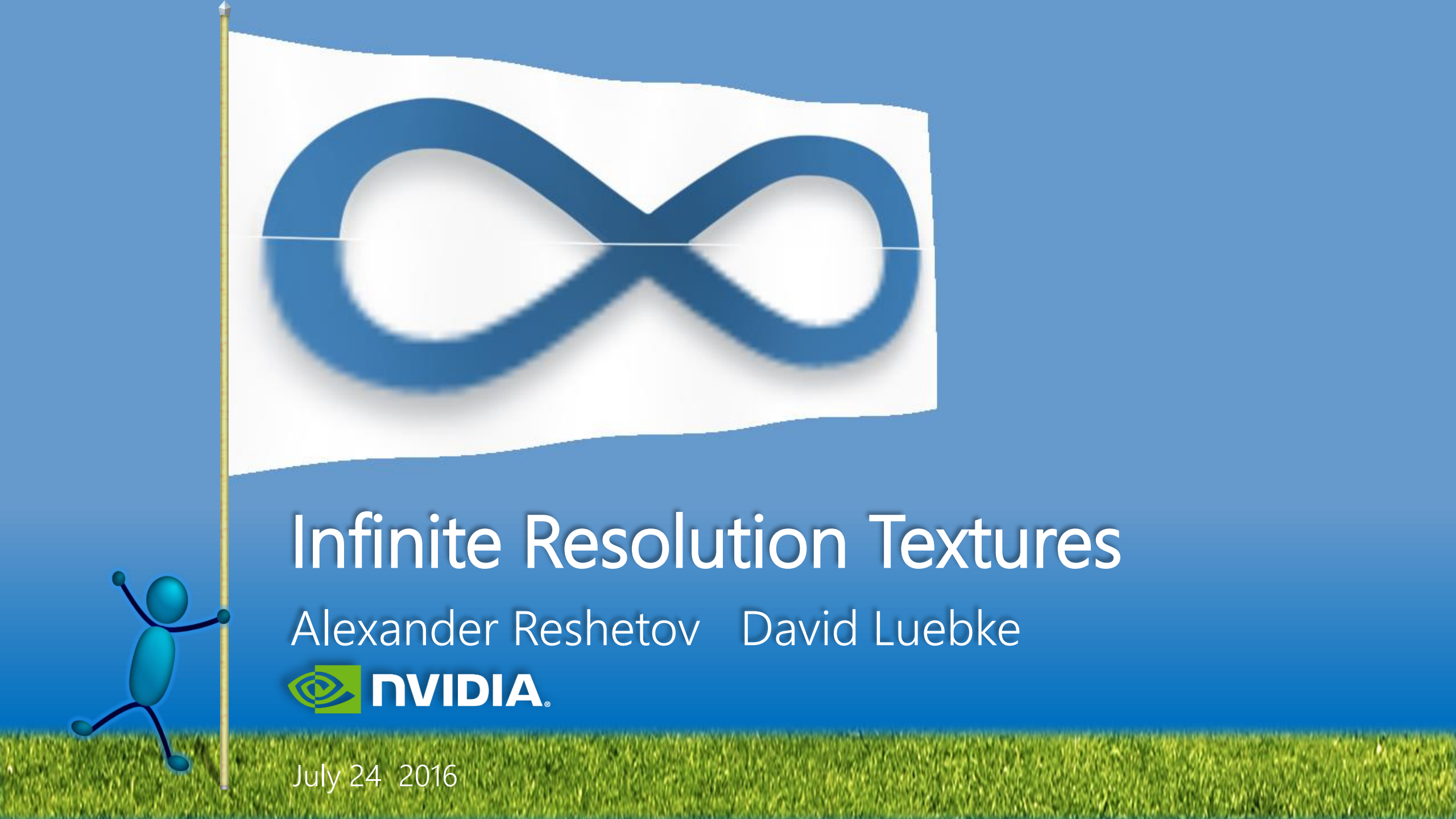


Infinite Resolution Textures

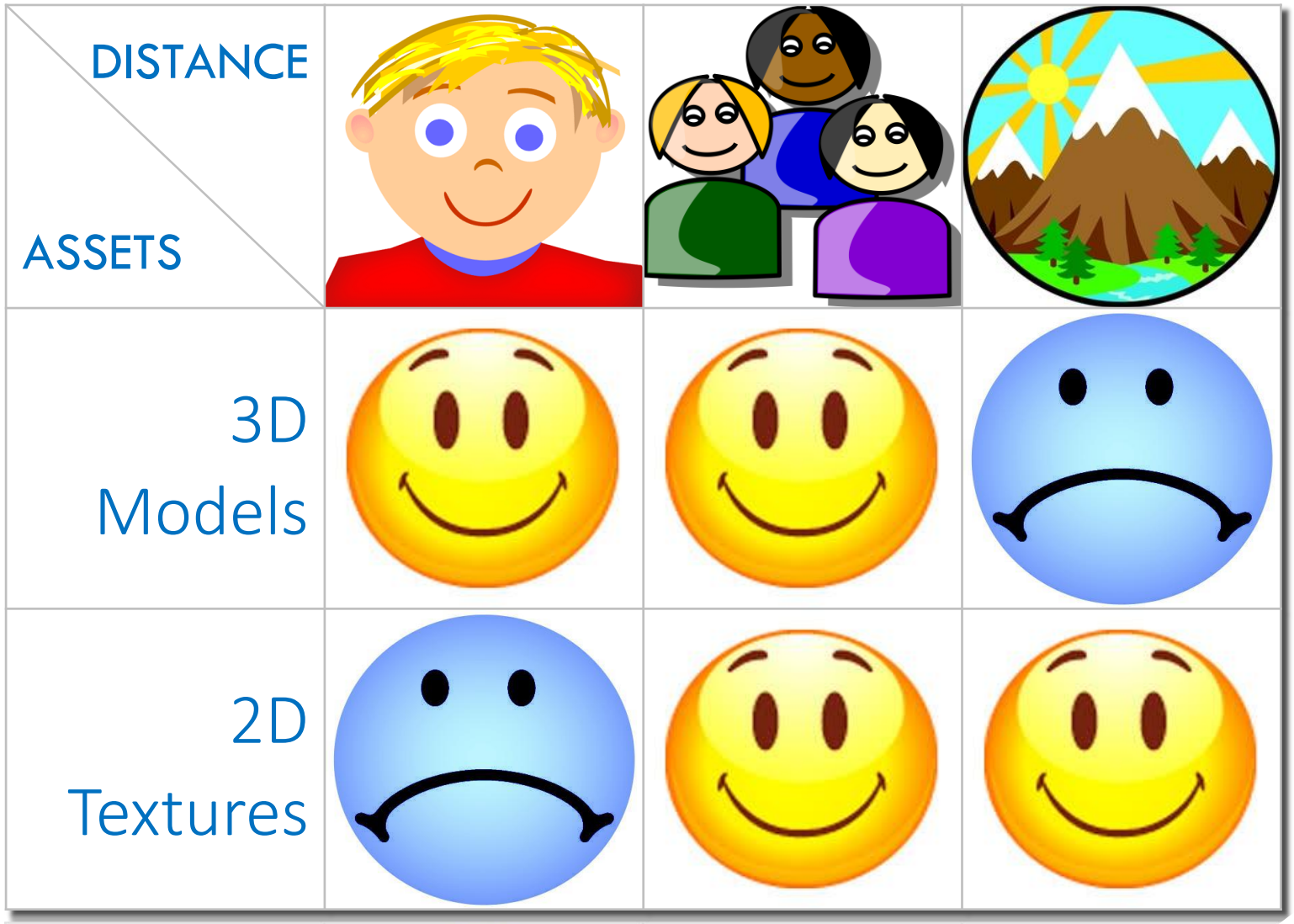
Alexander Reshetov David Luebke

NVIDIA

July 24 2016

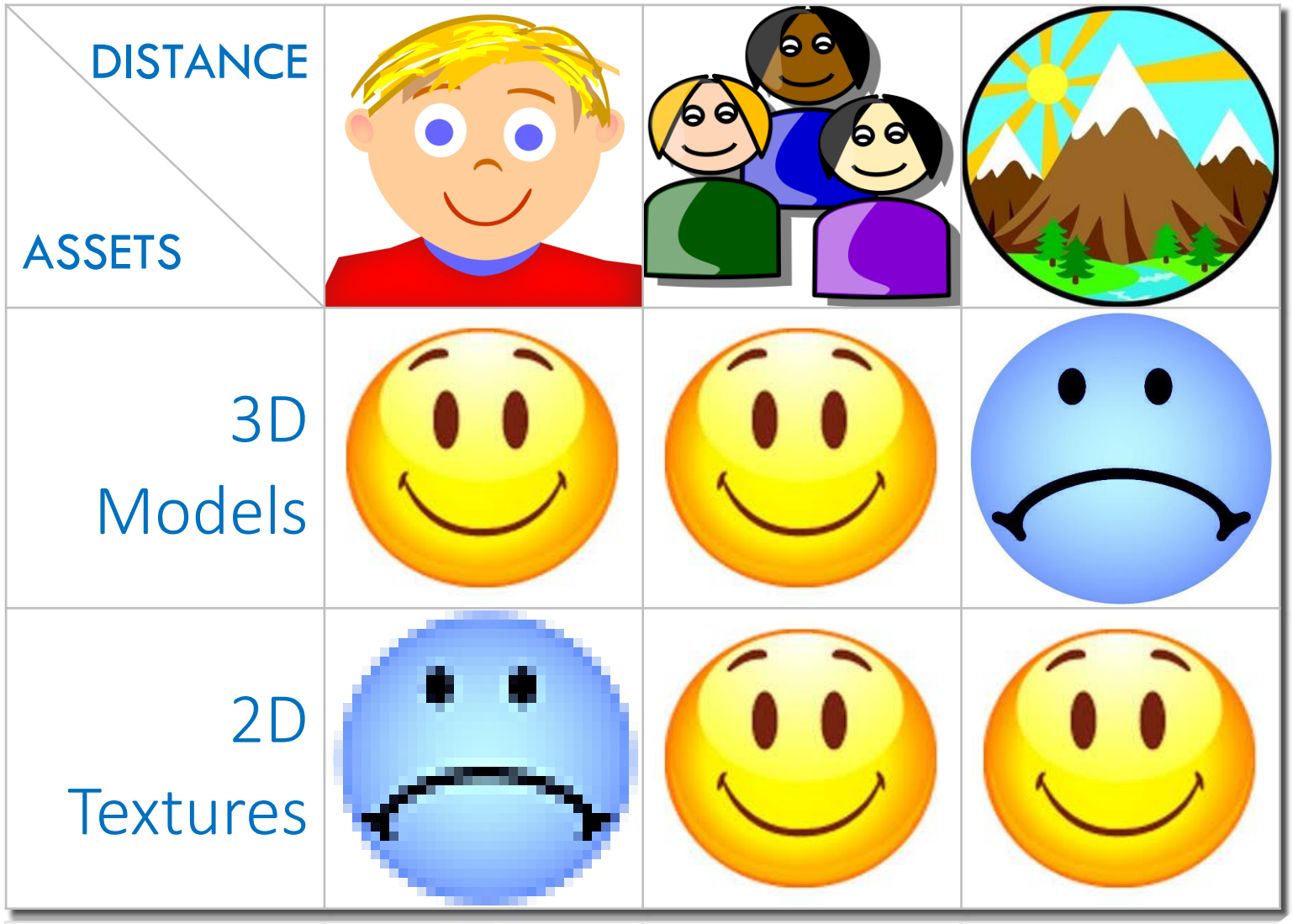
Motivation:

Computer Graphics at 30,000 feet



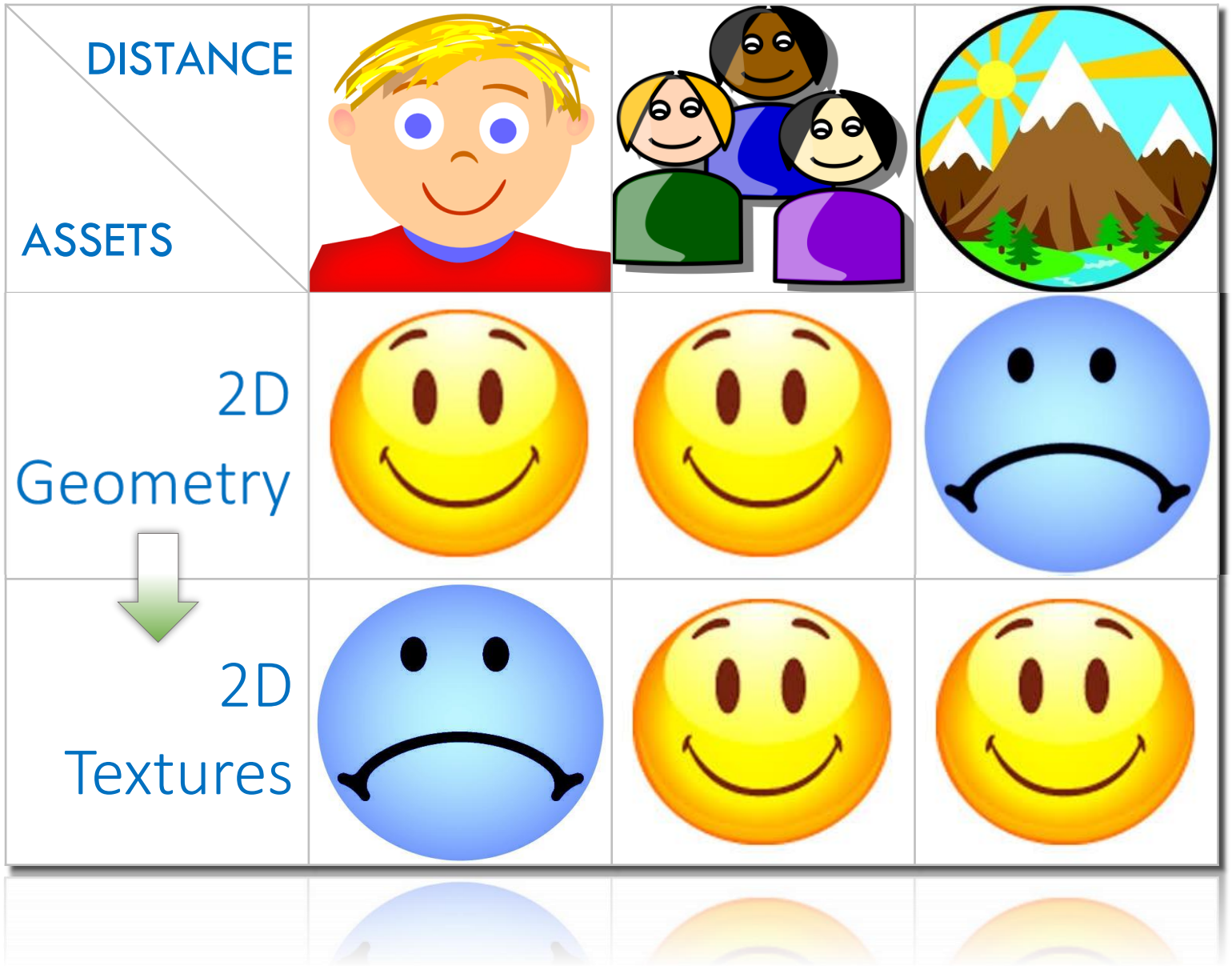
Motivation:

Computer Graphics at 30,000 feet



DISTANCE / ASSETS			
			
3D Models			
Infinite Resolution Textures			

An alternative approach



An alternative approach

DISTANCE / ASSETS			
3D Models			
2D Geometry			

Vector Graphics Evolution

- Before ‘70s — All Graphics is 2D Vector Graphics
  replaced with hardware-accelerated raster graphics but continue existence in professional applications
- 2005+ — GPU acceleration
  - Loop & Blinn — Curve Rendering
  - Kilgard & Bolz — Path Rendering
  - Ganacim et al. — Massively-parallel Vector Graphics
- 2016 — Ellis et al. — 3D Rendering to Vector Graphics

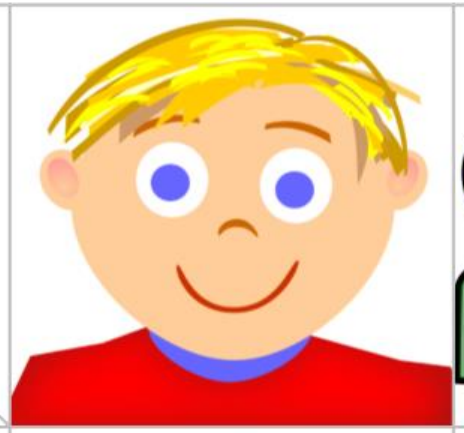
Before ’70s, all computer graphics was actually 2D vector graphics. It was changed with a hardware-accelerated texture sampling. 2D graphics continue to proliferate in professional applications where it was rendered in software.

It all changed in this century, when GPUs become universal enough to accelerate rendering of smooth curves, as proposed by Loop & Blinn.

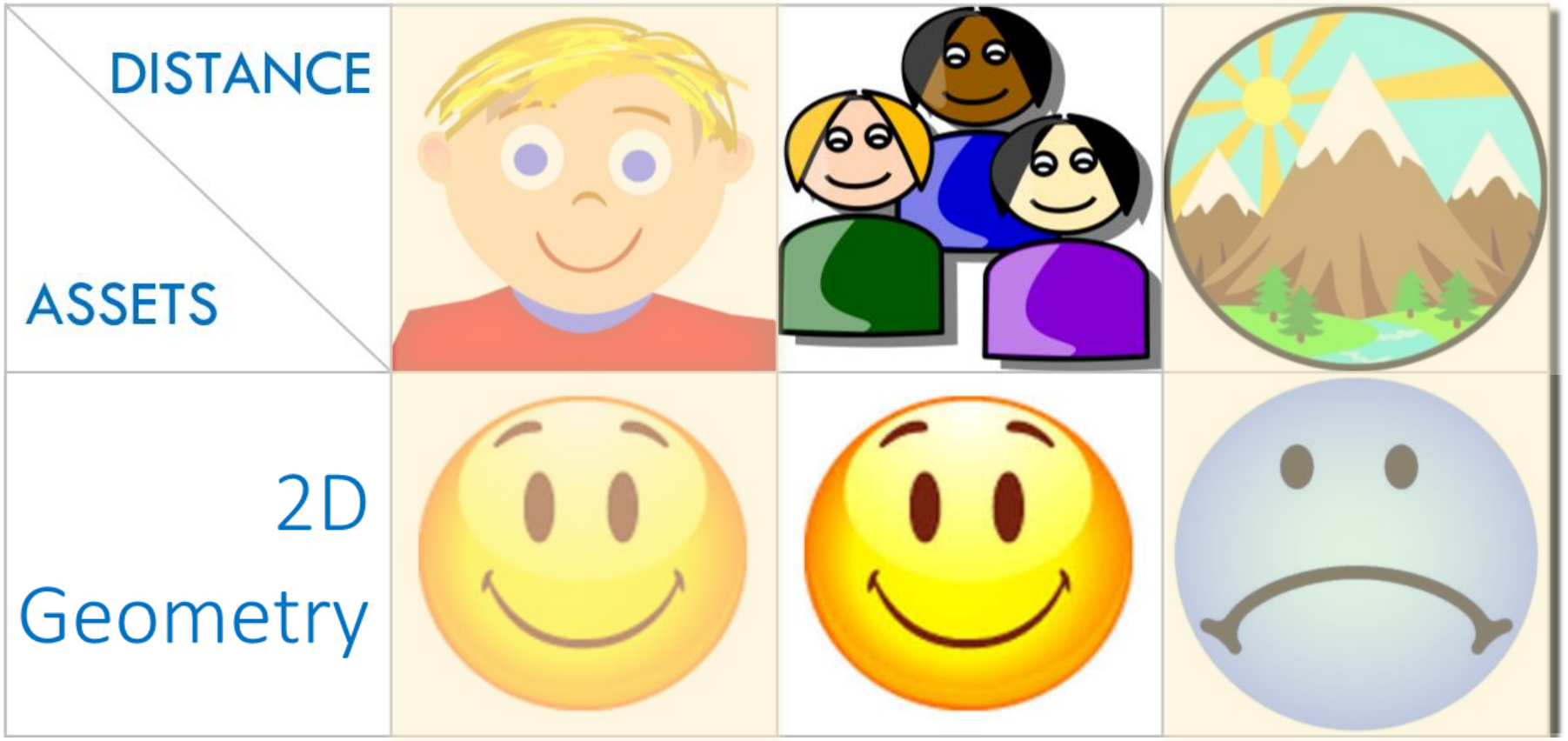
Kilgard and Bolz introduced a two-step Stencil, then Cover approach, allowing efficient GPU rendering of general vector textures. Ganacim et al. went further, employing an acceleration structure whose traversal enabled rendering parts of the image.

Now it is a part of Adobe products and you could also download NV Path Rendering SDK which is a part of GameWorks.

One of the most interesting – and unusual – papers at HPG was one by Ellis et al. who described a system that allows converting 3D scenes to vector graphics directly.

DISTANCE / ASSETS			
			
2D Geometry			

still a problem



no random sampling

Kilgard & Bolz ≅ rasterization

Ganacim et al. ≅ tiled rasterization

still a problem

We aim at a more general approach seamlessly combining raster and vector representations.

IR texture = raster image + silhouettes @ grid

```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```

IRT from an application standpoint

- instead of

```
float4 color = colorMap.SampleLevel(colorSampler, uv, lod);
```

- use

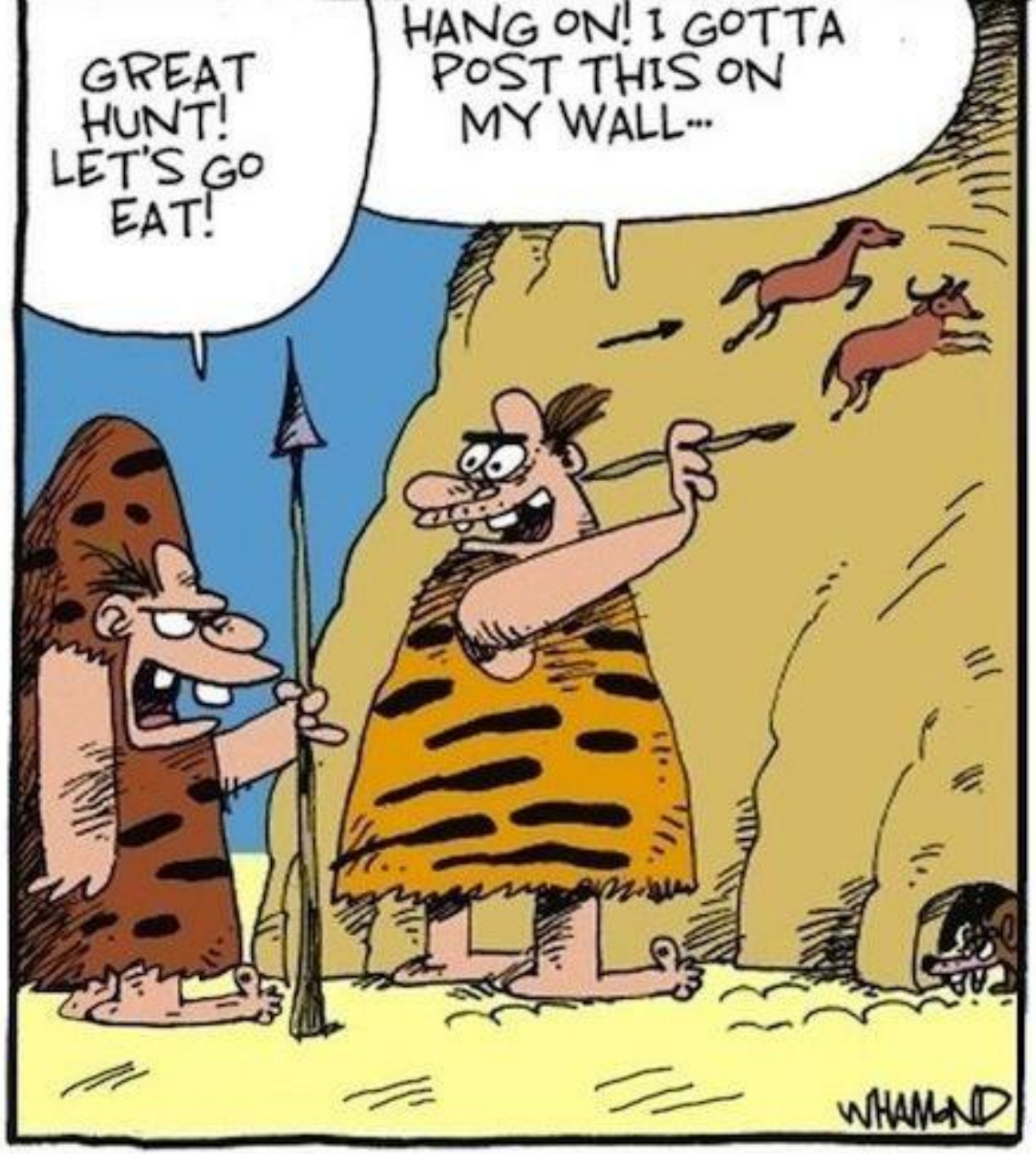
```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```

IRT calculates duv at runtime by evaluating distances to the precomputed silhouette edges

Just by tempering duv, we can blend between

- IRT (@ closeups) and
- traditional textures at a distance

Prior Art



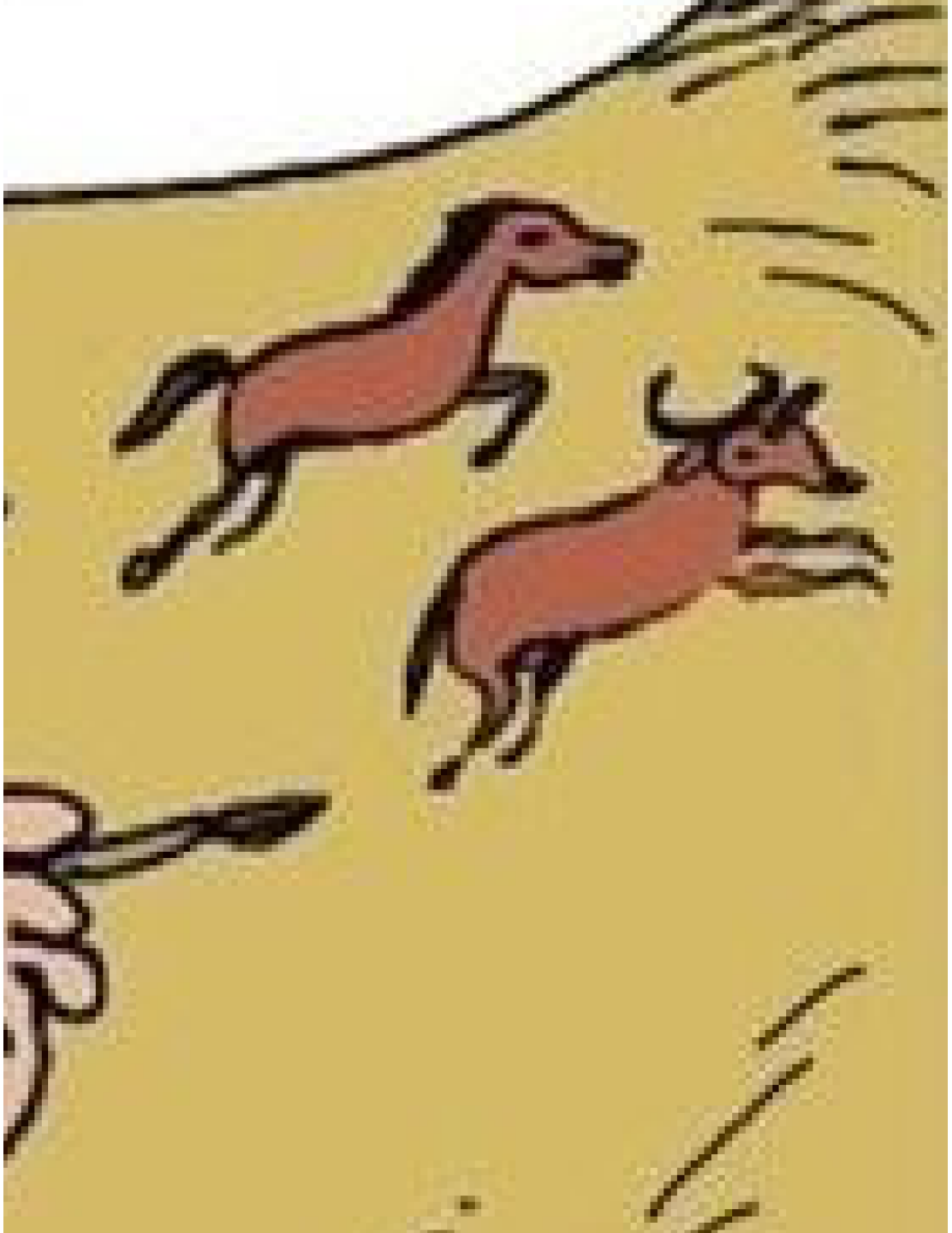
Early Facebook

https://www.pinterest.com/mizzchanty/facebook-quotes
pinned from sharenator.com

IRT = edges + raster image

hi freq

low freq

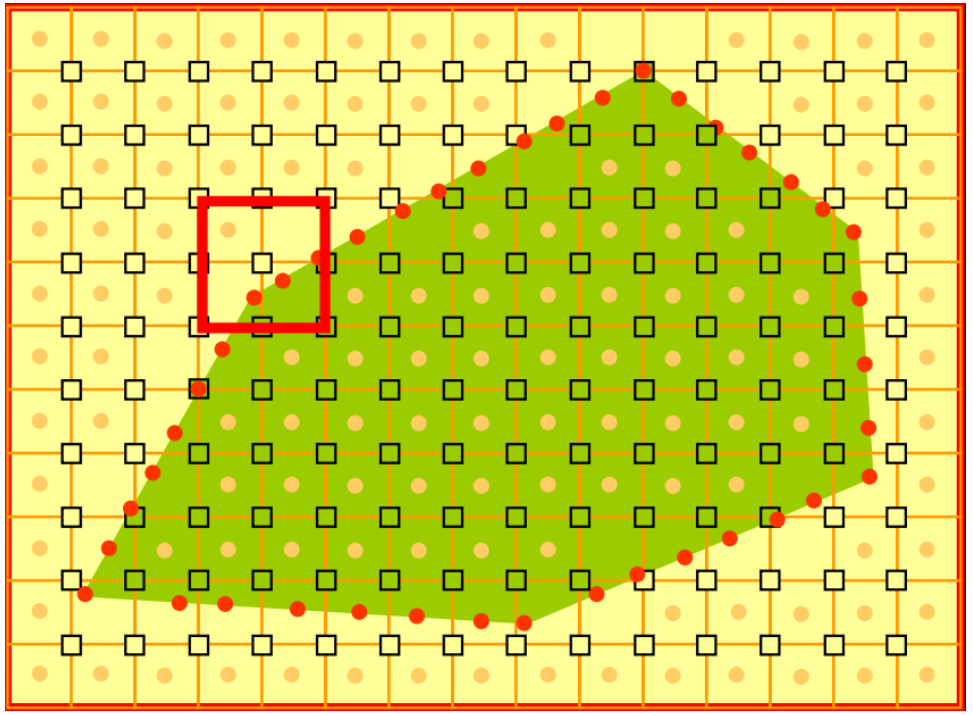


$\mapsto$
IRT

pinned from Scarlett Image

Prior Art circa 2005

- **Silmaps**
  Pradeep Sen
- Bixels
  Jack Tumblin,
  Prasun Choudhury
- Vector Texture Maps
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson
- Pinchmaps
  Marco Tarini,
  Paolo Cignoni



- piecewise-linear edges
- always interpolating colors on the same side of the edge
- with a custom interpolation scheme

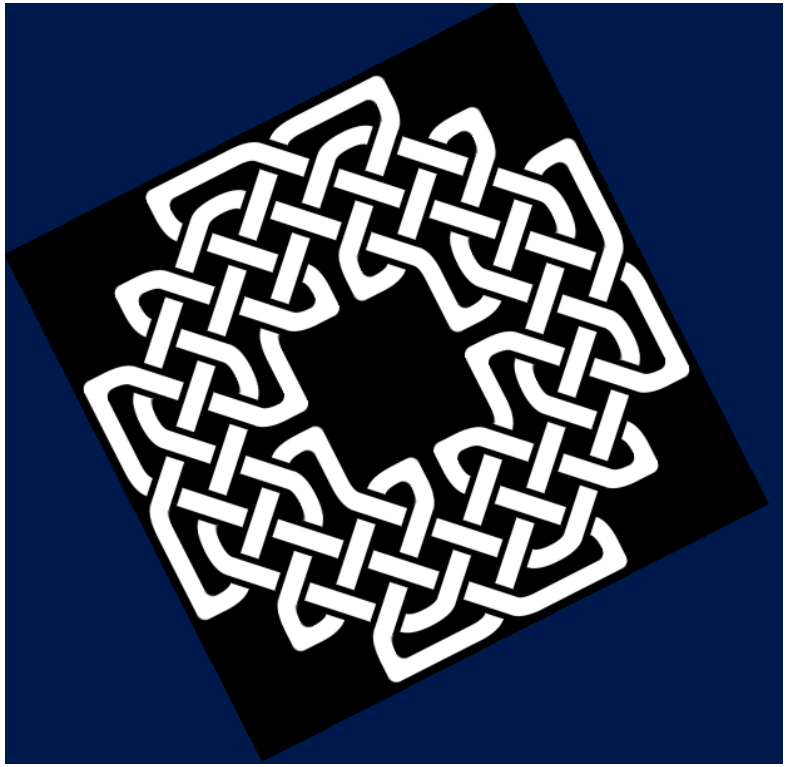
Prior Art circa 2005

- Silmaps
  Pradeep Sen
- **Bixels**
  Jack Tumblin,
  Prasun Choudhury
- Vector Texture Maps
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson
- Pinchmaps
  Marco Tarini,
  Paolo Cignoni

- decompose the texture plane into patches with straight boundary segments
- 10 patch functions

Prior Art circa 2005

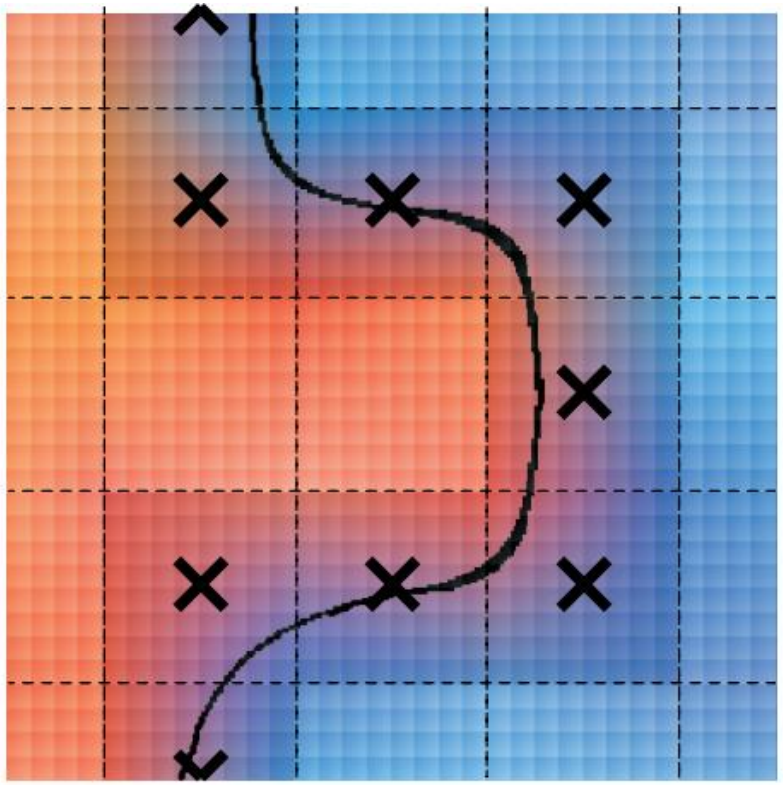
- Silmaps
  Pradeep Sen
- Bixels
  Jack Tumblin,
  Prasun Choudhury
- **Vector Texture Maps**
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson
- Pinchmaps
  Marco Tarini,
  Paolo Cignoni



- implicit cubic polynomials for edges
- binary classification function defines a patch

Prior Art circa 2005

- Silmaps
  Pradeep Sen
- Bixels
  Jack Tumblin,
  Prasun Choudhury
- Vector Texture Maps
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson
- **Pinchmaps**
  Marco Tarini,
  Paolo Cignoni



- a single quadratic silhouette edge per *pinchmap* texel
- use distance to the edge to compute new uv

pinchmaps

IRT

pinchmaps

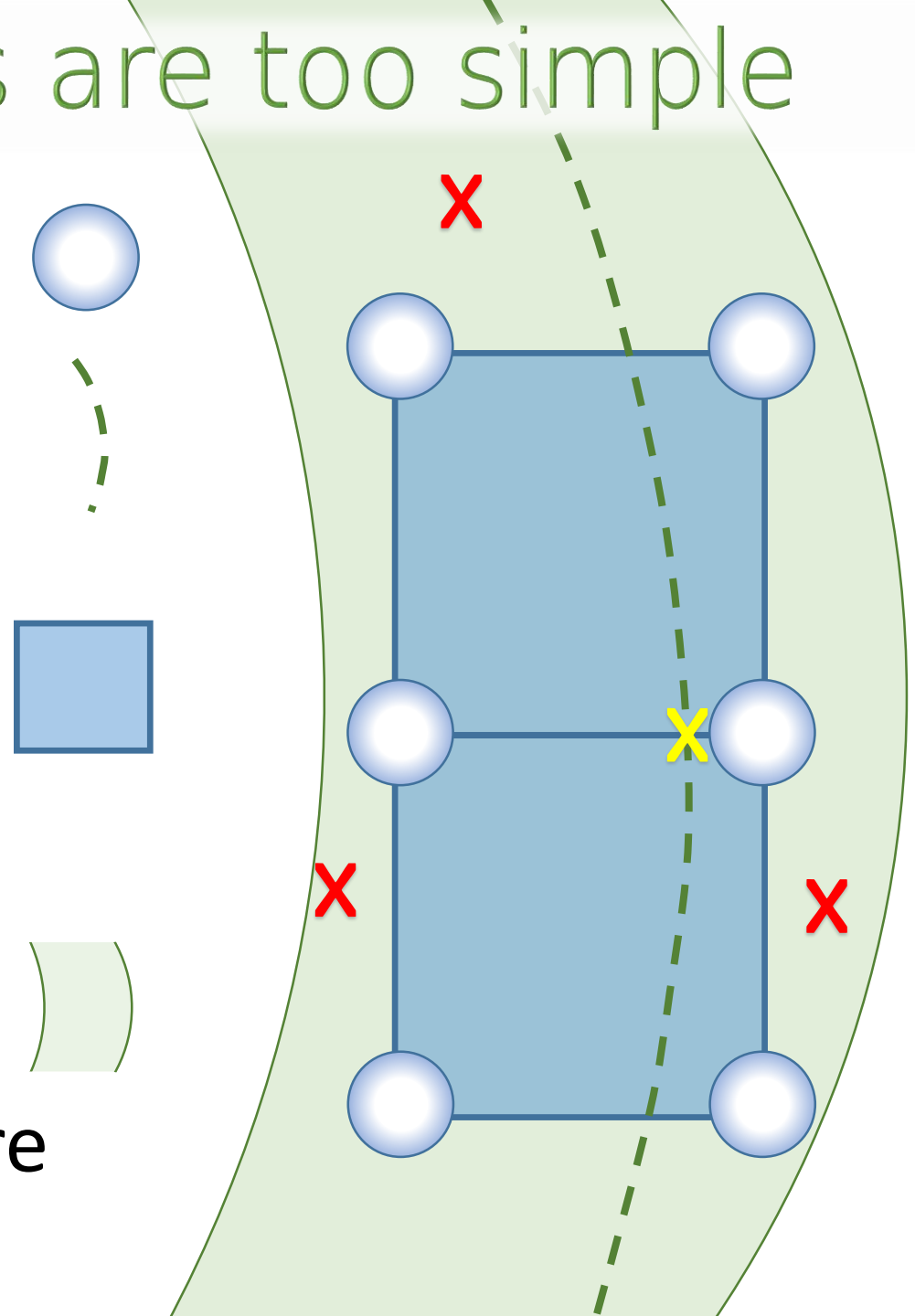
IRT

Occam vs Einstein

- **Occam's Razor**
  the simpler one is usually better
- **Einstein Principle**
  "a scientific theory should be as simple as possible, but no simpler"

Pinchmaps are too simple

- pinchmap texels
- define an implicit quadratic curve,
- so all samples that have 4 pinchmap texels…
- …will be resampled from the original texture



Issues

- No intersections

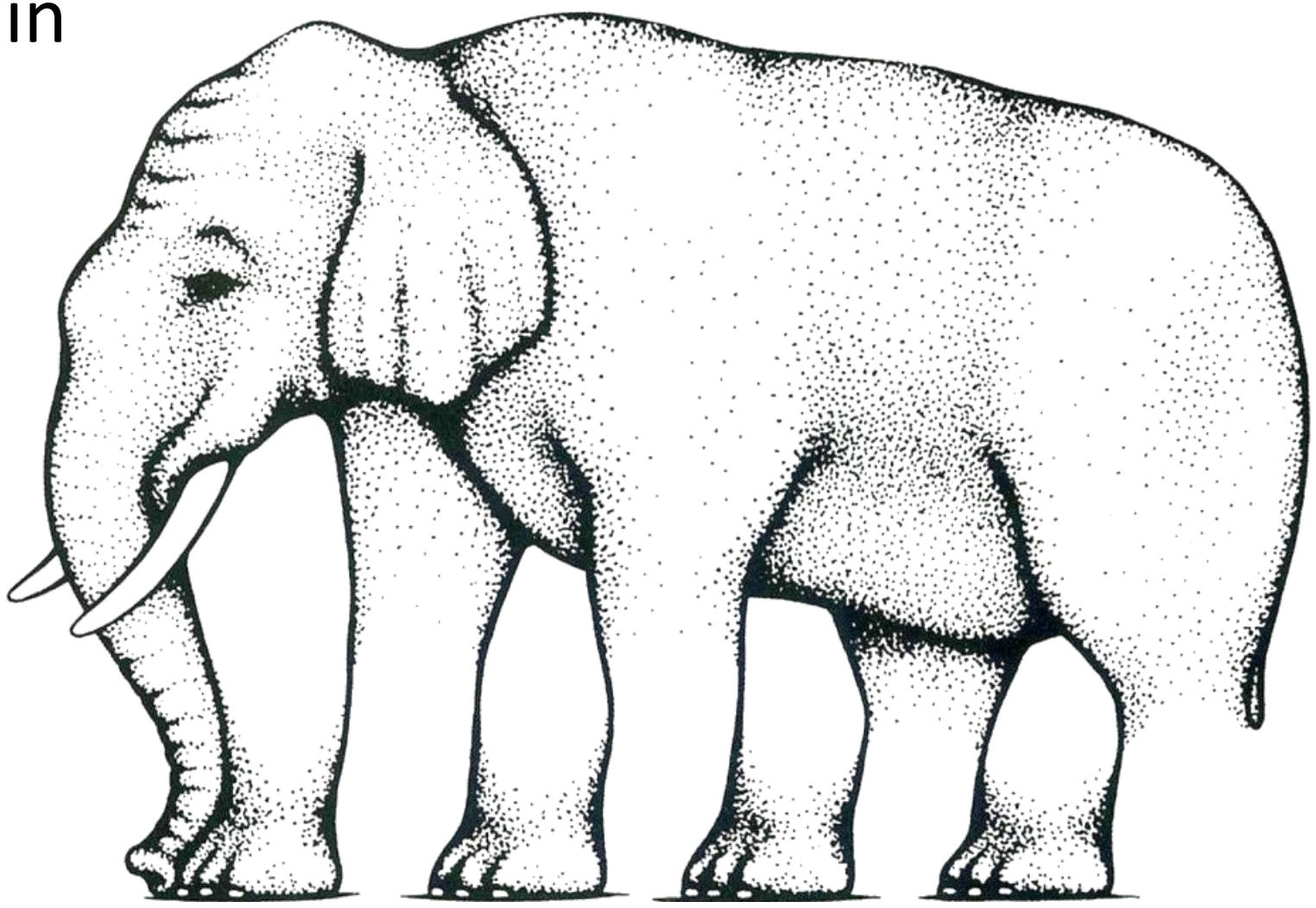
x 2nd degree only

x zero adjustment for all ‘outside’ samples ⇒ discontinuous duv

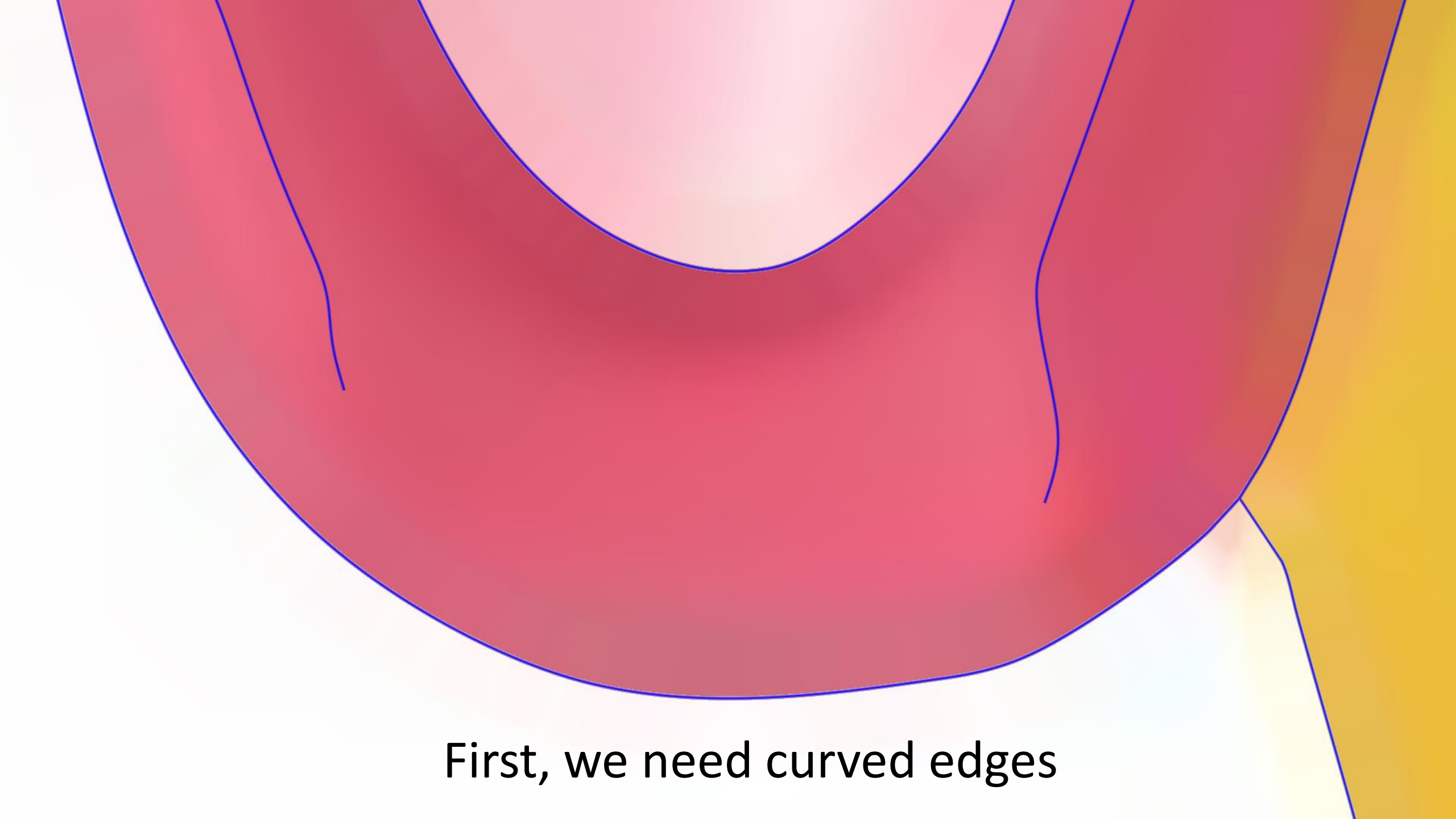
To address these problems…

…IRT uses more evolved processing…

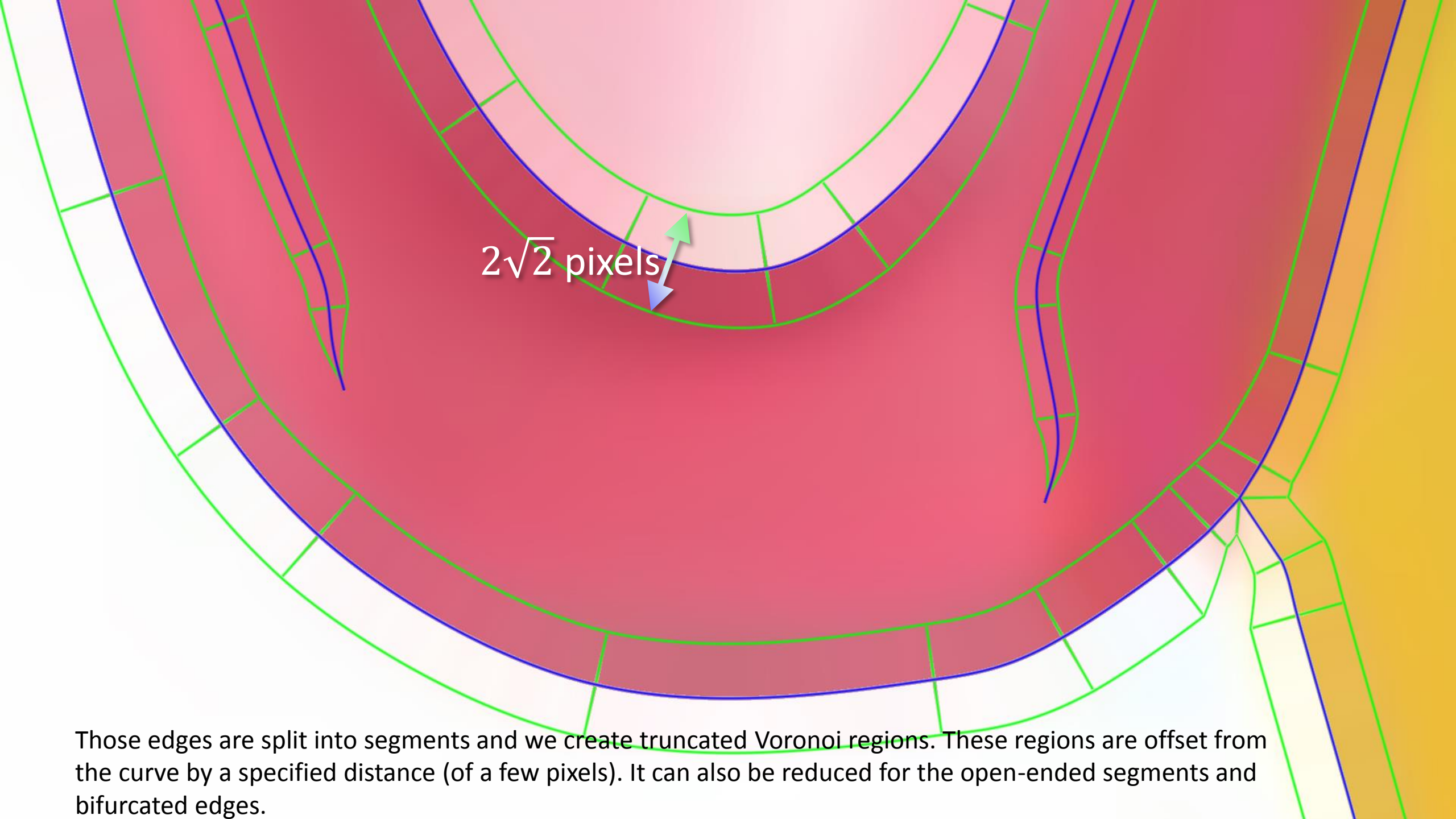
…that is easier to explain



legs-isential quandary
by Roger N. Shepard

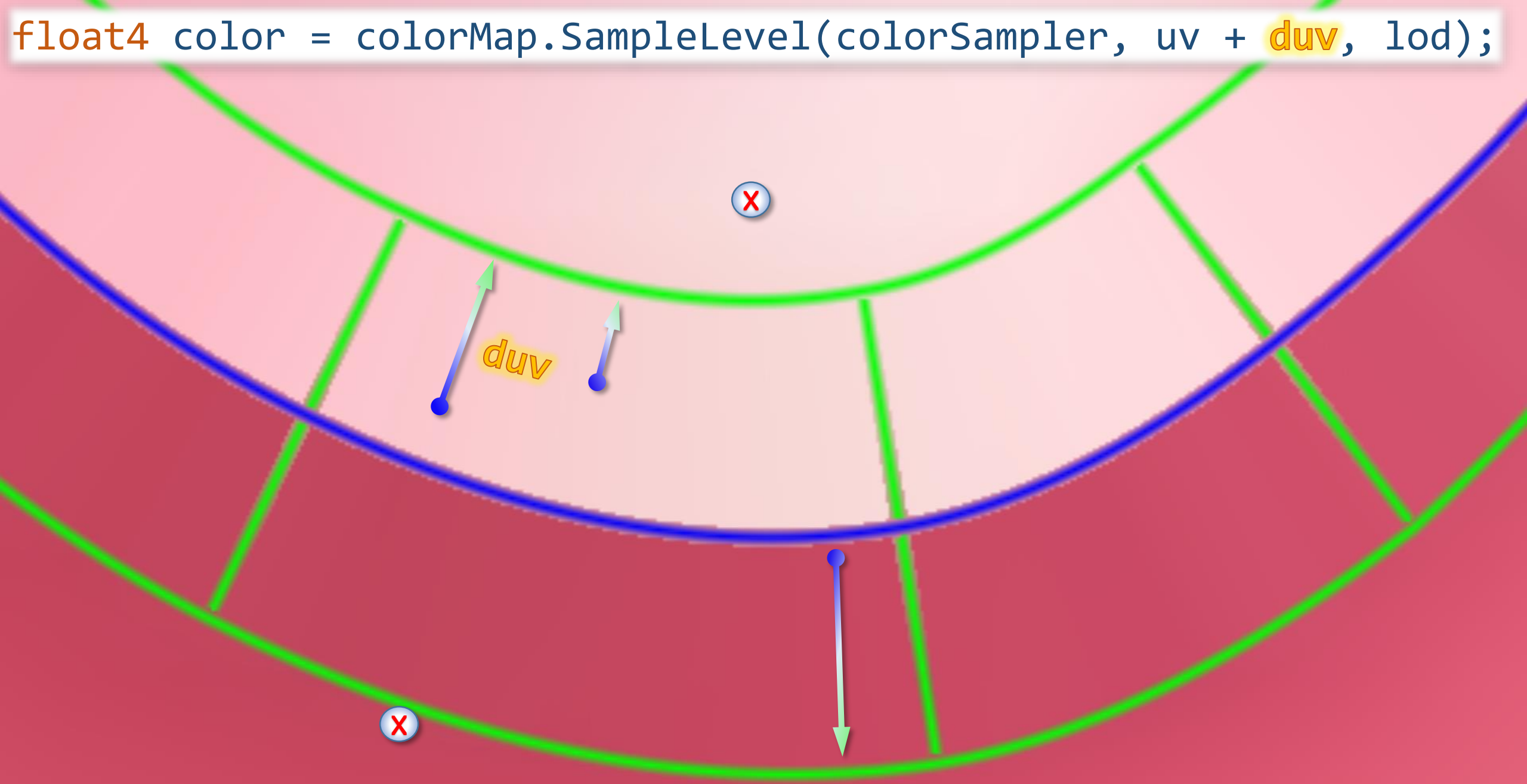


First, we need curved edges



Those edges are split into segments and we create truncated Voronoi regions. These regions are offset from the curve by a specified distance (of a few pixels). It can also be reduced for the open-ended segments and bifurcated edges.

```
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```



At run time, we just move the sample away from the edge. The samples outside Voronoi regions (X) will have zero duv

opportunities

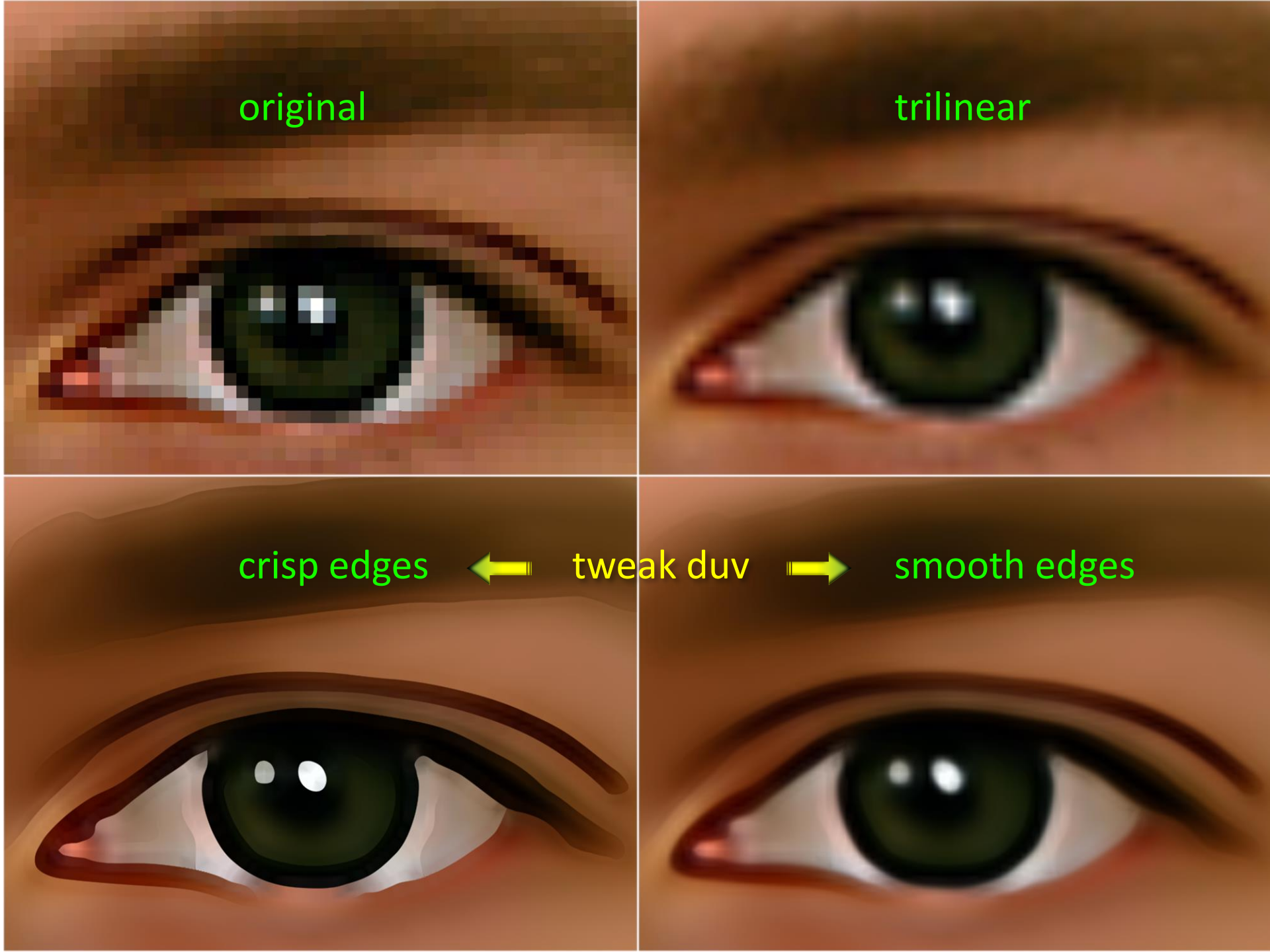
1. Temper* raster and vector modes just by scaling the texture coordinate adjustment `duv` using pixel/texel ratio as

```
float  pixratio = 0.5*length(fwidth(uv * texdim));
duv *= min(1, 2 * (1 - pixratio)) / texdim;
```

2. Perform antialiasing in a single fetch by adjusting `lod`
3. Do whatever we like with it (like ‘soft landing’)

* having the elements mixed in satisfying proportions
http://www.merriam-webster.com/dictionary/tempered

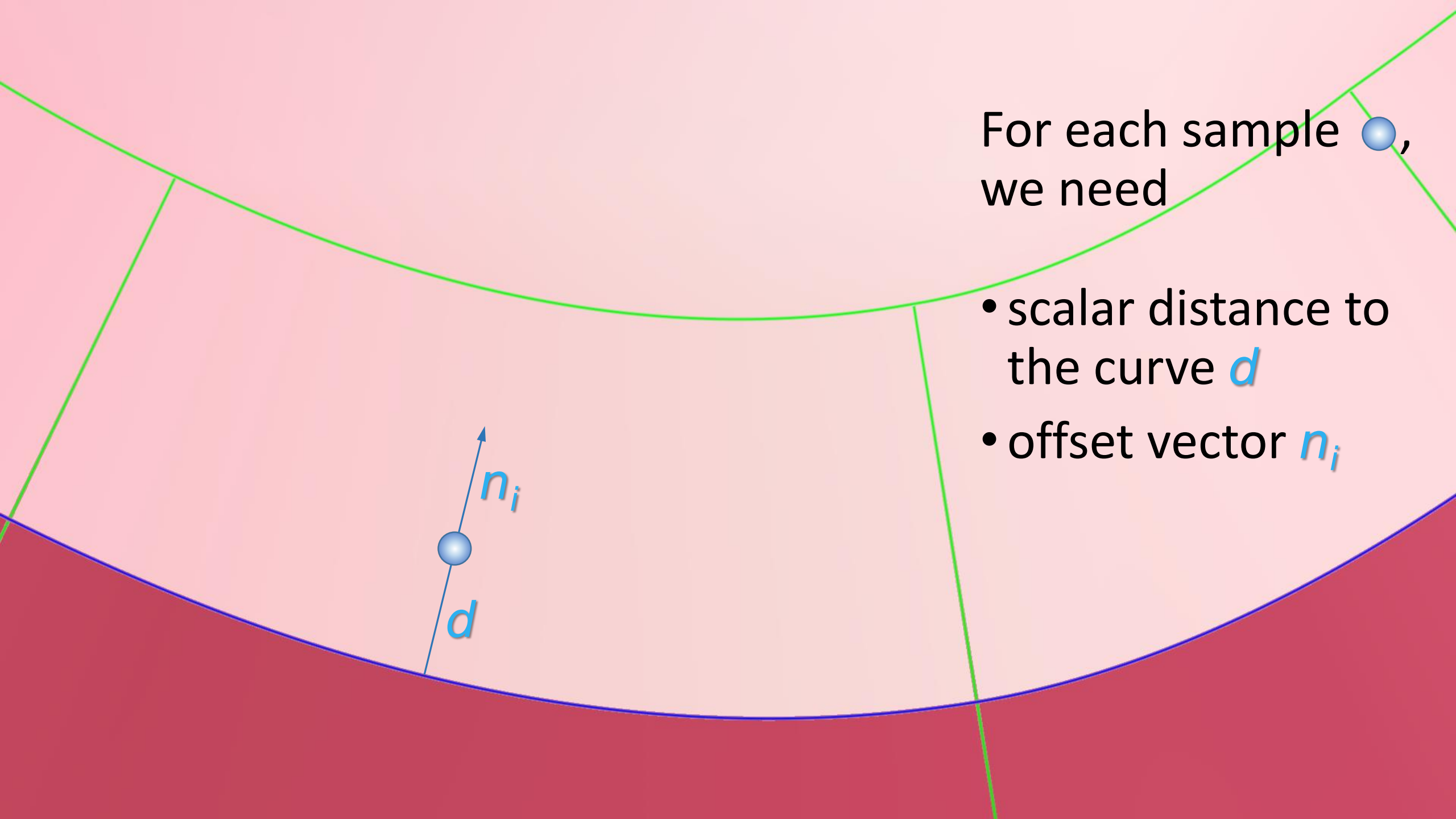
soft landing

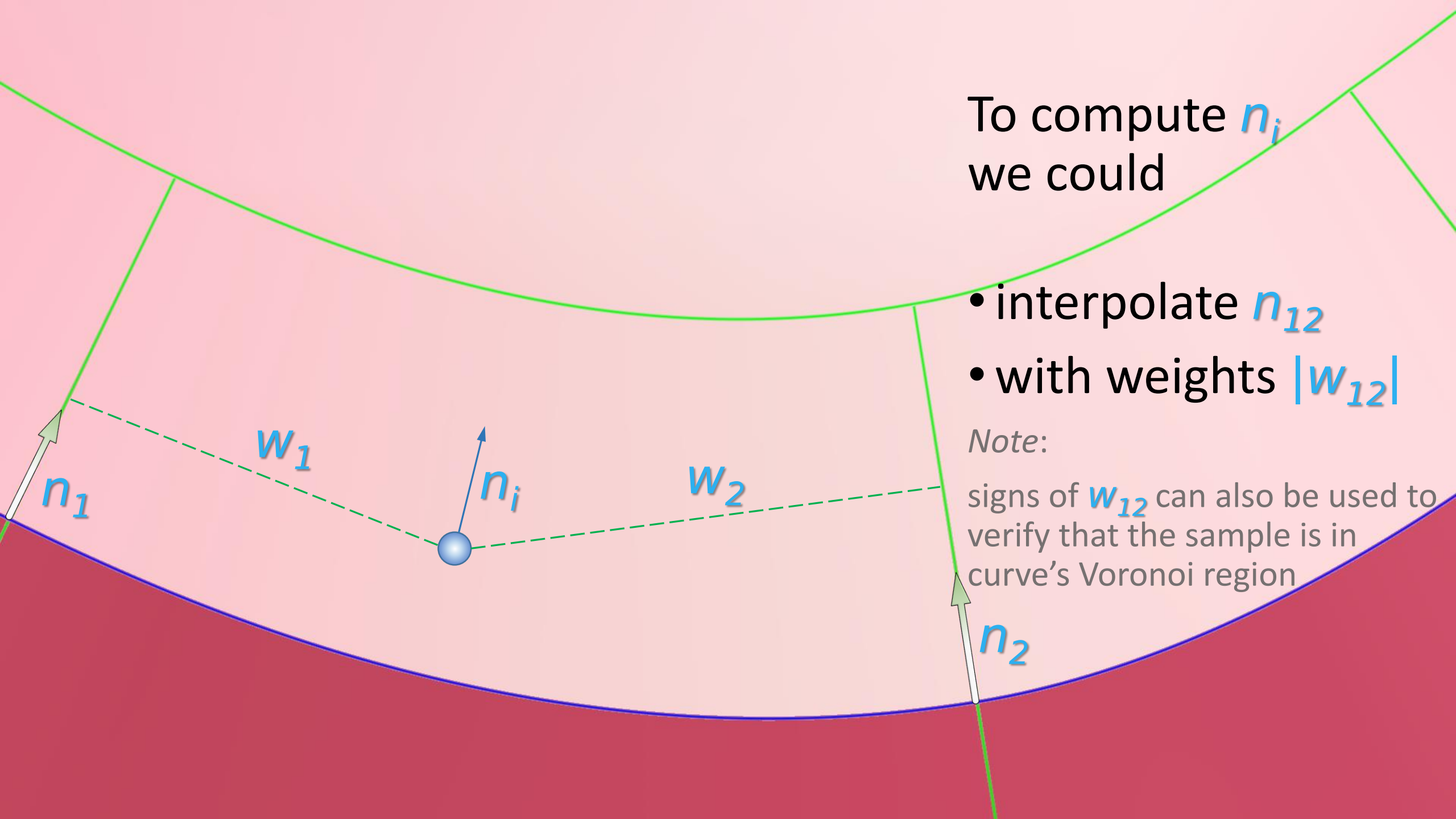


details

For each sample , we need

- scalar distance to the curve d
- offset vector n_i





To compute n_i we could

- interpolate n_{12}
- with weights $|w_{12}|$

Note:

signs of w_{12} can also be used to verify that the sample is in curve's Voronoi region

Distance to the curve

- ∃ numerous prior art approaches
- To compute it even faster, we propose two algorithms:

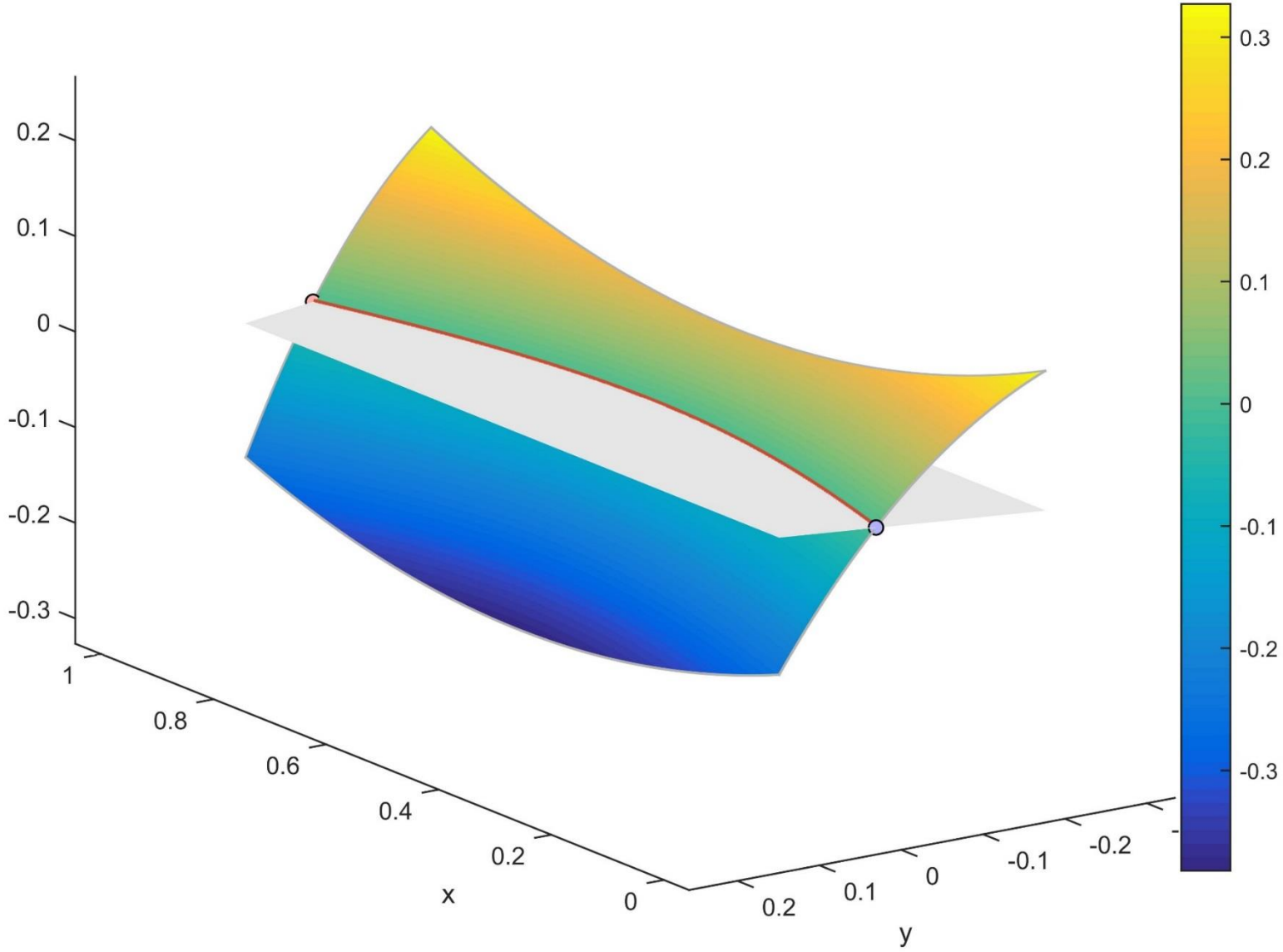
1. Implicit representation of cubic Bézier curves – using barycentric coordinates (savings: 6 terms instead of 10)

   (see also “Rendering Cubic Curves on a GPU with Floater's Implicitization” by Ron Pfeifle in JGT 2012)

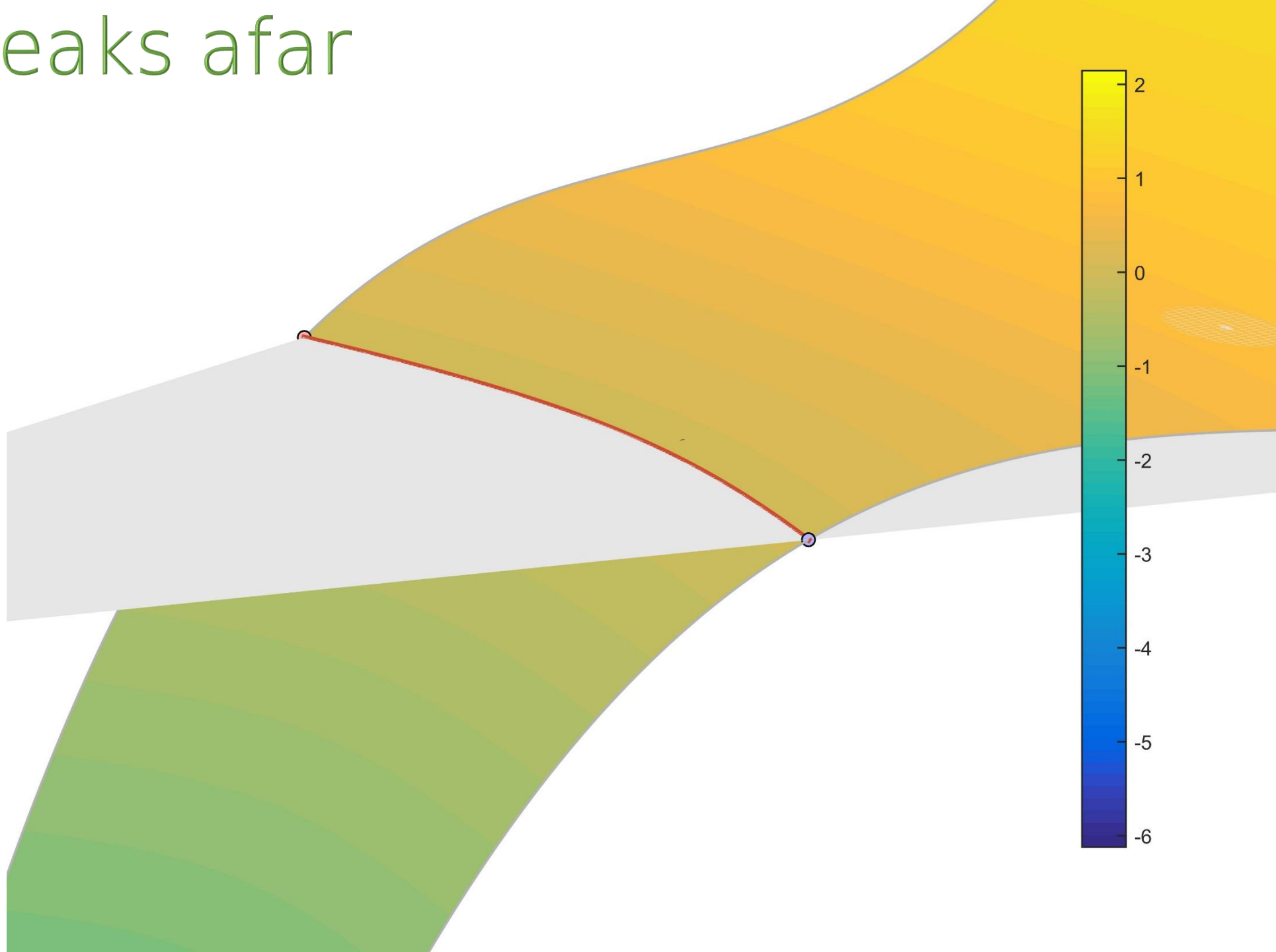
2. A quotient of two multivariate polynomials over variables that we choose (to make life easier)

   ≈ beefed up Phong interpolation in 1D

Using a value of implicit cubic polynomial as a distance proxy works near the curve…

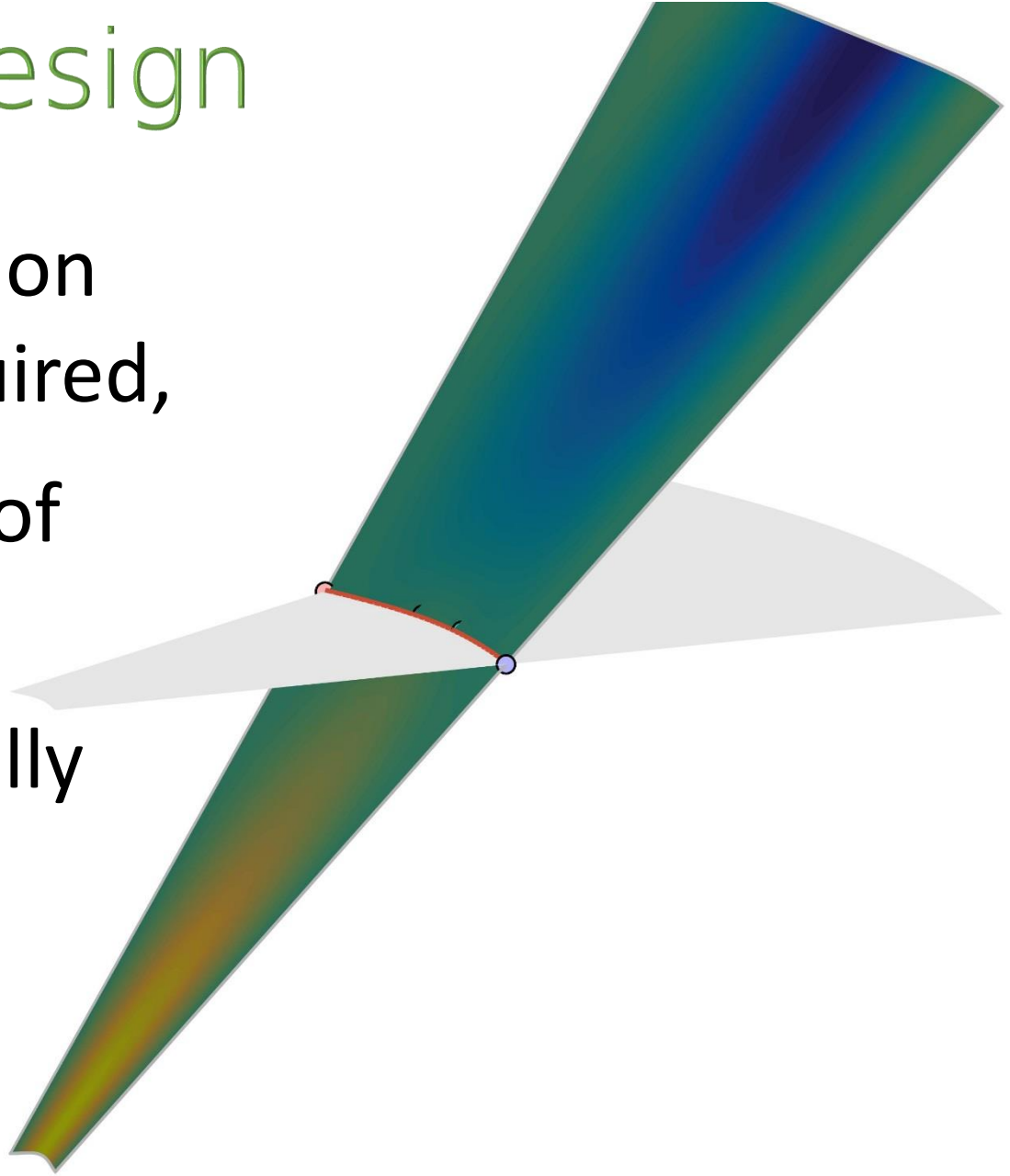


…but breaks afar

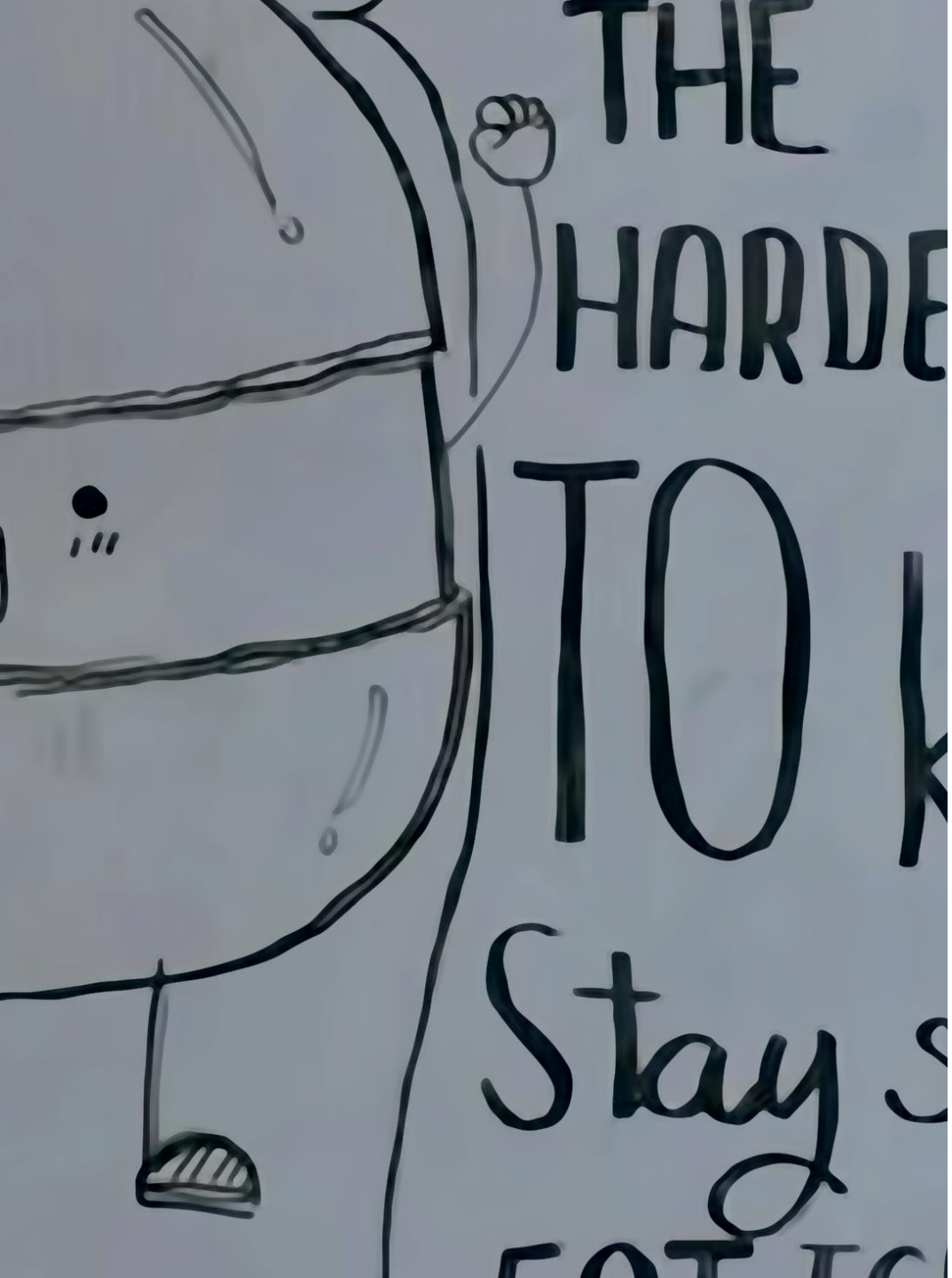


P_{n+1}/P_n is stable by design

- unless strict reproduction of Bézier curves is required,
- √ it should be a method of choice
- √ since it is unconditionally stable; there are other interesting possibilities as well

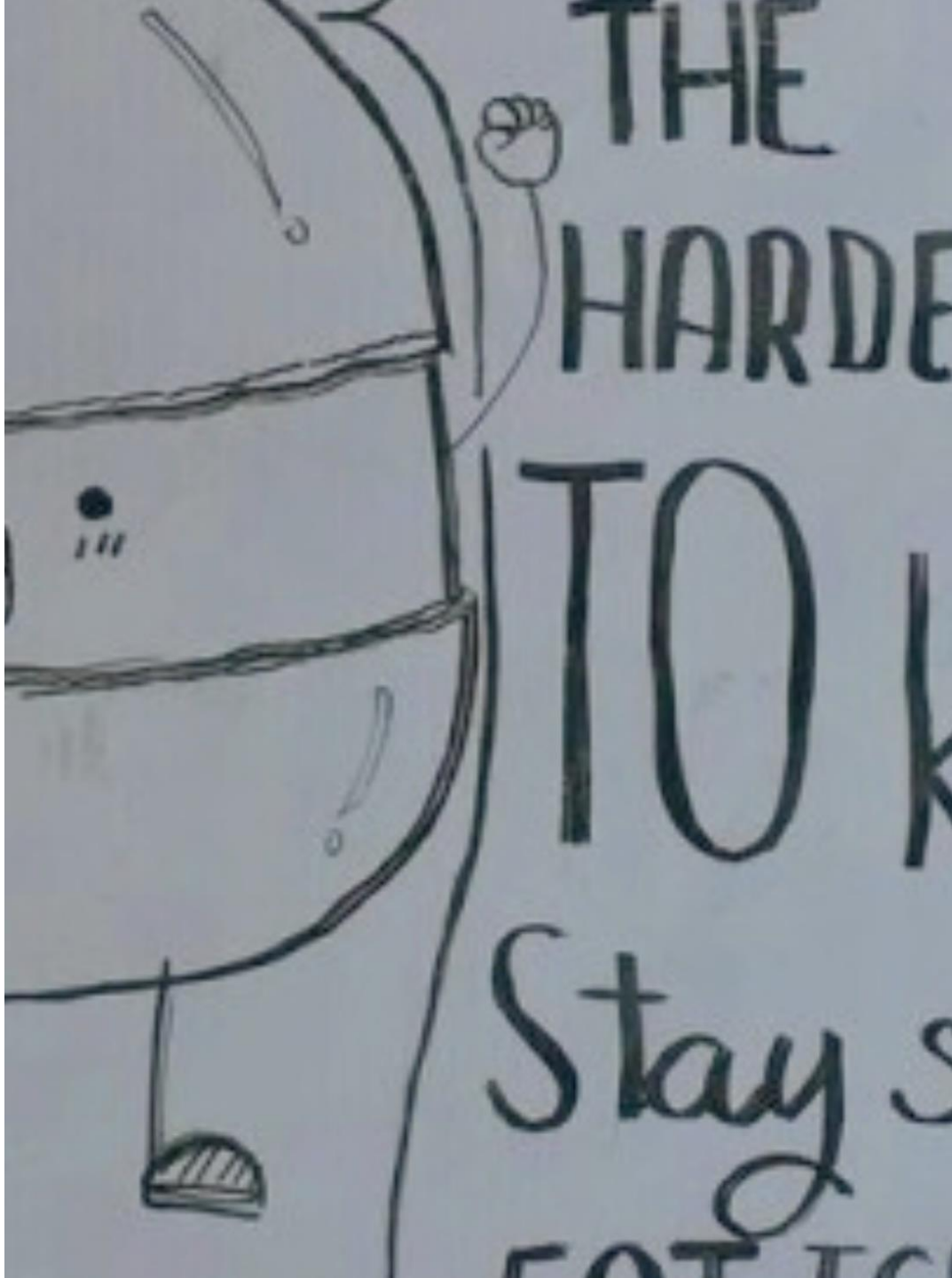


Expectations *vs* Reality



good

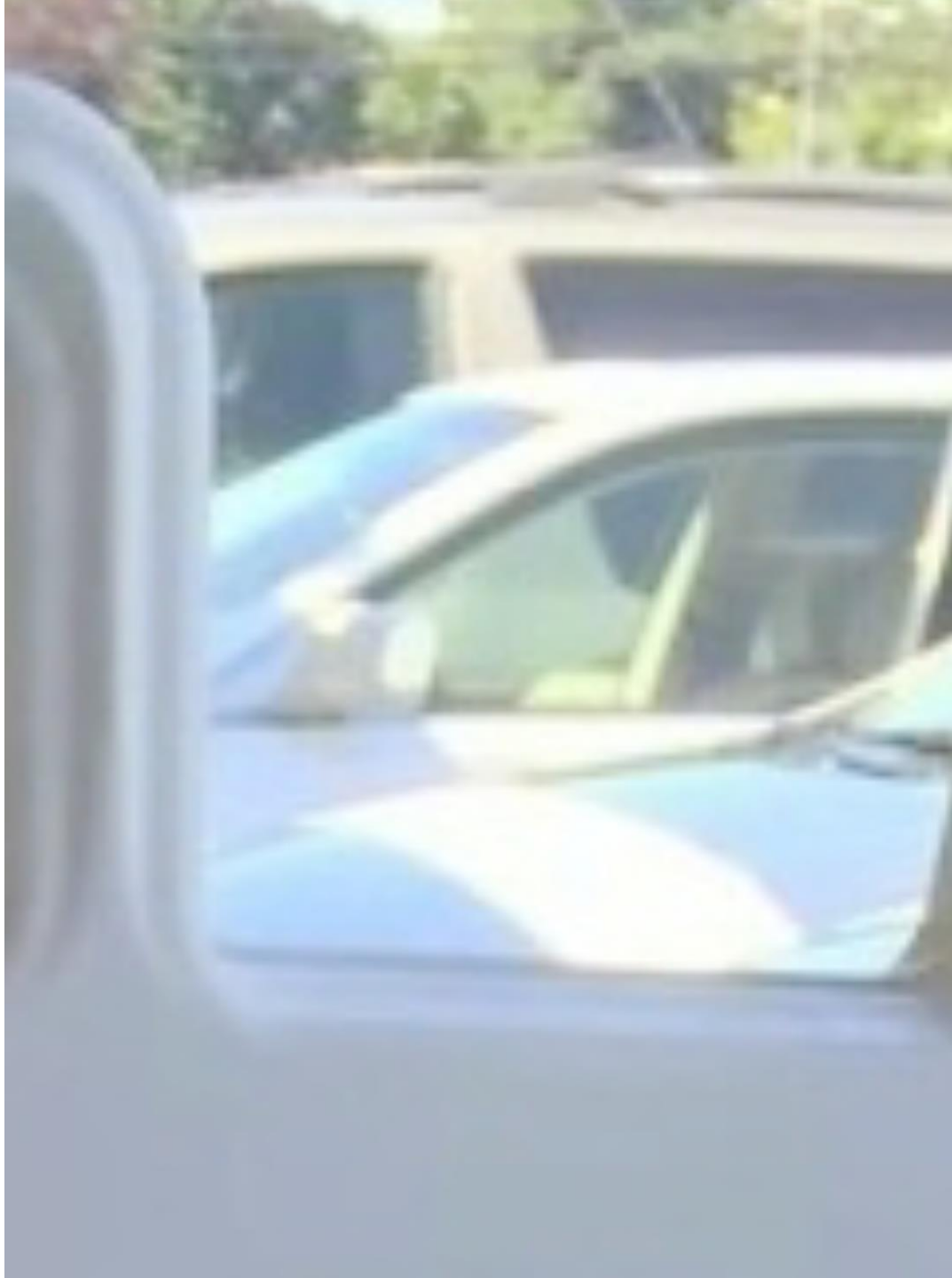
↤ IRT
raster ↦



bad

↤ IRT

raster ↦



Plans