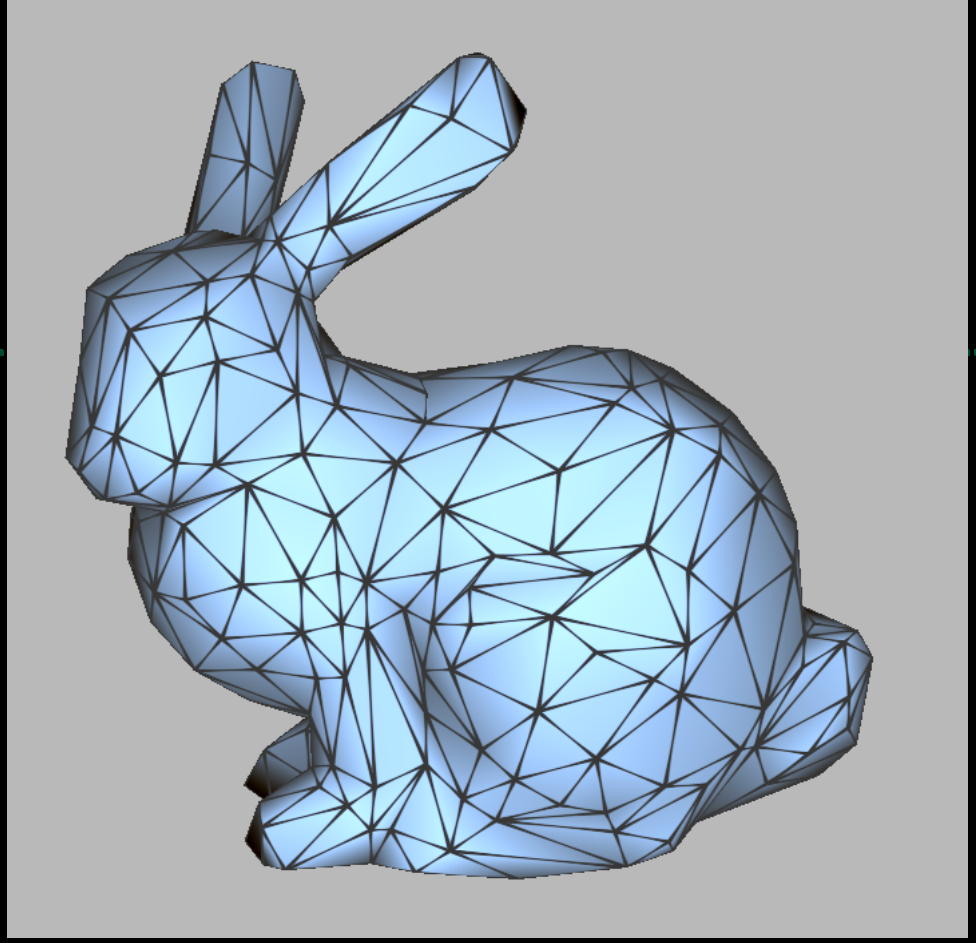
SIGGRAPH2006



SIGGRAPH2006

Single-pass Wireframe Rendering

with Hidden Surface Removal

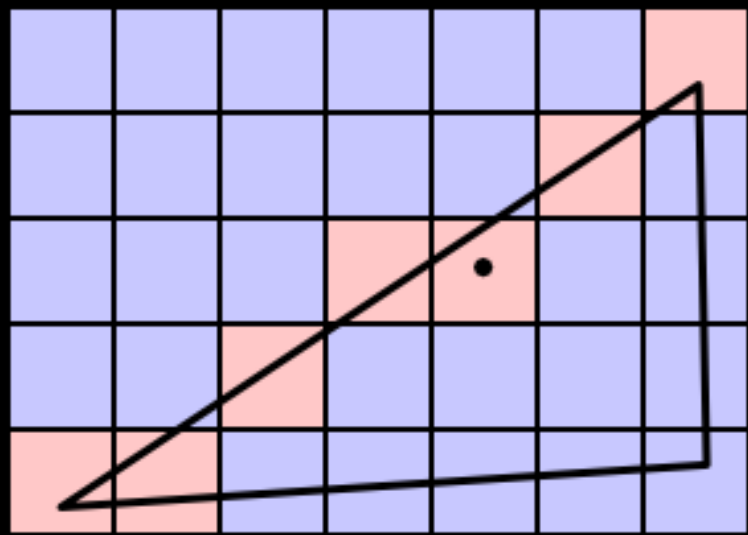
Andreas Bærentzen, Steen Lund Nielsen, Mikkel Gjøl, Bent D. Larsen and Niels Jørgen Christensen

Overview

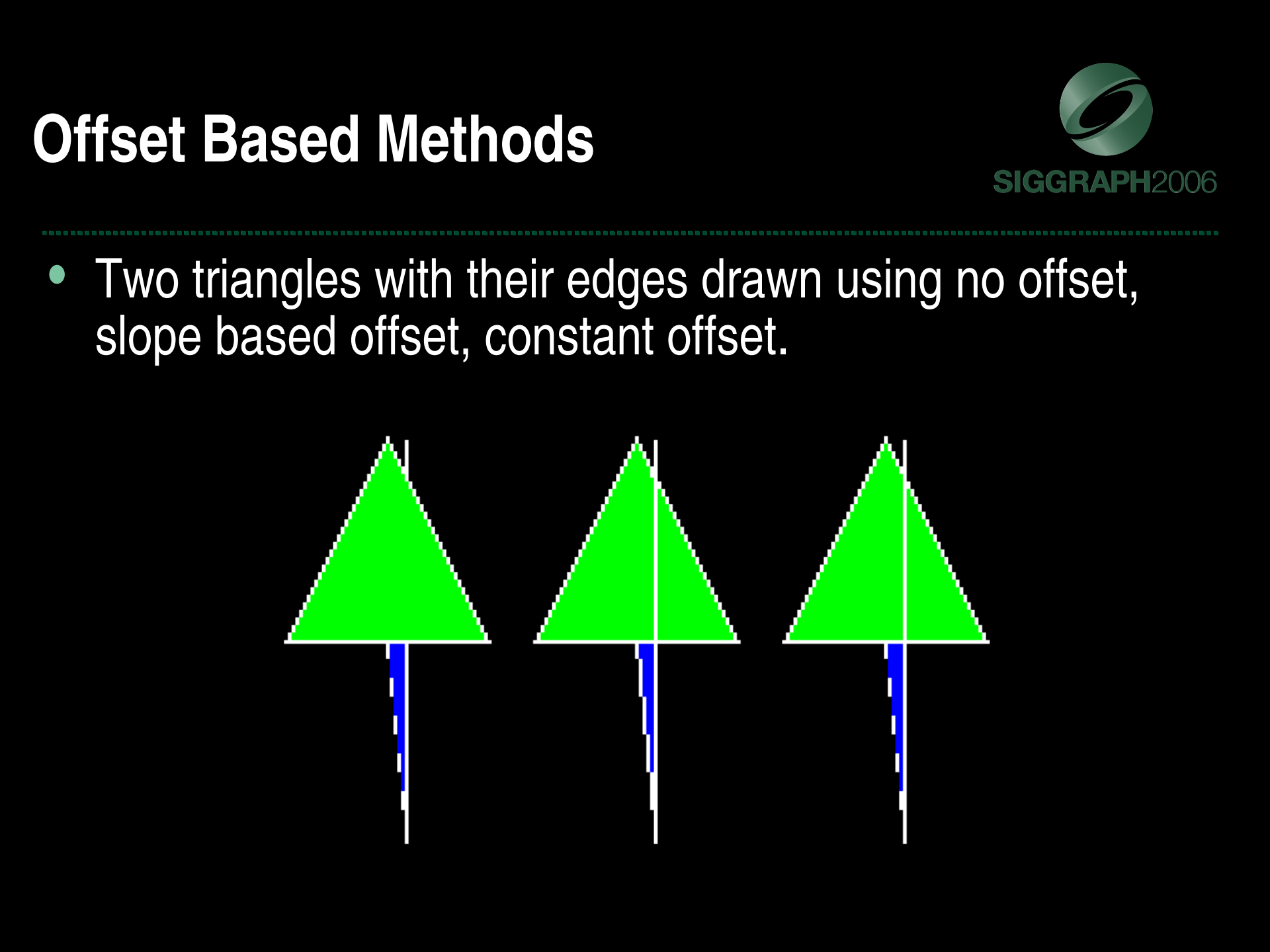
- What is wrong with the traditional **offset based methods** for wireframe drawing?
- A **single-pass** method for wireframe drawing with hidden surface removal.
- Drawing **prefiltered** lines.
- Results
- Discussion and future work.

Offset Based Methods

- Offset based methods have two passes:
  - Pass 1: Draw mesh as filled polygons.
    Z write and comparison enabled.
  - Pass 2: Draw mesh as wireframe with Z offset.
    Z write disabled. Z comparison enabled.
- Offset ensures filled polygons do not occlude wireframe.



Offset Based Methods



- Two triangles with their edges drawn using no offset, slope based offset, constant offset.

Offset Based Methods

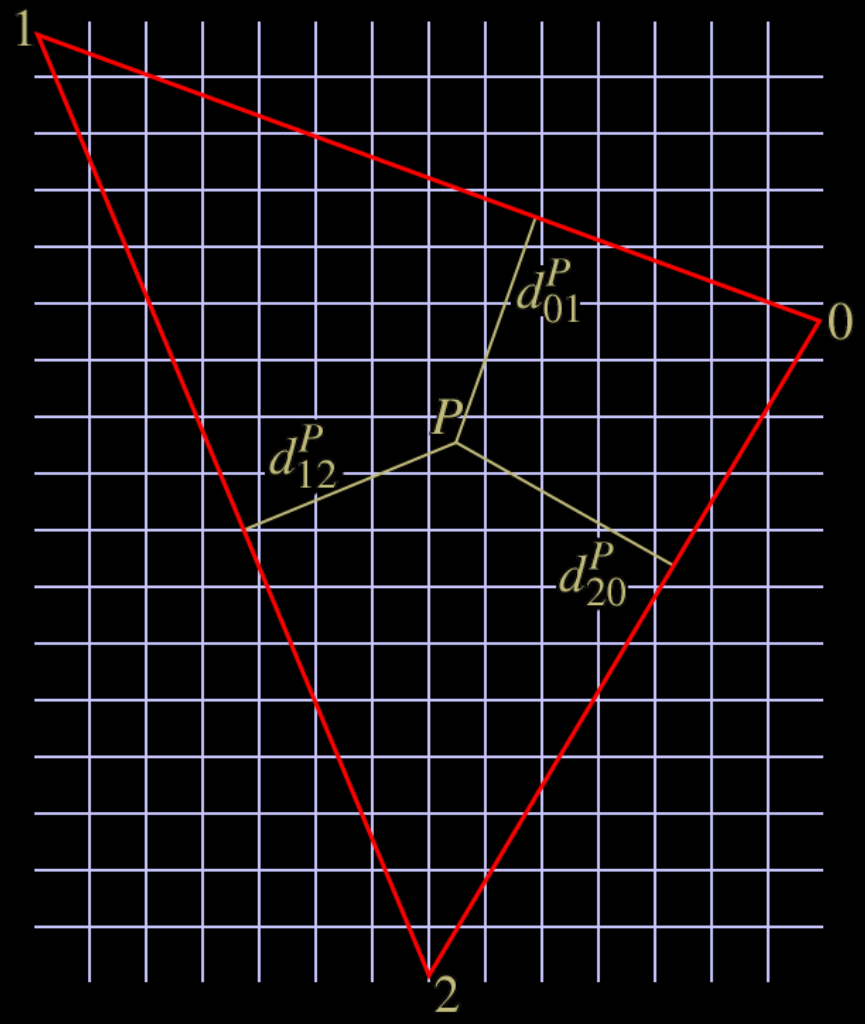
- use line drawing which is slow.
- are inherently two pass (even slower)
- require *antialiasing* to look good (slower still)
- have artifacts because it is impossible to pick a perfect offset in most cases.

- Admittedly, we can live with these methods, but we don't have to!

Approaches

- Edge is drawn only if the pixel already contains a fragment from the corresponding filled polygon. E.g. [Herrel et al. 1995].
- **Our method**: Draw wireframe directly as you are drawing filled polygon. All in one pass.

Single-Pass Method



- For each fragment, compute color from distance to closest edge:

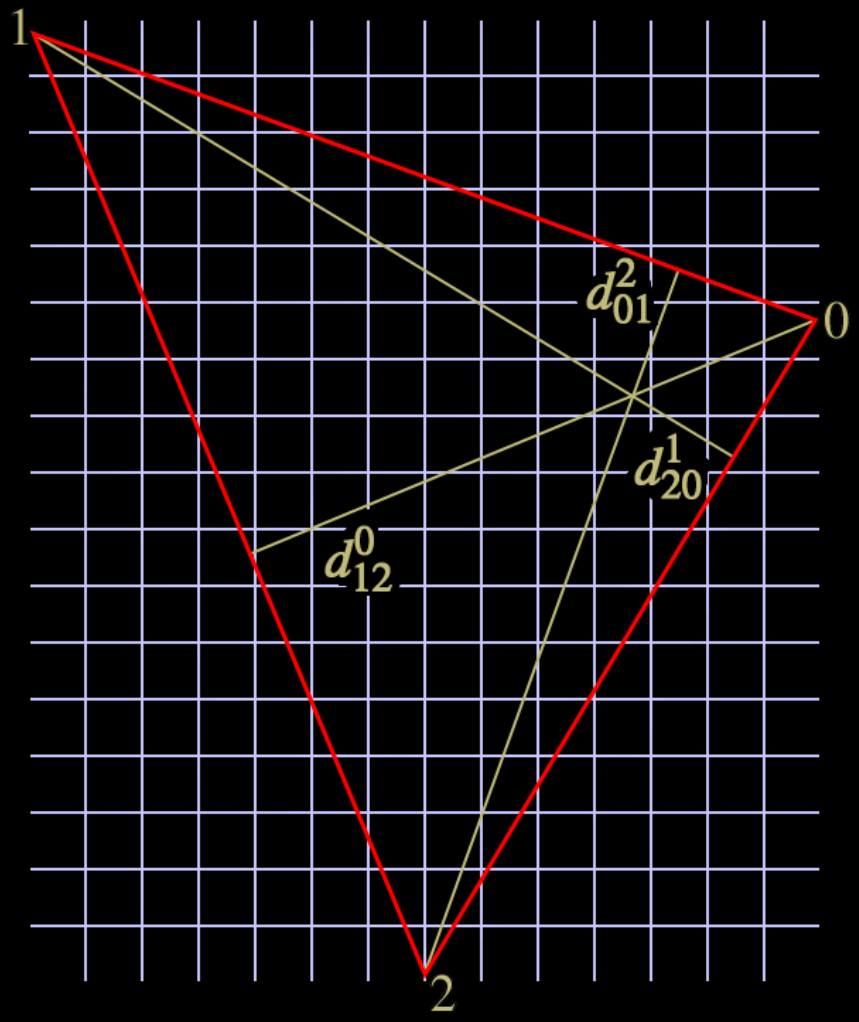
$$d = \min(d_{01}^P, d_{12}^P, d_{20}^P);$$

```
if (d < d_thresh)
    col = wire_col;
else
    col = fill_col;
```

Computing Distance

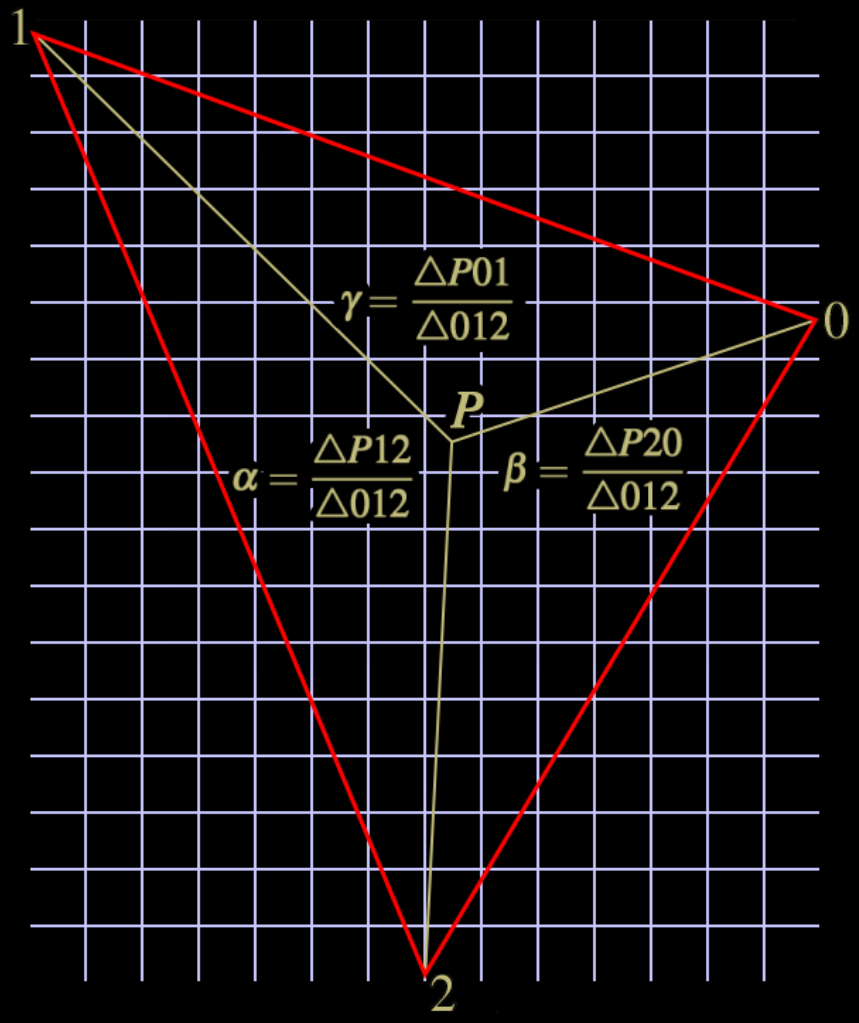
- We need 2D window space distance.
- In 2D, distance to line is an affine function.
- Linear interpolation reproduces affine functions.

Computing Distances



- Per vertex:
  - Input: All vertices of triangle.
  - Vertex shader projects all vertices of triangle.
  - Distances to all edges are computed.
  - Output: vector of distances to all three edges.

Computing Distances



- Per fragment:
  - Interpolate distances.

$$\begin{bmatrix} d_{01}^{P} \\ d_{12}^{P} \\ d_{20}^{P} \end{bmatrix} = \alpha \begin{bmatrix} 0 \\ d_{12}^{0} \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 0 \\ d_{20}^{1} \end{bmatrix} + \gamma \begin{bmatrix} d_{01}^{2} \\ 0 \\ 0 \end{bmatrix}$$

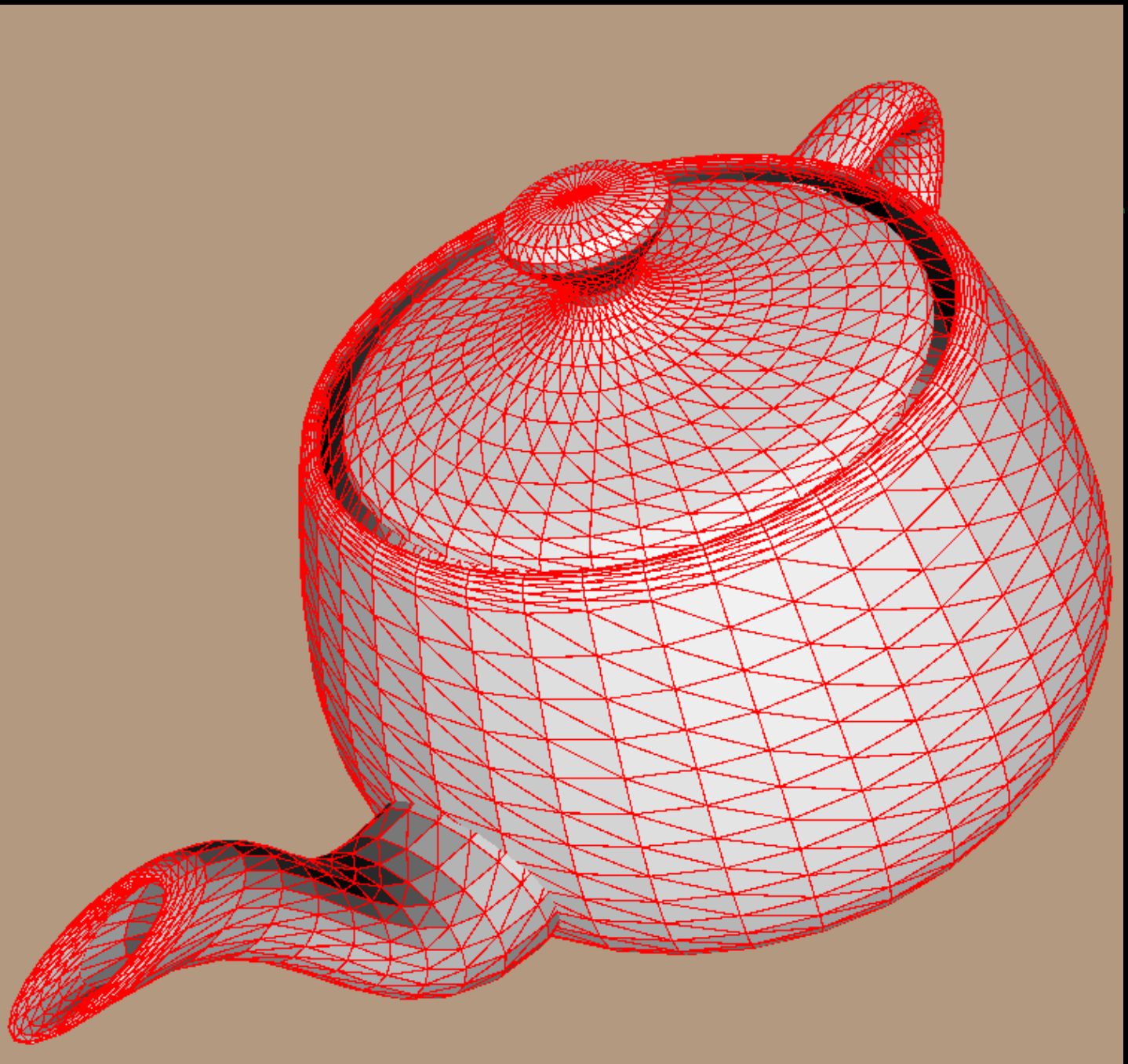
  - Compute color using previous fragment program.
- Note: This method extends to quads.

Perspective Correction

- Oops! Graphics cards perform perspective correct interpolation

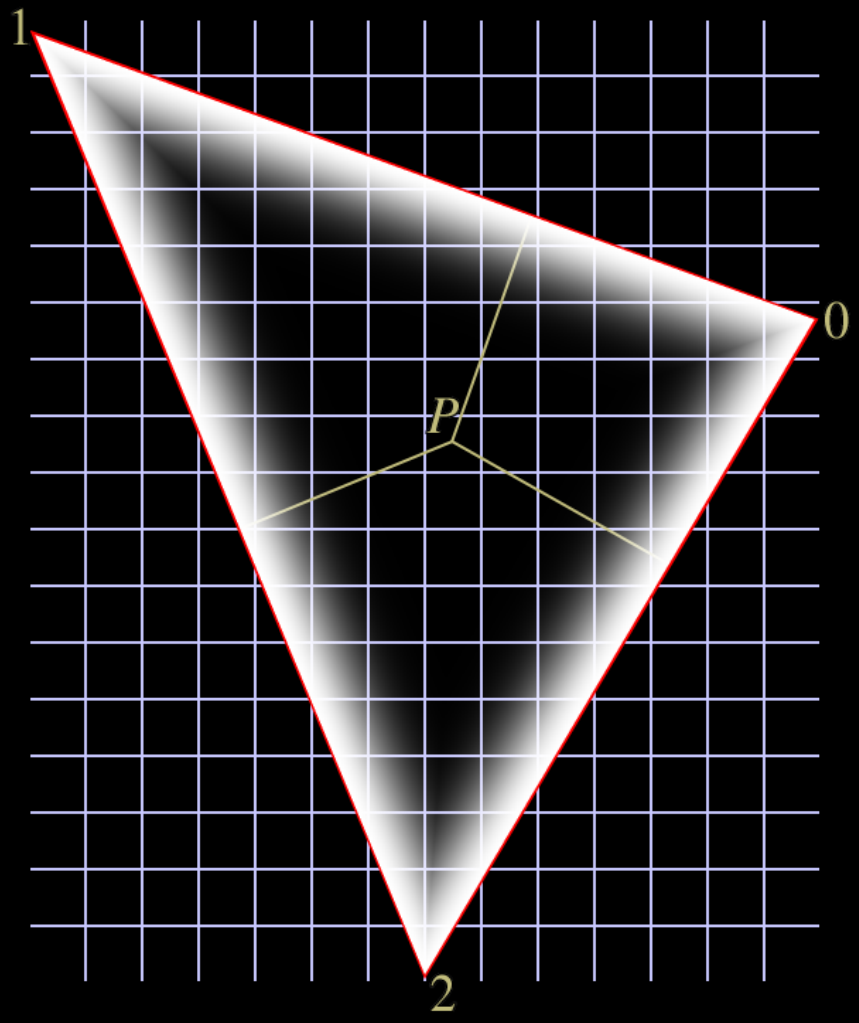
$$f = \frac{af_a/w_a + bf_b/w_b + cf_c/w_c}{a/w_a + b/w_b + c/w_c}$$

- However, it is easy to fix:
  - Negate perspective correction.
  - Wait for D3D 10 or OpenGL 3.0 hardware.



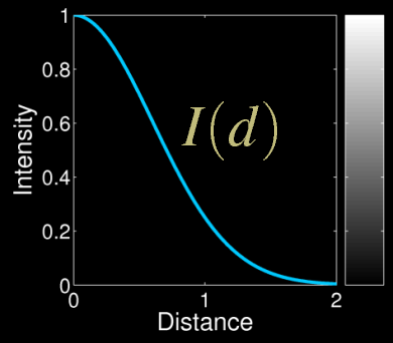
- Teapot using our method ... so far.

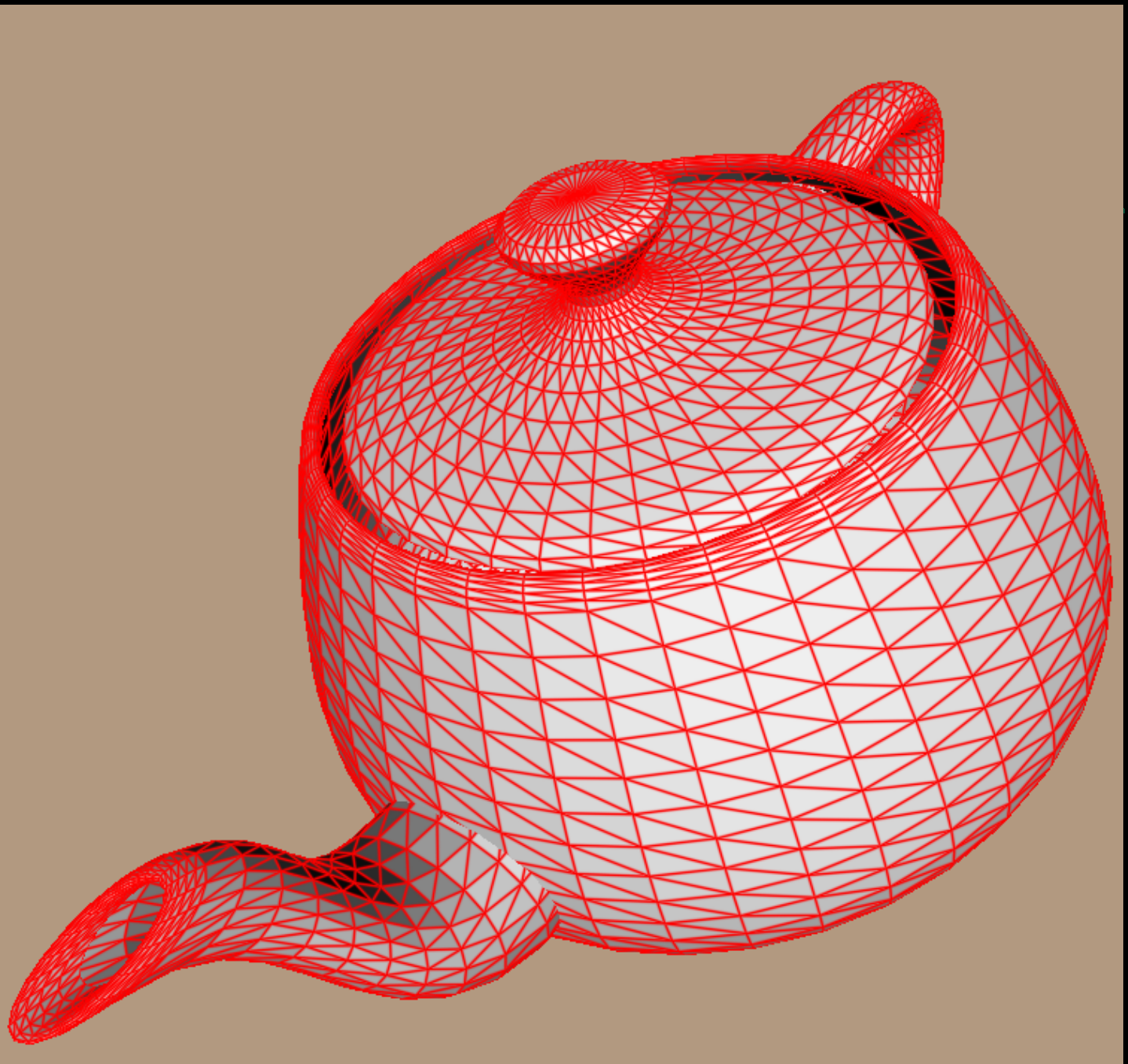
Preintegrated lines



- Color is now a function of distance:

$$d = \min(d_{01}^P, d_{12}^P, d_{20}^P);$$
$$\textbf{if } (d < d_{\text{thresh}})$$
$$\quad \text{col} = I(d) * \text{wire\_col} + (1 - I(d)) * \text{fill\_col};$$
$$\textbf{else}$$
$$\quad \text{col} = \text{fill\_col};$$

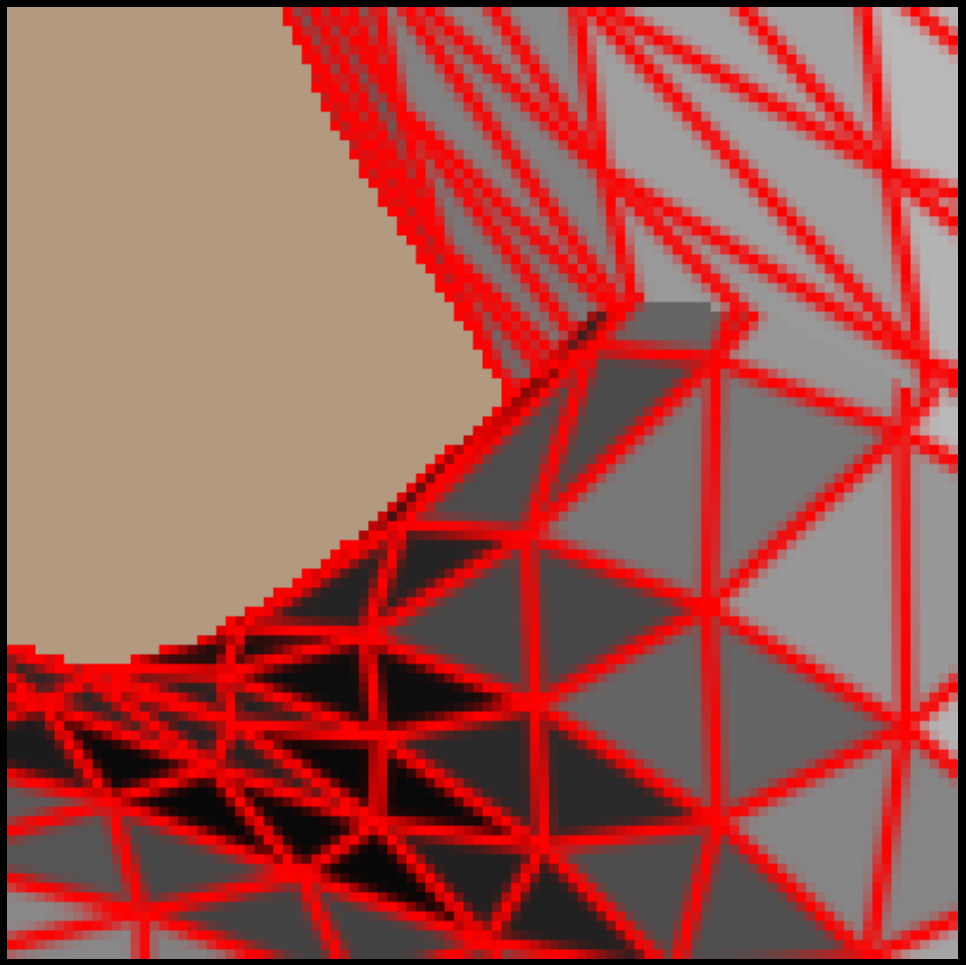




SIGGRAPH2006

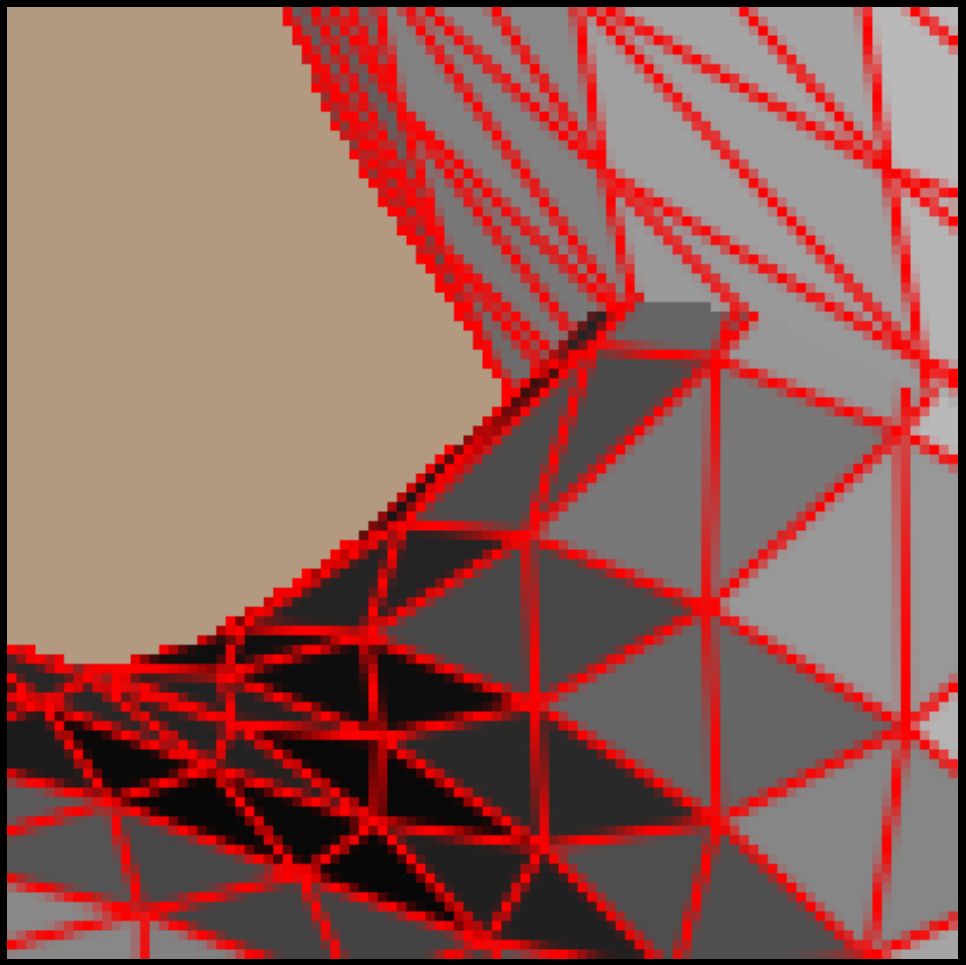
- Teapot using our method

Single-pass Method



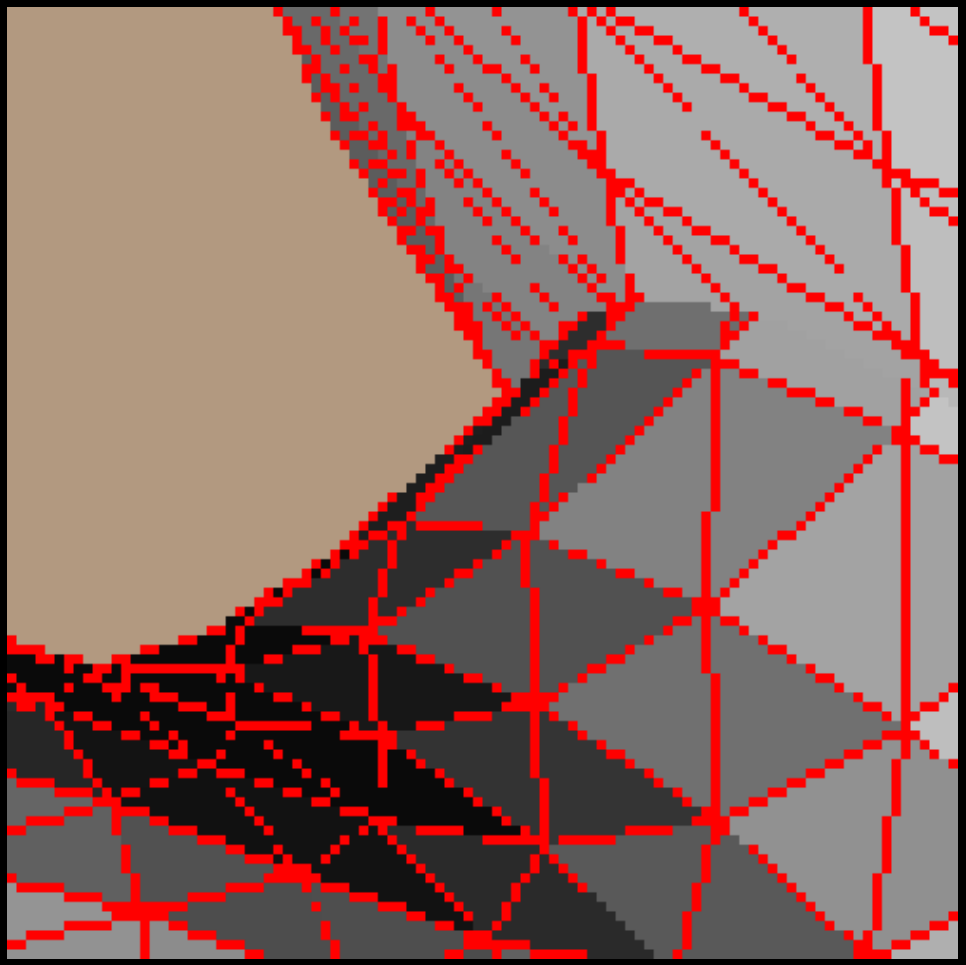
- Detail of teapot
- Notice smooth lines
- Perhaps lines are a little thick?

Single-pass Method



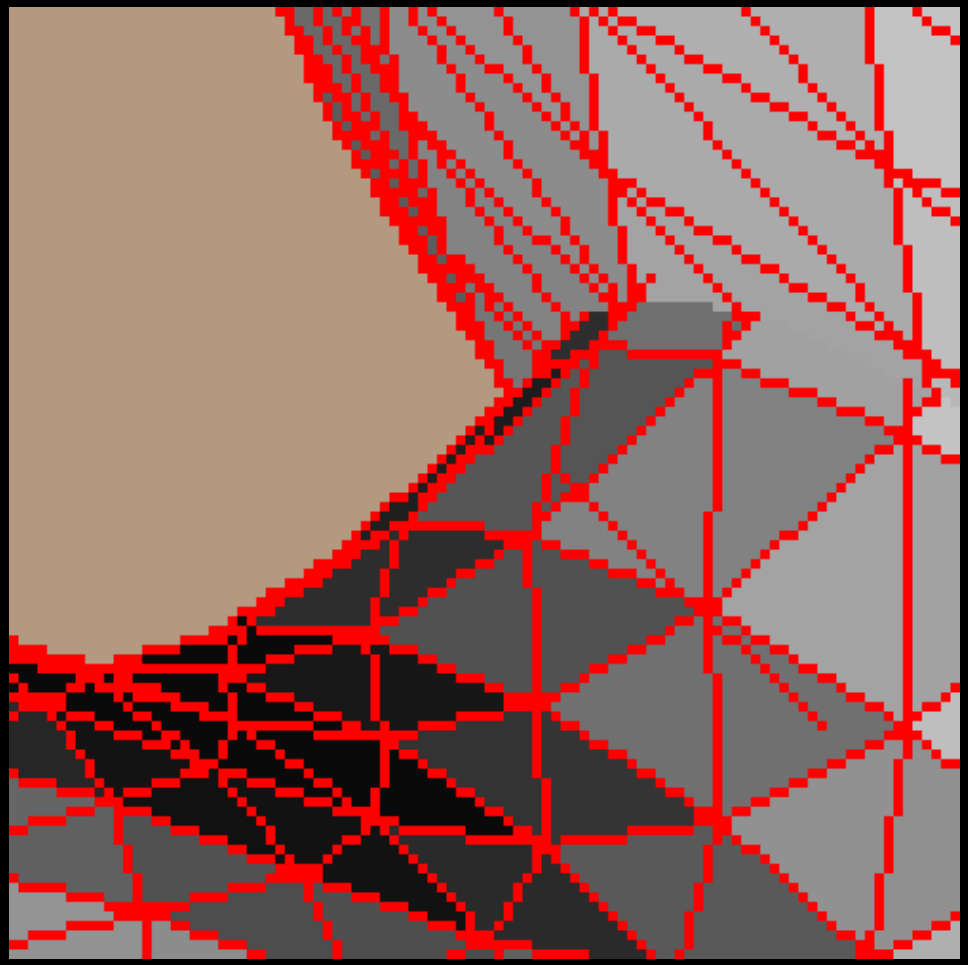
- Thinner lines are easy.
- However, we trade thinness for a little aliasing.

Offset-based method

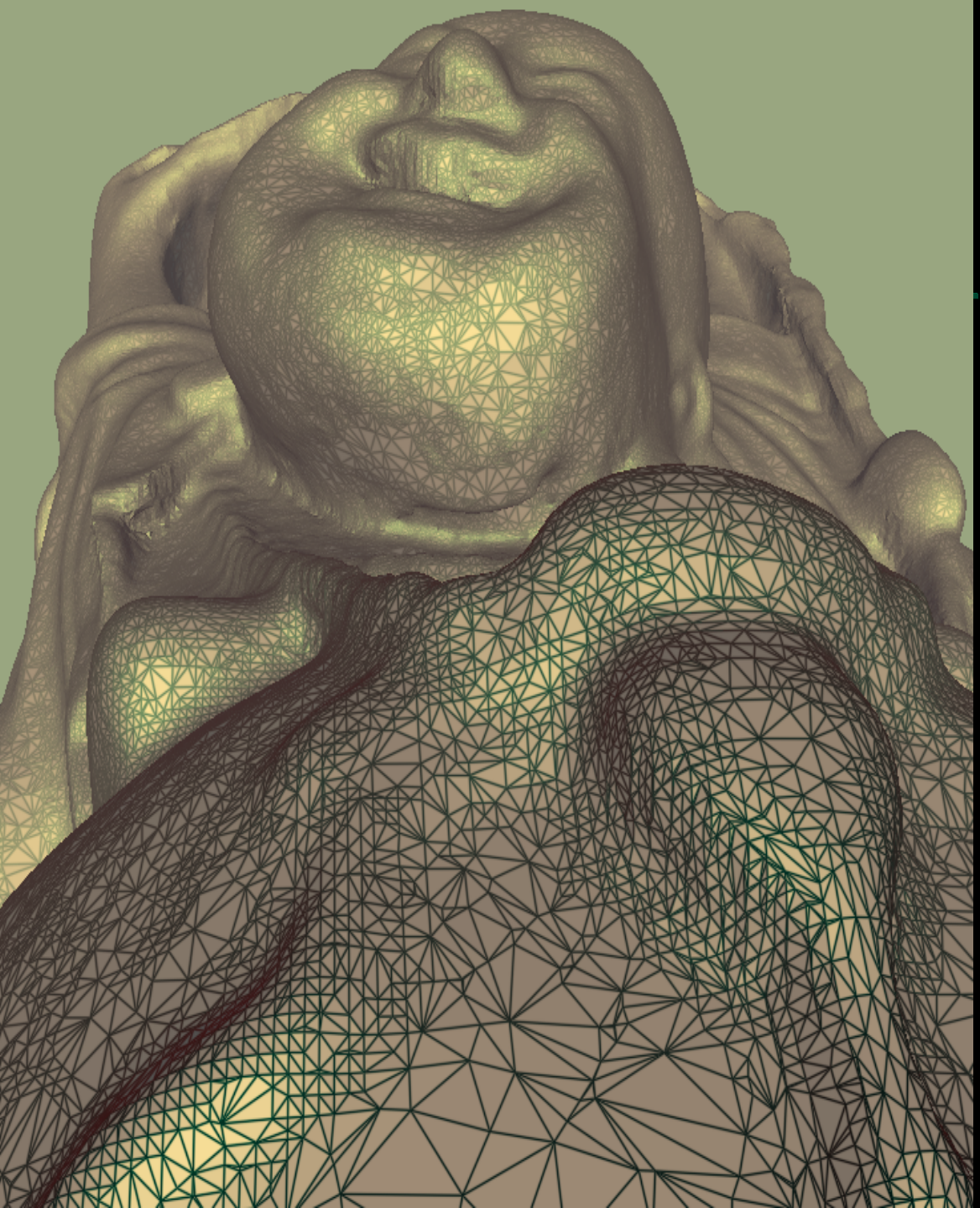


- Const offset = 2000
- Notice stippling
- Offset could be changed but there is no perfect value

Offset-based method

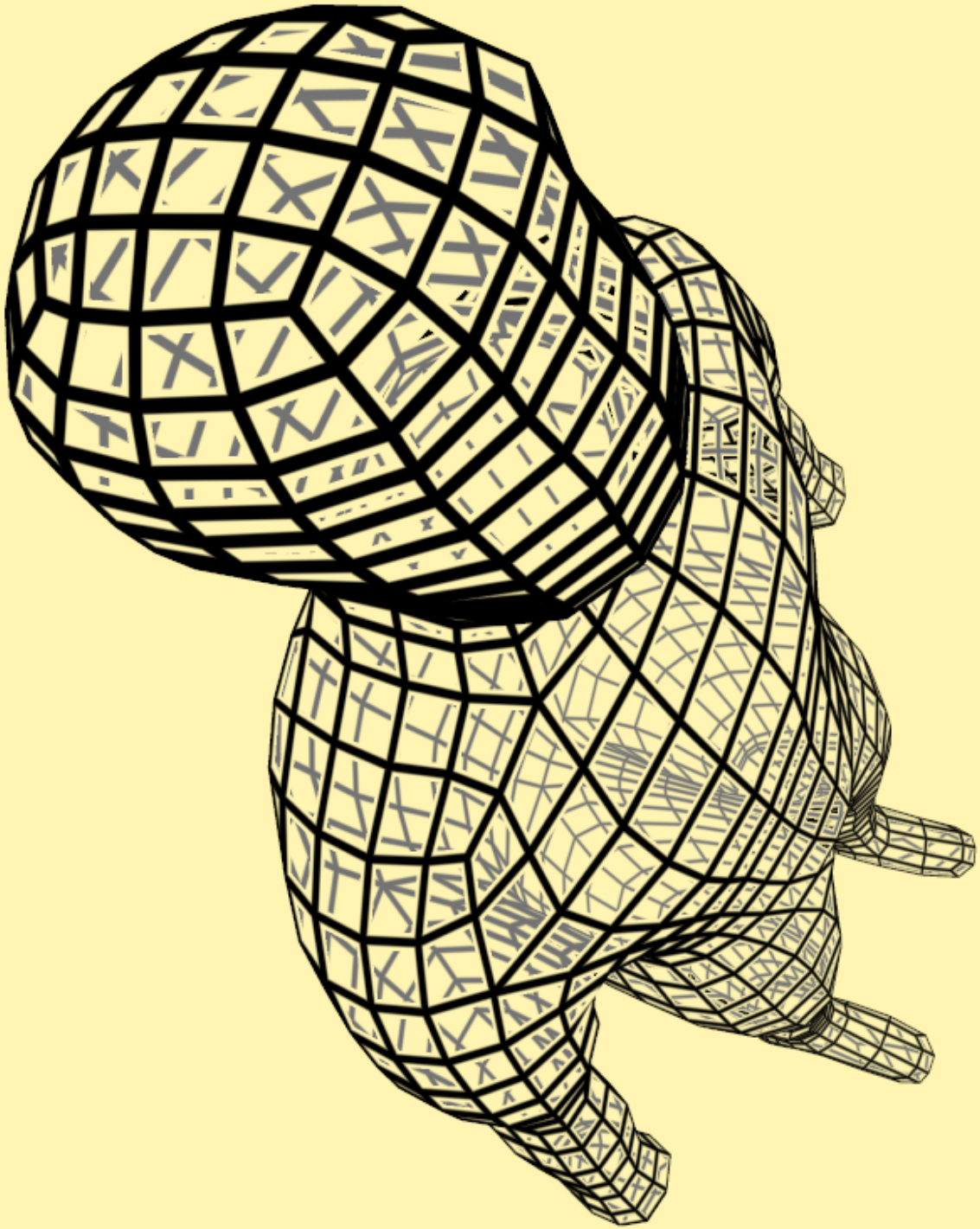


- Slope based offset
- Notice disocclusion artefacts



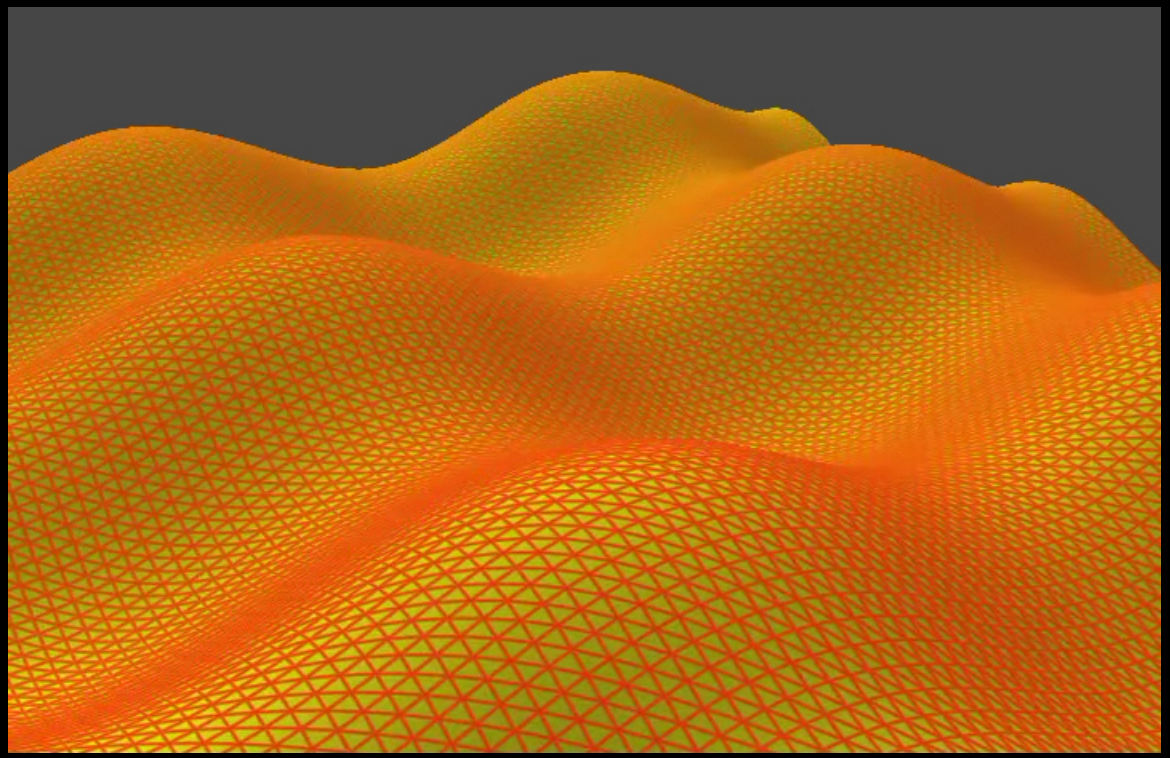
SIGGRAPH2006

- Use depth to fade out line color.



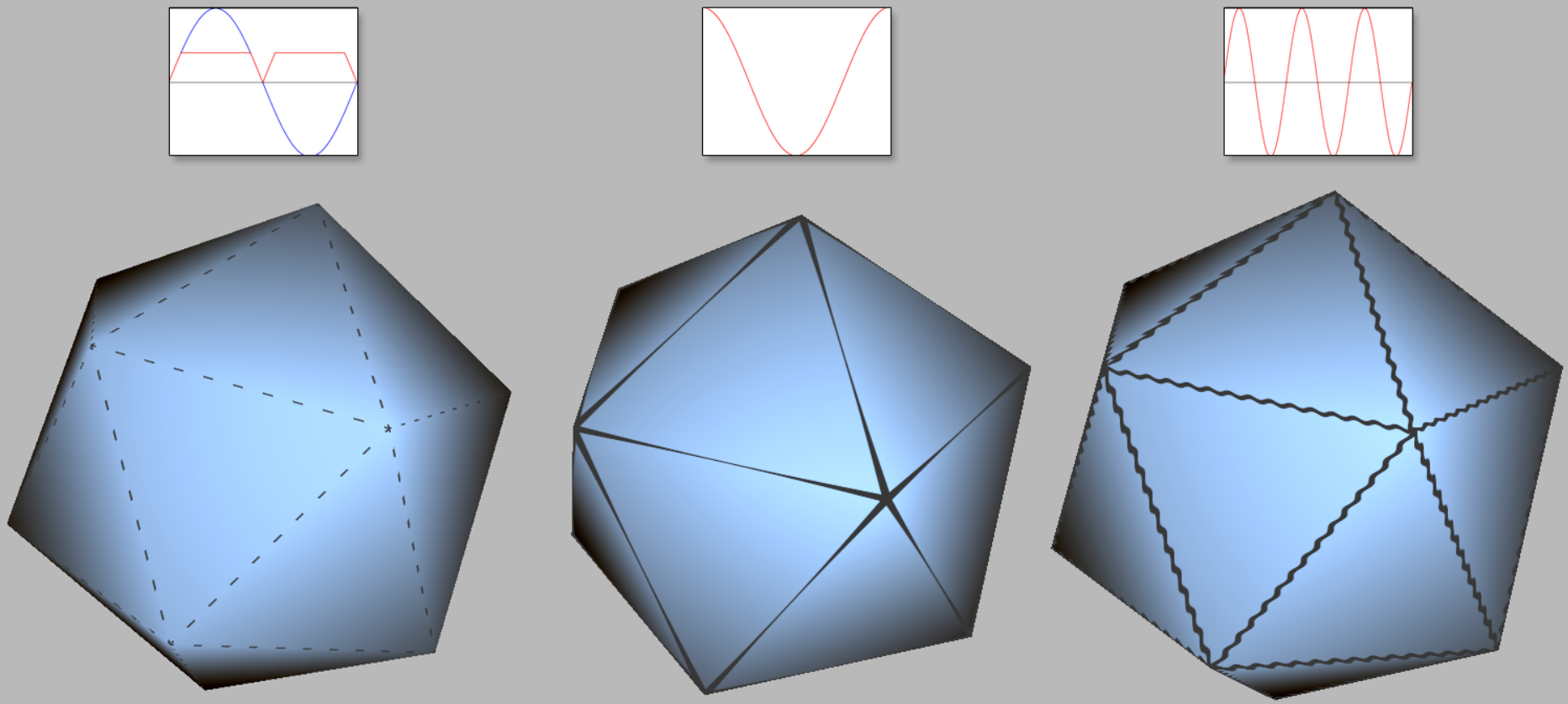
- Use depth to decide line thickness.
- Use alpha testing to remove interior of polygon.

Indexed Primitives



- Using the geometry shaders of D3D 10 allows us to draw wireframe triangle strips.

- Line Styles



Results

- Performance

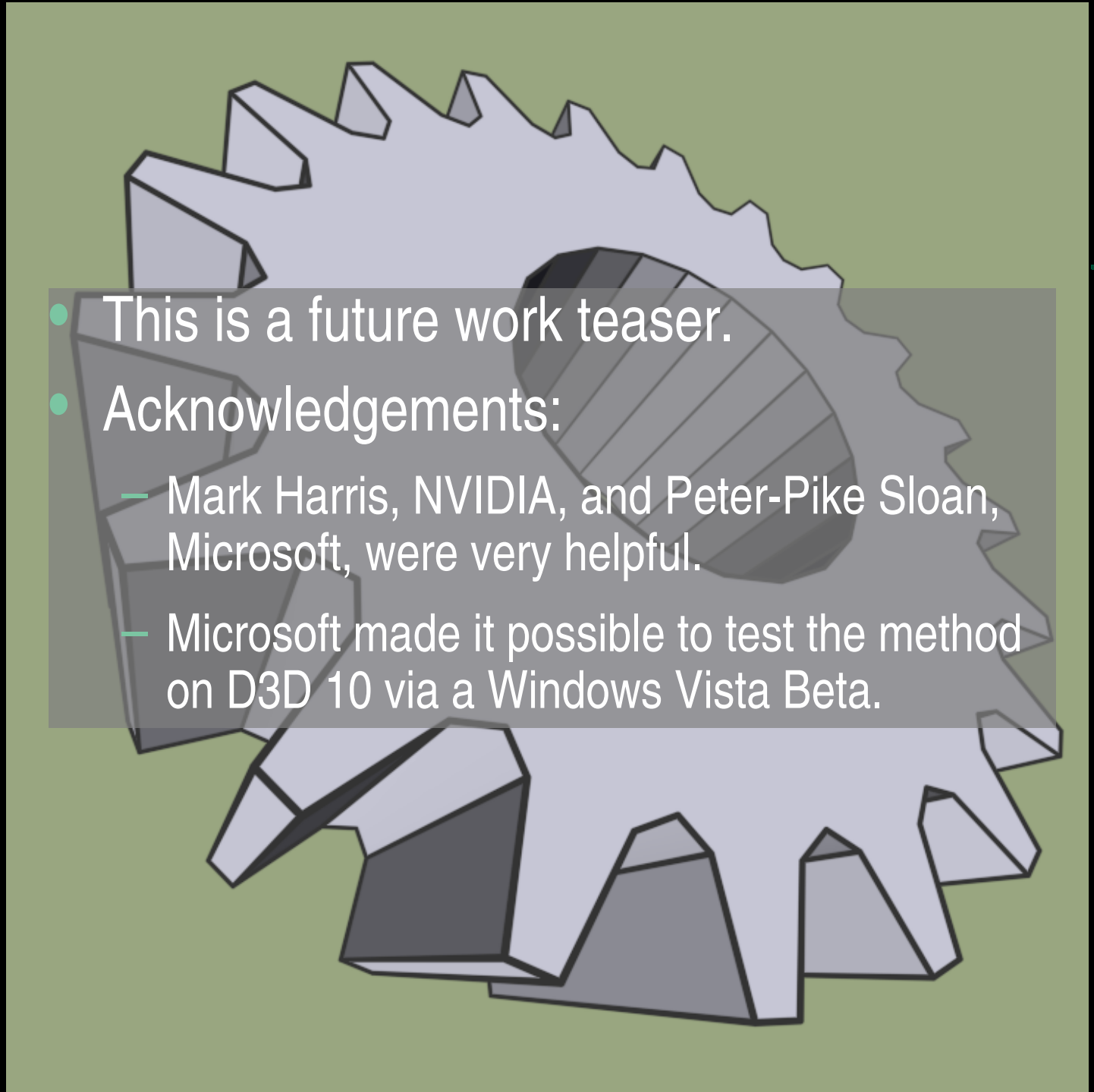
	GeForce FX5900XT	Quadro FX3000	GeForce 6600GT	GeForce 7800 GTX
Teapot, 6320 triangles				
offset	209.31	734.76	363.04	482.68
single	254.22	195.81	589.59	1512.09
Bunny, 69451 triangles				
offset	44.26	138.73	65.12	66.44
single	97.68	103.65	146.7	348.22
Buddha, 1087716 triangles				
offset	3.3	9.96	4.23	4.55
single	6.56	7.07	5.31	24.19

Issues

- The method does not support indexed geometry
- Primitive/geometry shaders alleviate this!
- Silhouette edges drawn from one side only
- Only noticeable on small meshes
- This is essentially a method for triangles and quadrilaterals
- yes – but that is a **very** common special case

Merits

- Method is fast
  - Single pass
  - Avoids line primitive
  - Little per-fragment computation
- High quality lines
  - Prefiltered
  - No Z issues
- Tweakable
  - Many line styles are possible



SIGGRAPH2006

- This is a future work teaser.
- Acknowledgements:
  - Mark Harris, NVIDIA, and Peter-Pike Sloan, Microsoft, were very helpful.
  - Microsoft made it possible to test the method on D3D 10 via a Windows Vista Beta.

Line types