Adaptive Terrain Tessellation on the GPU

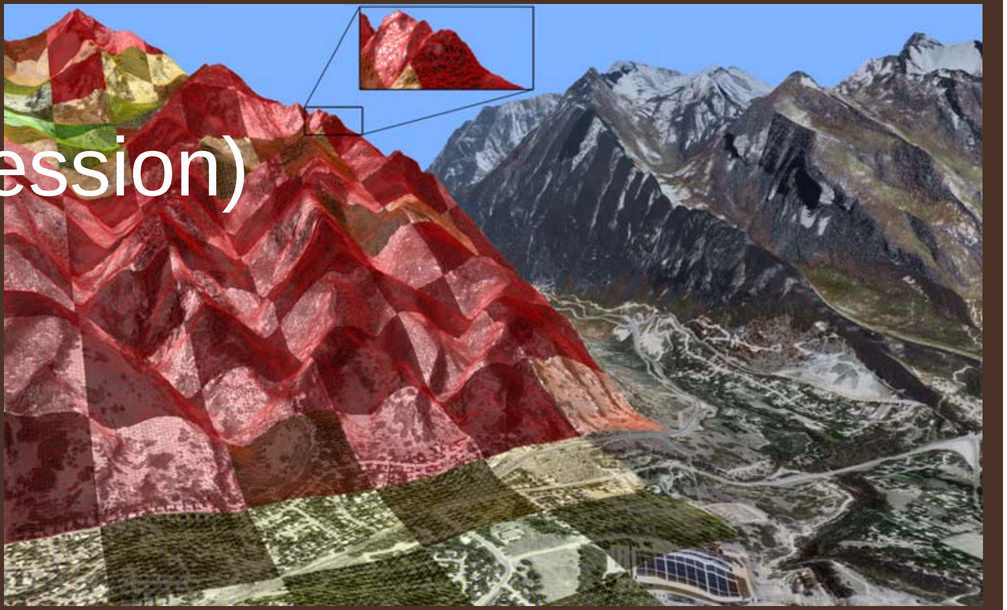
SIGGRAPH2008

Iain Cantlay

NVIDIA

Motivation

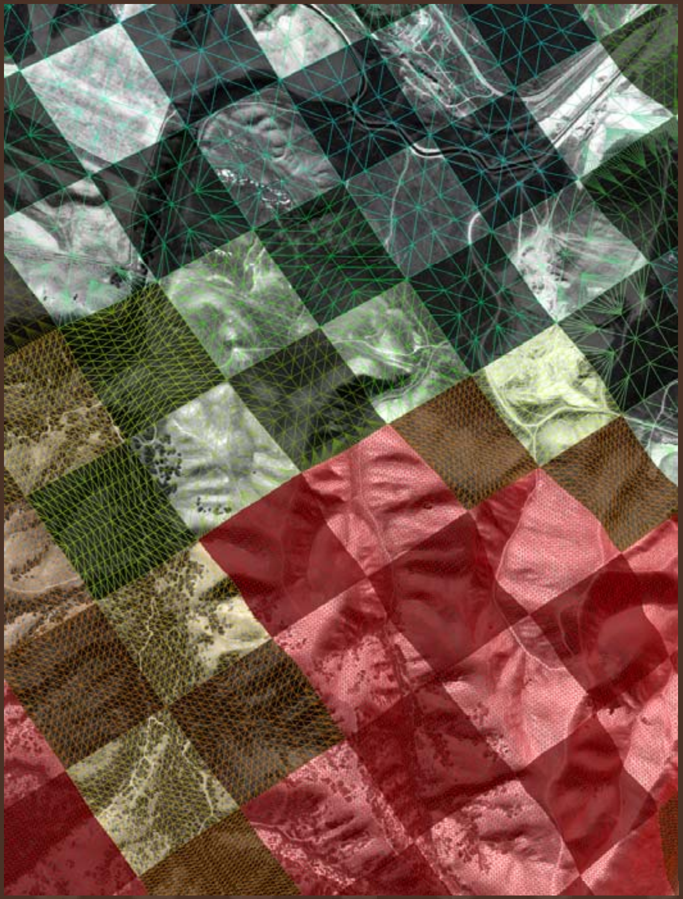
- Long view distances & large data sets
  - wide range of LOD
- Higher detail (compression)
- Unconstrained eye
  - highly dynamic
- Not GPU-friendly
  - ROAM [*Duchaineau 1997*]



SIGGRAPH2008

Basic tessellation use

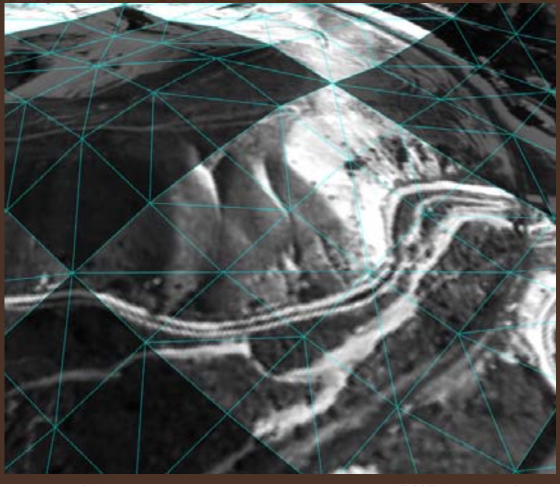
- Square, regular, flat patches
- LOD in Hull Shader
- Integer 2^n edges
- Displacement map:
  - Scalar displacement in Domain Shader (DS)
- A natural fit

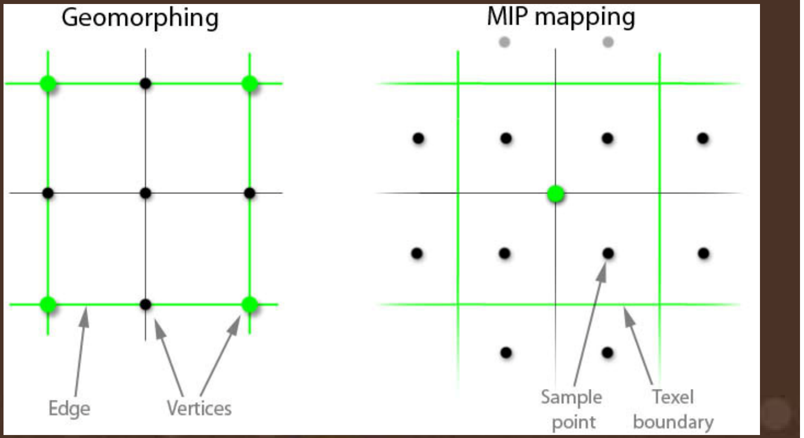


SIGGRAPH2008

Geomorphing & LOD

- Smoothly blend displacements between LODs [*Ulrich, 2002*]
- MIP sampling h/w blends
- MIP level per LOD
- Sample locations don’t match
- Nyquist: must over-sample

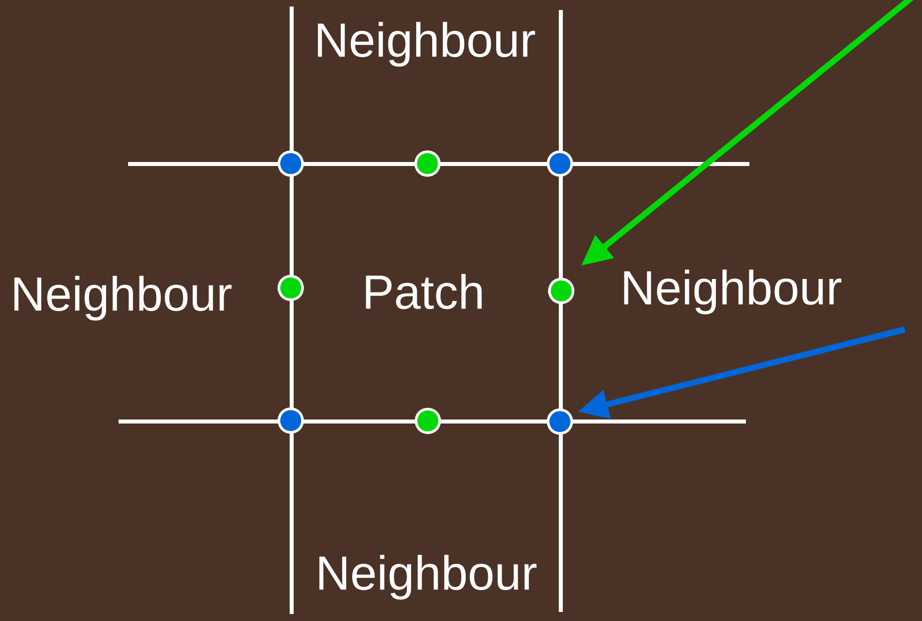




Crack Avoidance

- Neighbouring edges must match:
  - tessellation is function of edge mid-point
- Likewise for displacement MIP level:
  - Function of shared vertex position
- No math, no MAD – only displacement addition ☺
- Biases complicate later

Crack Avoidance

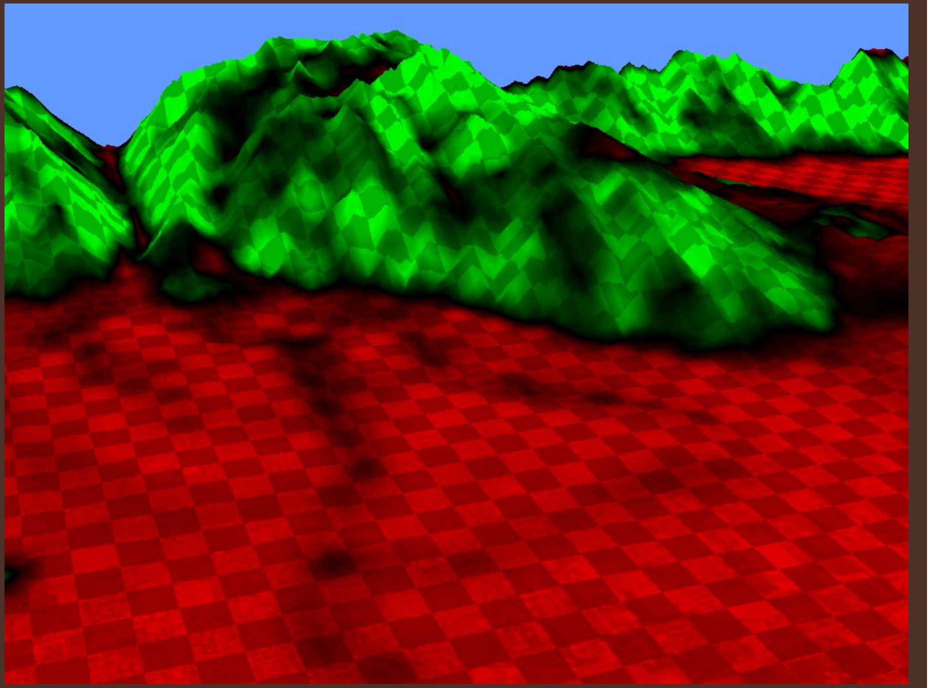


Tessellation level is a function of edge mid-points. All patches can trivially agree.

MIP map sample level is output **per-vertex** from hull shader. Again, all patches can trivially agree.

Roughness Bias

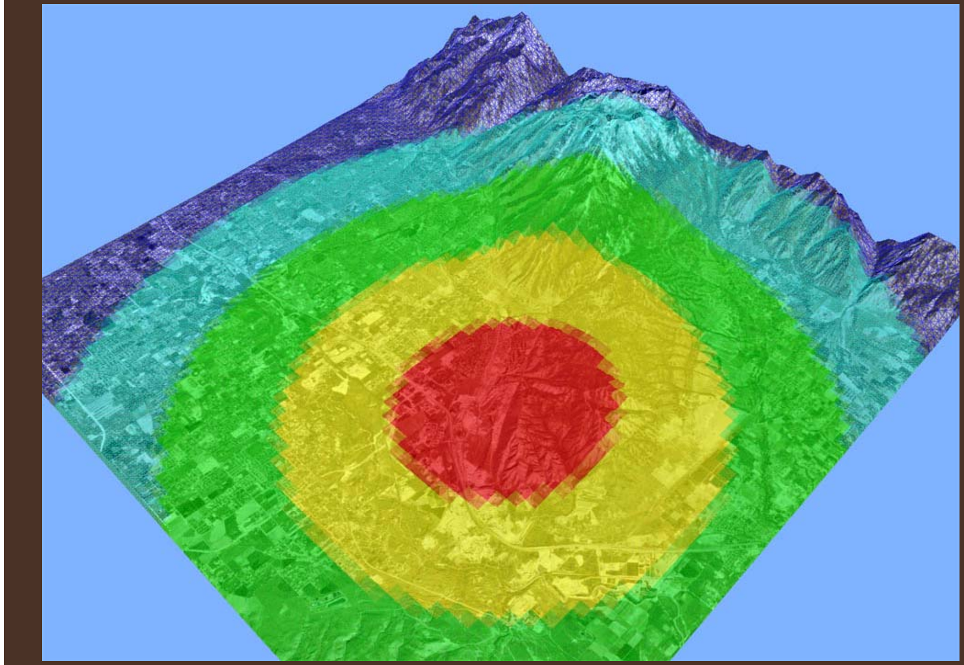
- Interesting areas – increase tessellation
- Flat boring – decrease
- abs(2nd order height differences)
- Pre-computed using CUDA

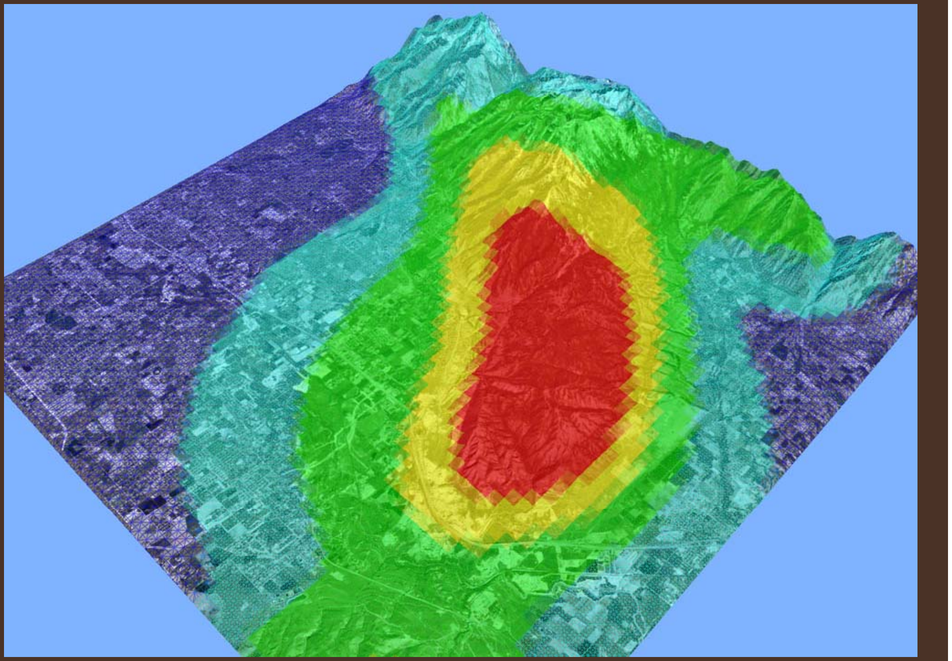


Green – rough; red – flat; black - neutral

Roughness Bias Results

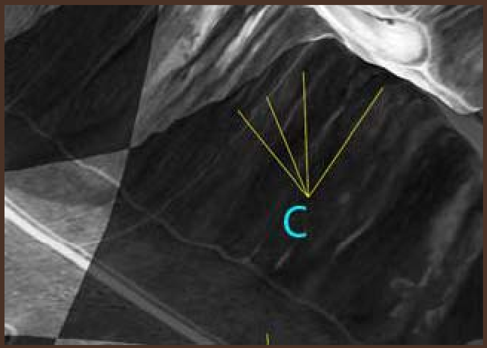
- Note increased blue, low-LOD

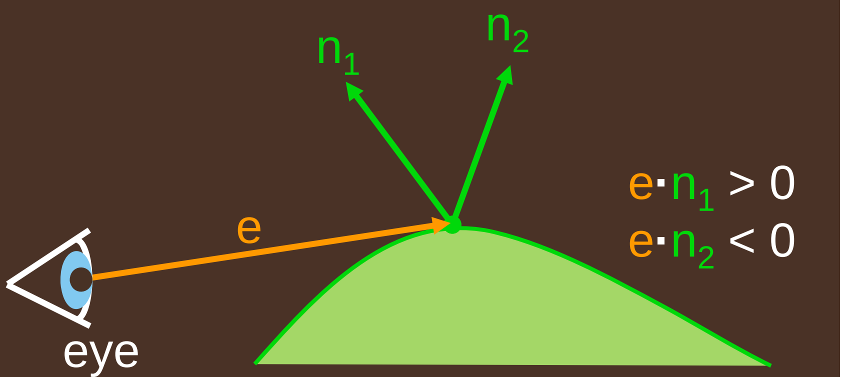




Silhouette Detection

- Is a patch a silhouette?
- Average orientation of height field
- K-means clustering of normals [*MacQueen, 1967*]
- Usual eye/normal math

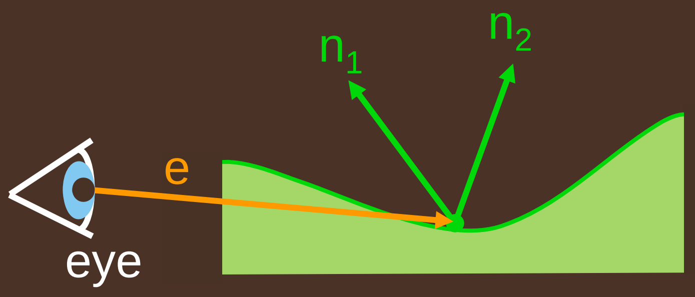




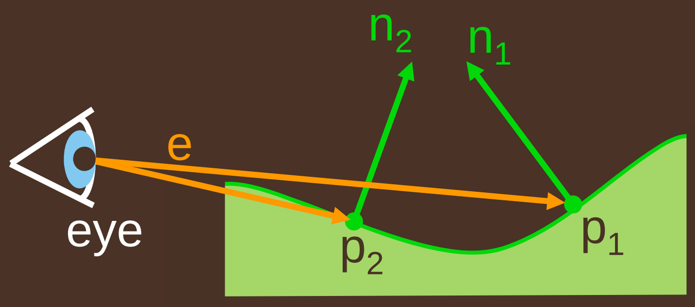
$e \cdot n_1 > 0$
$e \cdot n_2 < 0$

Concave Normals

- Concave normal pairs
  - give false positives
  - cannot form silhouette

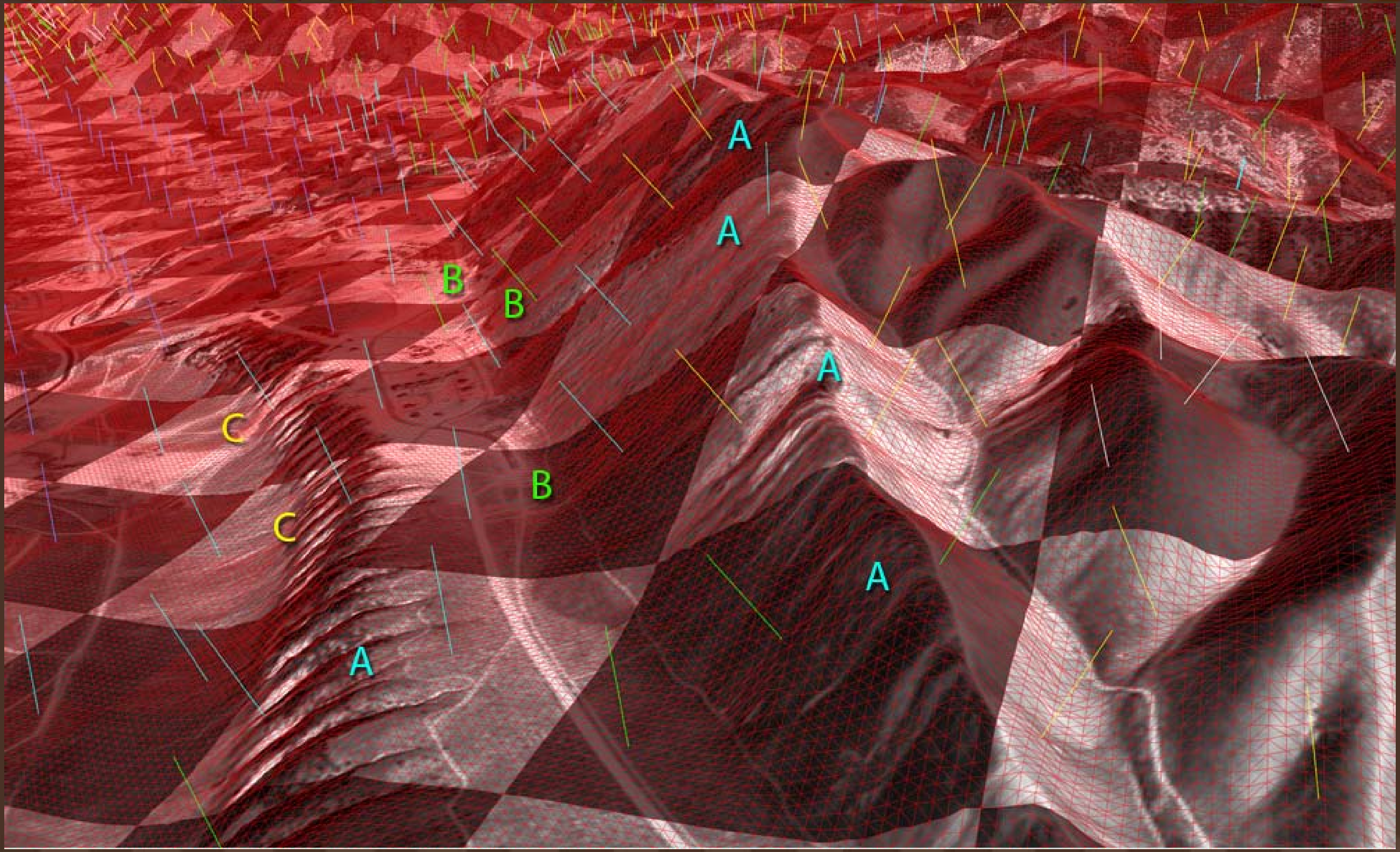


- Need normals' base positions



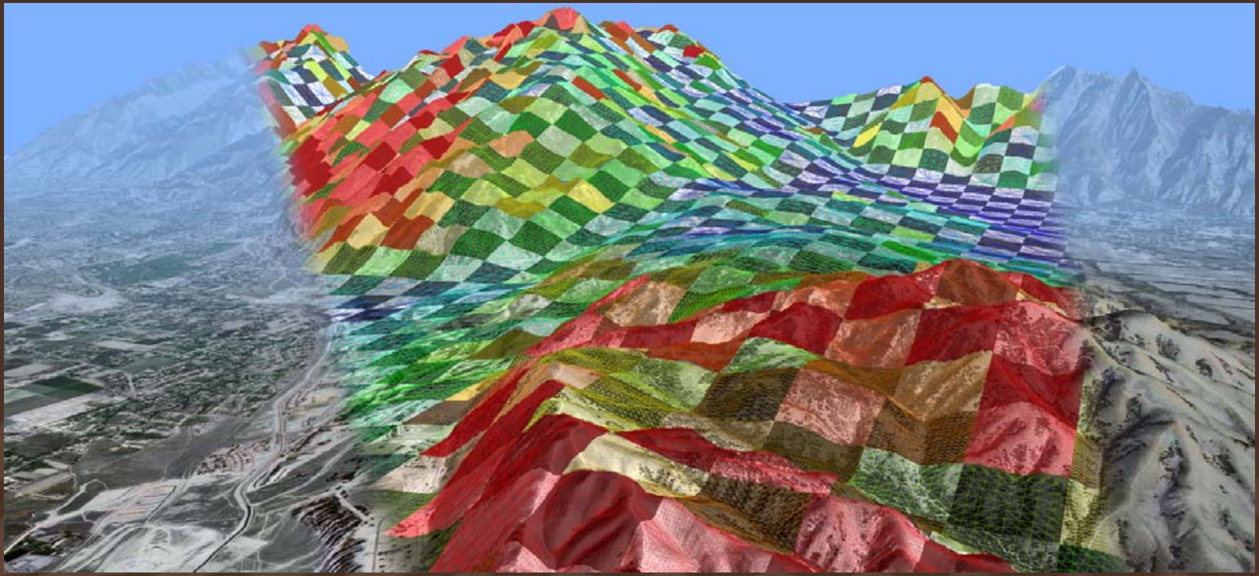
- Add position to clustering "distance"
- Discard in pre-process

Concave Normals



Silhouette Detection Results

- Mostly finds correct silhouettes, but…
- Too many “interior” silhouettes
- Misses silhouettes on patch boundaries



Conclusion

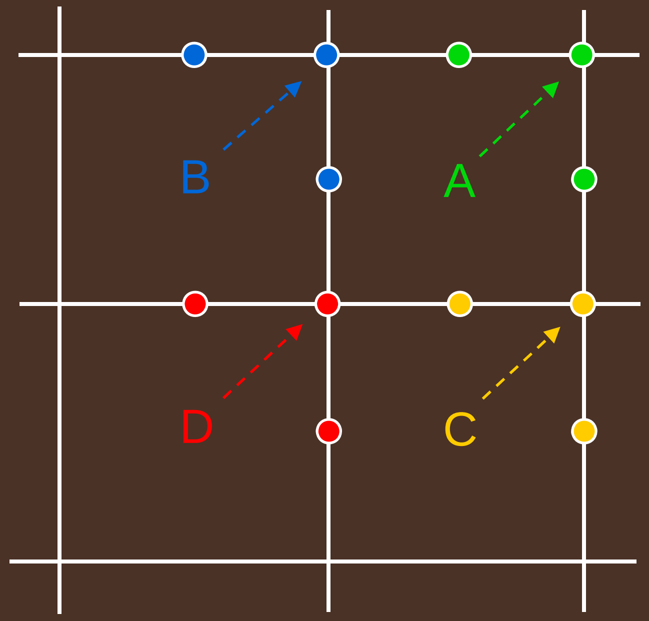
- Tessellated terrain – natural fit
- Easy engine integration
  - Displacement mapping – simple content requirements
  - Shaders – flexible patch arrangements
  - Shaders – flexible LOD

Not enough time,
too much material

Crack Avoidance With Biases

- Roughness bias
  - Texture based – patches must agree on sample points
  - Use edge mid-points and patch vertices
- Silhouette bias is more complex
  - Basic idea remains: agree on calculations at shared edges. Diagram…

Silhouette Bias Crack Avoidance



- Compute silhouette bias per-patch, A, B etc.
- Assign to vertices and edges as shown
  - only approximately correct
- Consider patch A
  - Must compute 4 biases for patches A,B,C & D

Detail Noise

- Oversampling and bilinear filtering leads to smooth look
- Add noise from a small texture
- Proportional to roughness measure
  - Suggested by fractal self-similarity
  - Works well

Limitations and further work

- Tessellation levels up to 64
  - 6 LODs
  - Hierarchical base patches
- Decals
  - parameterization/screenspace
- Lighting
  - Tessellated normals?

References

- “Rendering Massive Terrains”, Thatcher Ulrich, Siggraph 2002
- “ROAMing Terrain: Real-time Optimally Adapting Meshes”, Mark Duchaineau et al, IEEE Visualization 1997
- “Some Methods for classification and Analysis of Multivariate Observations”, J. B. MacQueen, 1967
- http://developer.nvidia.com