal = R*vertex normal
  - Rotate 2nd-order PRT coeffs by R
  - Convolve with lighting environment
  - Final lighting: Add result to convolution result from vertex shader

Specular Occlusion

- Visibility Aperture
  - Cone that approximates visibility from a point
  - Defined by bent normal and aperture radius
- first two bands of PRT data give visibility aperture estimate for free
- Useful for specular occlusion
  - masking cube map reflection lookups



Cube Map Setup in Whiteout

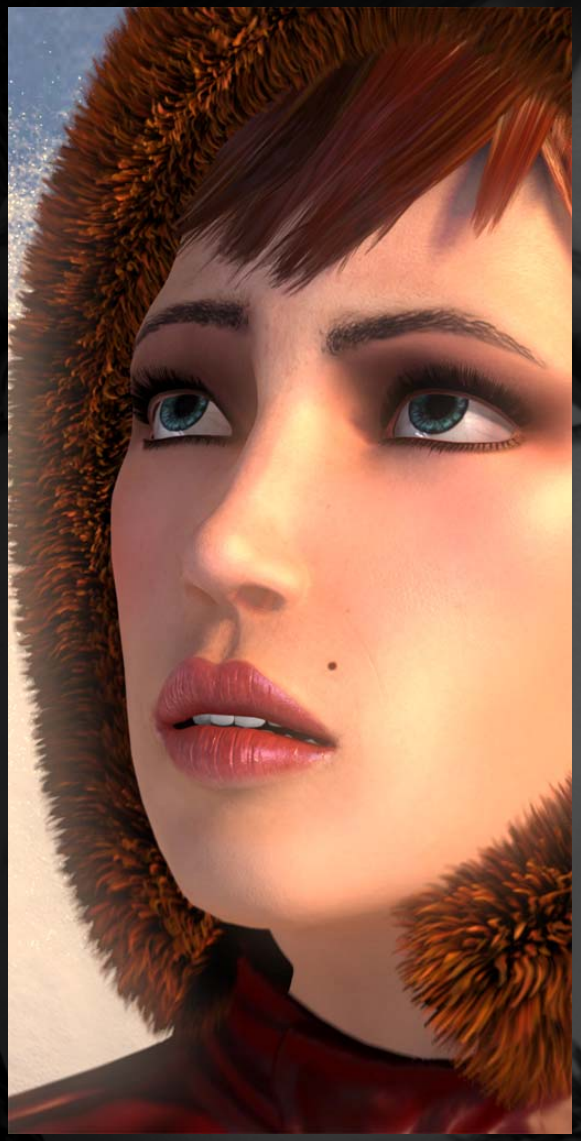
AMD Smarter Choice

- Static cube maps
  - Mountains
  - Transport Interior
  - Transport Exterior
  - Pilot's Visor
- Dynamic cube maps
  - Ruby
- Sun removed from HDR sky map and added back in as a direct light.
  - Extreme HDR lighting value painted down

Direct Lights

AMD
Smarter Choice

- Sun
- Glow from Pulse Explosion
- Rim Lighting on Ruby in HQ
- Glow coming off the HQ Console
- Supplemental lighting

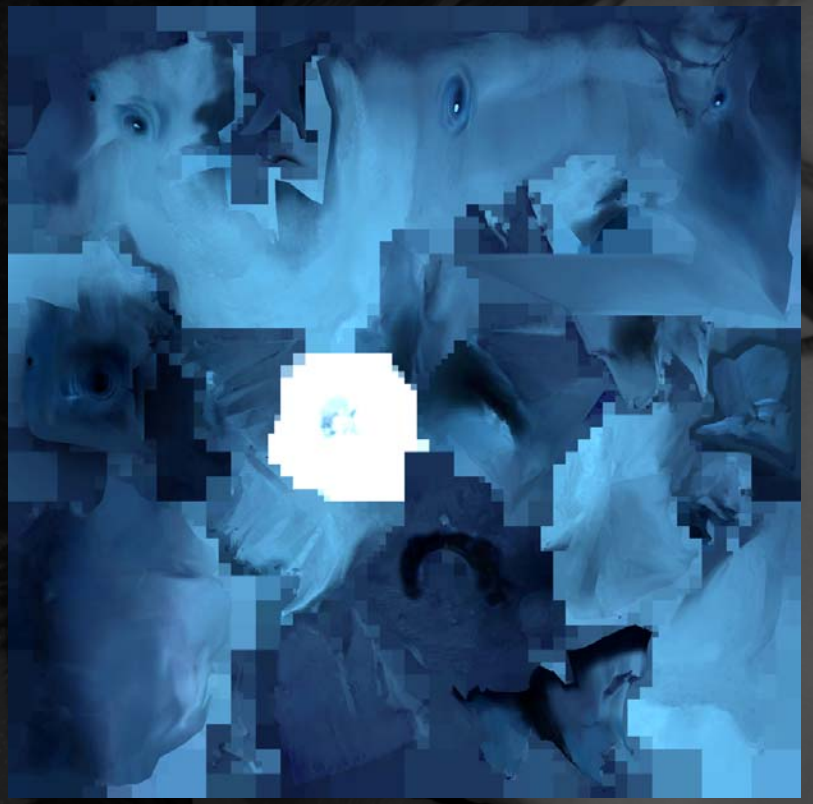


Shadow Light

- Dynamic depth mapped shadows
- Shadow light manually placed and animated in scenes for optimal usage of shadow buffer resolution
- Rigged the shadow spotlight to a camera in Maya
- Easily visualize frustum and clipping planes

Lightmaps

- Advantages and disadvantages
- Entire HQ is light mapped
- Shape of the HQ environment did not lend itself well to lighting only from the sky map
- Lights Ruby via the PRT cube map
- Rendered offline so uses GI and SSS



PRT Light Cards

- Dynamic cube map is only 64x64 per face
- Problematic for very small light sources
  - Flicker
  - Complete dropout
- Compensate with Light Cards
- Only render into the PRT cube map buffer

PRT Light Cards (cont.)

AMD
Smarter Choice

PRT Light Cards (cont.)

AMD
Smarter Choice

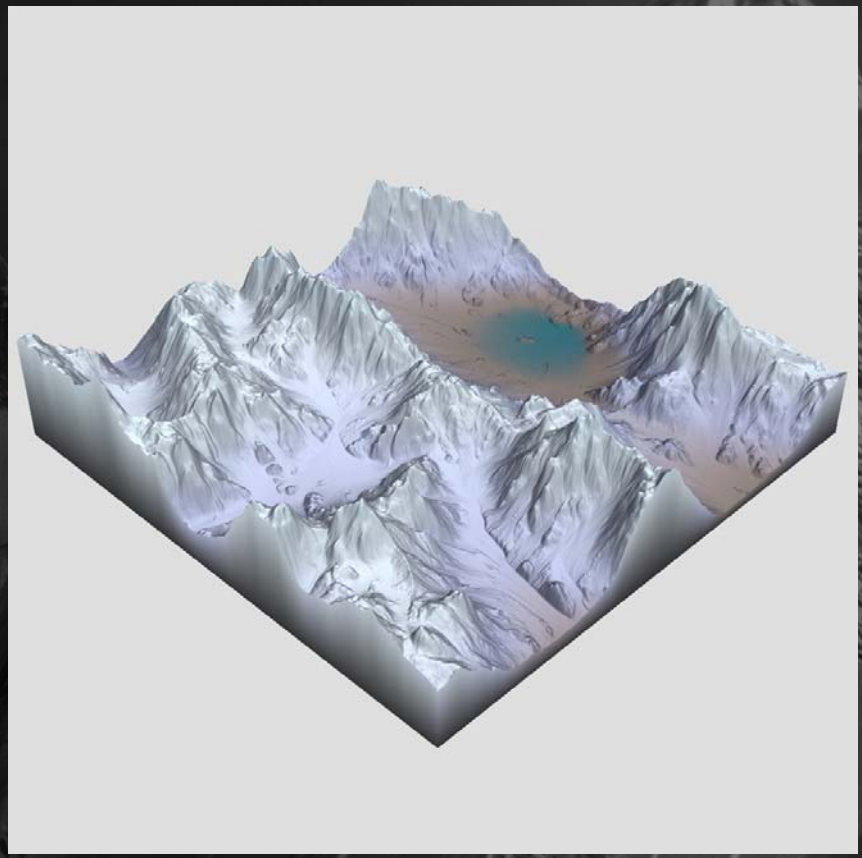
Environments

AMD Smarter Choice

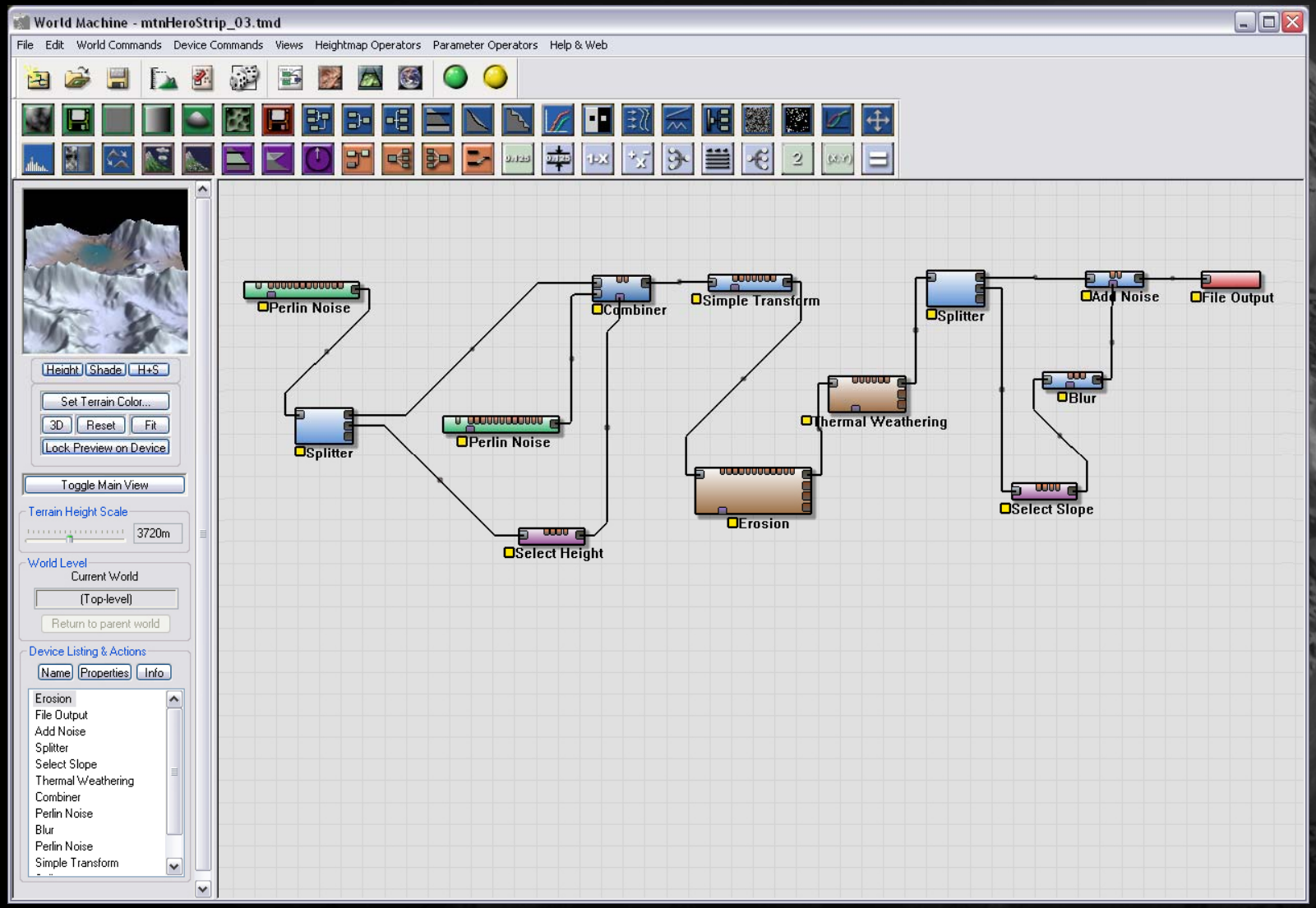
- Challenges
  - Expansive Scenes
  - Viewing Distances
  - Repetition
  - Memory Limitations
- Solutions
  - Procedural Authoring and Shading
- Examples
  - Mountains
  - Ice Cave

Mountain Creation

- World Machine
  - Procedural terrain tool
- Flow based interface
- Start with noise patterns
- Or import your own height map
- Outputs height map (up to 32 bit)
- Mountains, canyons, craters, etc.
- So I have a height map… now what?

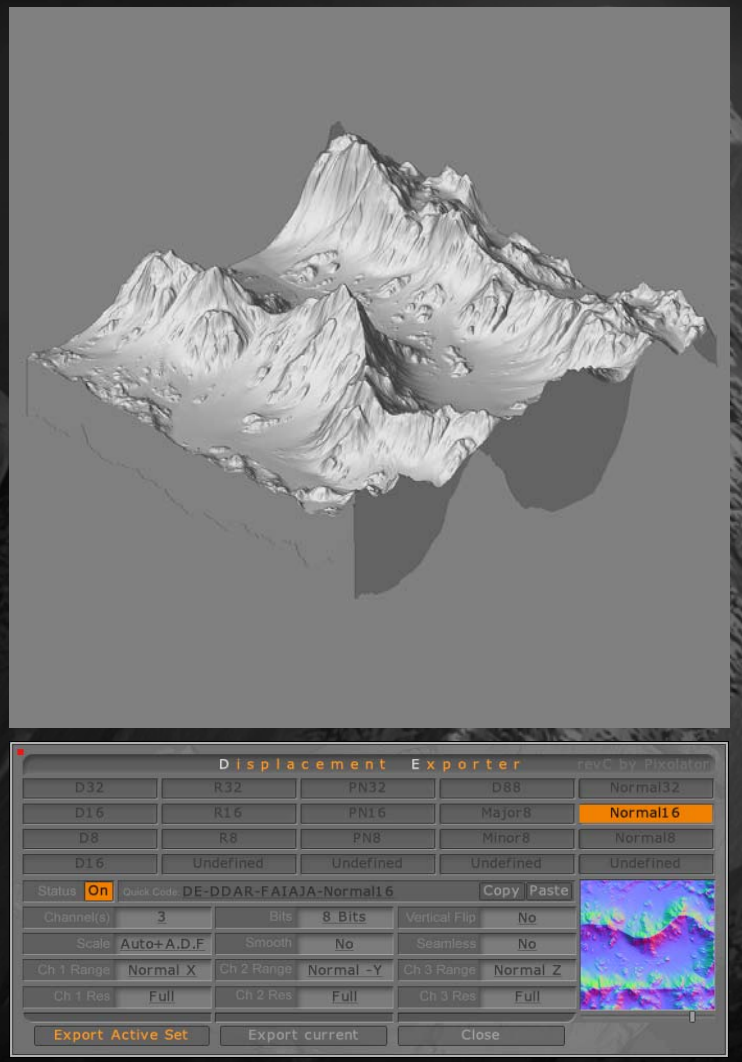


Mountain Creation



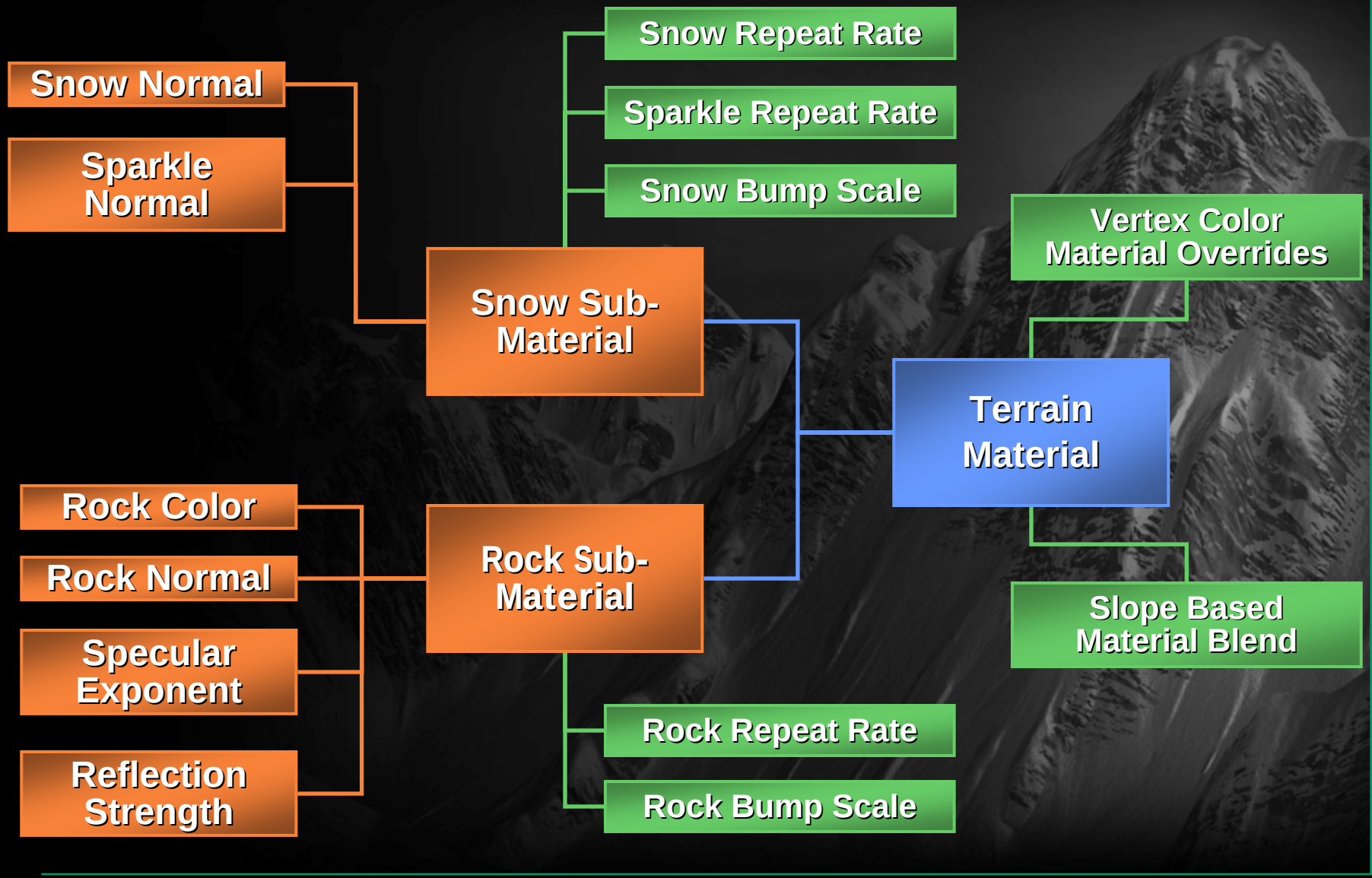
Mountain Creation (cont.)

- Import height map into ZBrush
- Subdivide and displace
- Used ADE plug-in to generate normal map
- Output high polygon mesh to OBJ
- Optimize mesh for use in-engine
- Only one UV set



Main Mountain Shader

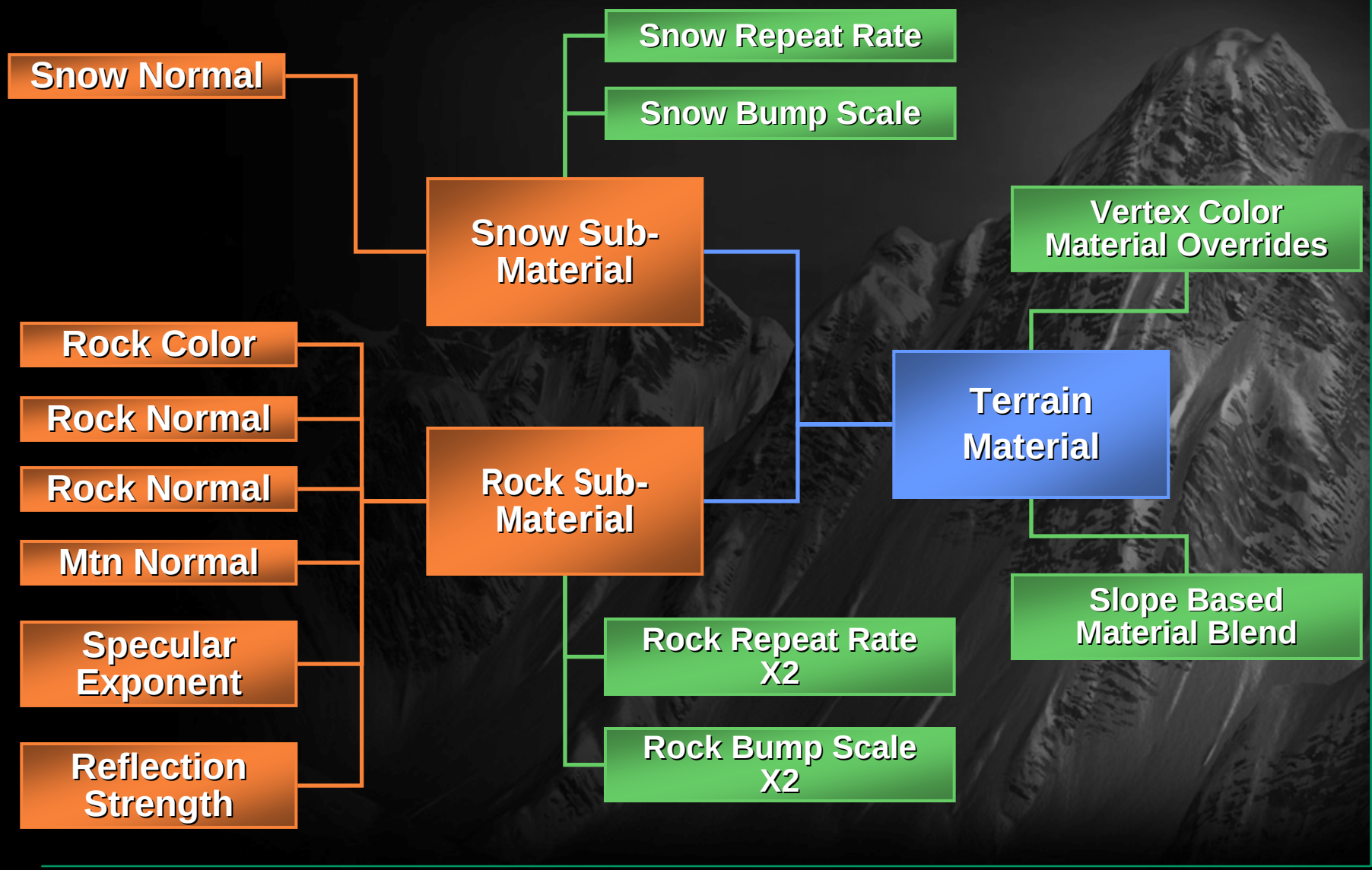
Main Mountain Shader



Background Mountain Shader

AMD
Smarter Choice

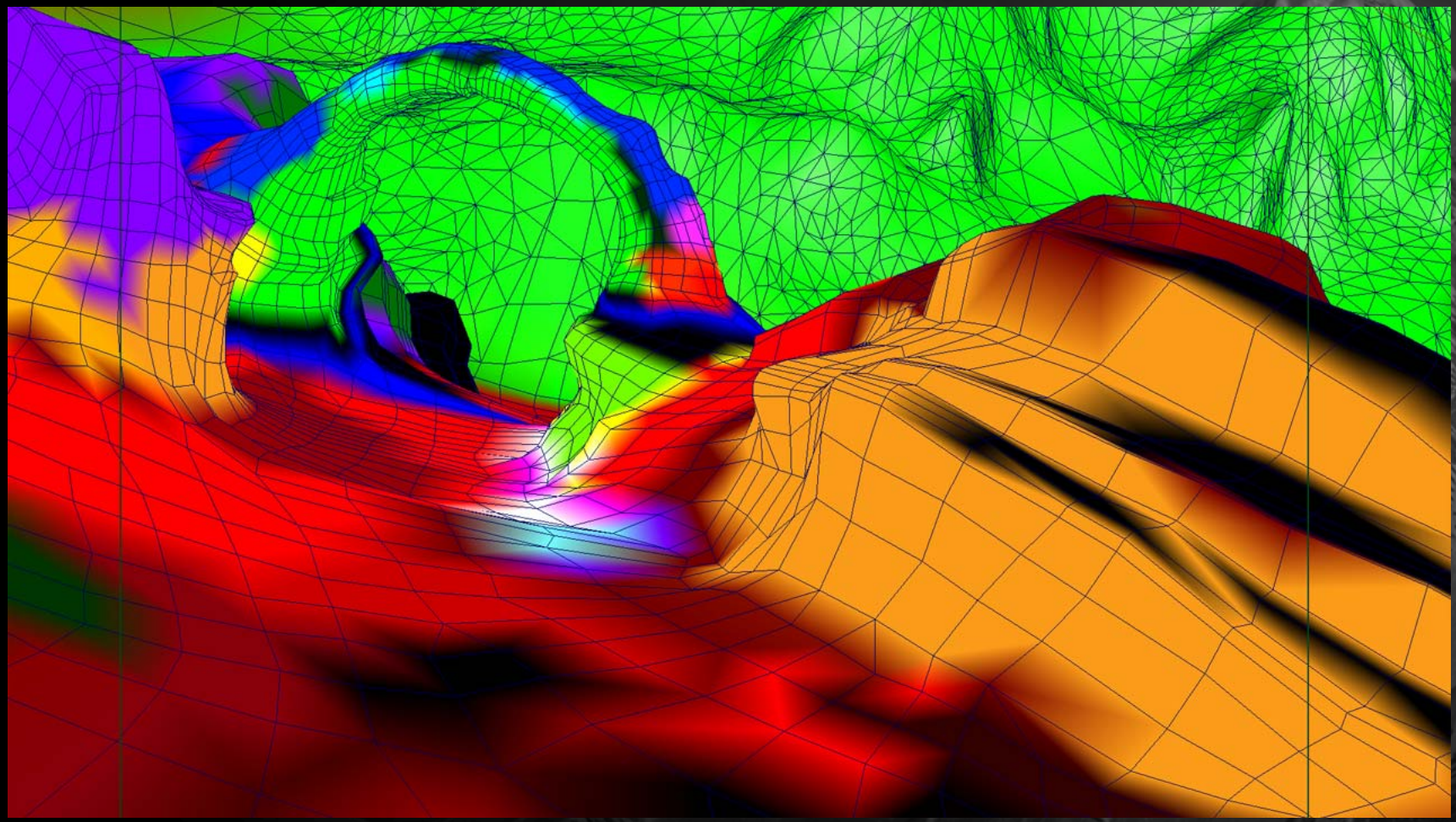
Background Mountain Shader



Vertex Color Setup

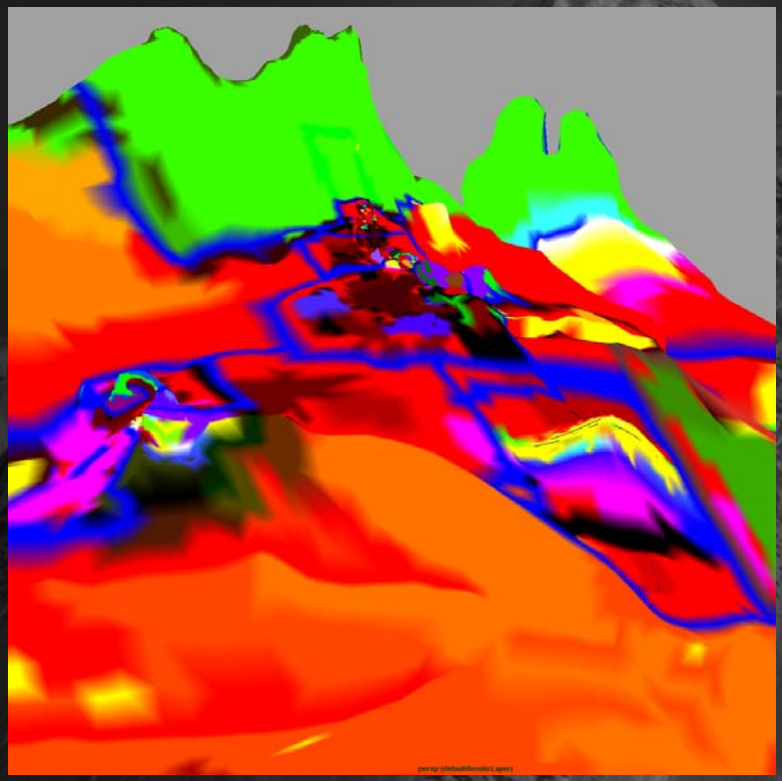
Vertex Color Setup

AMD
Smarter Choice



Vertex Color Setup

- Red = Force Bumpy Snow
- Green = Force Bumpy Rock
- Blue = Force Smooth Snow



- Overrides programmable slope based blend
- Hides UV seams
- Allows for blending of shader variations


AMD
Smarter Choice
DEMO
Ruby: Whiteout
Snow Accumulation Demo
AMD

Ice

- Challenges
  - Highly reflective
  - Refracts light
  - Subsurface scattering
  - Perceived depth
  - Needs to integrate with snow

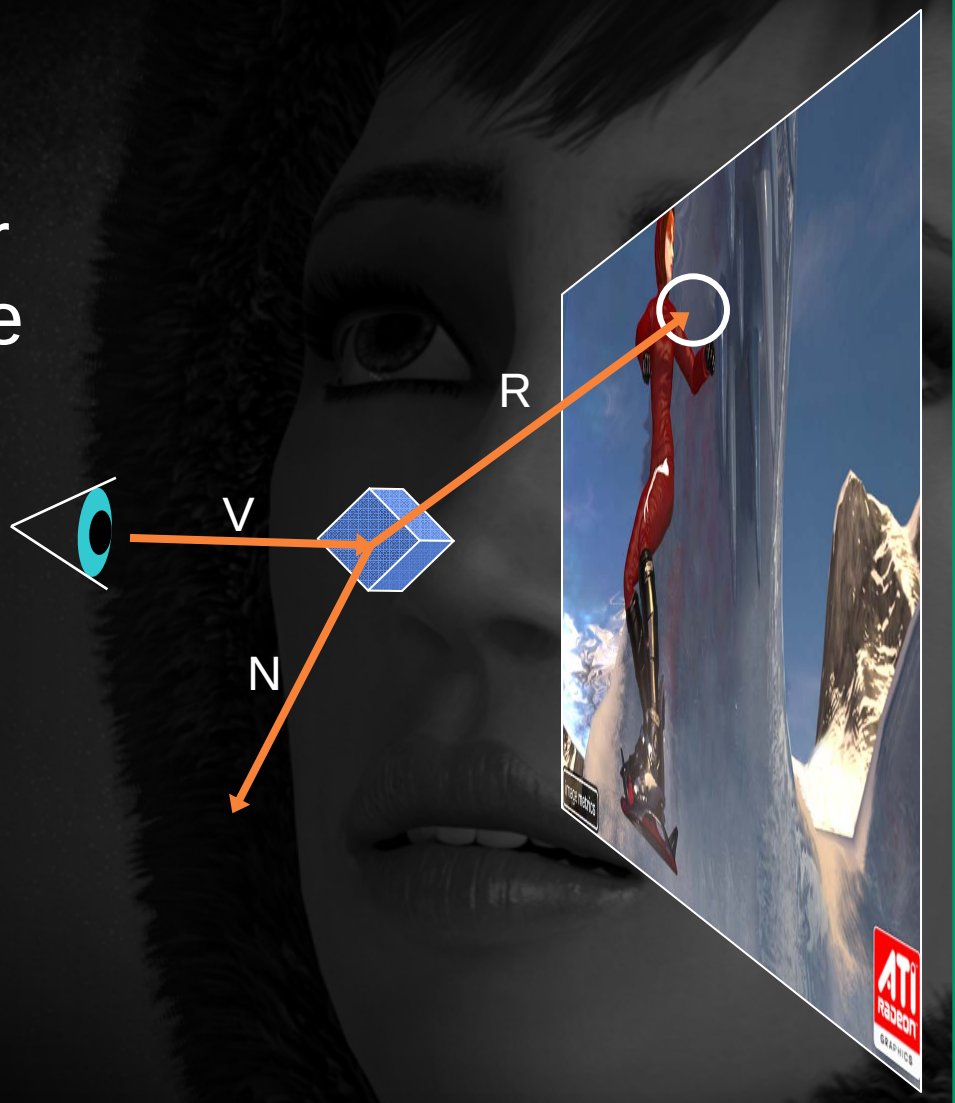
Reflecting Ruby

AMD Smarter Choice

- Ruby's reflection in the ice is dynamically ray traced
- By "Ruby" we mean her bounding box
- Calculate reflection vector for each ice pixel
- 12 rays (samples) cast per pixel in a cone arrangement
- Lerp in red based on the number of samples that intersect the box

Refracting Ruby

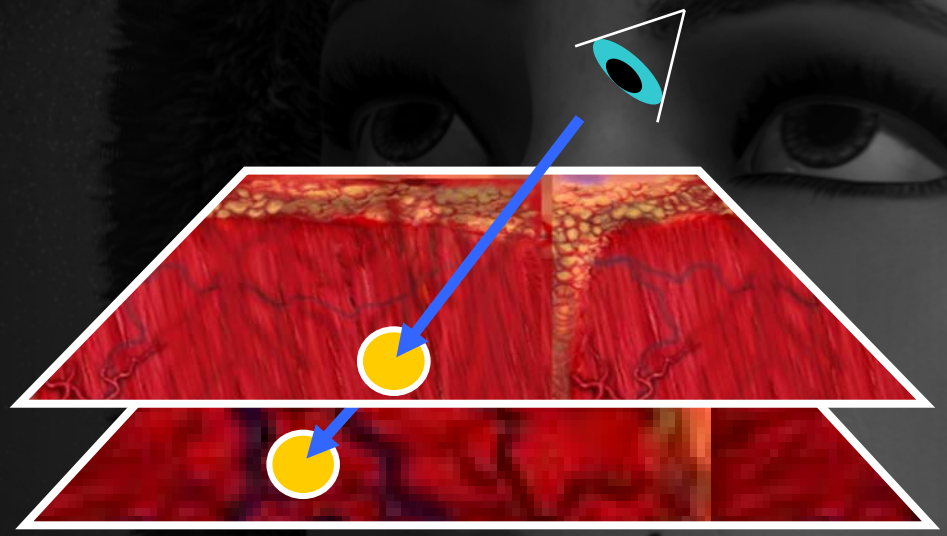
- Take a copy of the backbuffer before rendering icicle layer and stuff into texture in memory
- Calculate refraction angle based on surface normal
- Use refraction angle to offset texture sample in UV space



Parallax… the Illusion of Depth

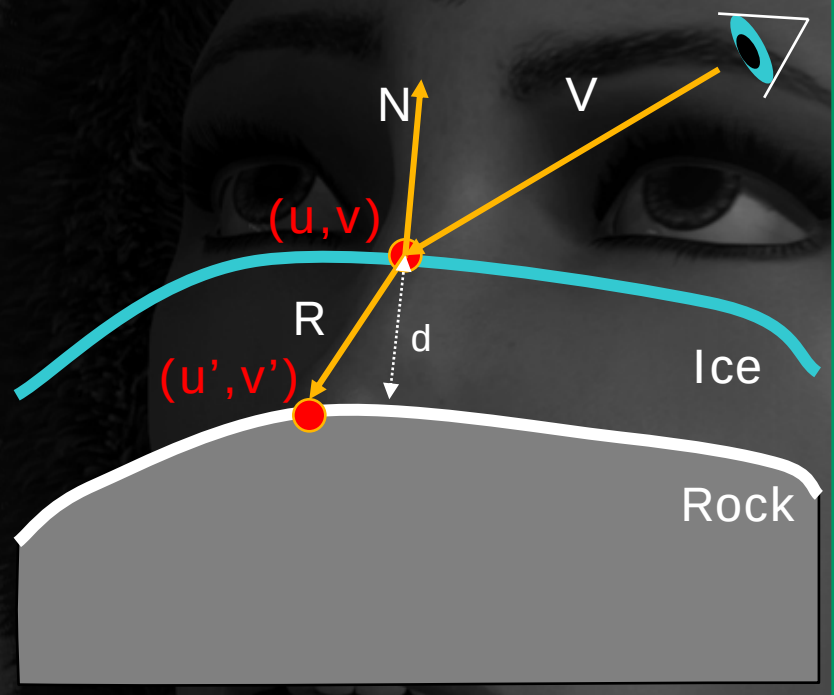
AMD Smarter Choice

- Uses Chris Oat's multi-layer material technique
  - See SIGGRAPH 2006 course notes
- Added refraction
- Use offset texture coordinates to sample from inner layer's texture



Depth and Refraction

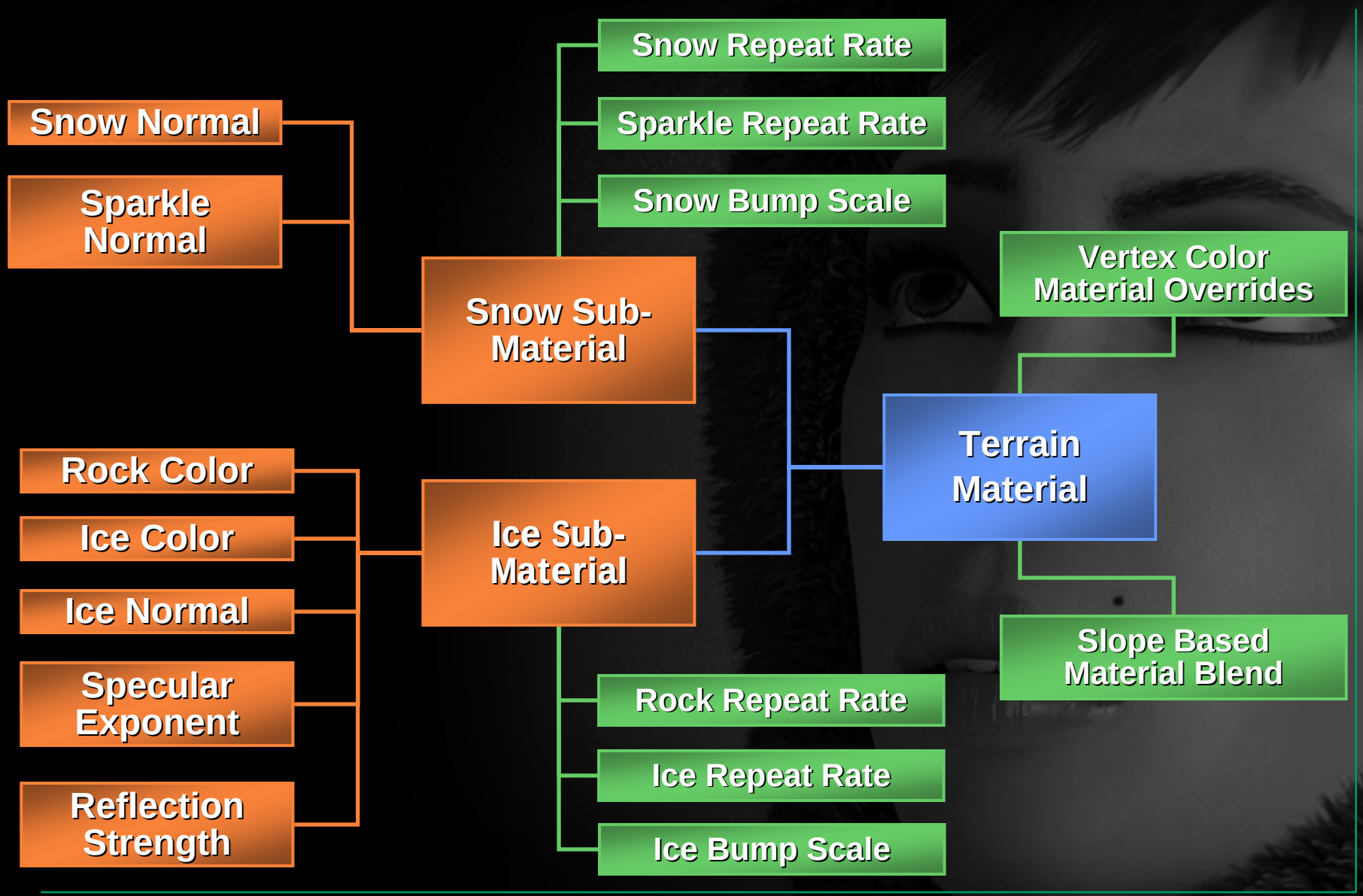
- Eye ray V intersects ice layer at texcoord (u,v)
- Shade ice layer normally
- In tangent space:
  - Compute refracted eye ray R
  - Compute intersection with rock layer, assuming constant ice thickness d
  - Convert to new texcoord (u', v')
- Shade rock layer with (u', v') and blend with ice



Ice Shader

AMD
Smarter Choice

Ice Shader


Snow Repeat Rate
Sparkle Repeat Rate
Snow Bump Scale
Snow Normal
Sparkle Normal
Snow Sub-Material
Vertex Color Material Overrides
Terrain Material
Rock Color
Ice Color
Ice Normal
Specular Exponent
Reflection Strength
Ice Sub-Material
Slope Based Material Blend
Rock Repeat Rate
Ice Repeat Rate
Ice Bump Scale

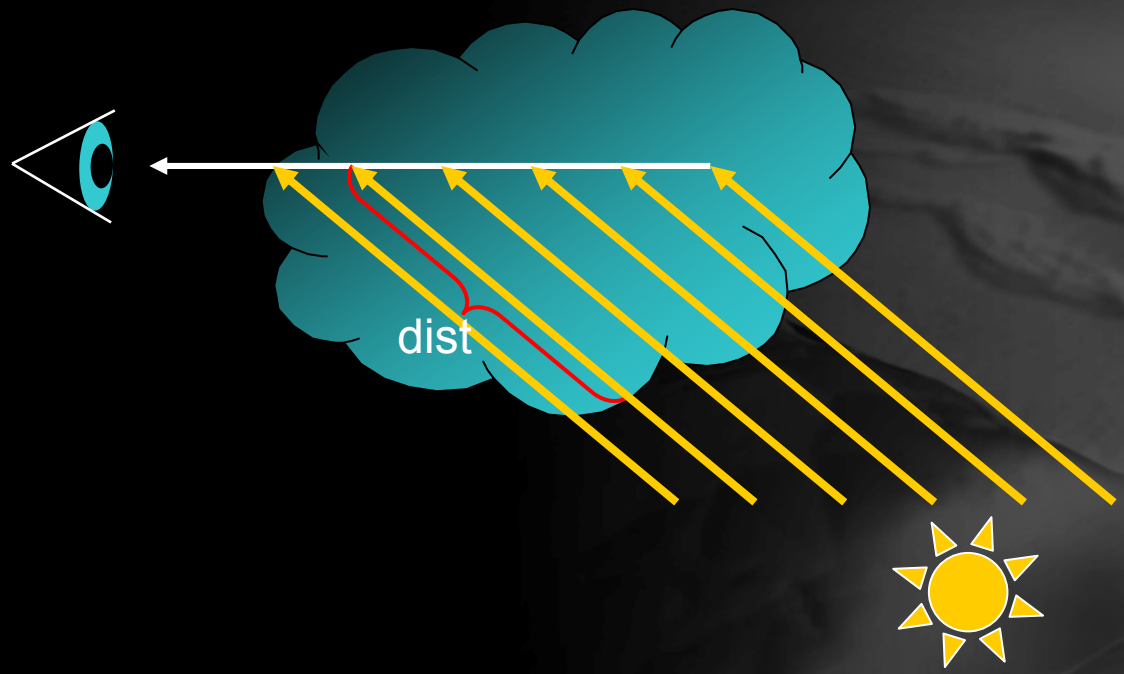

AMD
Smarter Choice
DEMO
Ruby: Whiteout
Ice Demo
AMD

Particle and Hair Self-Shadowing

Particle and Hair Self-Shadowing

Particle and Hair Self-Shadowing

Volumetric Lighting (Simplified)



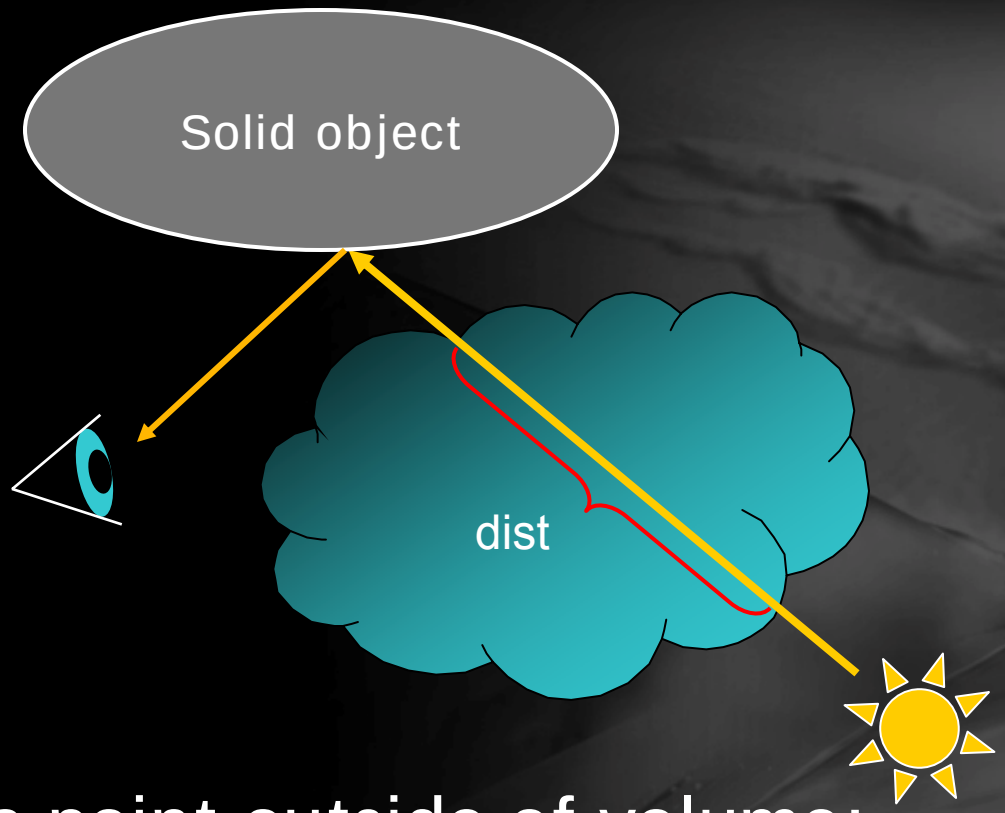
- Light enters volume and is attenuated
- Scatters towards viewer

$$attenuation = e^{-\int_{ray} density(p)dp}$$

Assuming uniform density:

$$\approx e^{-dist \cdot density_{avg}}$$

Volume Shadows



- Shadowing a point outside of volume:
- Need to know volume thickness
  - Entry- and exit point distance from light source

Particle and Hair Self-Shadowing

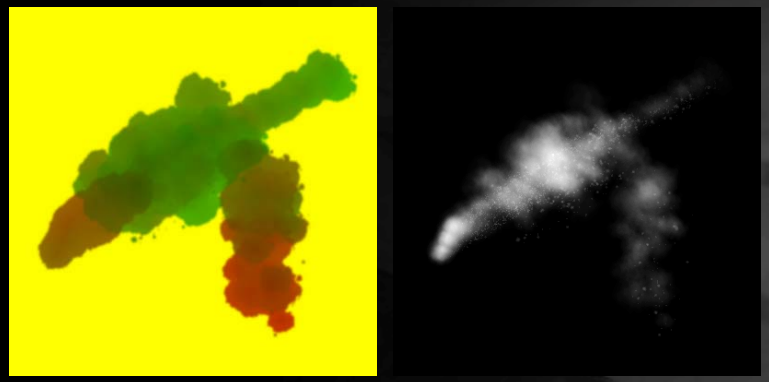
AMD Smarter Choice

- Fast thickness estimate of particle volume
- Thickness = Max Z – Min Z
  - Two depth buffers
  - From light's point of view
- Generate both depth buffers in a single pass
- Compute average volume density for a ray:
  - Additive blending of particle density/opacity
  - Divide by thickness
- Also works for mesh data
  - no need for closed volume (e.g. Ruby hair geometry)

Implementation

- Render min depth and max depth from light's POV in a single pass
- Pixel shader output:
  - Z in R
  - 1-Z in G
  - particle opacity in A
- RGB blend func: MIN
- Alpha blend func: ADD

Visual Breakdown



Density

G

R

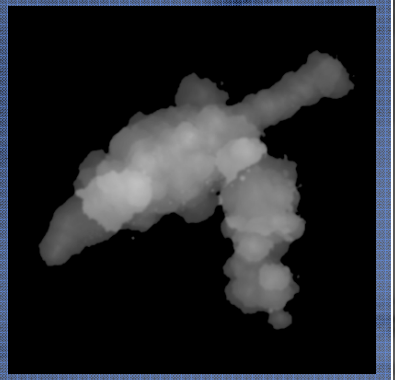
Min 1-Z

Max Z

-

Min Z

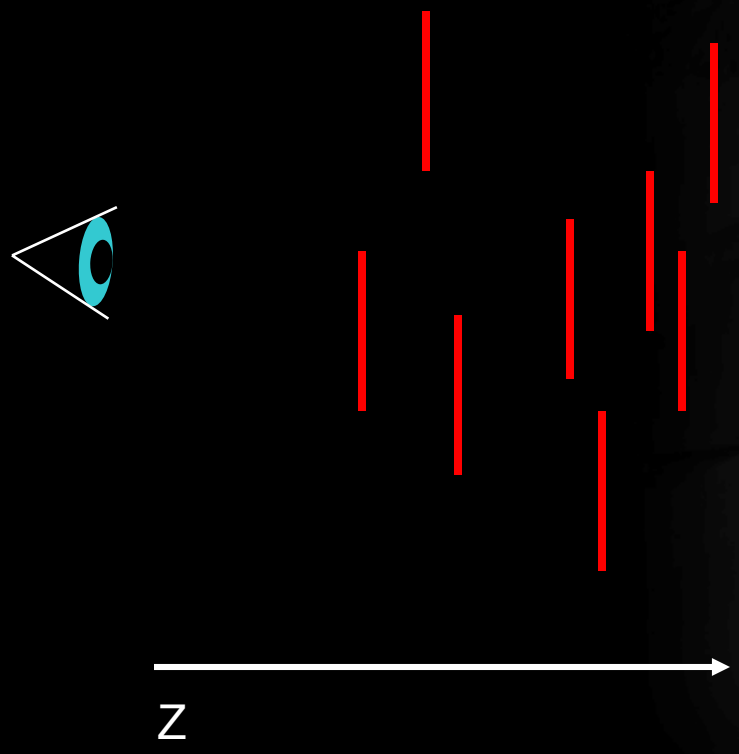
=



Thickness

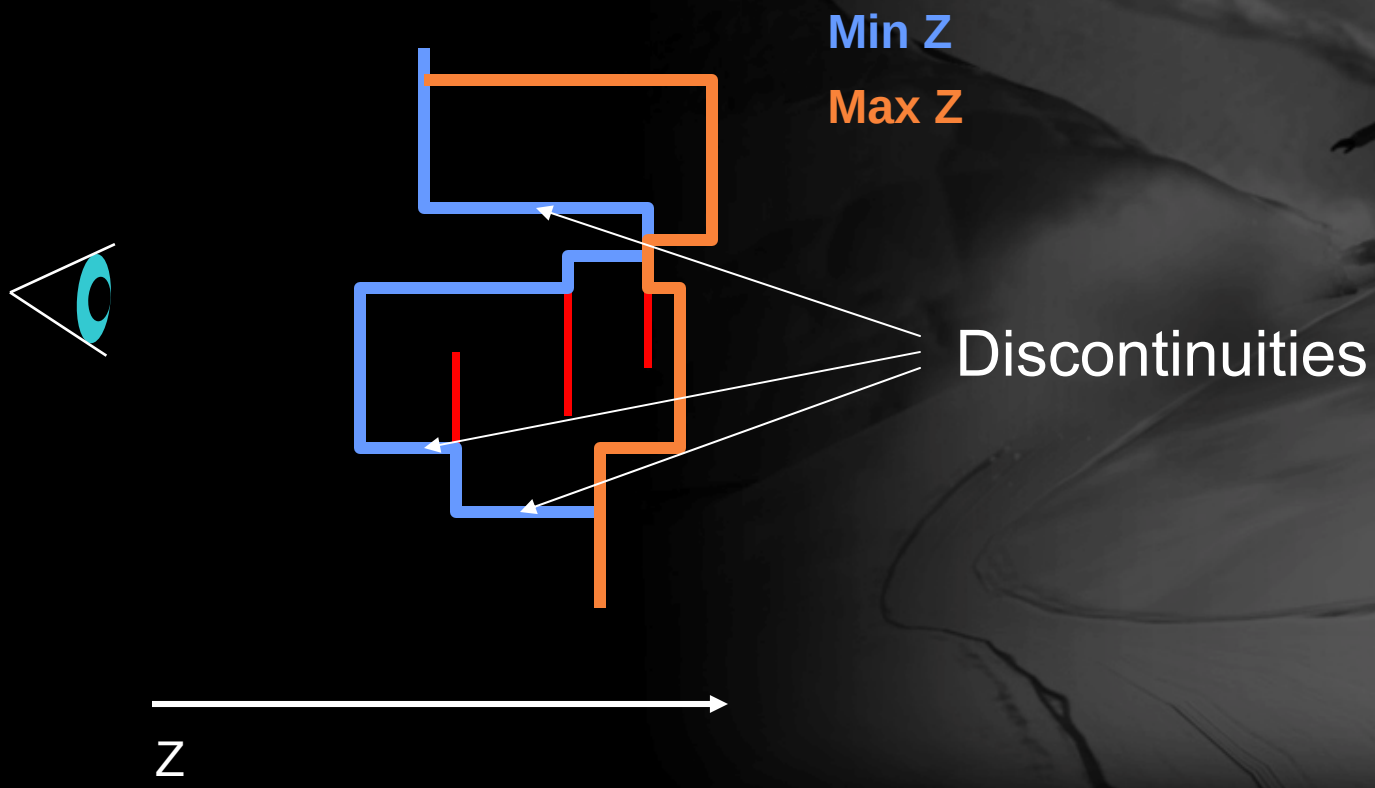
Particle Depth

- Particles are textured quads: (side view)



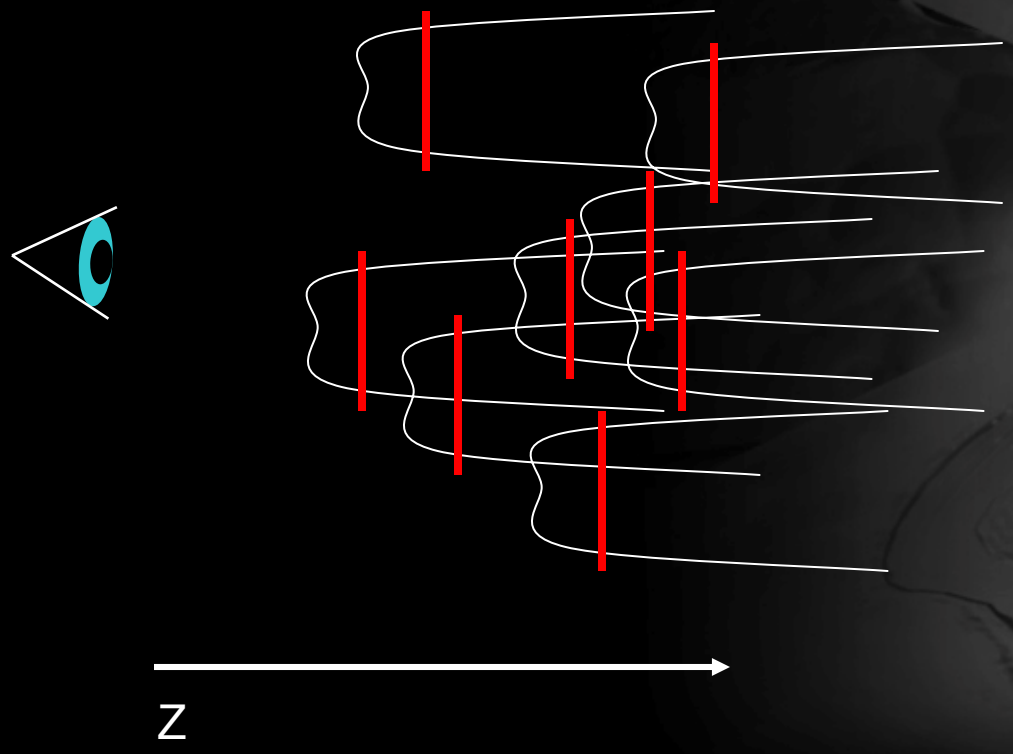
Particle Depth

- Discontinuities cause lighting artifacts



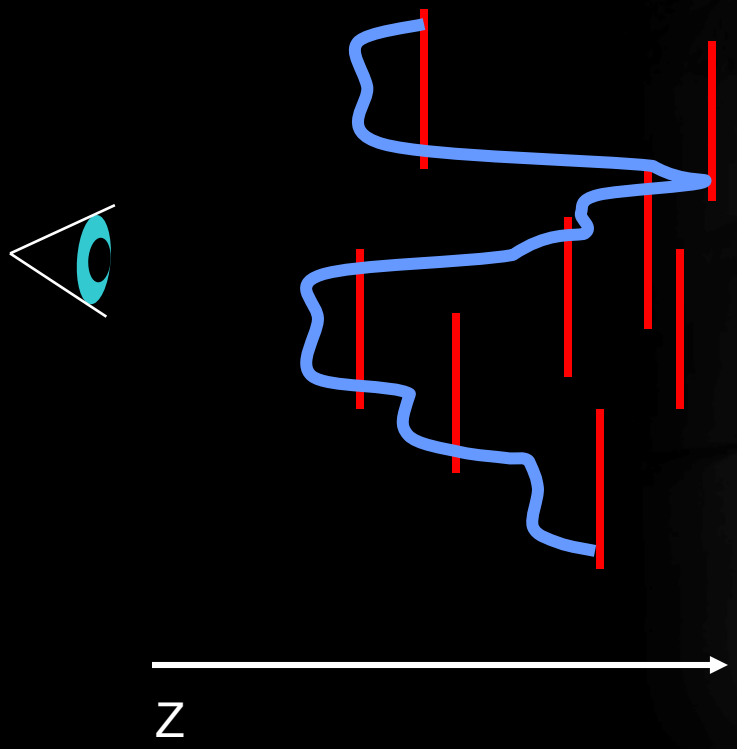
Particle Depth

- Adding depth offset maps for Min Z



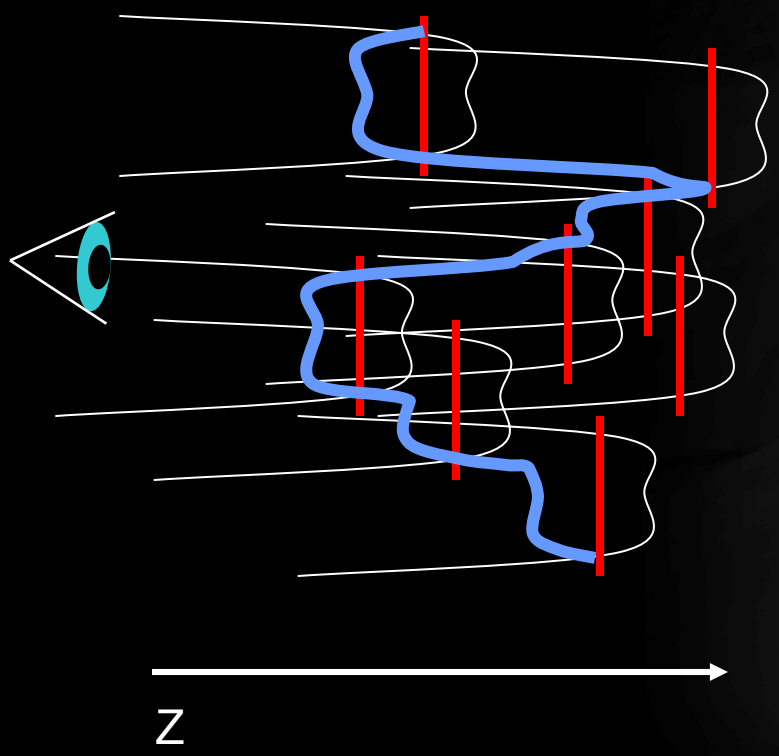
Particle Depth: Smoothing Z

- Add displacement/depth offset map
- Smoothed, no bad discontinuities



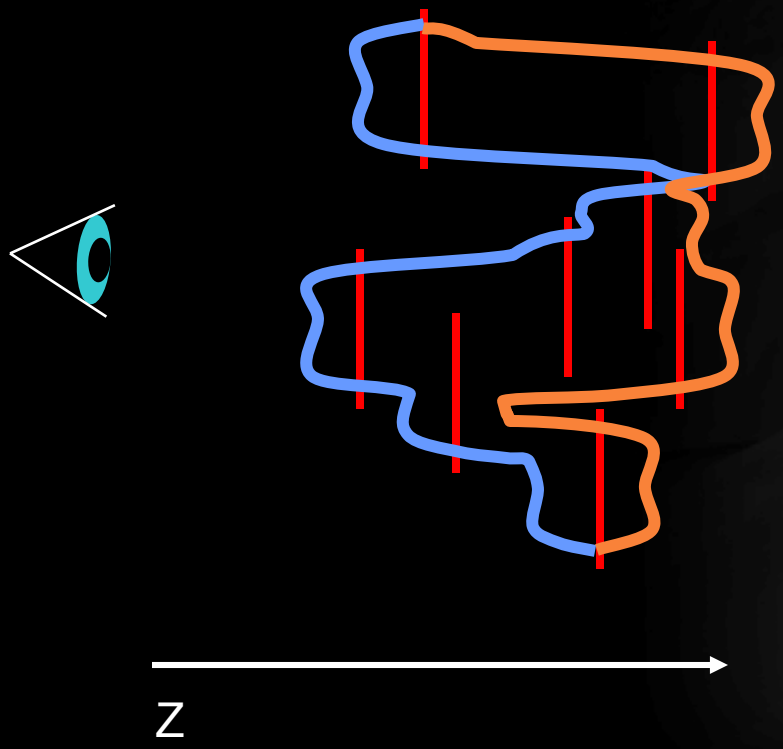
Particle Depth

- Adding depth offset maps for Max Z



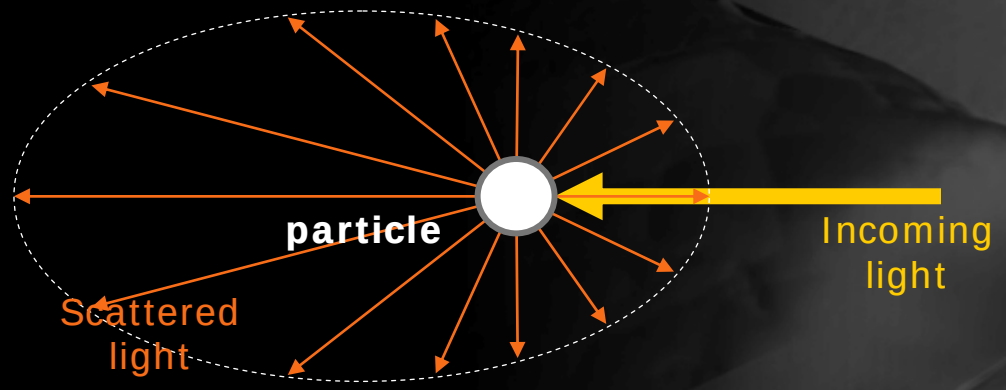
Particle Depth

- Smooth min and max Z
- Additional blur of depth buffer



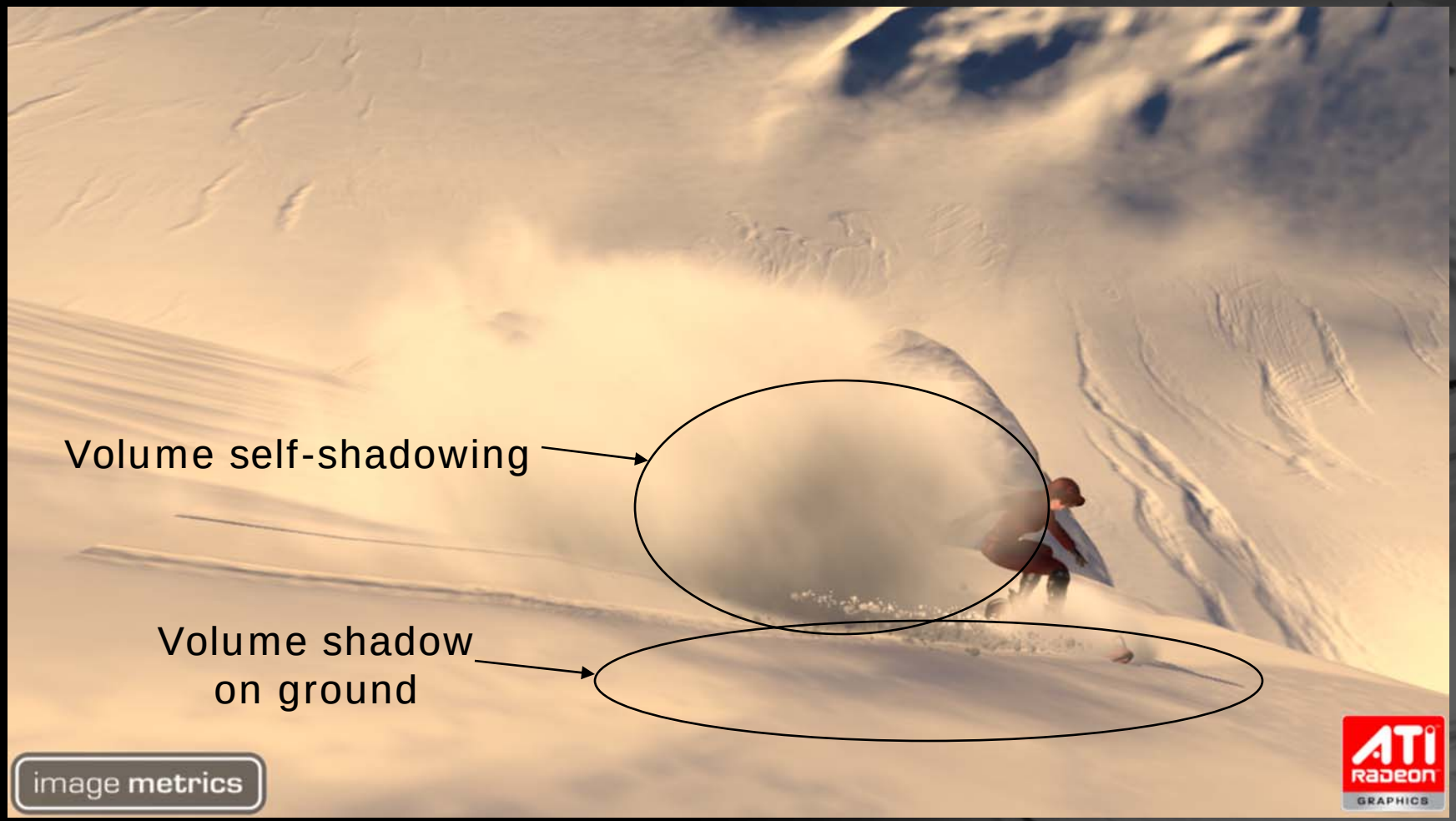
Particle Shading

- Direct light: Henyey-Greenstein phase function to simulate Mie scattering
  - Forward scattering = Nice rim lights in backlit situations



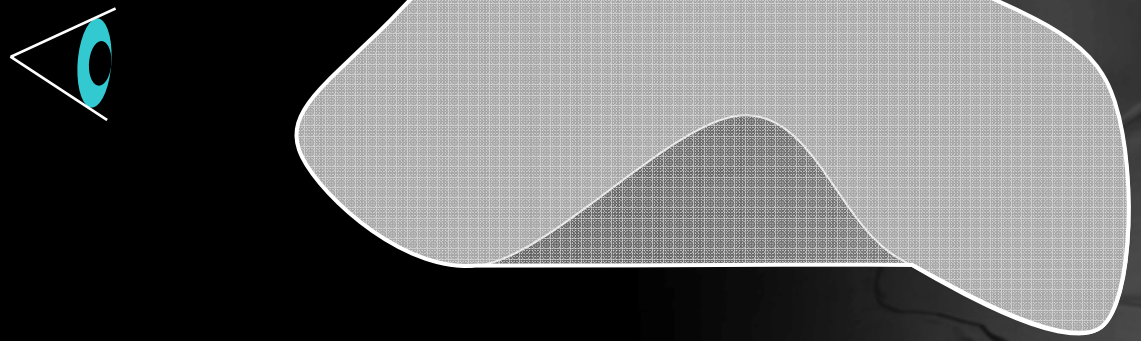
- Add per-vertex SH lighting environment lookup as ambient term

Example


Volume self-shadowing
Volume shadow
on ground
image metrics
ATI
RADEON
GRAPHICS

Limitations

- Complex shapes aren't represented correctly
- Assumes uniform volume density per ray
- …but works well in practice!



More Particle Tricks

AMD Smarter Choice

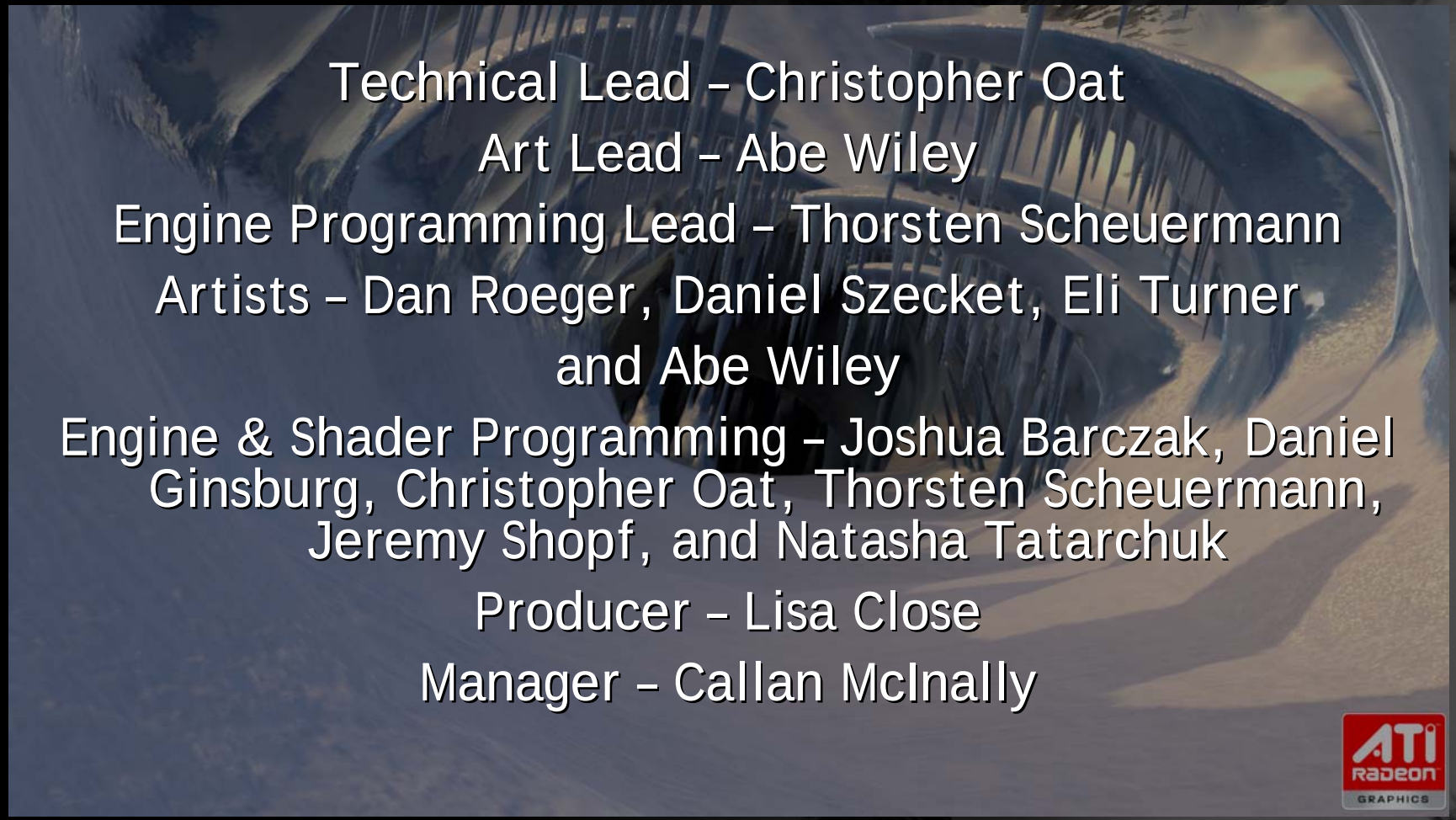
- Soft particles
  - Fade particle alpha based on `saturate(scale*(z_buffer_depth - particle_depth))`
- Shadowing from “normal” shadow map (not self-shadowing)
  - High overdraw of particle systems make filtered shadow map lookups very expensive
- Just sample shadow map *per vertex* for corners of particle quads
  - Saves a ton of texture fetches
  - Get “soft shadows” for free from vertex attribute interpolation

Clothing Wrinkles

- DX10 Geometry Shader gives you access to vertex info
- Compare size of the deformed triangle to original bind pose size
- Calculate “Wrinkle Weight”… average change in size per vertex
- If smaller, blend in wrinkle map
- Blends per vertex for smooth blending


AMD
Smarter Choice
DEMO
Clothing Wrinkle Demo
AMD

The Team



Technical Lead – Christopher Oat

Art Lead – Abe Wiley

Engine Programming Lead – Thorsten Scheuermann

Artists – Dan Roeger, Daniel Szecket, Eli Turner and Abe Wiley

Engine & Shader Programming – Joshua Barczak, Daniel Ginsburg, Christopher Oat, Thorsten Scheuermann, Jeremy Shopf, and Natasha Tatarchuk

Producer – Lisa Close

Manager – Callan McInally

Questions?

http://ati.amd.com/developer

abe.wiley@amd.com
thorsten.scheuermann@amd.com