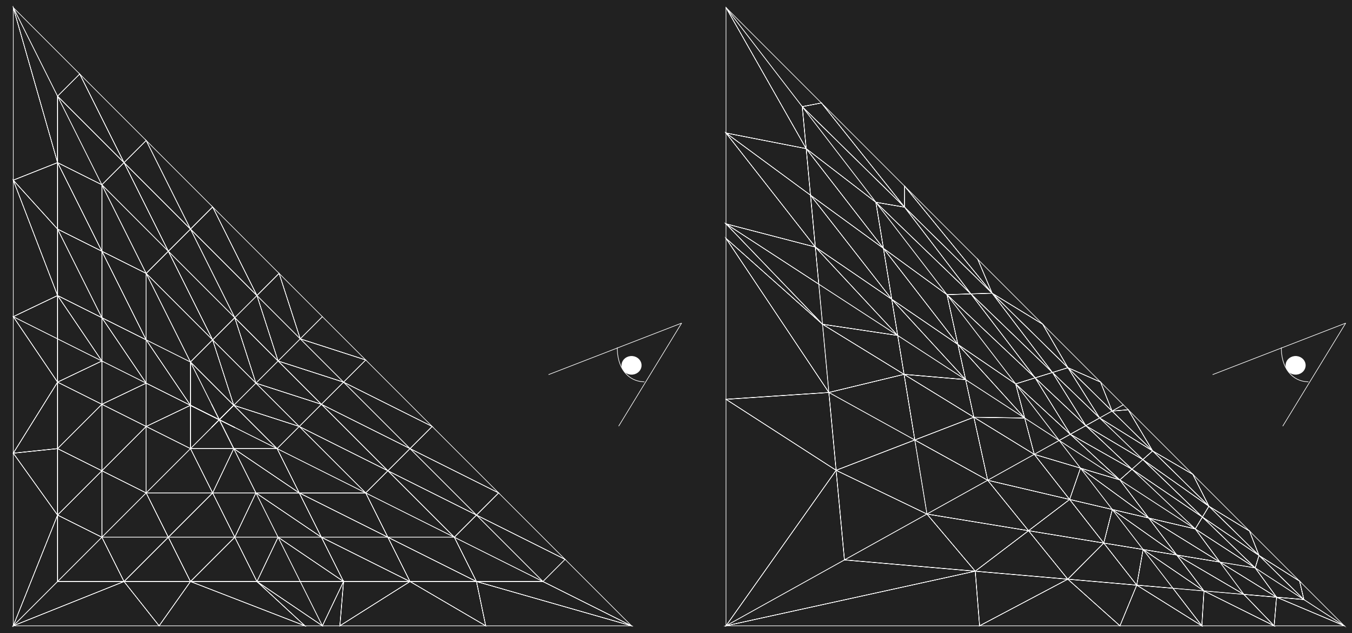
Non-Uniform Fractional Tessellation

Jacob Munkberg, Jon Hasselgren and
Tomas Akenine-Möller
Lund University

Simple idea

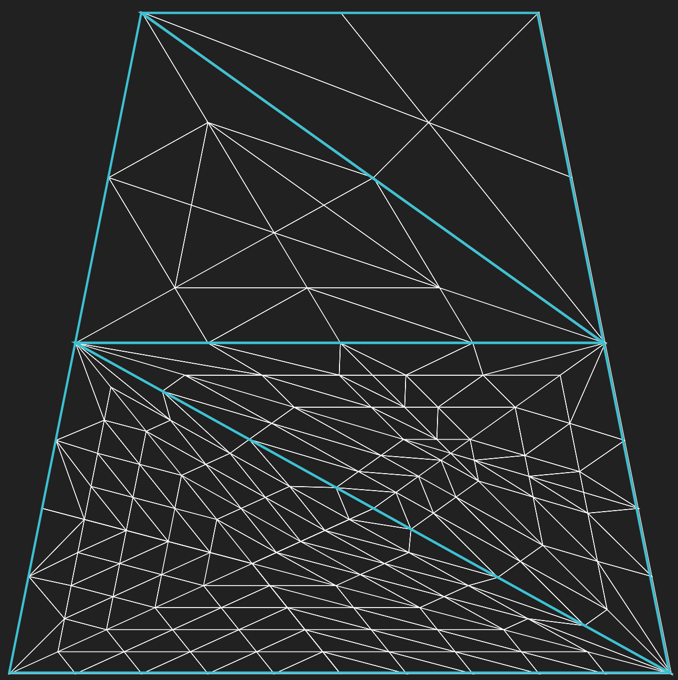
- We want triangles evenly distributed in screen space
  - Modify the tessellation pattern in current GPUs
  - Before the vertex shader is invoked



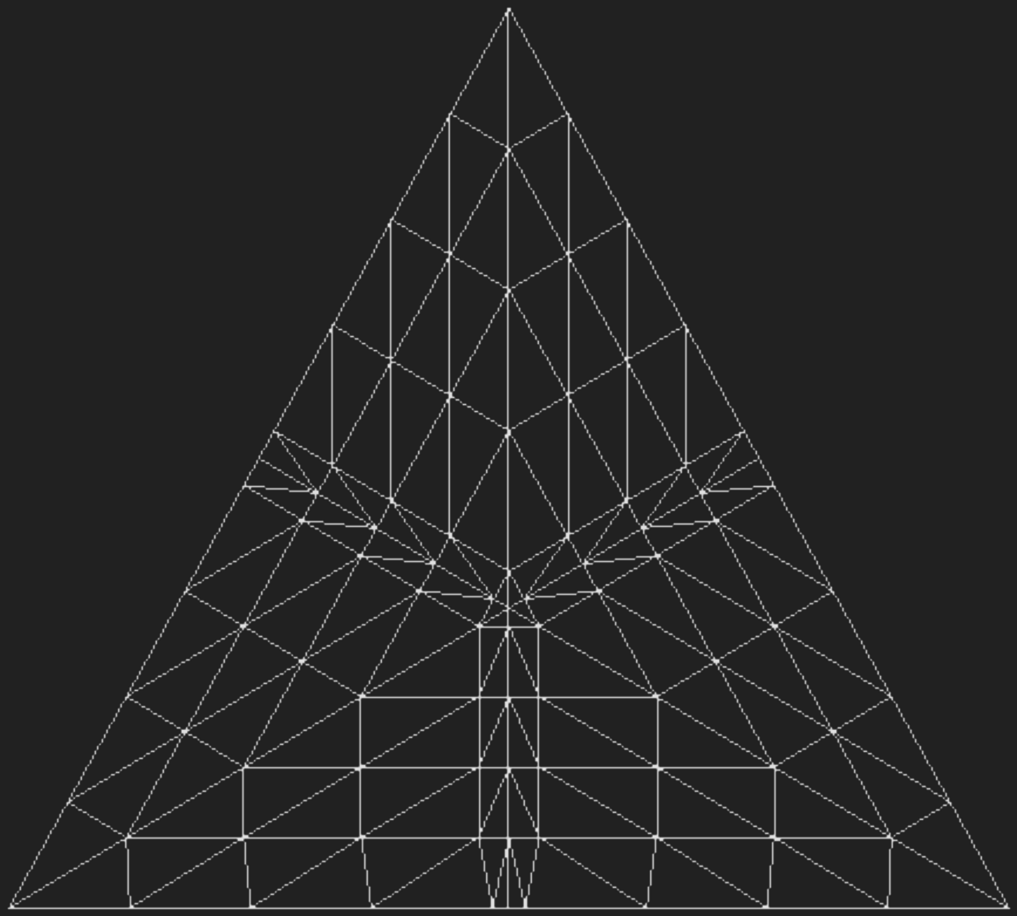
Regular Our

Regular Fractional Tessellation

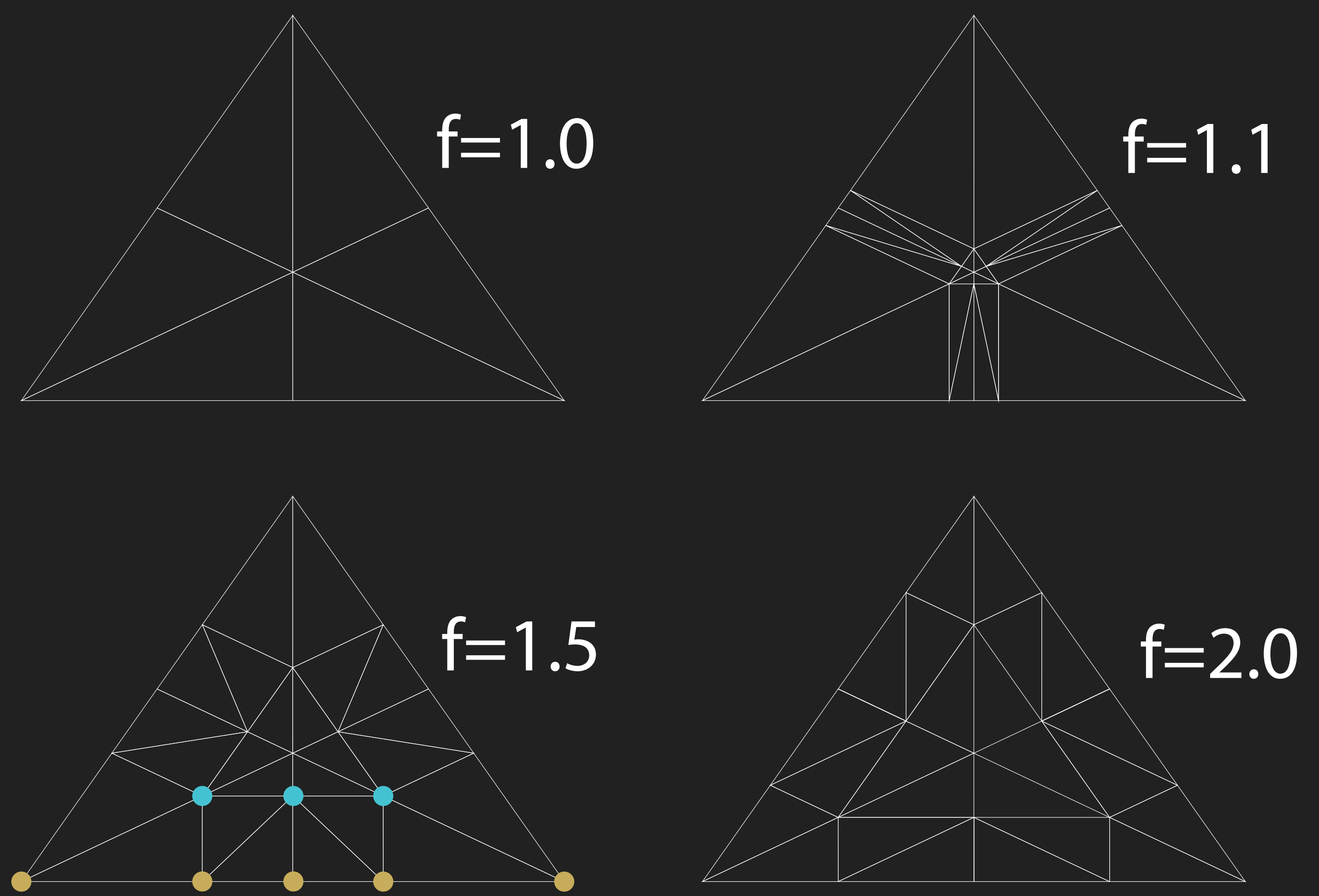
- Introduced by Moreton [2001]
- Continuous Tessellation Scheme
  - Floating point edge weights
- Allows for continuous level of detail
  - New vertices emerges from the center of each edge
  - No cracks or T-junctions



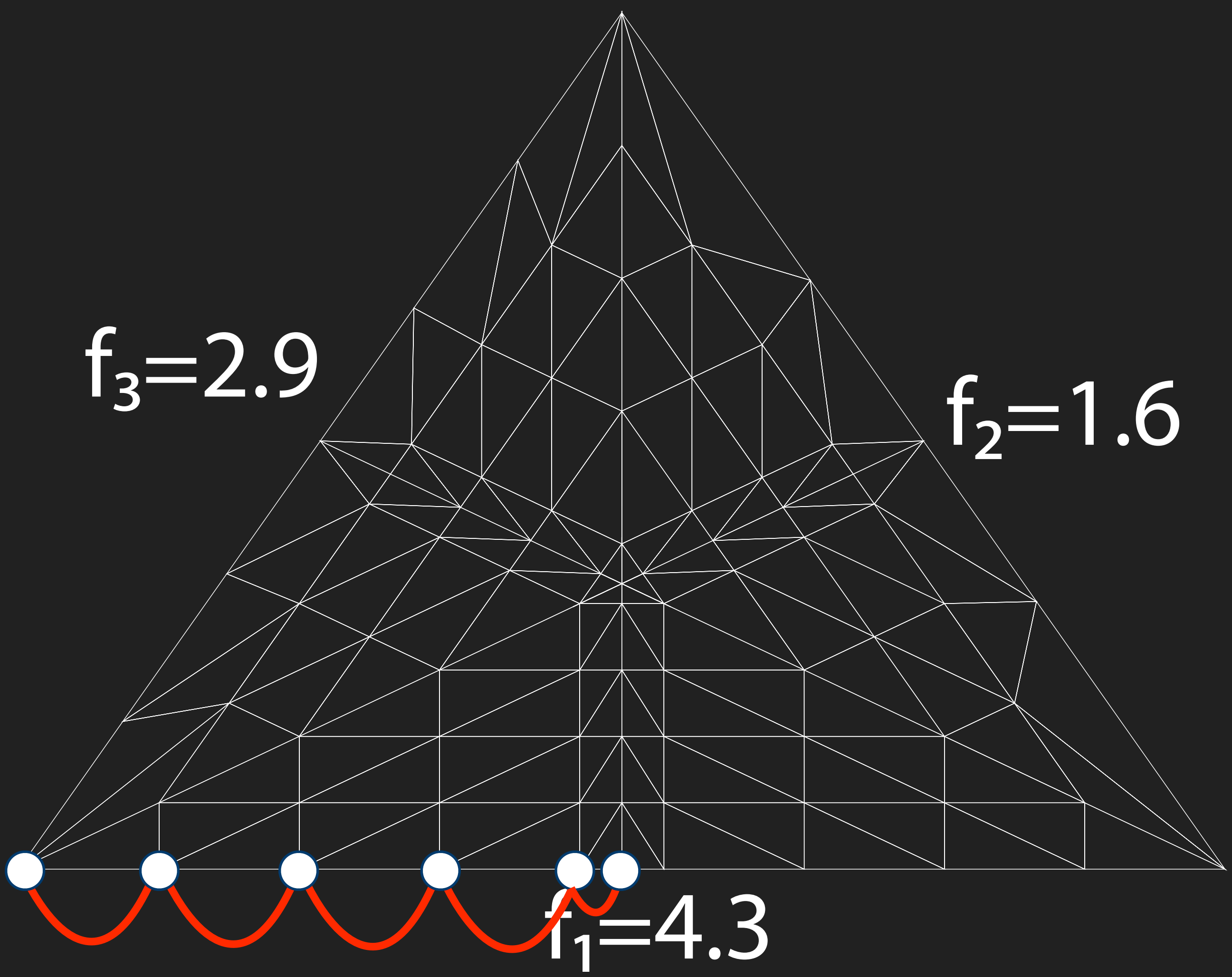
Animation



Edge Factors

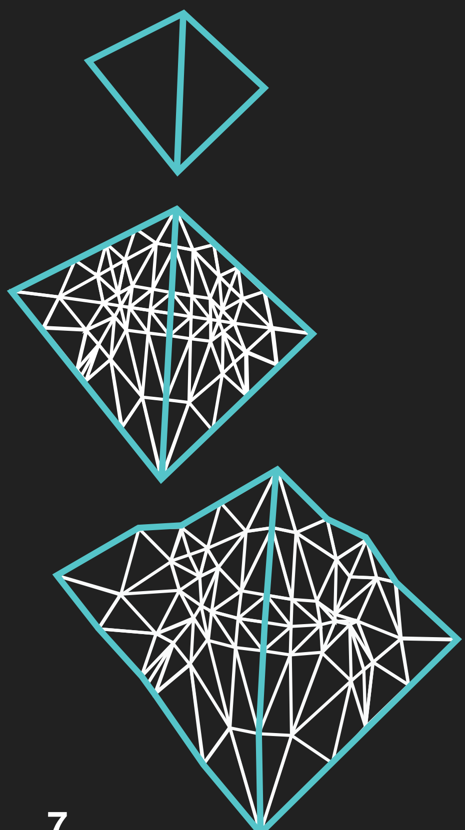


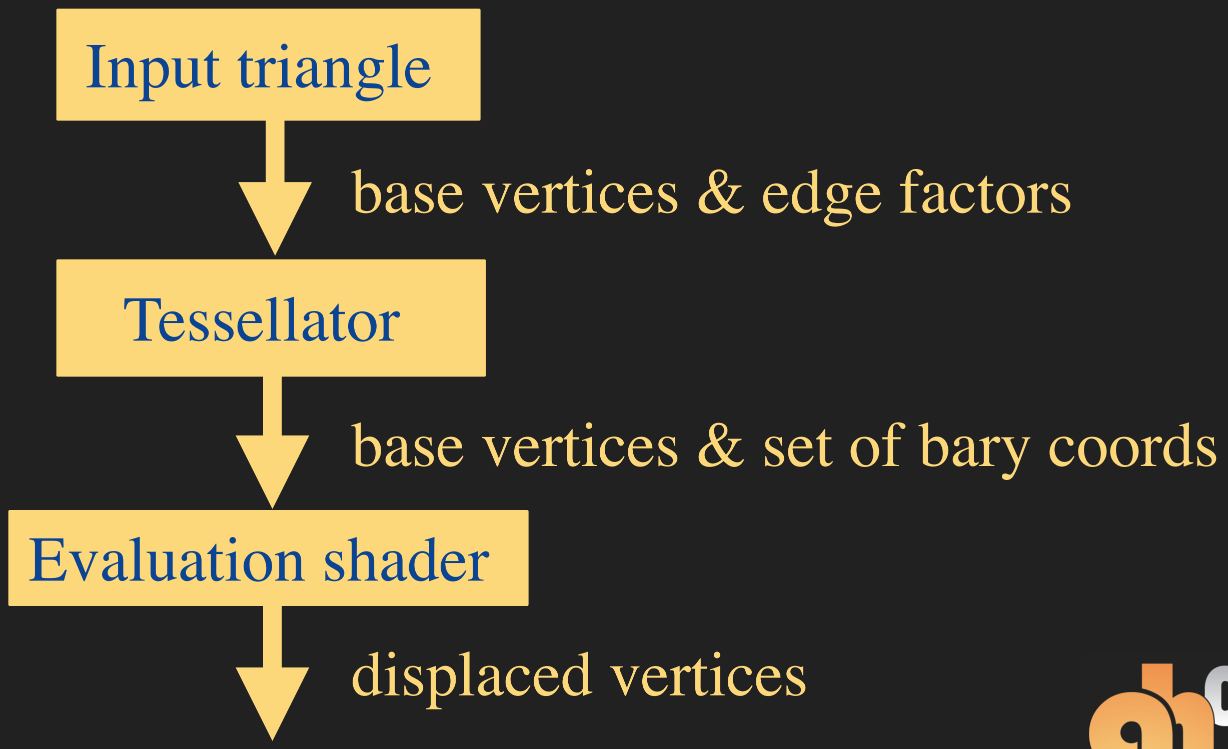
Unique edge factors



Fractional Tessellation on GPUs

- New AMD cards support fractional tessellation
- DX11 is likely to support tessellation

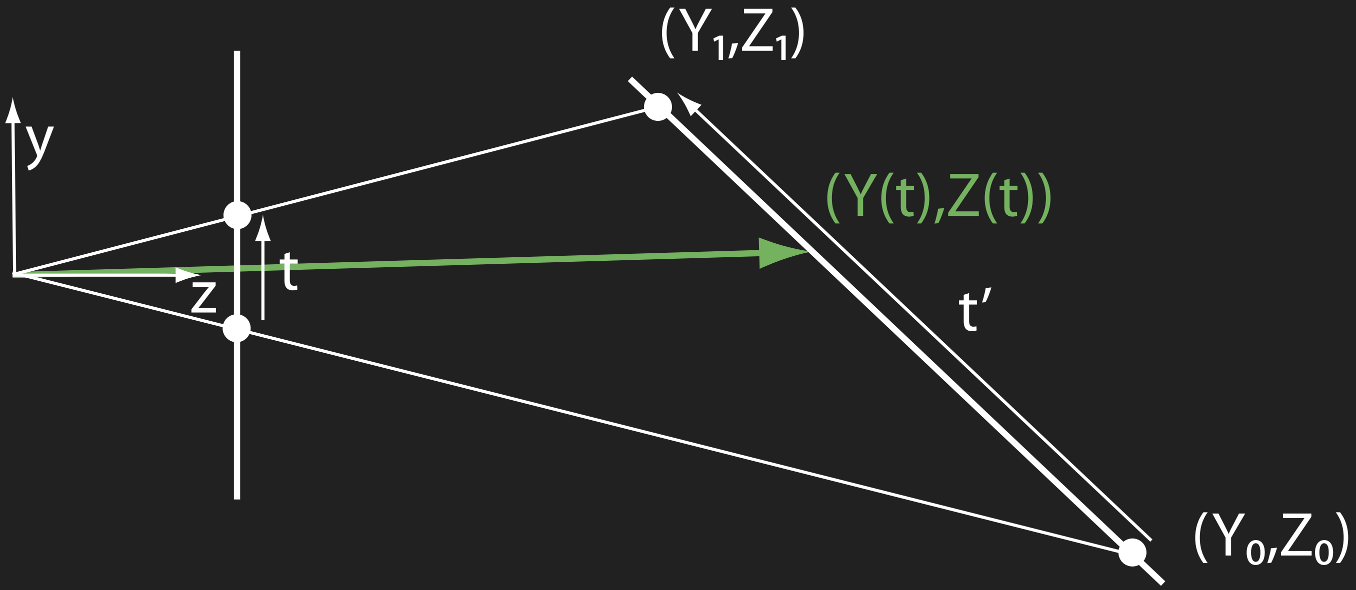




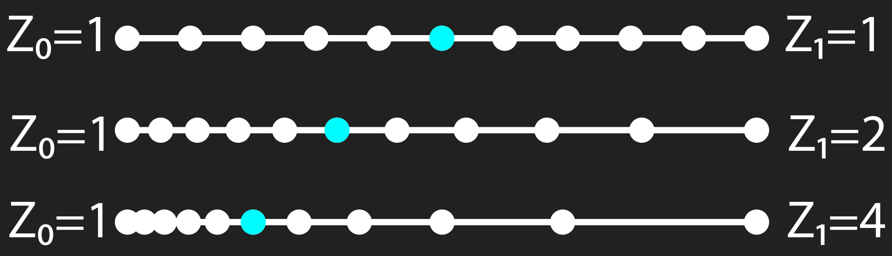
Evaluation/Vertex Shader

- Black box
  - Includes displacement lookups, surface evaluations, etc...
  - Moves vertex positions arbitrarily
  - We don’t know the exact evaluation shader
- But...it often contains a projection into clip space!
  - Exploit this
  - We want to reverse the effect of this projection, for more uniform tessellation in screen space

Perspective interpolation recap



$$t' = \frac{t/Z_1}{t/Z_1 + (1-t)/Z_0}$$



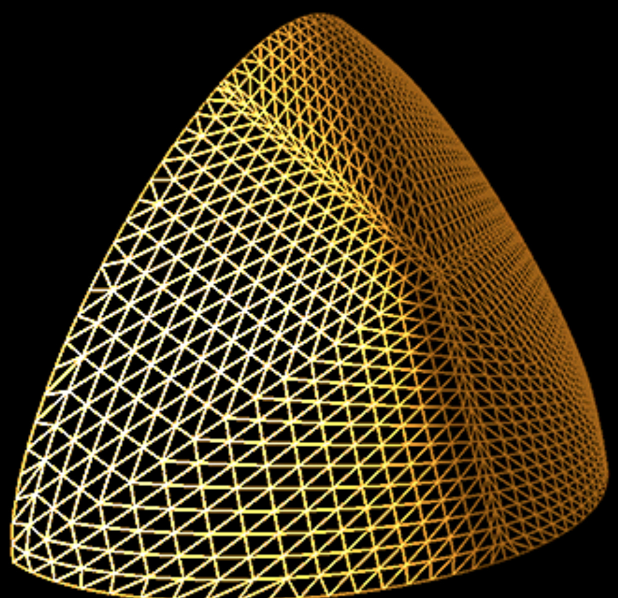
In the triangular domain...

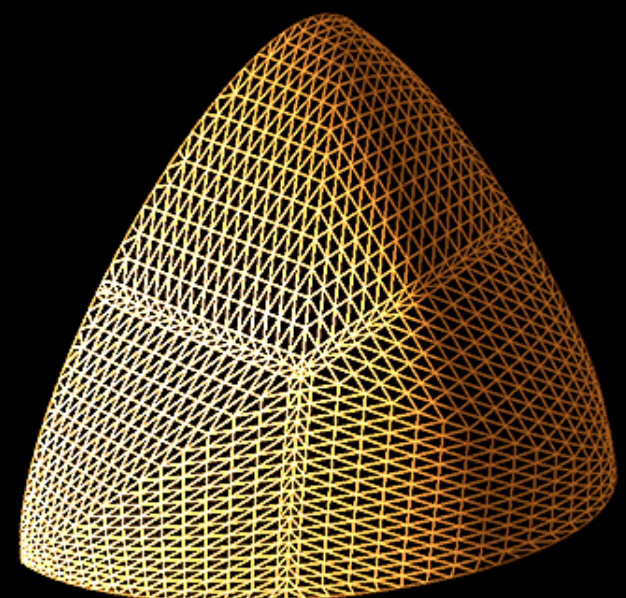
- The GPU tessellator generates a uniform distribution in the parametric space of the triangle
- We want a uniform distribution in screen space
- Use the perspective remapping!

$$u' = \frac{u/Z_1}{(1-u-v)/Z_0 + u/Z_1 + v/Z_2},$$
$$v' = \frac{v/Z_2}{(1-u-v)/Z_0 + u/Z_1 + v/Z_2}.$$

- Add this to beginning of evaluation shader
  - ~11 additional shader instructions

Comparison - wireframe



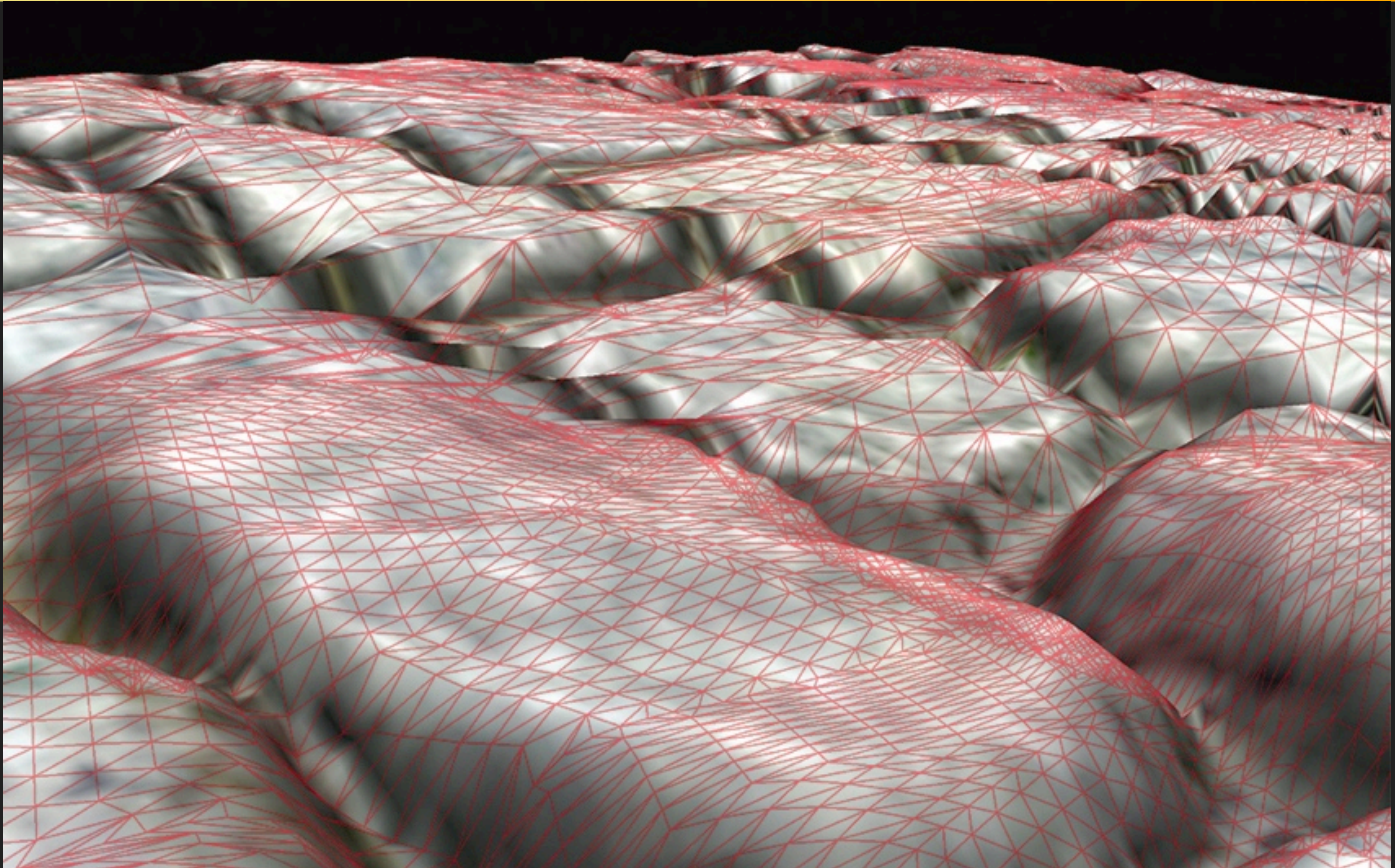


Equal #tris

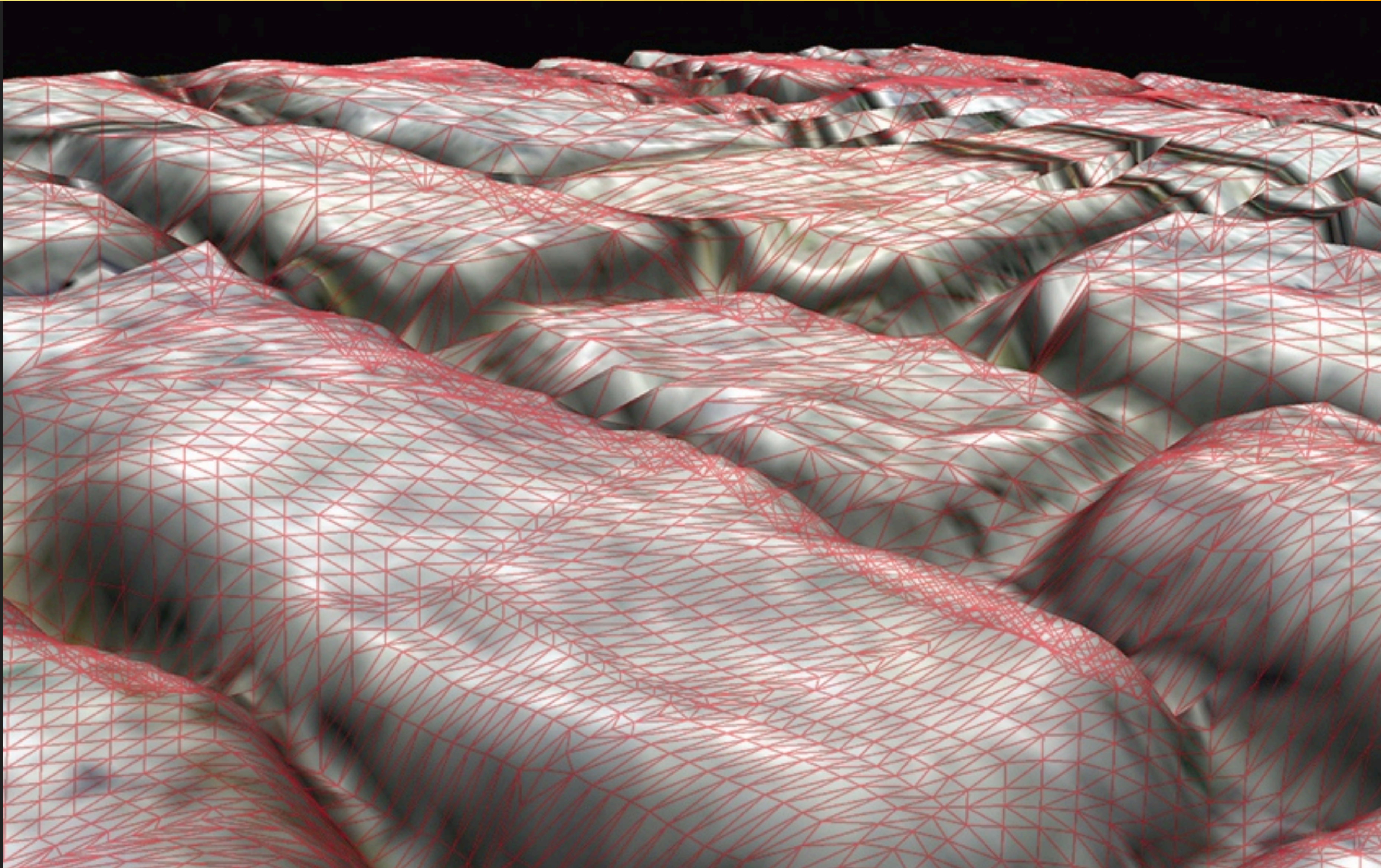
Regular

Our

Brick road - Regular

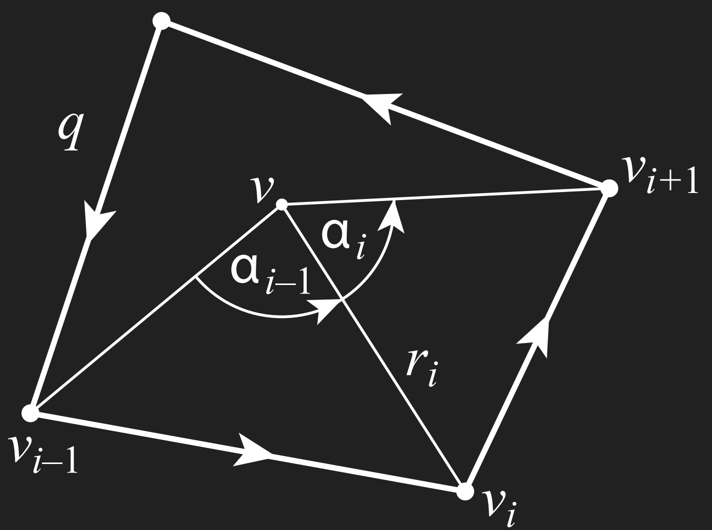


Brick road - Our



Quad Patches

- A Quadrilateral Rendering Primitive [Hormann and Tarini GH2004]:
  - Mean value coordinates λ_i can be used as barycentric coordinates for quad patches [Floater 2003]



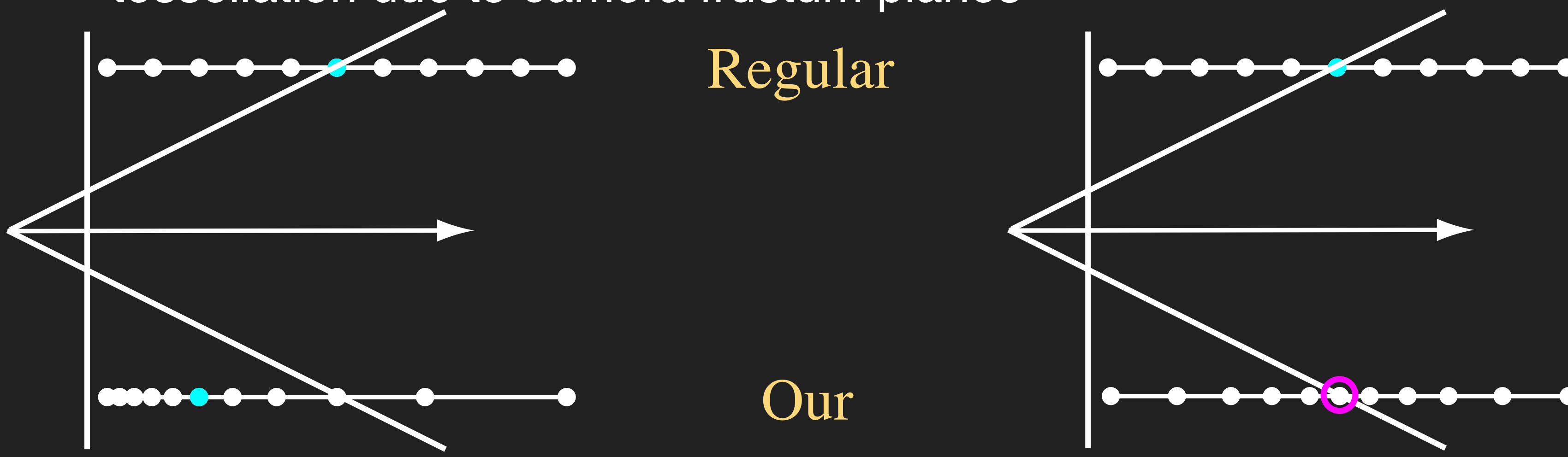
$$\mu_i(v) = \frac{\tan(\alpha_{i-1}(v)/2) + \tan(\alpha_i(v)/2)}{r_i(v)}$$

$$\lambda_i(v) = \frac{\mu_i(v)}{\sum_{j=0}^{n-1} \mu_j(v)}$$

$$\lambda_i'(v) = \frac{\frac{\lambda_i(v)}{w_i}}{\sum_{j=0}^{3} \frac{\lambda_j(v)}{w_j}}$$

All good?

- No!
  - Perspective interpolation flips when triangles straddles the Z=0 plane (division by zero, and/or negative Z-values)
  - Further: A risk that we get worse results than regular fractional tessellation due to camera frustum planes

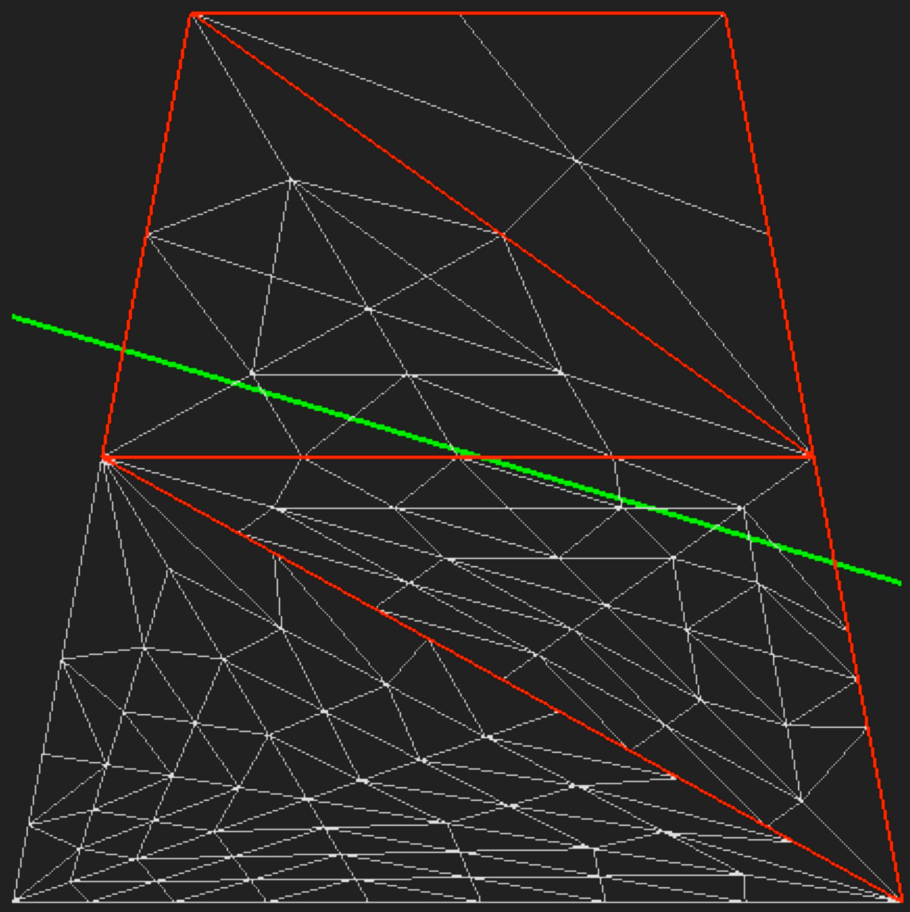


  - Clipping against entire view frustum helps

Straddling Triangles

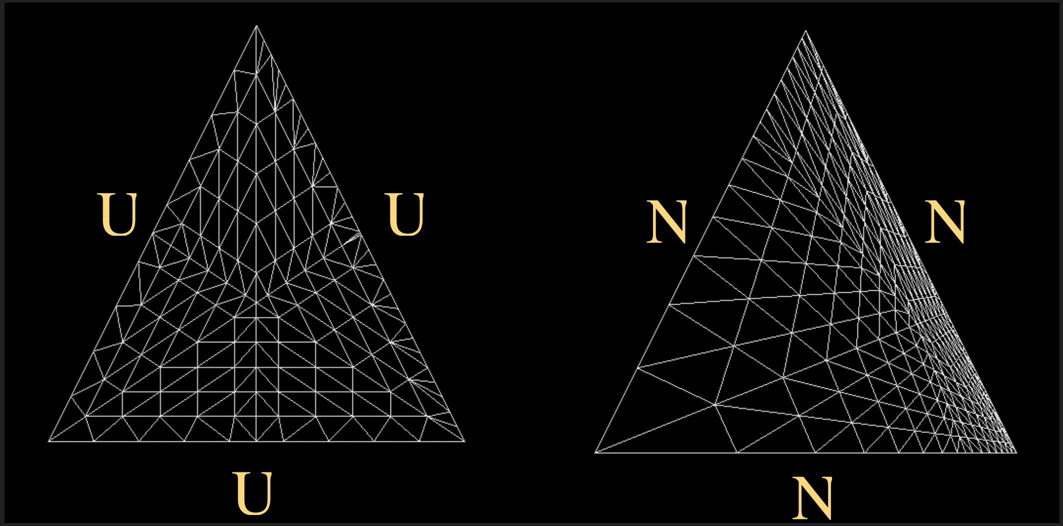
- Clipping is costly
  - Must clip against all frustum planes, not only near plane
  - Only performed on the base mesh
  - May introduce additional sliver triangles
- Alternative:
  - If triangle intersect a frustum plane
    → revert to regular fractional tessellation

Cracks



Edge interpolation

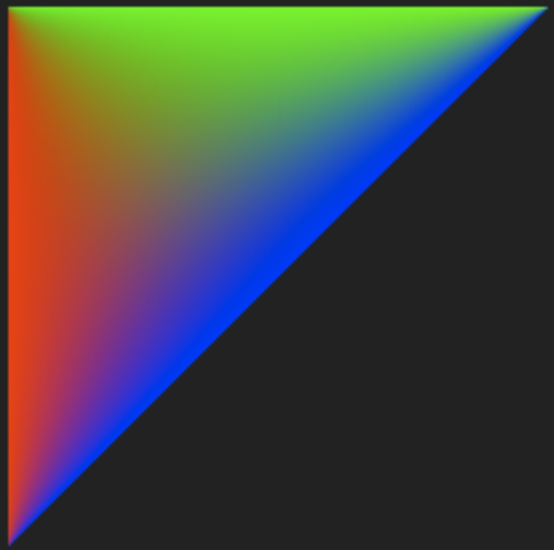
- Tag each edge of the triangle
  - either uniform (U) or non uniform (N)



- We want to blend between them
  - fully uniform or fully non-uniform on respective edge
  - varying smoothly over the triangle surface

Edge interpolation

- Color example:
  - A constant color along an edge, and a smooth blend in the interior of the triangle

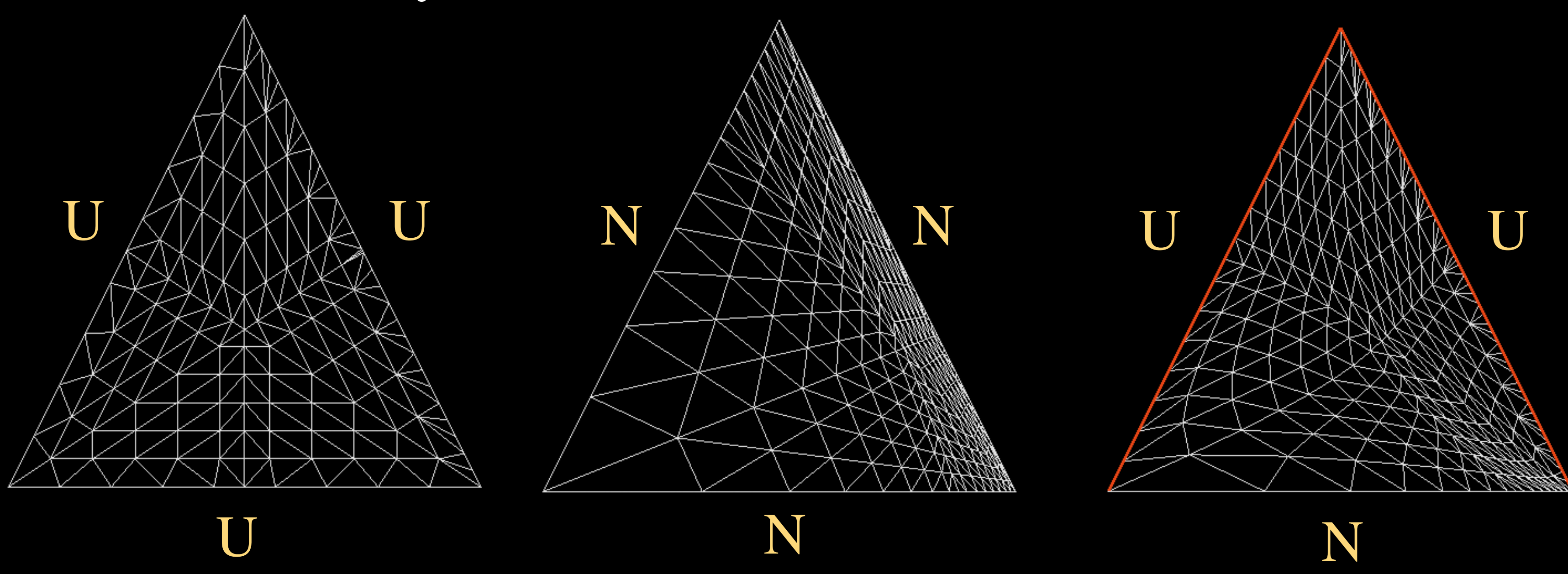


$$
\begin{aligned}
\alpha &= (1-u)vw \\
\beta &= u(1-v)w \\
\gamma &= uv(1-w) \\
Color &= \frac{\alpha R + \beta G + \gamma B}{\alpha + \beta + \gamma}
\end{aligned}
$$

Edge Interpolation example

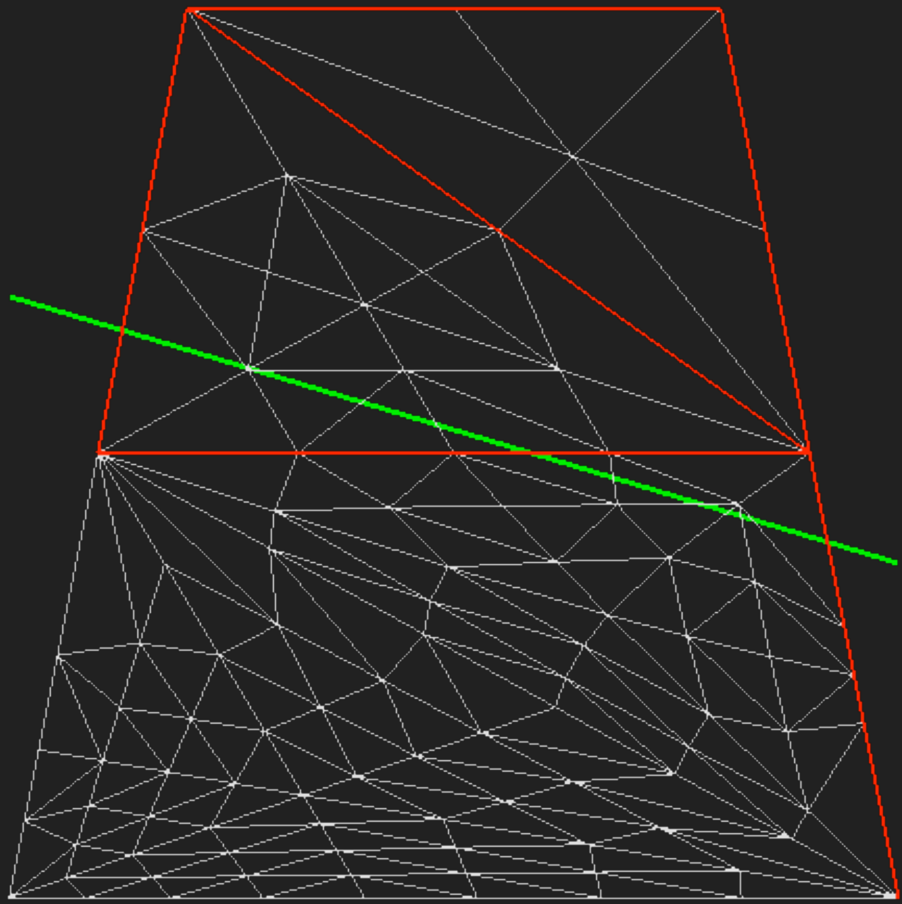
U: use standard barycentric coordinates (u,v) = $(u,v)_U$

N: use PC barycentric coordinates (u',v') = $(u,v)_N$



$$(u,v)_I = \frac{\alpha \cdot (u,v)_U + \beta \cdot (u,v)_U + \gamma \cdot (u,v)_N}{\alpha + \beta + \gamma}$$

Edge Interpolation animation



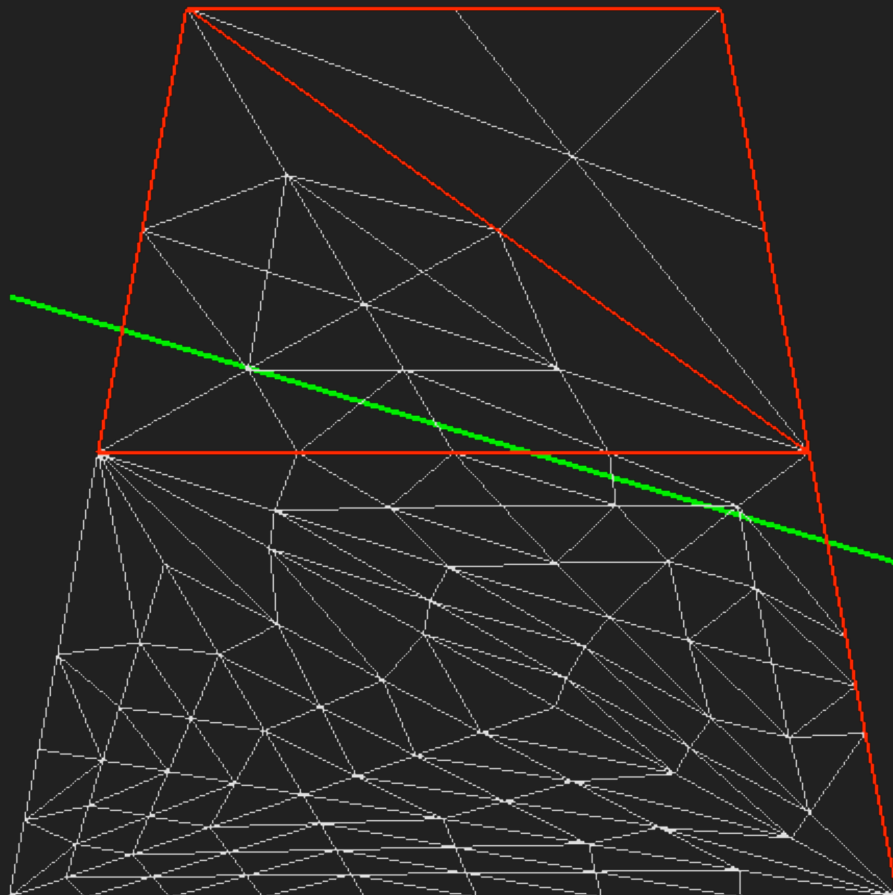
Smooth Warping

- The warping must be introduced gradually

- One more interpolation, in a guard zone, when a triangle edge intersects a frustum plane

- The guard zone is expressed as a fraction of the base triangle edge length

Smooth Warping animation



Video Example - Vertex swimming

Conclusions

- Simple technique
  - Added control of fractional tessellation with vertex weights
  - Redistribution of the tessellation pattern by warping the barycentric domain
  - Easily generalized to quad primitives
- But
  - Most useful for objects with large difference in Z
  - Many difficult cases must be handled in practice...

Future Work

- Tessellation APIs will become available!
  - Nice to try it out in real time!

- Vertex weights do not have to be depth values
  - Perspective-correction is only one application example
  - Other useful warping function might be possible
  - Each edge can have a unique warping function