SIGGRAPH2005

SIGGRAPH2005

Filtering Cubemaps

Angular Extent Filtering and Edge Seam Fixup Methods

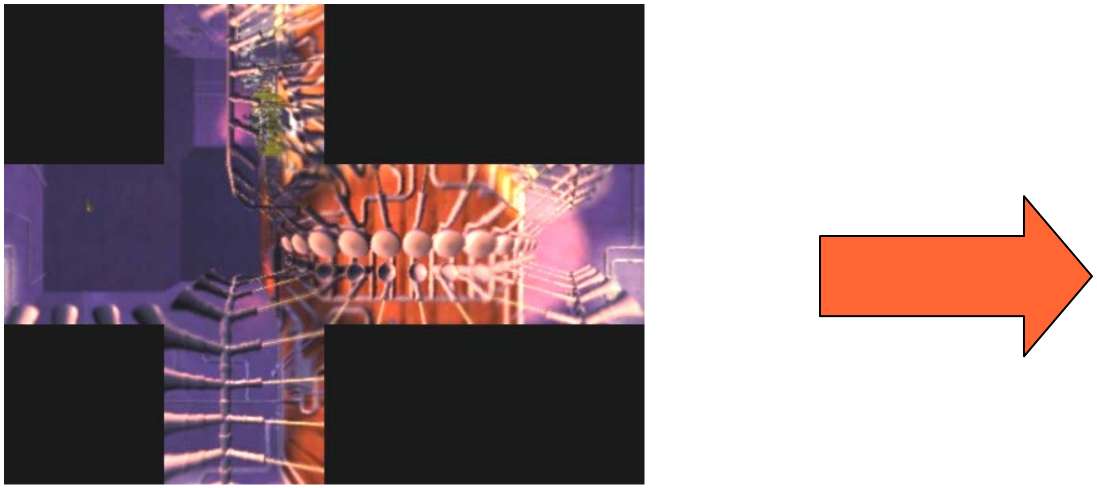
John R. Isidoro

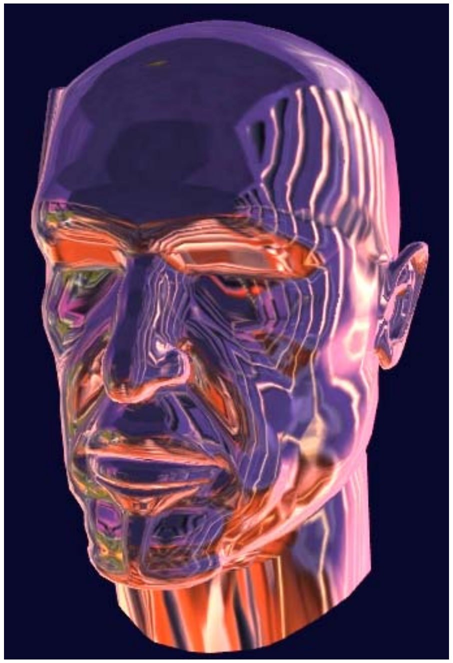
3D Application Research Group

ATI Research

Introduction

SIGGRAPH2005





- Hardware cube mapping is ubiquitous.
  – Straightforward hardware implementation, fast!
  – (Unnormalized) direction vector used directly to fetch from texture.
    - Fewer ALU ops than other environment map parameterizations.
  – Single texture fetch for any environment map direction.
  – Solid angle subtended by texels doesn't vary as much as it does in other single fetch parameterizations such as spherical maps, lat-long maps, or angular maps.

Motivation

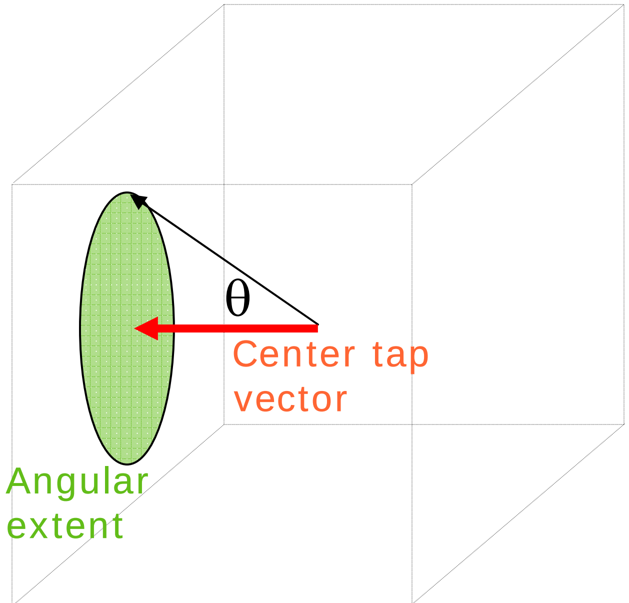
- However, two main issues remain troublesome for cube map users.
  - Virtually no current graphics hardware provides bi/tri linear filtering across cube faces directly.
    - Visible cubemap face edge seams in many cases. (miplevels)
    - Complaints about this from quite a few people.
    - Expensive to do correctly in hardware. Per-face texture borders are another possibility but are rarely supported.
  - Current approaches treat each cube face as a flat 2D texture for filtering.
    - It would be useful to take into account the varying solid angle of cube map texels across a face.

Overview

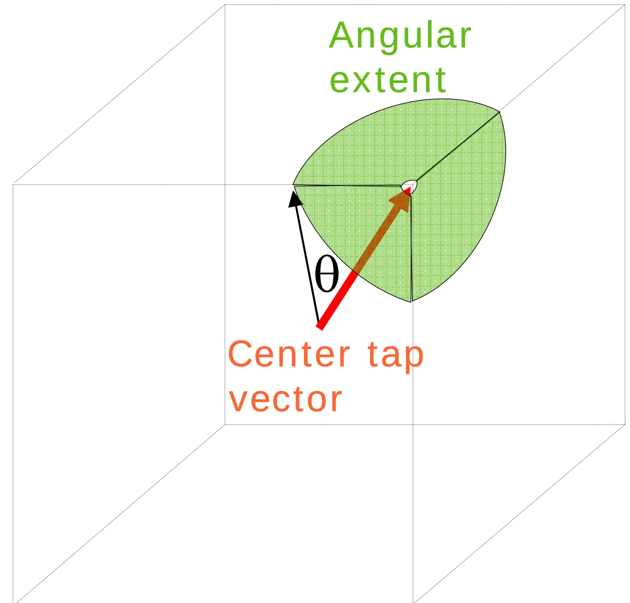
- The techniques described in this sketch **Angular Extent Filtering** and **Edge Seam Fixup** attempt to address these issues.
- These methods are used for preprocessing and mipchain generation for existing static cubemaps.

Angular Extent Filtering (AEF)

SIGGRAPH2005



Filtering within Face



Filtering across faces

- Angular extent filtering processes all taps within a given angle of the 3D center tap vector.
  - Filter extent is not texel size dependent, rather filter extent has constant solid angle, (e.g. constant area when projected onto the unit sphere).
    - Filter may (and should) encompass a different number of texels for different center tap vectors.
  - This allows for filtering kernels that span across cube face edges.

Efficient angular extent filtering

- Precompute solid angle and direction vector lookup cubemap.
- For each output texel:
  - Determine bounding box regions for angular extent in each face on input cubemap.
  - Process all texels within each bounding box region.
    - Filter weights: lookup table indexed using dot product between tap vector and center tap vector.
    - Weight each texel using solid angle and weights of chosen filter function.

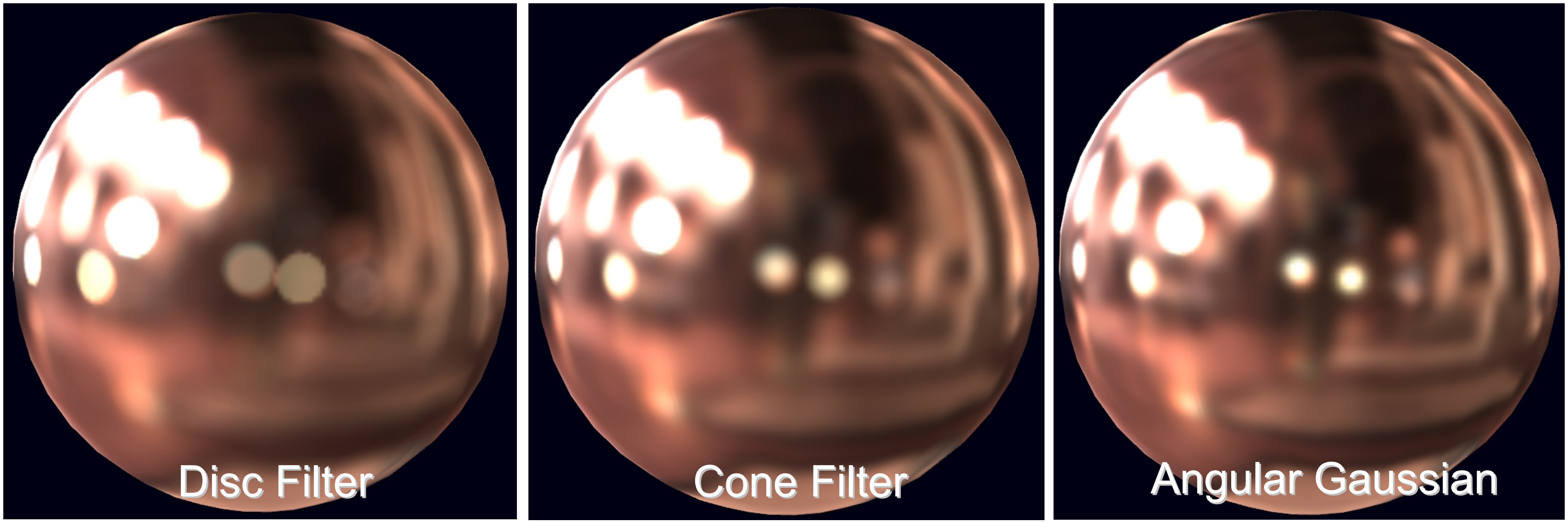
Advantages

SIGGRAPH2005

+ Every texel within angular extent is processed exactly once.
  - Very important for HDR imagery: Single very bright texels can significantly influence the filtered result.
+ Filter is circularly symmetric on the surface of the sphere.
  - Filter shape is independent of direction vector in spherical space.
  - The idea is applicable to other parameterizations as well,
  - The precomputed solid angle and direction vector map is only dependent on the resolution and the parameterization. Can be performed prior to filtering.

Angular Filter Types

SIGGRAPH2005



Examples of angular extent filtering:

- Disc Filter: All taps within a specified angle of the center tap weighted equally.
- Cone Filter: Linear falloff based on angle between tap and center tap.
- Angular Gaussian Filter: Gaussian falloff based on angle between tap and center tap.
  - 3 standard deviations within specified extent angle.

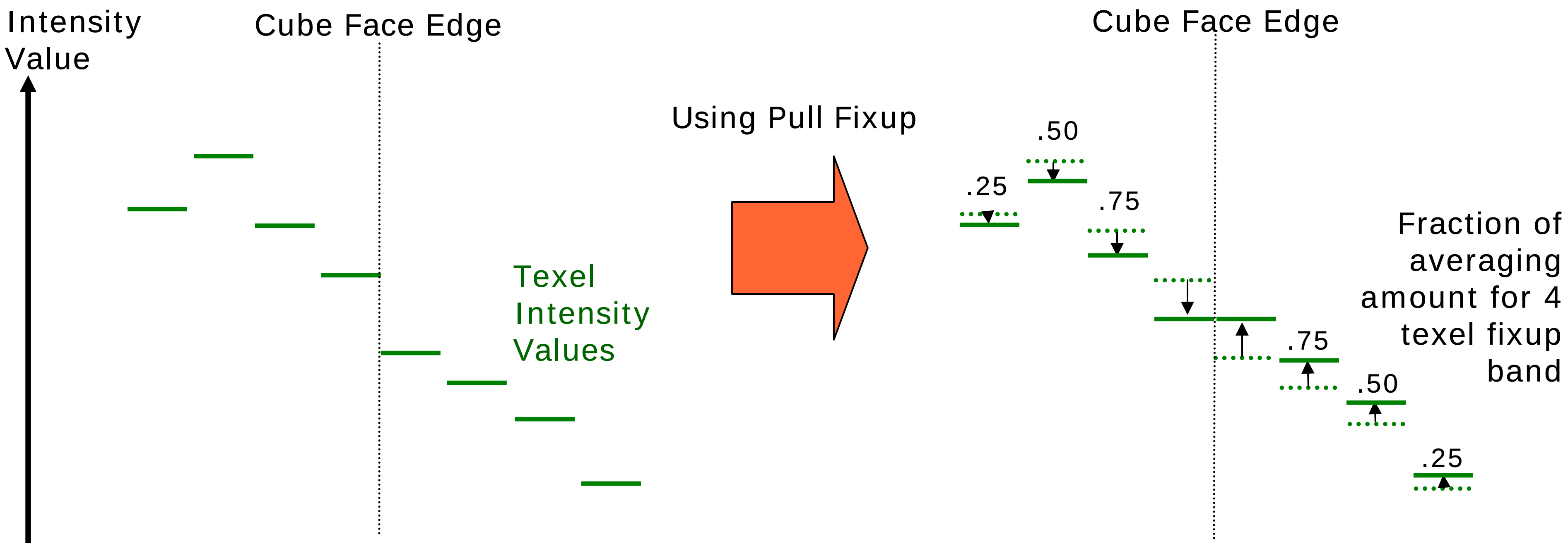
Edge Fixup (EF)

SIGGRAPH2005

- In addition to angular extent filtering, texels are averaged across edges, and corners after filtering.
  - Obscure seam artifacts from the HW not being able to filter across edges.
- To obscure the effects of the averaging, the averaging amount is blended into texels within a fixup region of the edge.
  - Two techniques to do this Pull Fixup and Smooth Fixup.

Edge Fixup: (Pull Fixup)

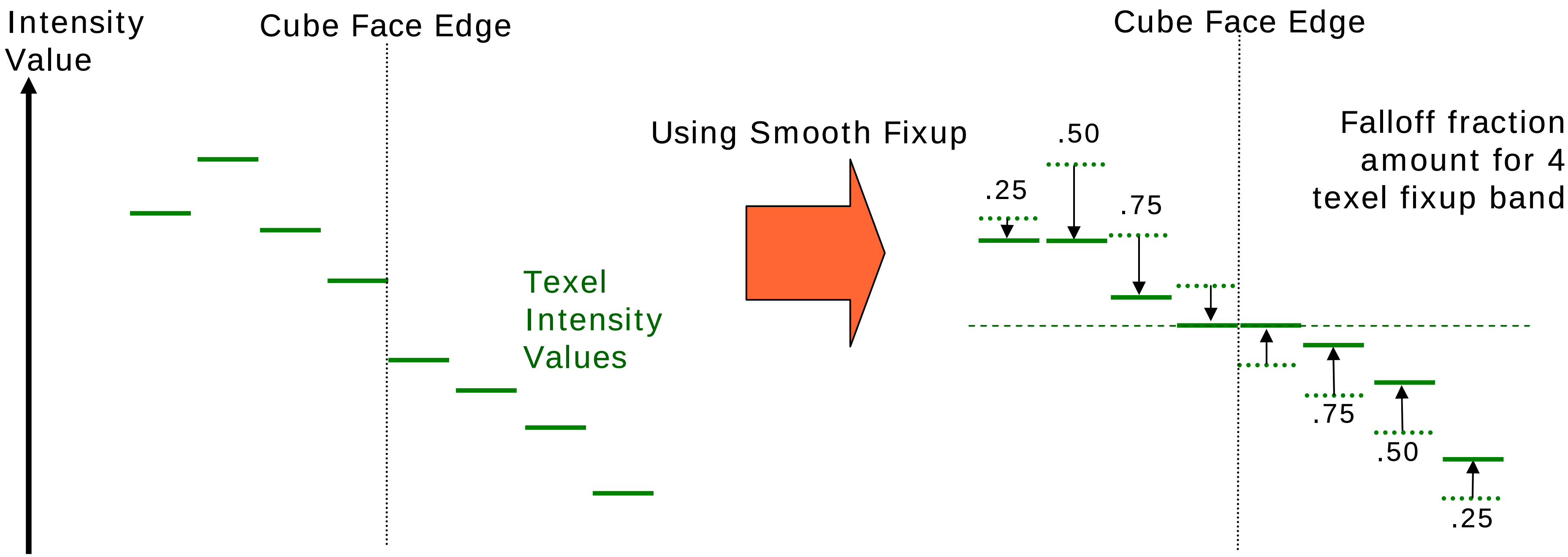
SIGGRAPH2005



- Pull Approach: looks at the **amount of intensity change** caused by averaging edge values.
- The intensity change is propagated and faded out over the texels within a few texel lengths of the edge.
  - Fade out is either linear or cubic.
- Preserves high frequency detail, while obscuring the hard seam.

Edge Fixup: (Smooth Fixup)

SIGGRAPH2005

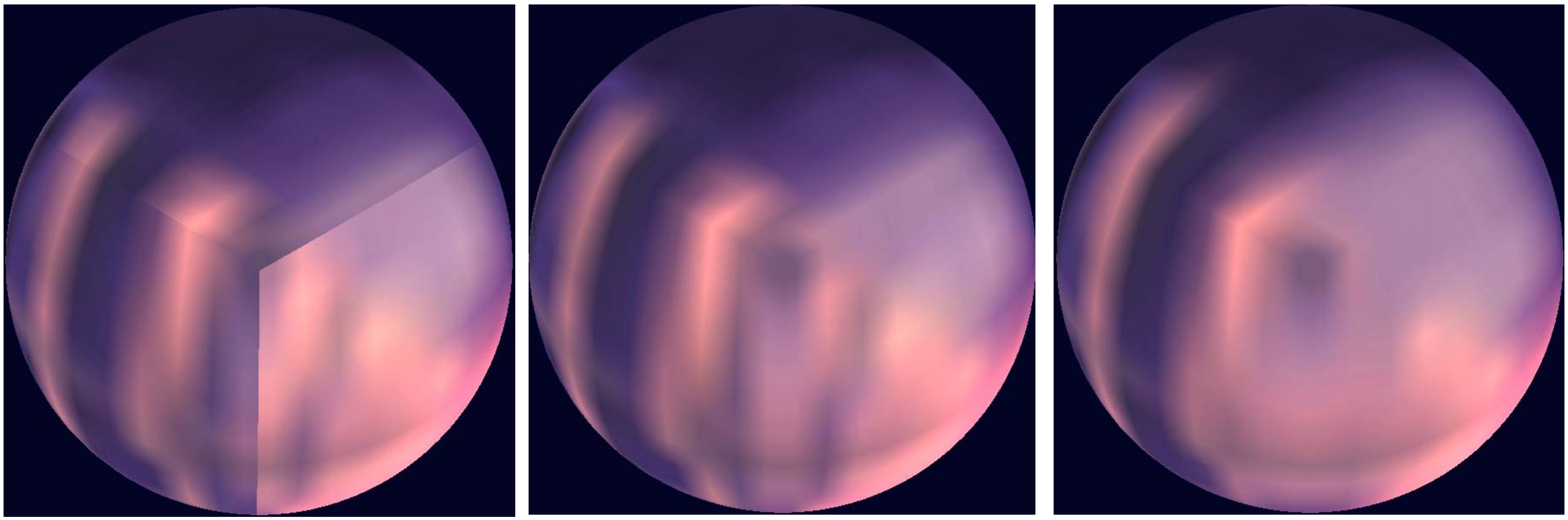


- Smooth Approach: looks at the **intensity value of the edge texel** caused by averaging edge values.
- The intensity value is averaged over the texels within a few texel lengths of the edge.
  - Fade out is either linear or cubic.
- Obscures the seam better, but loses high frequency detail.

Pull/Smooth Fixup Example

SIGGRAPH2005

16x16 miplevel, 4 texel fixup band

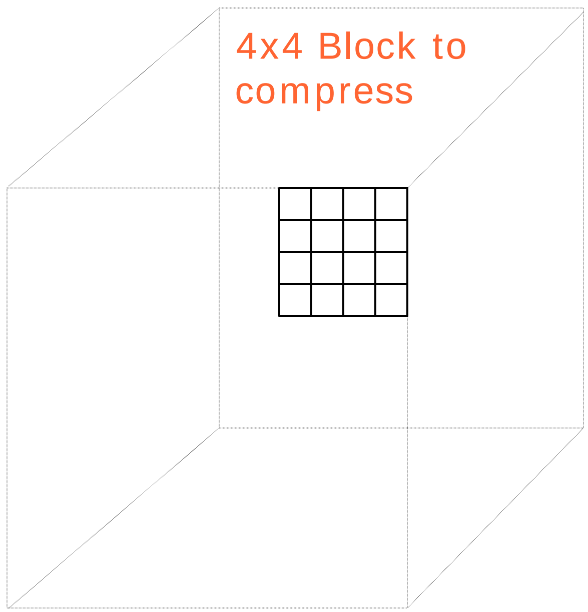


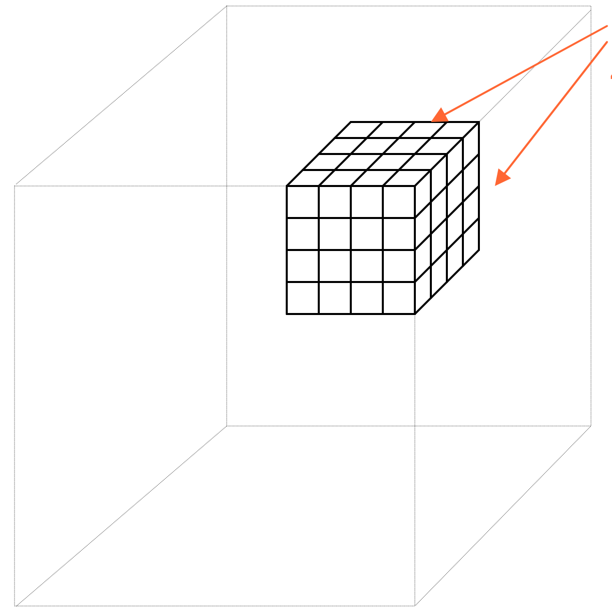
No Edge Fixup | Pull Fixup | Smooth Fixup

- Pull fixup preserves high frequency detail better.
- Smooth fixup obscures the edge seam better.

Using DXT Block Compression with edge fixup

SIGGRAPH2005





- DXT1 block compression encodes each 4x4 block of texels using two representative ramp colors and 2-bits per texel to linearly interpolate between them.
  - Each 4x4 block is independent of the others.
- To use edge fixup, estimate the ramp colors for a block using not only its own 4x4 neighborhood, but any 4x4 neighborhoods adjacent across a cubemap edge.
  - Since the ramp colors for across edge adjacent blocks are identical, the edge texel colors (post-compression) will be identical as well.

2x2 Miplevel (Standard)

- 2x2 mip-level without edge fixup and across face filtering
  - Strong edge artifacts.. 2x2 miplevel is unusable by itself .

2x2 Miplevel (AEF&EF)

SIGGRAPH2005

- AEF & EF allows for the 2x2 mip level to be used as a diffuse environment lighting term.

4x4 Miplevel (Standard)

SIGGRAPH2005

- Edge filtering artifacts make the 4x4 miplevel from standard mip filtering algorithms are only useful for mipmapping purposes.

4x4 Miplevel (AEF&EF)

SIGGRAPH2005

- AEF & EF makes cubemaps with a 4x4 miplevel that can also be used for rough metal shaders.

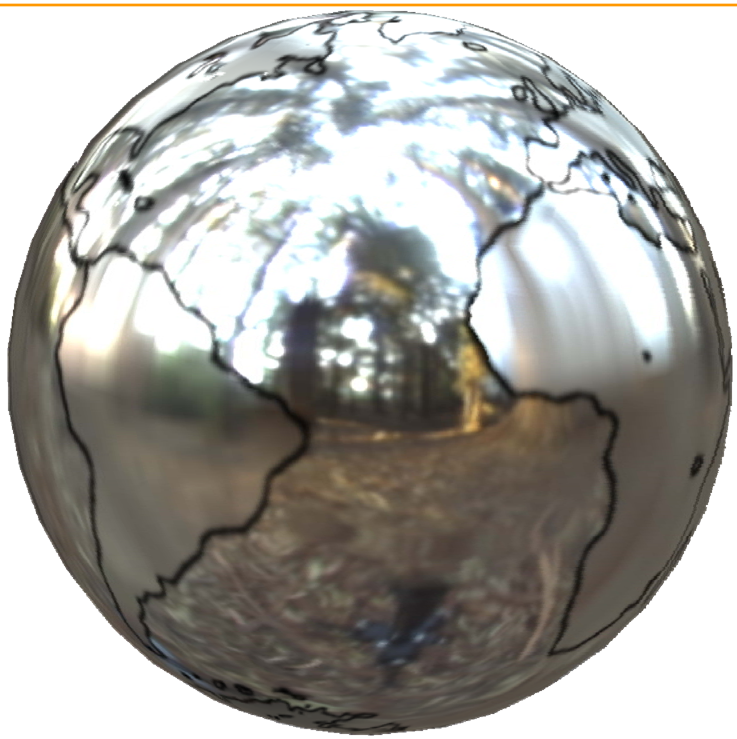
Other Miplevels (AEF&EF)

SIGGRAPH2005

- Different mip levels can be used to simulate different surface roughness [Ashikmin02]
  - Rougher surface → more blurry reflection
  - Use **MaxMipLevel** texture sampler state to clamp miplevel, not **texCUBEbias(…)**

Per-pixel Roughness Mapping

SIGGRAPH2005



- Per-pixel roughness mapping
  - Pack miplevel index into alpha channel of each cubemap miplevel
  - Use it to determine bias amount for **texCUBEbias(…)** to implement miplevel clamping in shader
  - Or use **texCUBElod(…)** on PS 3.0

Roughness Mapping Shader

SIGGRAPH2005

```
float fNumMipLevels;     // number of mip-levels in cubemap
float fBlurScale = 4.0;  // scale factor for blurriness

float4 main(float2 inUV   : TEXCOORD0, float3 inNormal : TEXCOORD1,
            float3 inView : TEXCOORD2) : COLOR0
{
  // Surface roughness stored in alpha of base map
  float4 cBase = tex2D(tBase, inUV);
  float fRoughness = cBase.a;

  float3 R = reflect (normalize(inView), normalize(inNormal));

  // Each cubemap stores miplevel index in alpha (scaled by 16/255).
  // Determine mip-LOD levels from 0 to fNumMipLevels (minification)
  float fMipLevelMinification = (255.0/16.0) * texCUBE(tCube, R).a;

  // Determine mip-LOD levels from -fNumMipLevels to 0 (magnification)
  float fMipLevelMagnification = (255.0/16.0) *
         texCUBEbias(tCube, float4(R, fNumMipLevels-1.0)).a;
[…]
```

- Determine the miplevel for the current texel by fetching the cube map with different mipbias levels and using the miplevel stored in the alpha channel.

Roughness Mapping Shader


```
[…]
  float fMipLevel = 0;

  //choose between magnification and minification range
  if(fMipLevelMinification == 0) { // 0 is the largest (base) miplevel
    // the whole cubemap is being magnified, compute "logical" miplevel
    // (which is negative)
    fMipLevel = fMipLevelMagnification - (fNumMipLevels - 1.0);
  }
  else {
    // the cubemap is being minified
    fMipLevel = fMipLevelMinification;
  }

  // compute final mip bias to clamp miplevel
  float fMipBias = max(fGlossScale * fRoughness - fMipLevel, 0.0);

  float4 cRefl = texCUBEbias(tCube, float4(R, fMipBias));

  return cBase * cRefl;
}
```

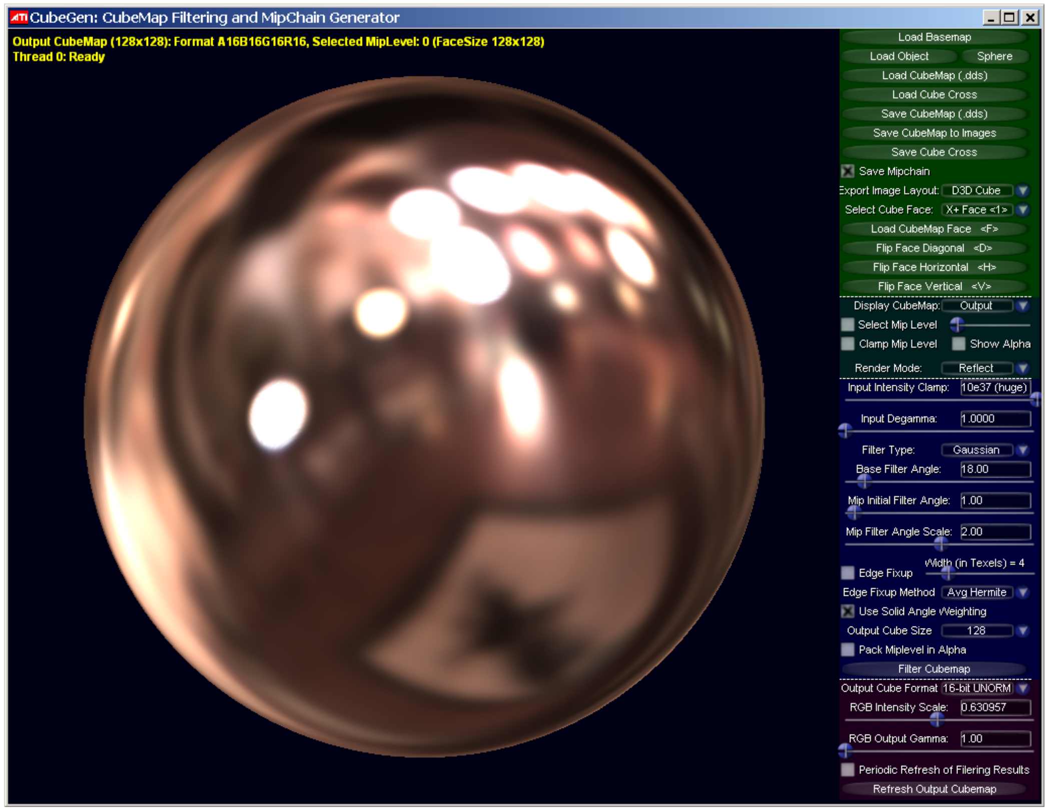
- The miplevel can be subsequently used to determine the mipbias amount needed to implement per-pixel mip-level clamping in the pixel shader.

Roughness Mapping

- Could use multiple subsequent mip-clamped fetches into cubemap for piecewise assembly of BRDF response.
  - Miplevel determination only needs to be computed once per-pixel shader invocation.

CubeMapGen Tool

SIGGRAPH2005



- CubeMapGen is a publicly available tool for cubemap filtering and mip-chain generation that uses angular extent filtering and edge seam fixup.
  - Available on http://www.ati.com/developer

References

- [Ashikhmin02] Ashikhmin, M. and Abhijeet, G. 2002. *Simple Blurry Reflections with Environment Maps*. In Journal of Graphics Tools, 7(4):3-8.
- [Kautz00] J. Kautz, P. P. Vázquez, W. Heidrich, and H.-P. Seidel *A Unified Approach to Prefiltered Environment Maps,* EG Rendering Workshop '00
- [Voorhies94] Voorhies, D. and Foran, J. *Reflection Vector Shading Hardware*. SIGGRAPH 1994, pp 163-166

- Some of the cubemaps used in this presentation can be found at:
  - http://www.debevec.org/Probes/
- Questions?