Real-Time Stroke-Based Halftoning

Vanessa Freudenberg

Doctoral Thesis Defense, Magdeburg, 2004-06-04

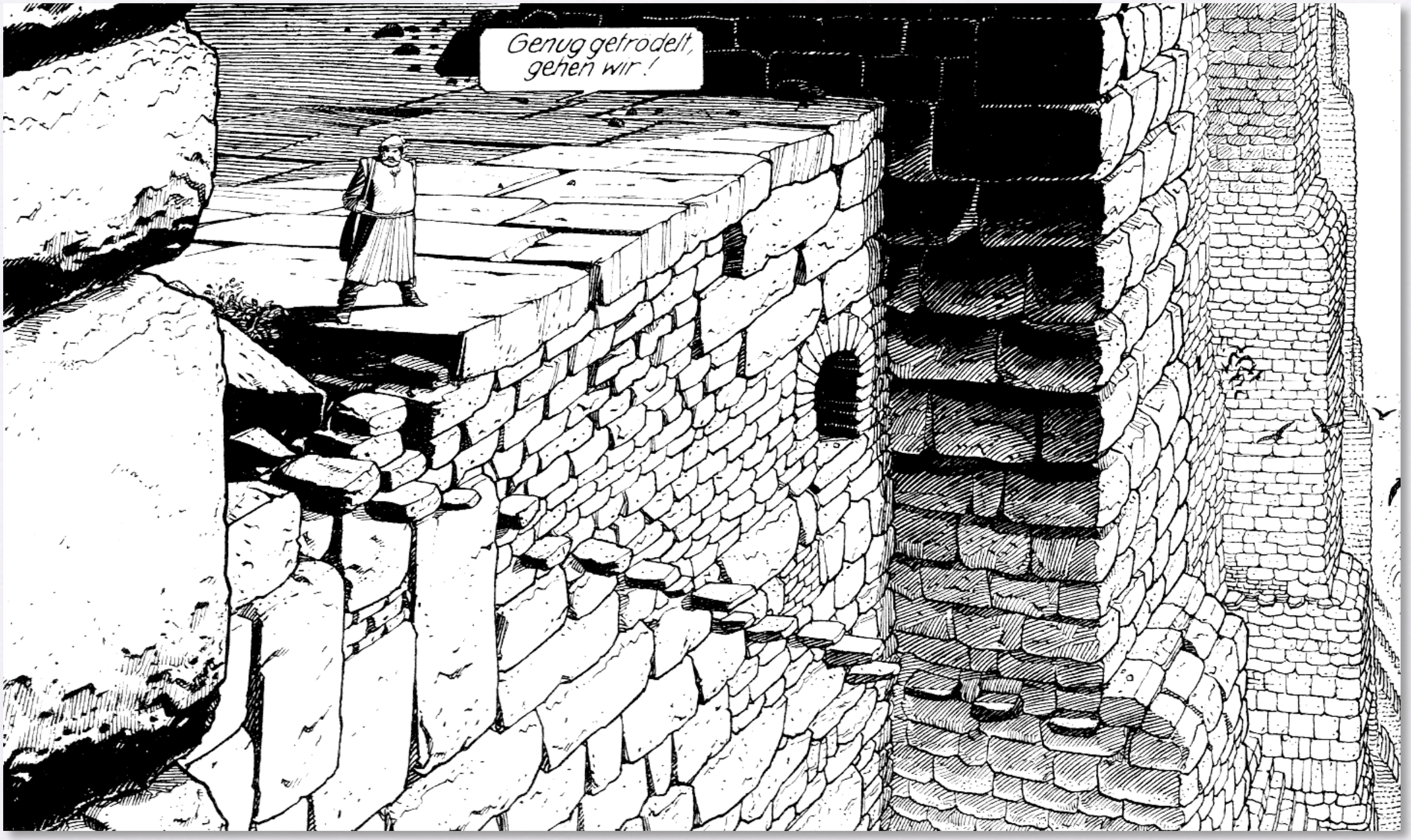
Real-Time

[Valve Software, 1998]

Motivation

[Valve Software, 2004]

Motivation



[Schuiten and Peeters, 1994]

Motivation

Goal: real-time non-photorealistic rendering

- requires hardware acceleration

Problem: hardware designed for photorealism

- PR: pixel-oriented
- NPR: larger primitives

Solution: rededicate hardware

- strokes instead of pixels

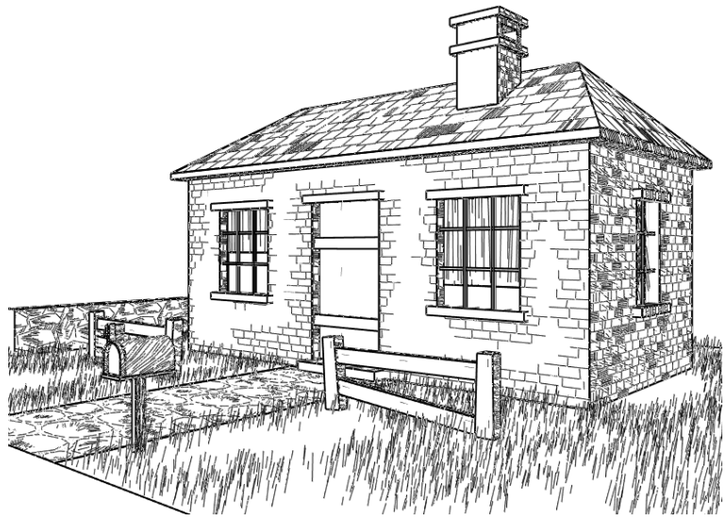
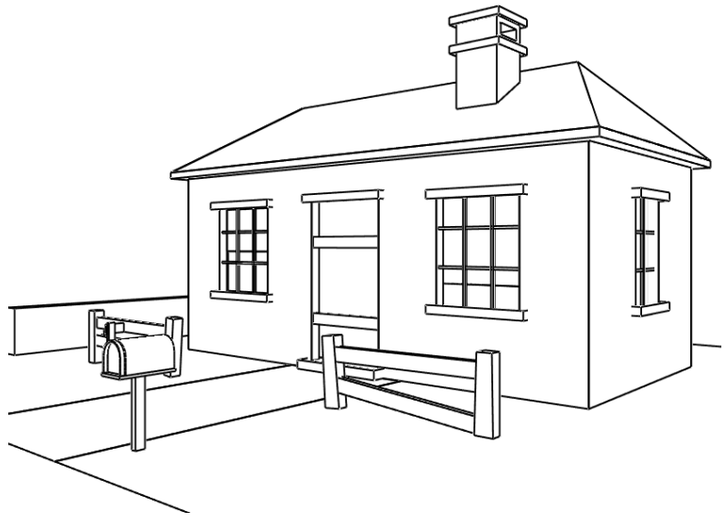
Inspiration: halftoning

- borrow means, though not goals

Related Work

- Winkenbach & Salesin (1994)
- Lake et al. (2000)
- Praun et al. (2001)
- Webb et al. (2002)

Overview

	texture-based (implicit)	geometry-based (explicit)
shading		
outlines		

Texture-Based Shading

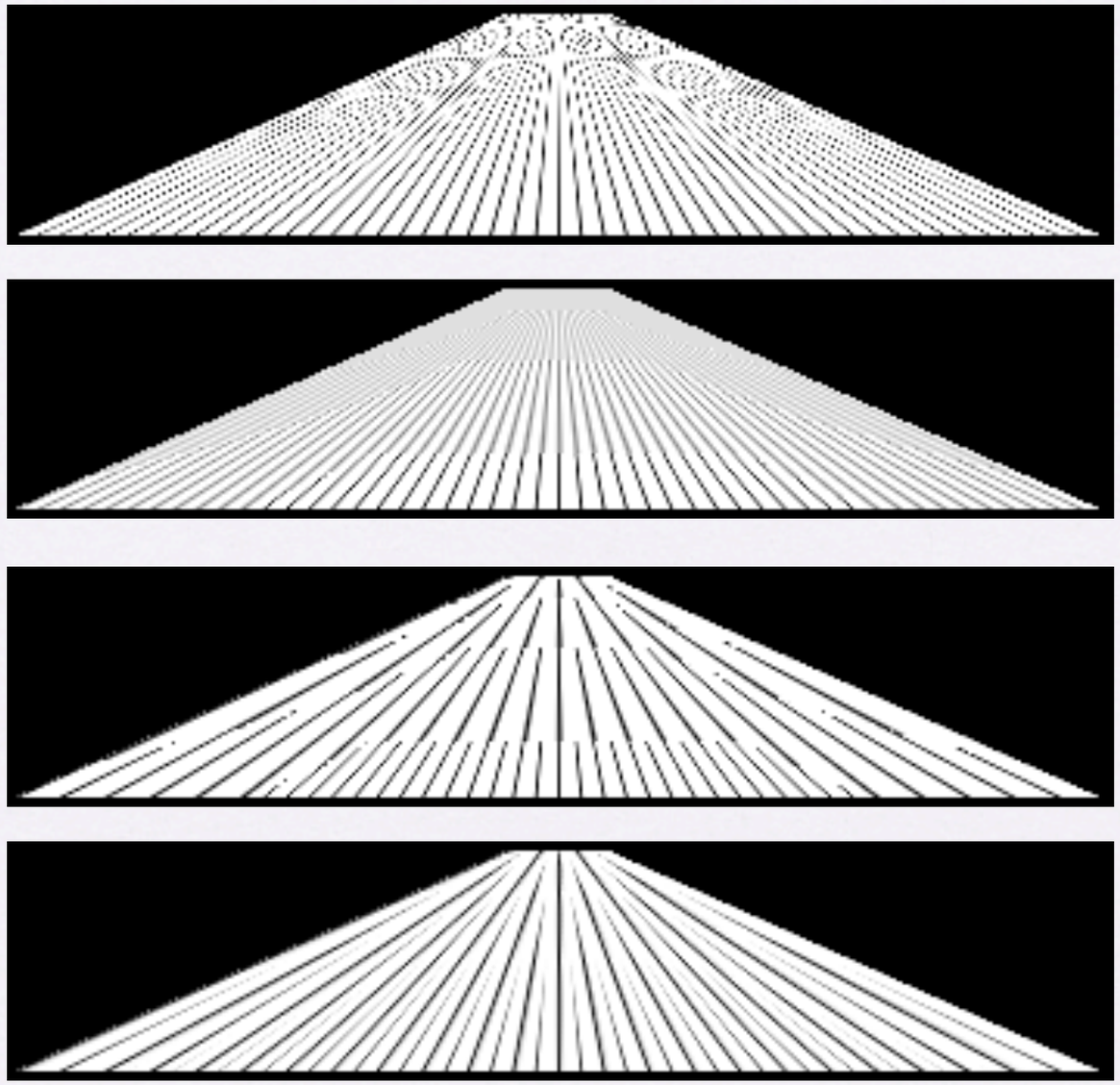
Goal: use texture mapping for stroke rendering

Problem: textures are scaled with distance

- distant lines clash
- want roughly constant density on screen

Solution:

- use mip-map texture filtering
- chooses between different stroke densities



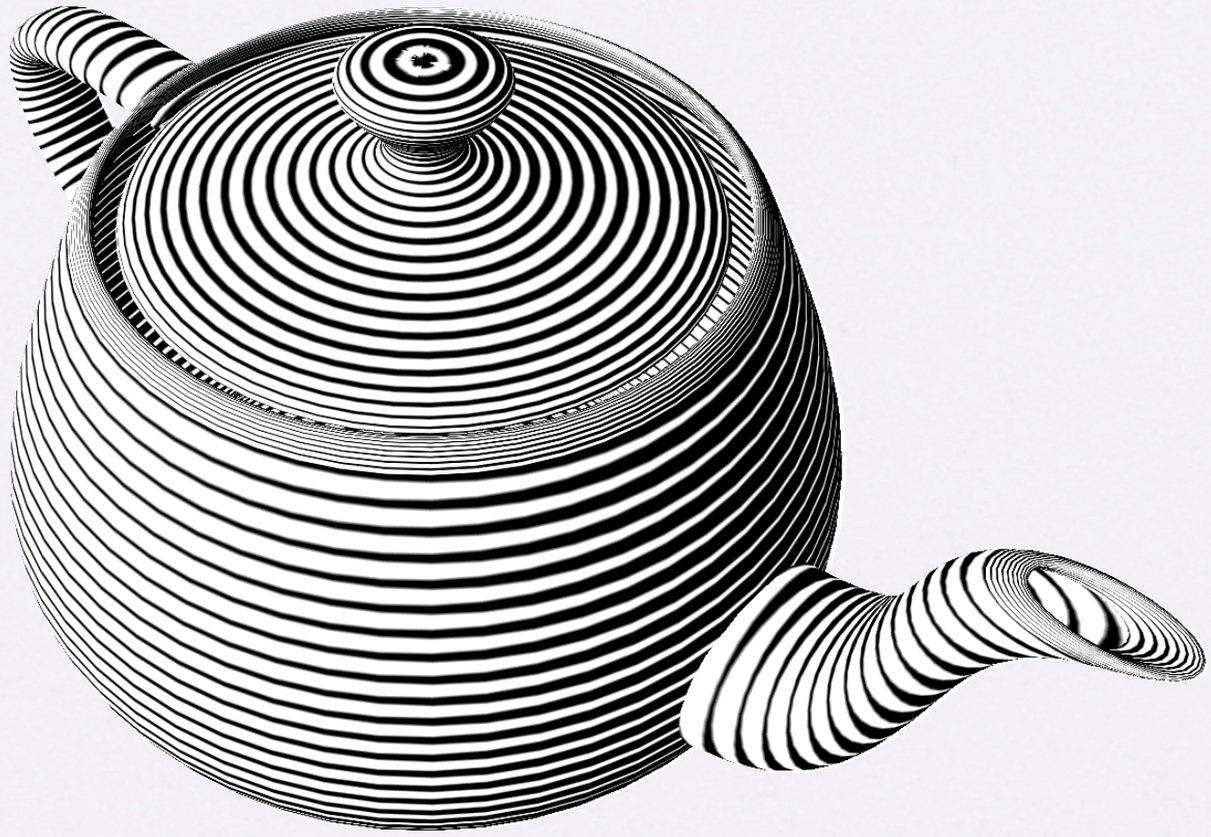
demo: mipmap

Texture-Based Shading

Goal: shading by varing stroke width

Problem: adjust width of strokes in texture

- texture image is fixed



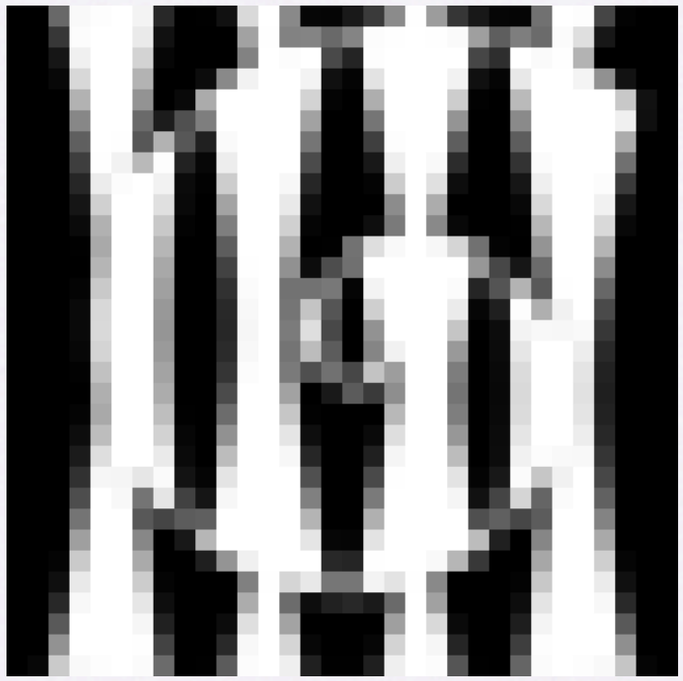
Solution:

- use halftone screen as texture
- configure texture unit to perform threshold operation

Texture-Based Shading

Problem: screen resolution too low

- aliasing, dot popping



Solution:

- smooth threshold operation

One texture stage only*

inv((inv(col) – tex) << 2)

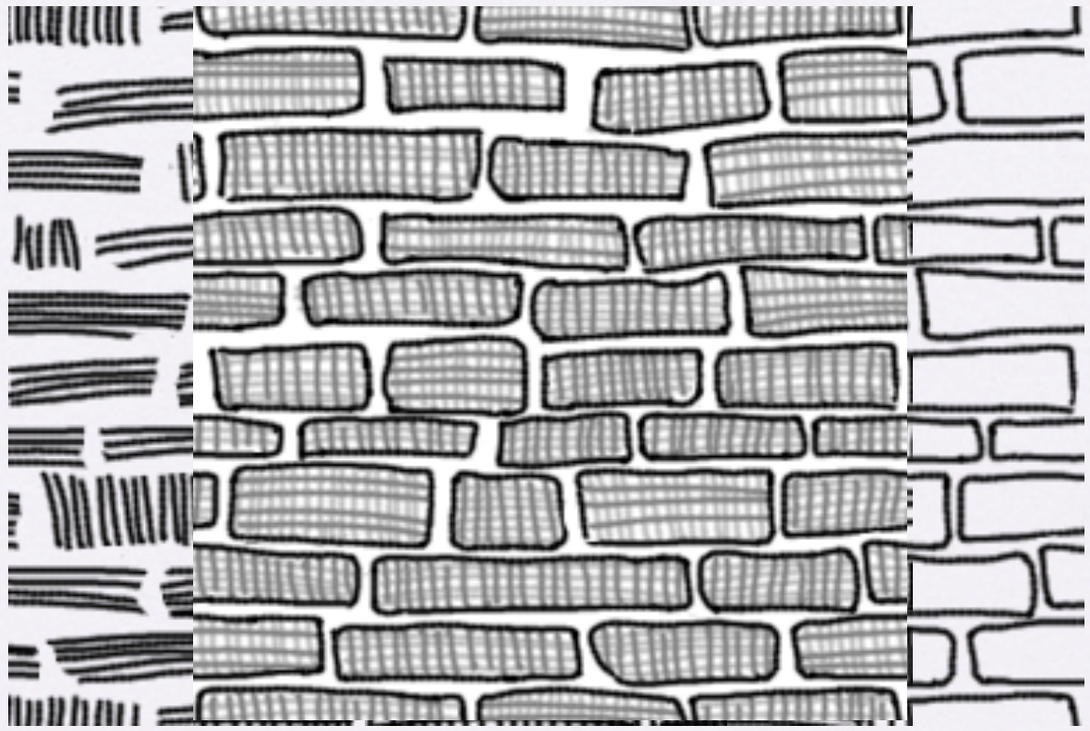
* outer inv() is input mapping of next stage

demo: rtstroke 1

Texture-Based Shading

Goal: render in pen-and-ink-style

Problem: dynamically adjust stroke density

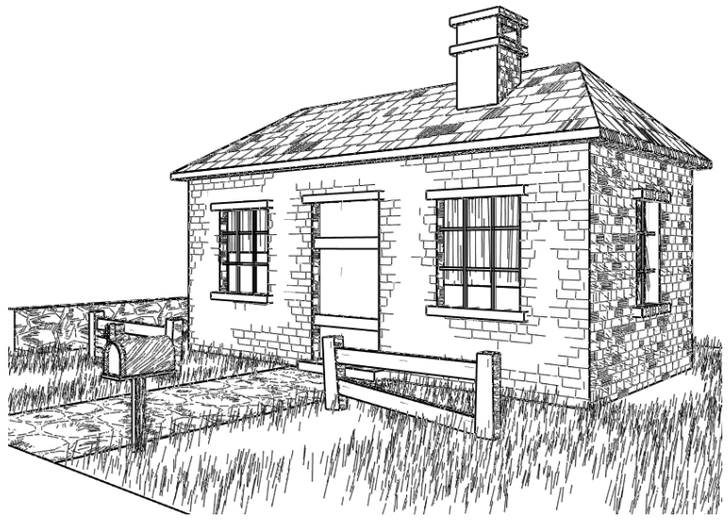
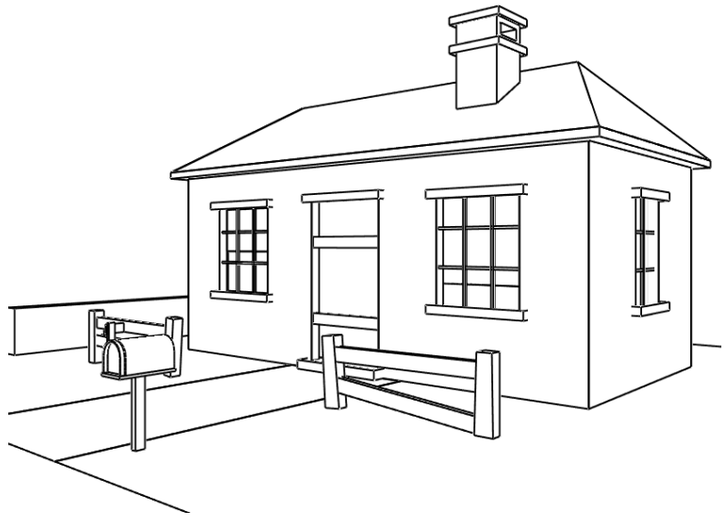


Solution:

- construct special halftone screens
- same threshold operation

demo: rtstroke 2

Overview

	texture-based (implicit)	geometry-based (explicit)
shading		
outlines		

Geometry-Based Shading

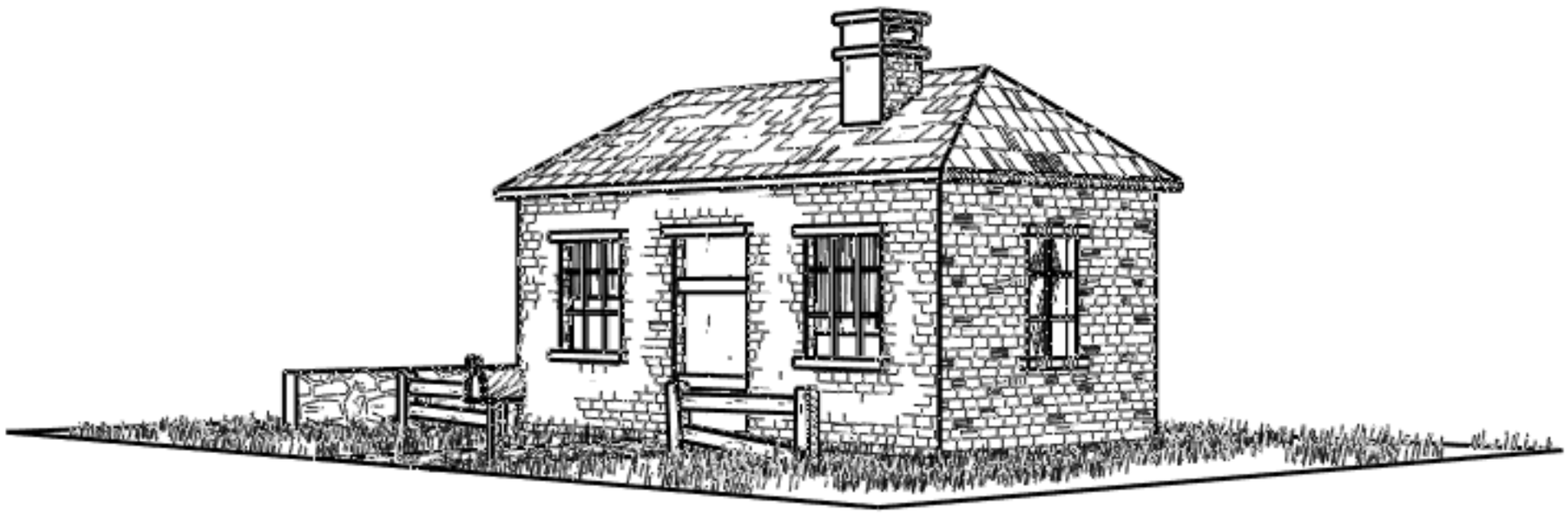
Goal: draw strokes until desired density is reached

Problem: graphics hardware too inflexible

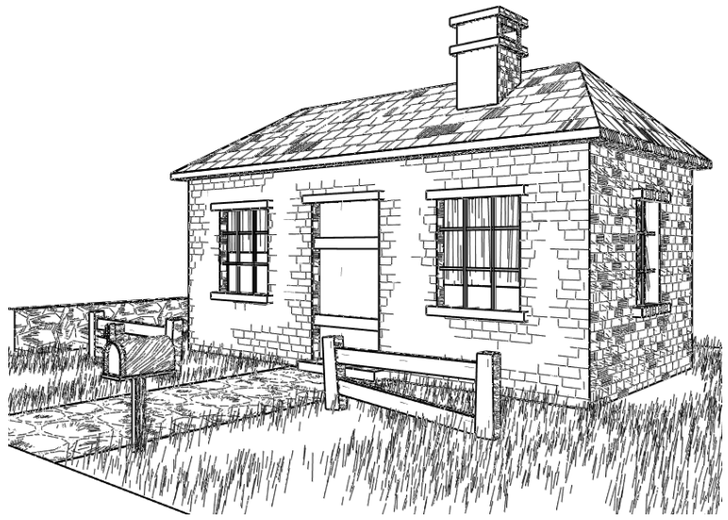
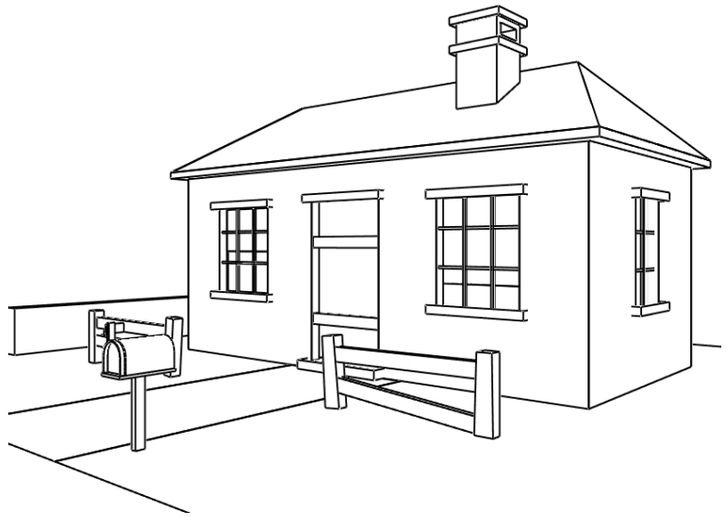
- can not create primitives
- one vertex at a time

Solution: thresholding

- submit all strokes
- replicate data in vertices
- discard excess strokes



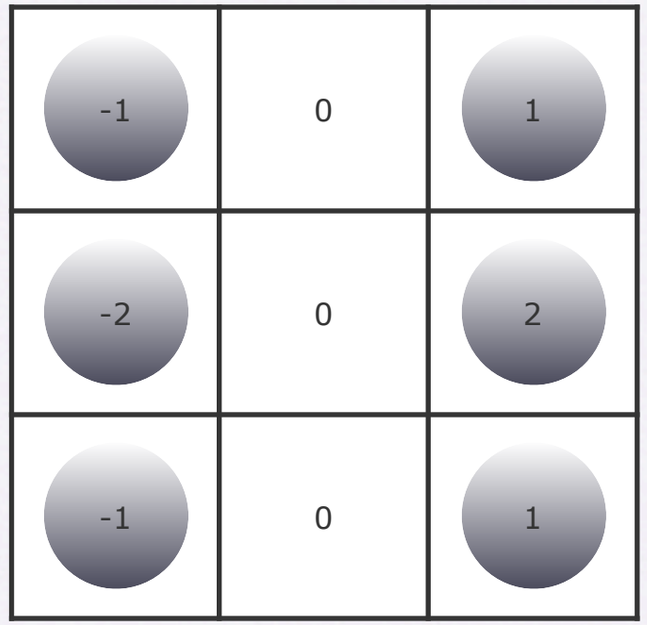
Overview

	texture-based (implicit)	geometry-based (explicit)
shading		
outlines		

Texture-Based Outlines

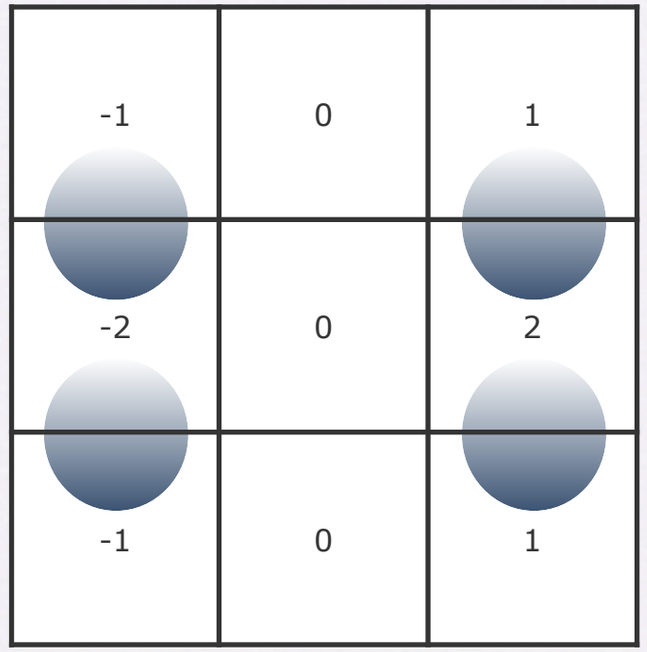
Goal: edge-detection filter on G-buffers

Problem: number of samples in filter kernel

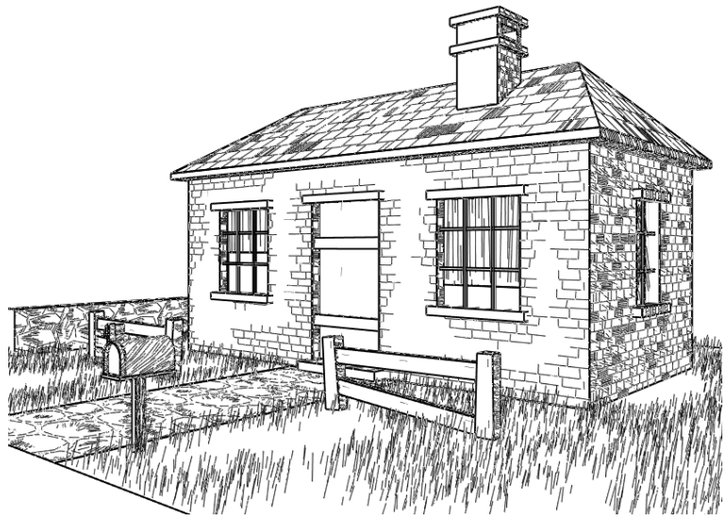
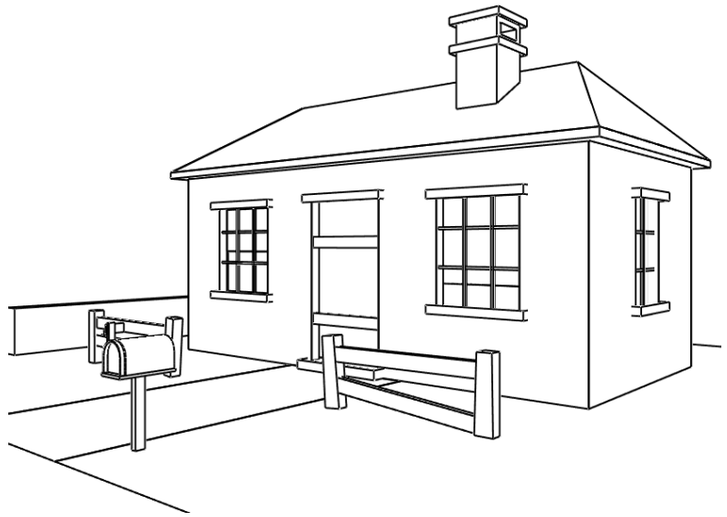


Solution: place sample points between texels

- samples two texels at once
- position determines weight



Overview

	texture-based (implicit)	geometry-based (explicit)
shading		
outlines		

Geometry-Based Outlines

Goal: determine silhouettes on GPU

Problem: access normals of adjacent faces

- only vertex-local data accessible

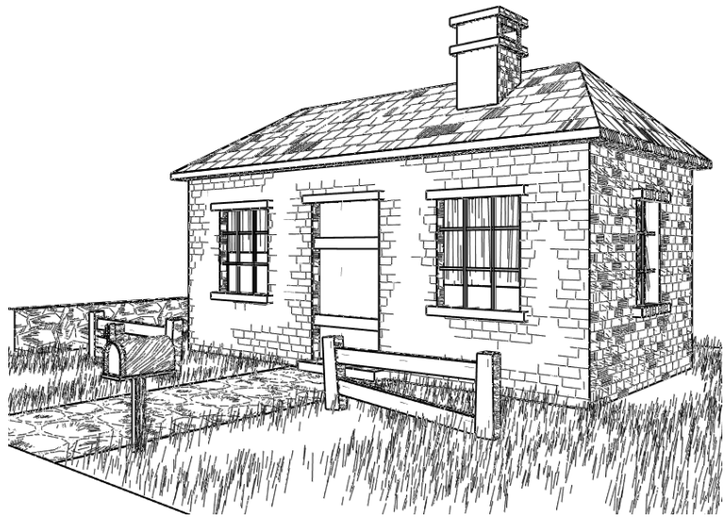
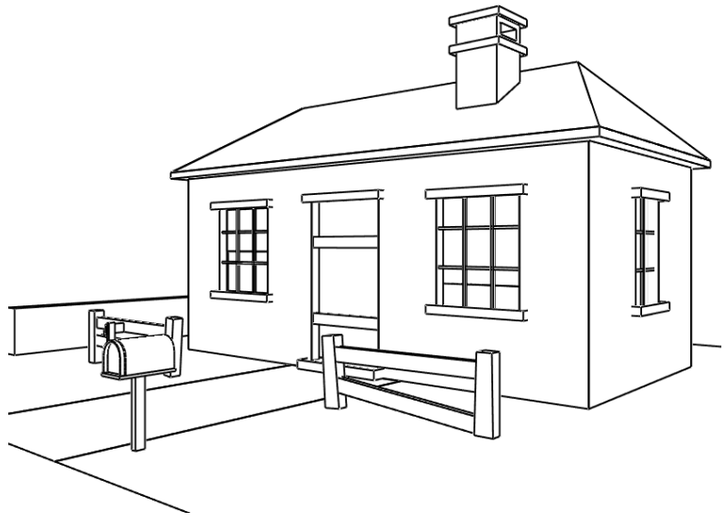
Solution: replicate data across vertices

- store normals with each edge vertex

Geometry-Based Outlines

```
void silhouette_test(SilhouetteVertex vertex,
                     out float4 position : HPOS,
                     out float4 color : COL0)
{
   position = mul(glstate.matrix.mvp, vertex.Position);
   float4 normalA = mul(glstate.matrix.invtrans.modelview[0], vertex.NormalA);
   float4 normalB = mul(glstate.matrix.invtrans.modelview[0], vertex.NormalB);
   float4 view = mul(glstate.matrix.modelview[0], vertex.Midpoint);
   float facingnessA = dot(view, normalA);
   float facingnessB = dot(view, normalB);
   color = facingnessA * facingnessB < 0.0 ?
      float4(0.0, 0.0, 0.0, 1.0) : float4(0.0, 0.0, 0.0, 0.0);
}
```

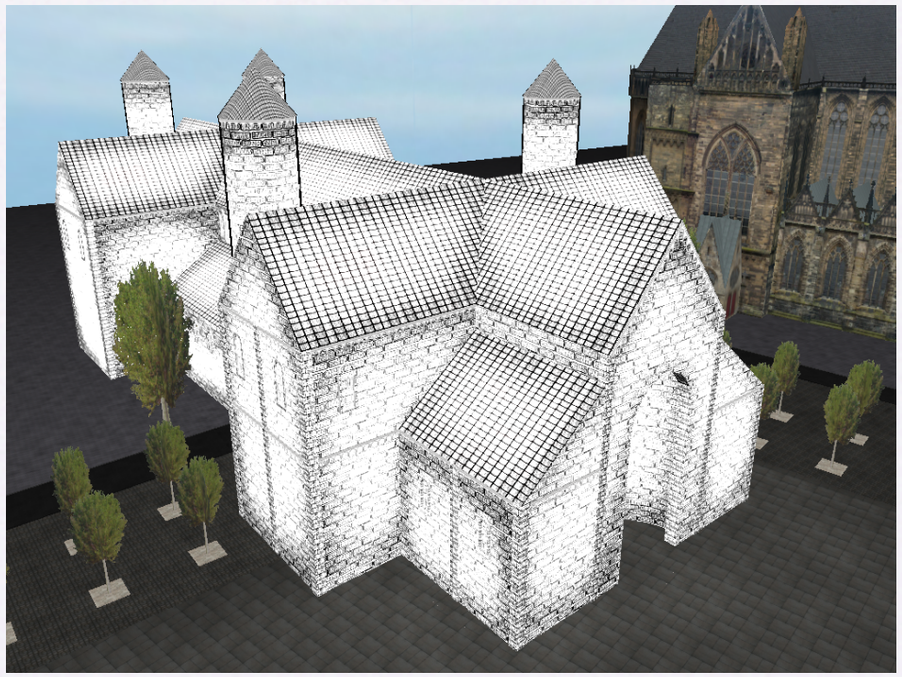
Overview

	texture-based (implicit)	geometry-based (explicit)
shading		
outlines		

Applications

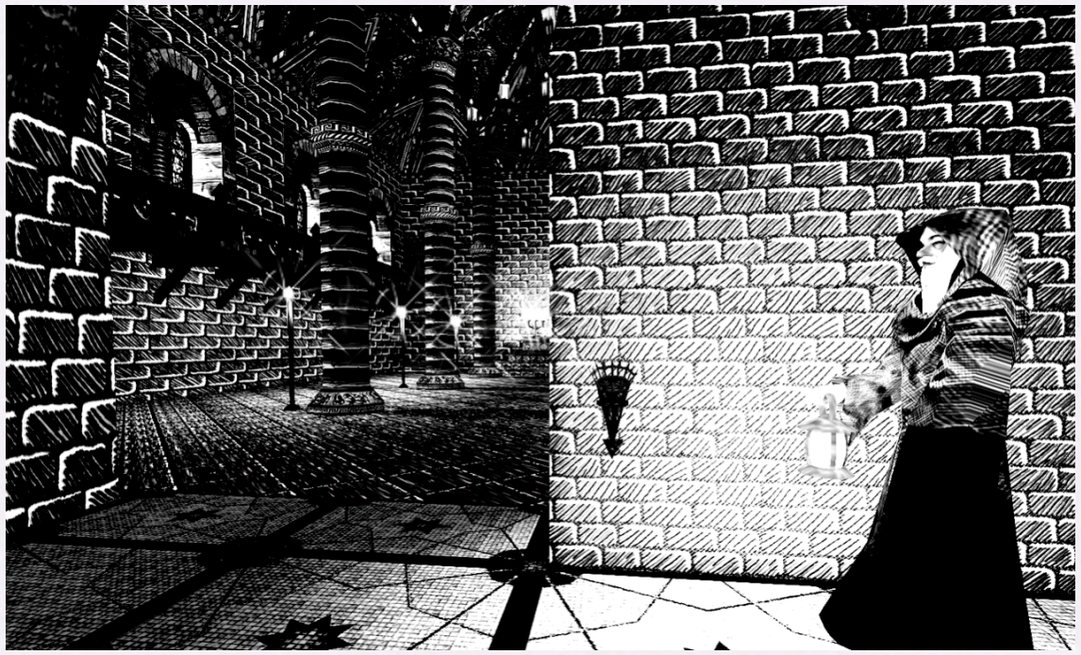
Archaeological Walkthrough

- veracity of virtual reconstruction
- depict uncertainty by non-realistic rendering style



Game Demo

- adapted from photorealistic style
- very few changes



demo: bfhtdemo

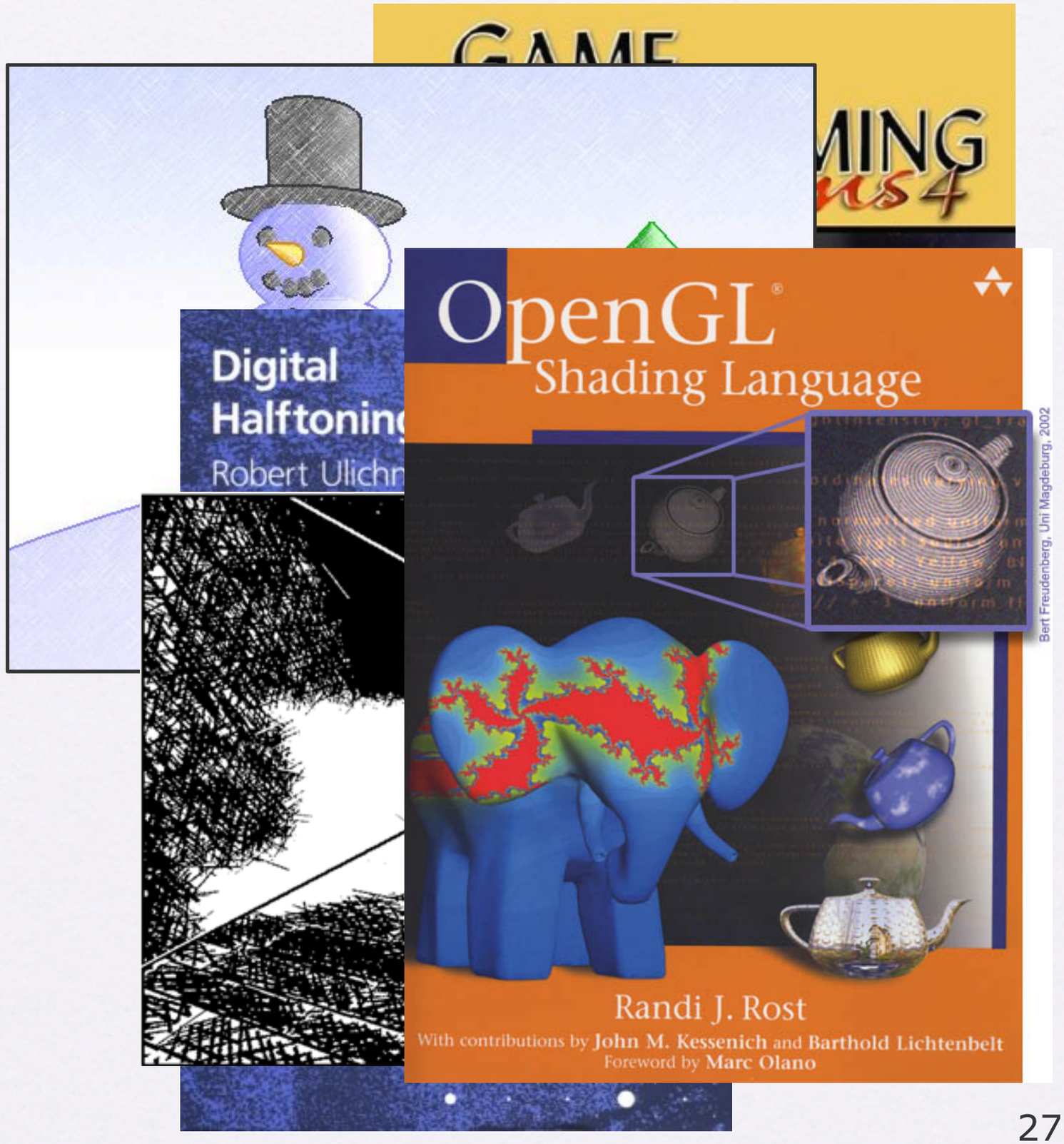
run

Conclusion

- practical method for fast non-photorealistic shading
  - as fast as photo-realistic methods
  - artist-controlled style
- rededicate hardware for non-photorealism
  - build on ideas from halftoning
- wider range of applications for NPR
  - interactive illustration
  - games

From Current to Future Work

- artists tools
- rendering in color
- exact tone reproduction
- intention-driven lighting model
- NPR hardware



THE END

old / additional slides

Geometry-Based Outlines

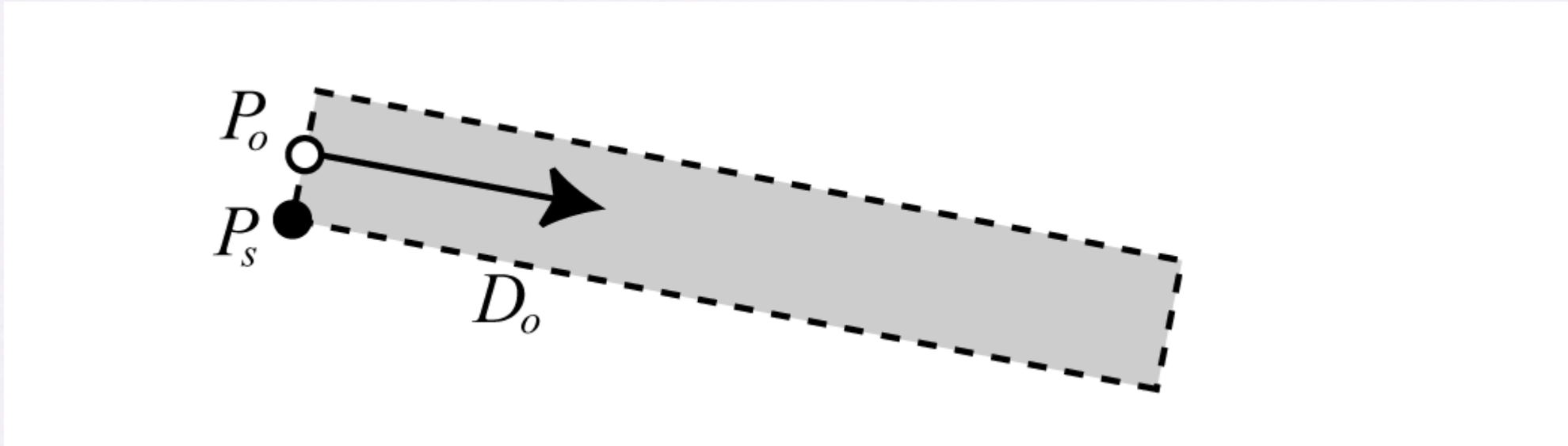
Goal: adjust stroke width

Problem: must depend on target width, not distance

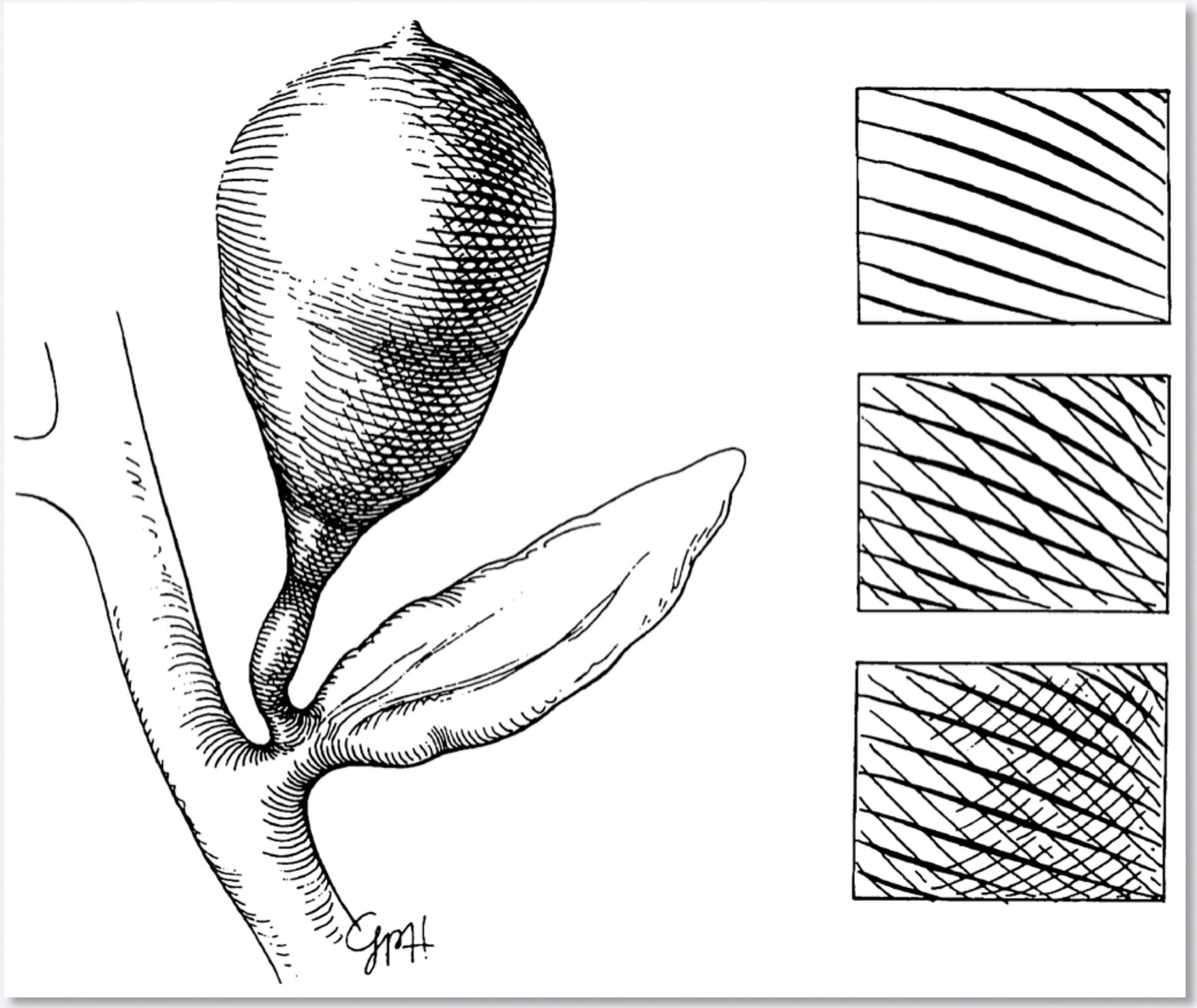
- polylines have fixed width
- quads are scaled with distance

Solution: draw quads in screen space

- two vertices per end, same P_o, differing D_o



Strokes



Pen-and-Ink Illustration [Hodges, 1989]

Strokes

Wood Cut [M. C. Escher]

Real-Time

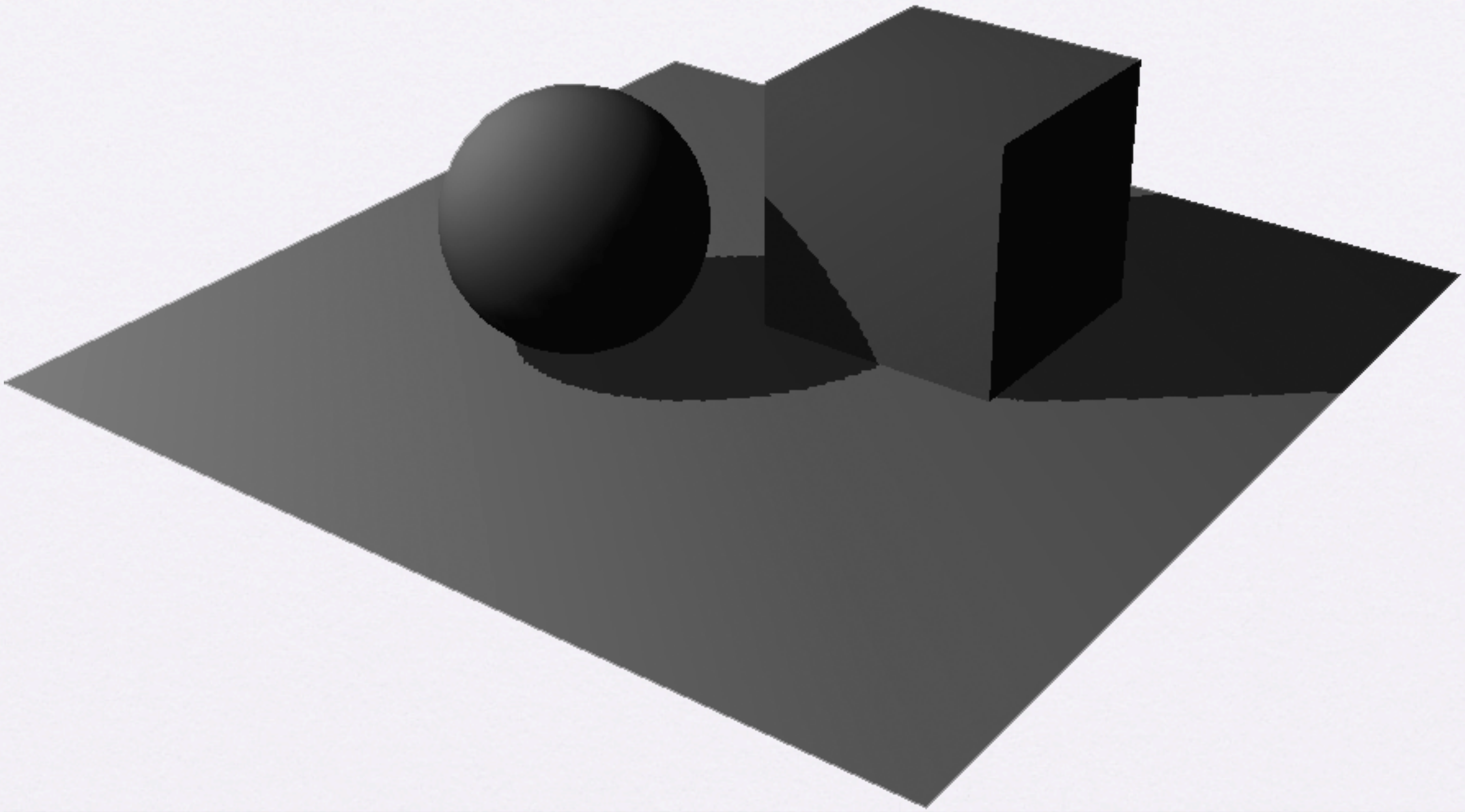
Halflife [Valve Software, 1998]

Real-Time

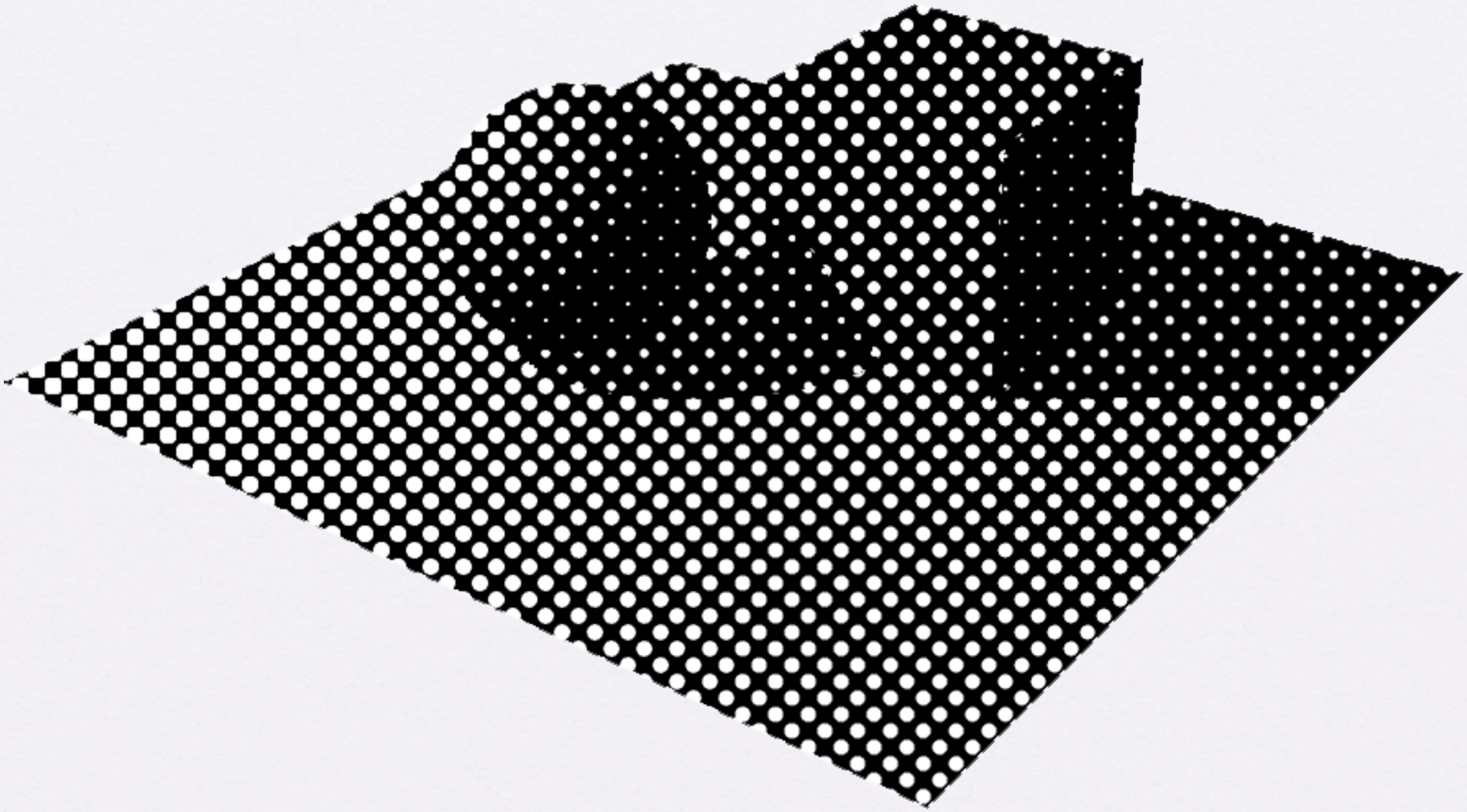
Halflife 2 [Valve Software, 2004]

Real-Time Stroke-Based Halftoning?

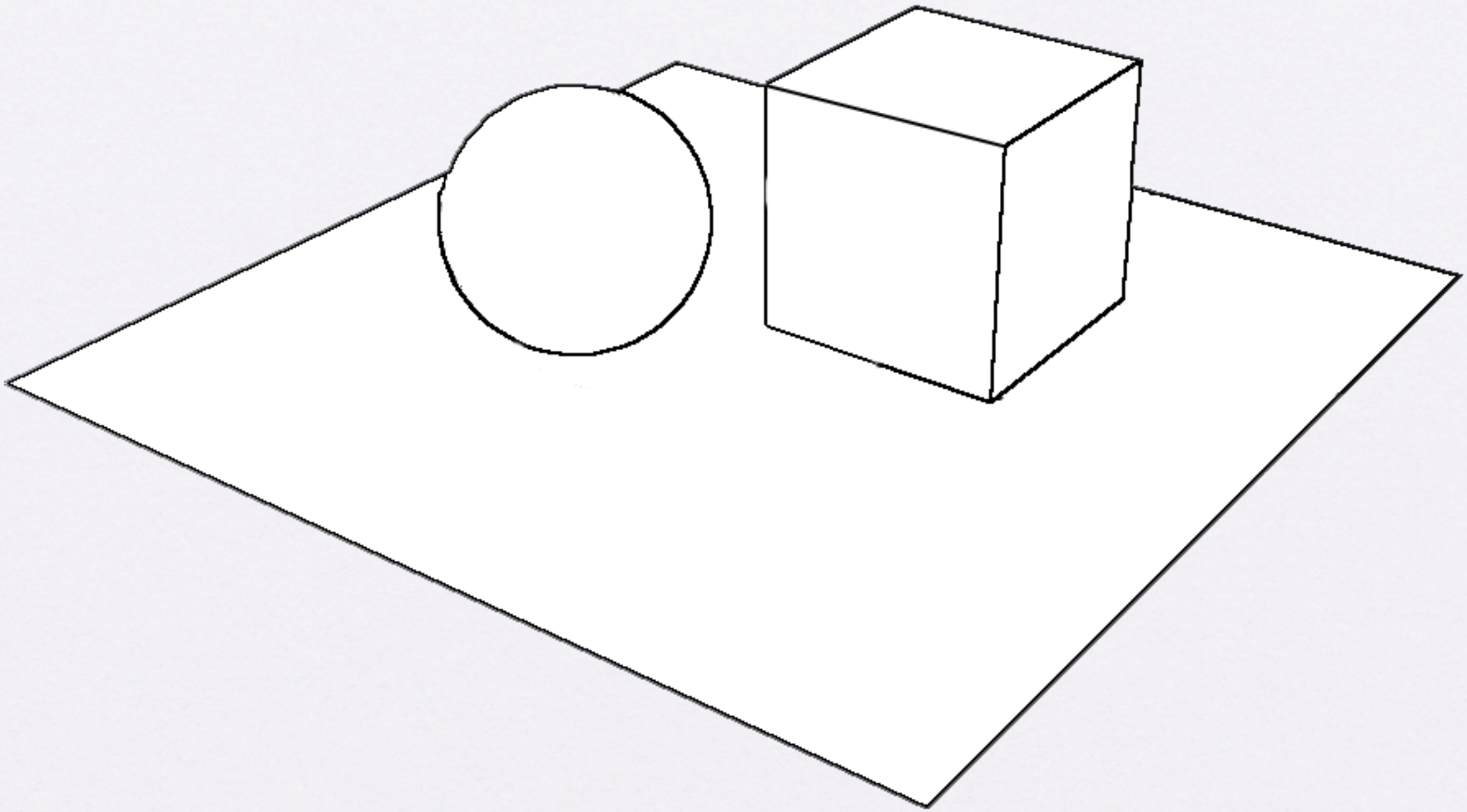
Real-Time Rendering



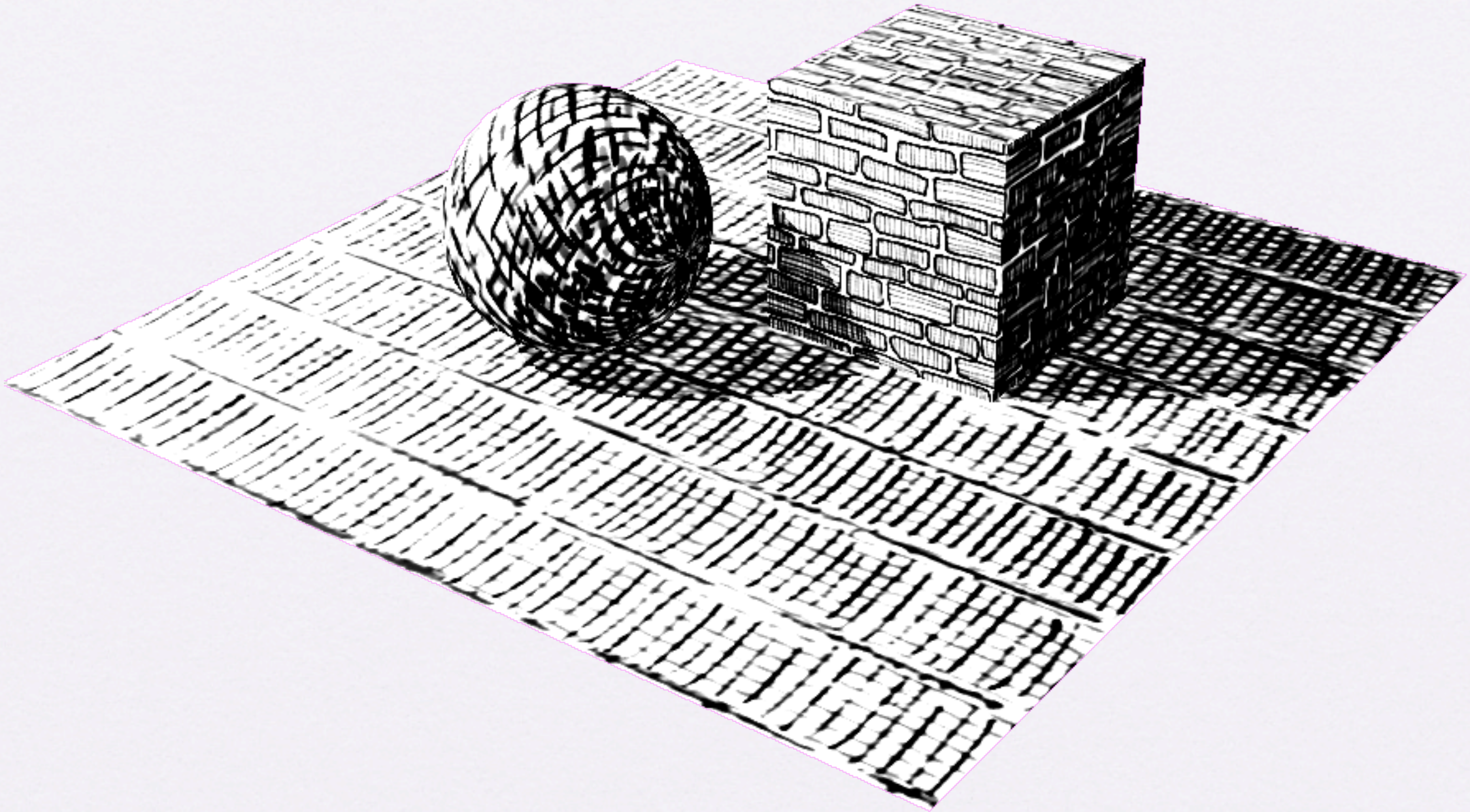
Real-Time Halftoning?



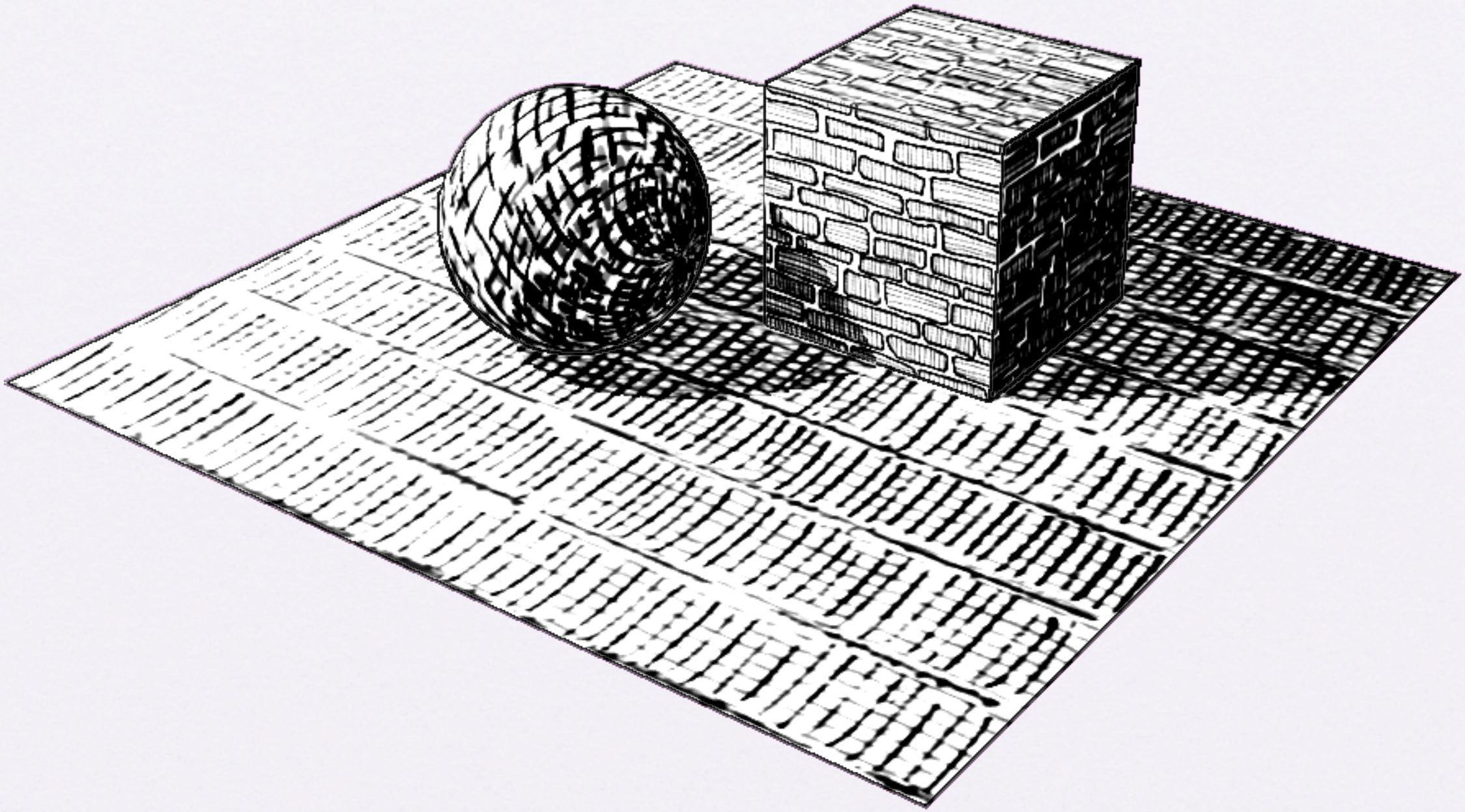
Stroke-Based Rendering



Real-Time Stroke-Based Halftoning



Real-Time Stroke-Based Halftoning



Overview

- Shading with Strokes
- Outline Rendering
- Applications
- Conclusion

Overview

- **Shading with Strokes**
- Outline Rendering
- Applications
- Conclusion

Shading with Strokes

- vast number of strokes
- adapt to changing lighting and perspective
- adjust density and width
- maintain frame-to-frame coherence

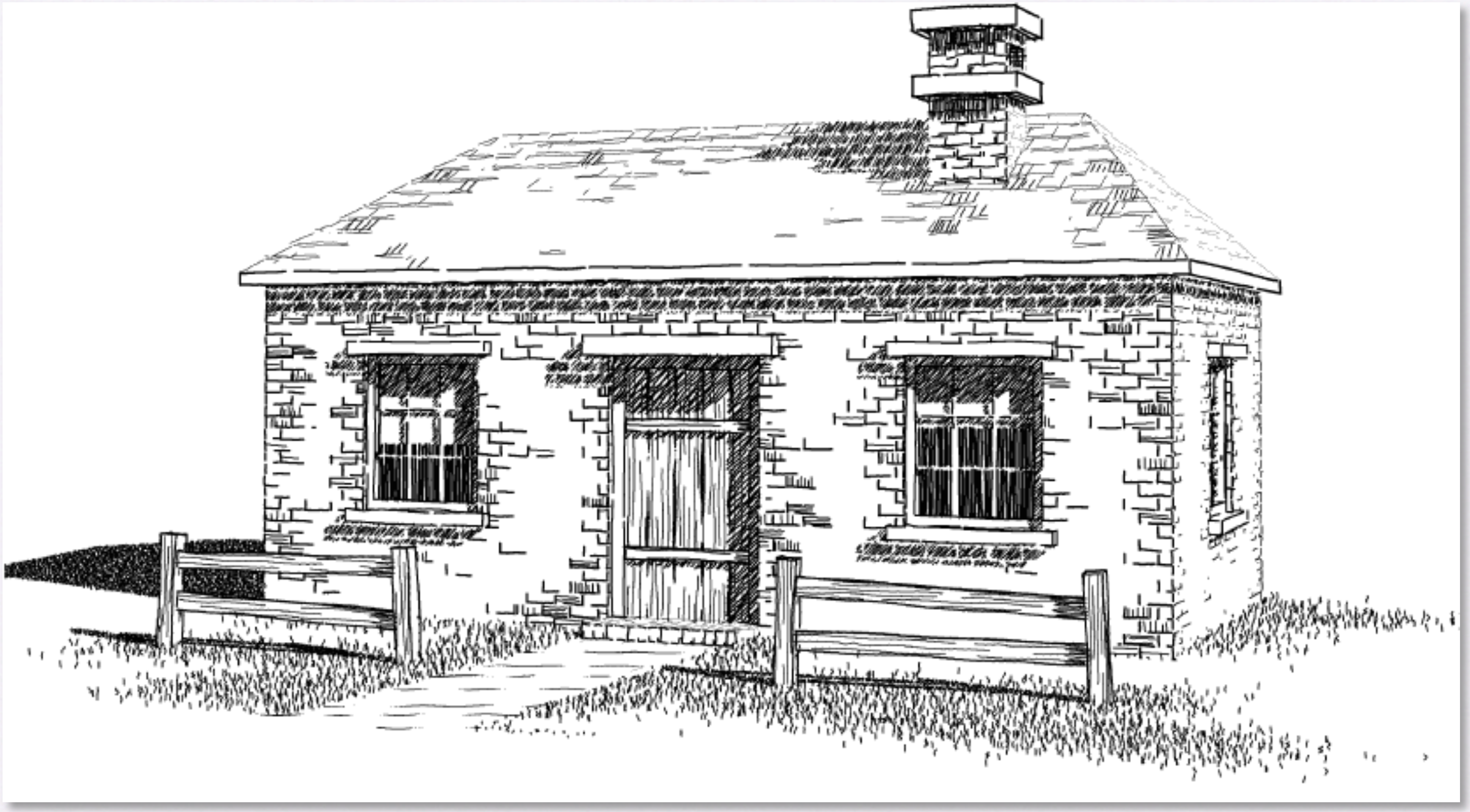
Related Work

- existing techniques for shading with strokes

Related Work

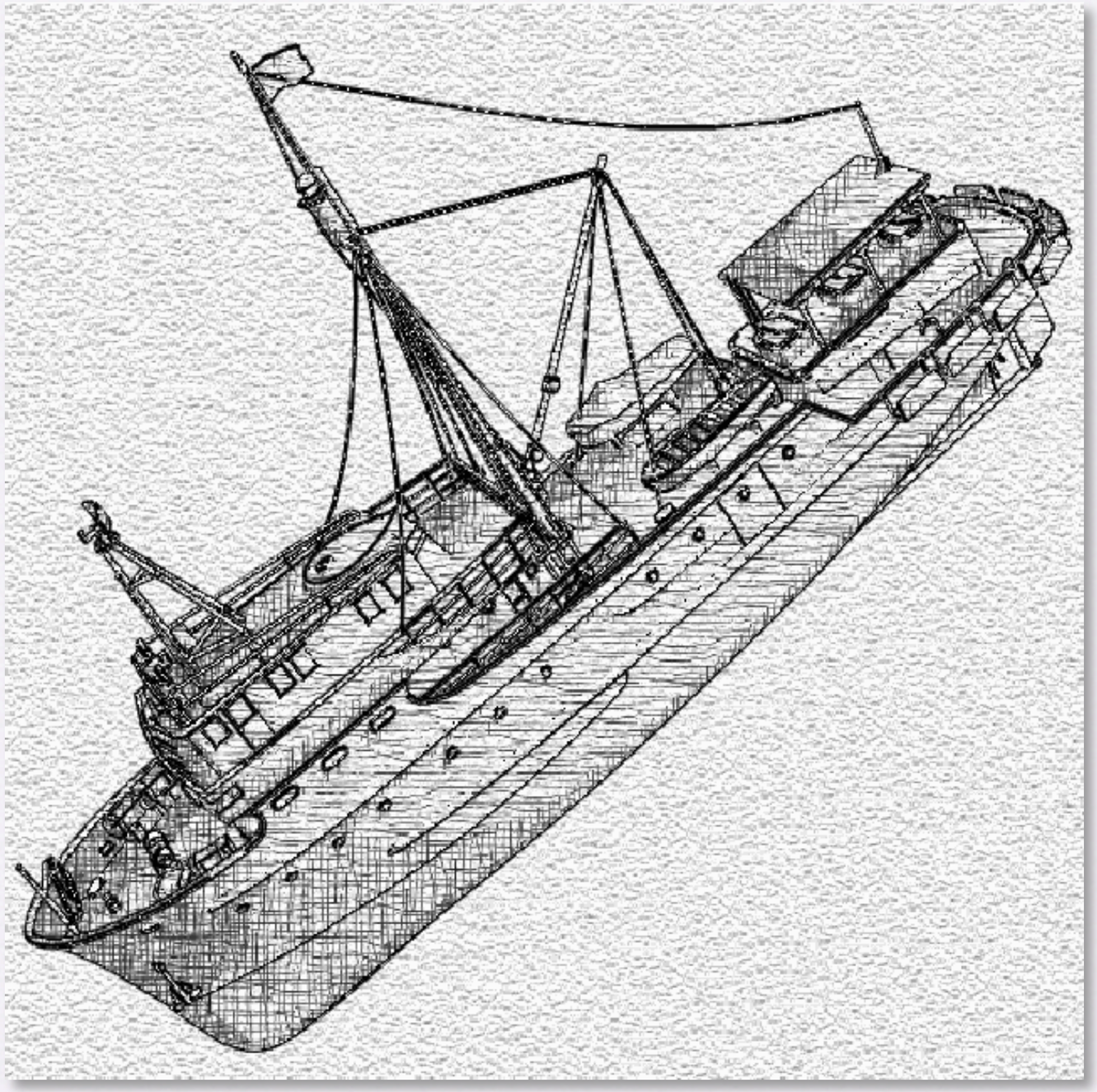
Winkenbach & Salesin (1994)

Related Work



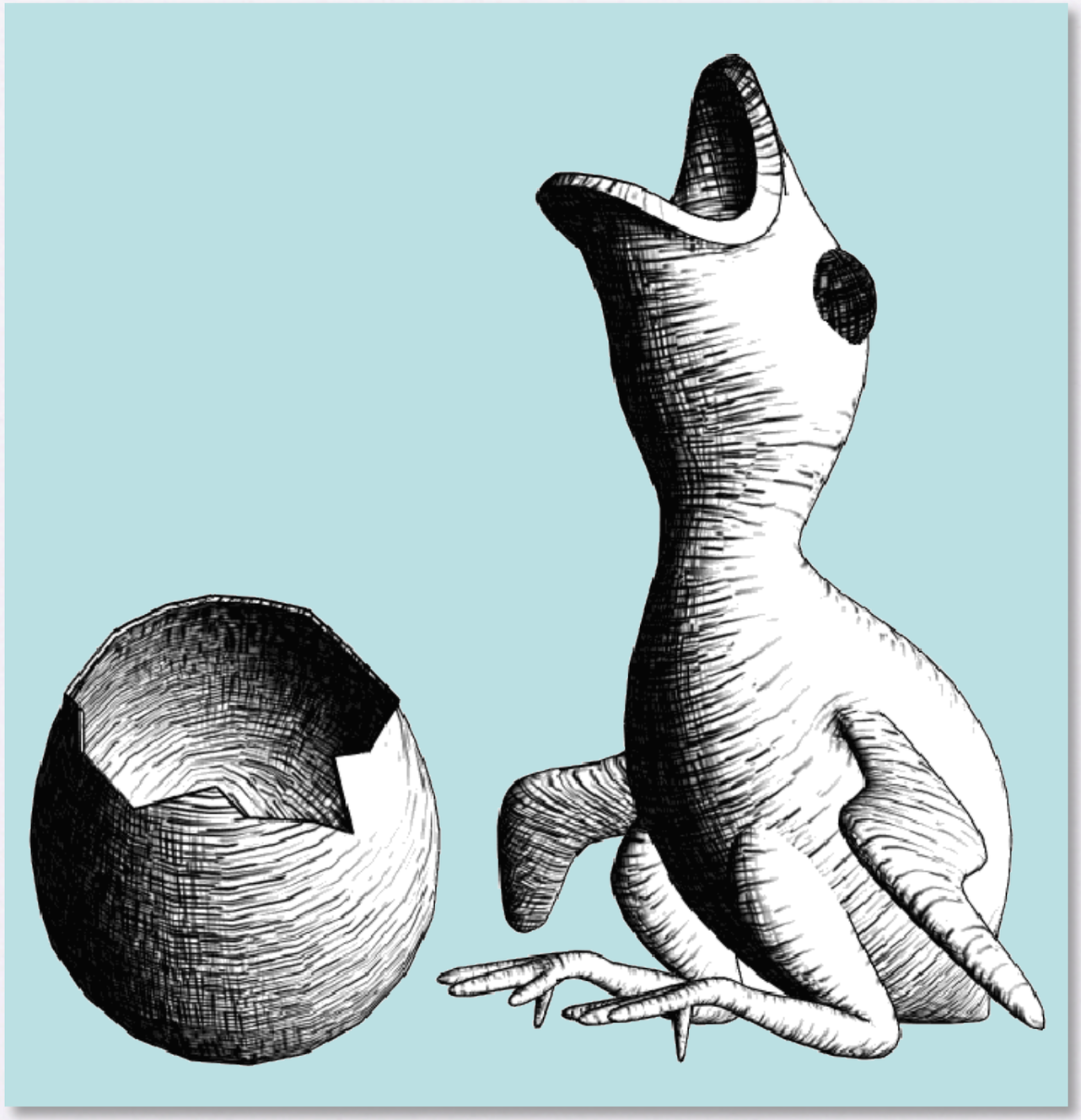
Winkenbach & Salesin (1994)

Related Work



Lake et al. (2000)

Related Work



Praun et al. (2001)

Related Work

Webb et al. (2002)

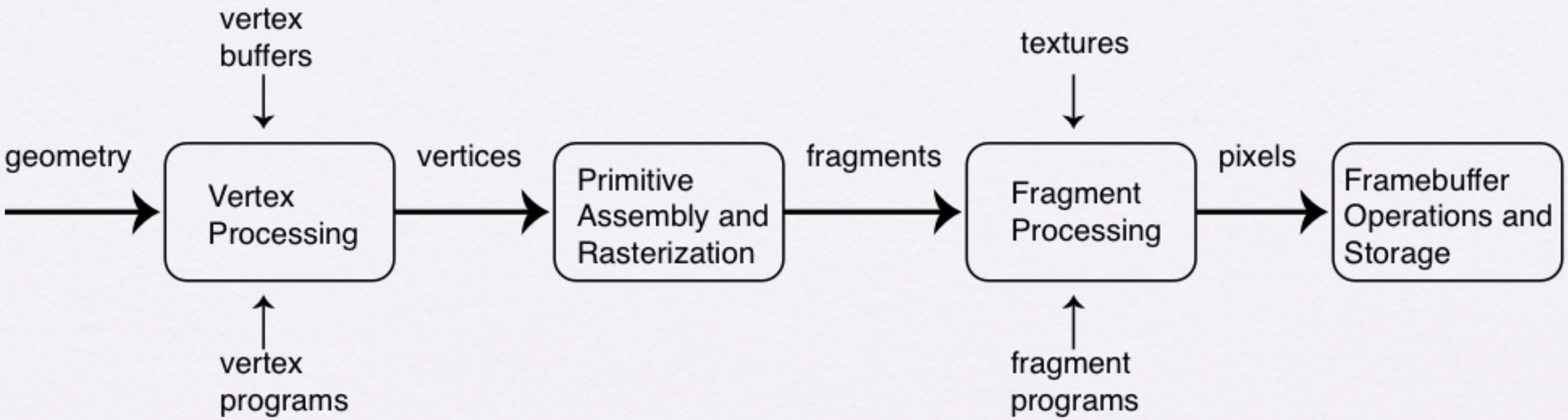
Related Work

- mostly automatically generated hatching
- slow and expressive vs. fast but restrictive

My Work

- explicit stroke generation
  - geometry-based
  - rendered by vertex program
- implicit stroke generation
  - texture-based
  - rendered by fragment program

Rendering Pipeline



Width and Density

- pen-and-ink:
  - constant stroke width
  - varying stroke density
- wood-cut:
  - varying stroke width
  - constant stroke density

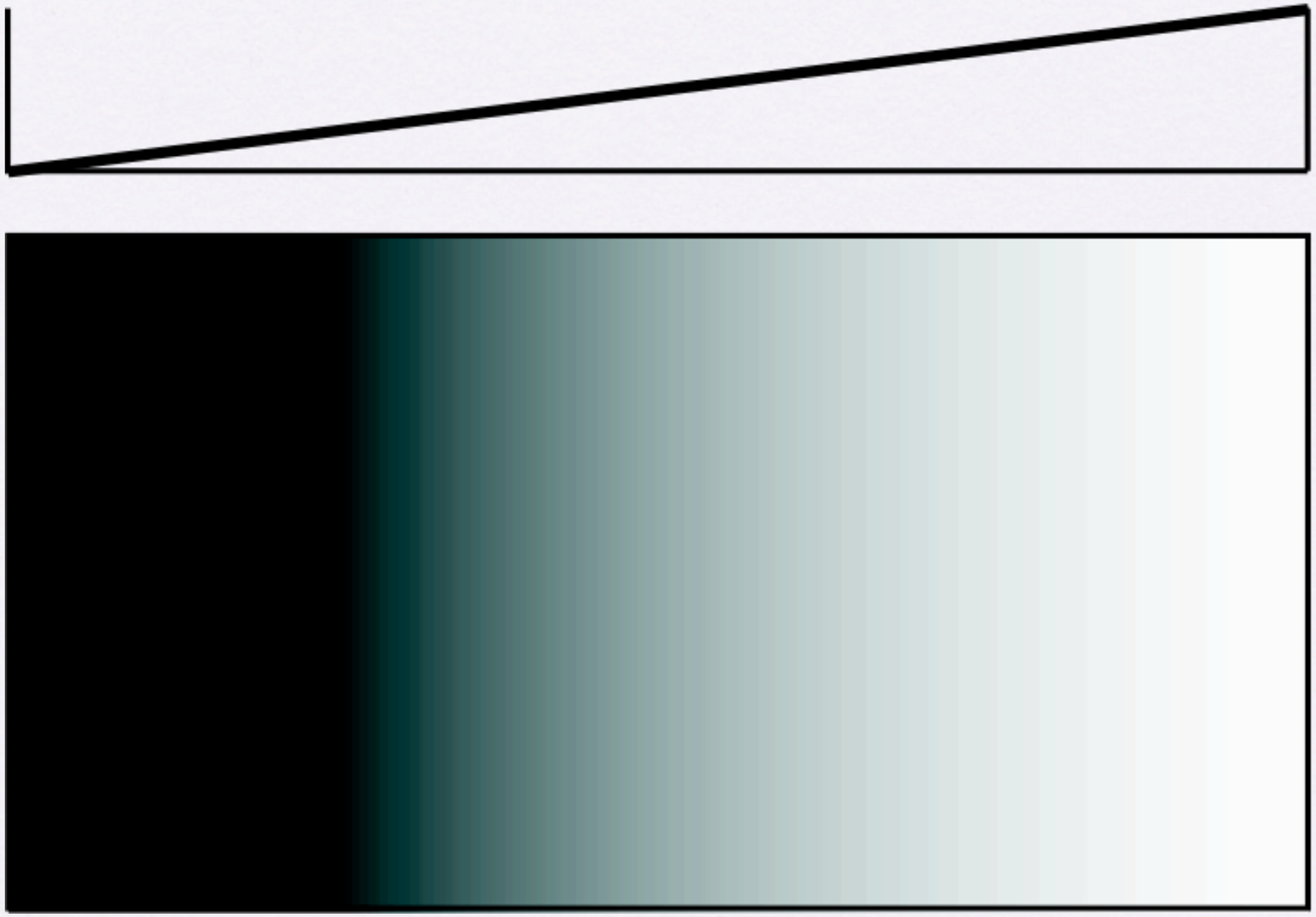
Stroke Width

- relatively simple to adjust
- must be independent of distance

Intensity-Based Stroke Width

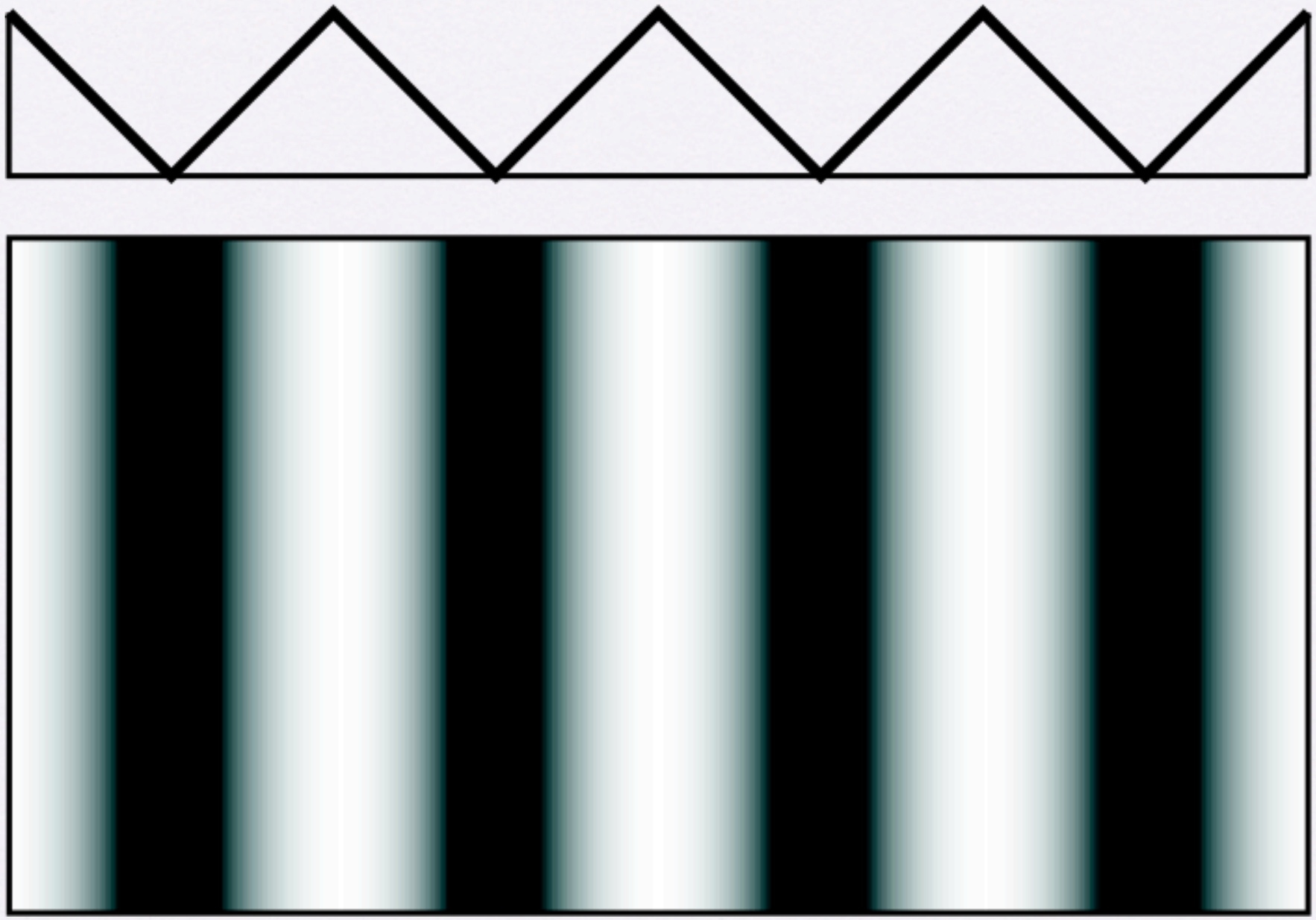
- halftone screen
  - contains multiple widths
- threshold operation
  - selects appropriate width

Intensity-Based Stroke Width



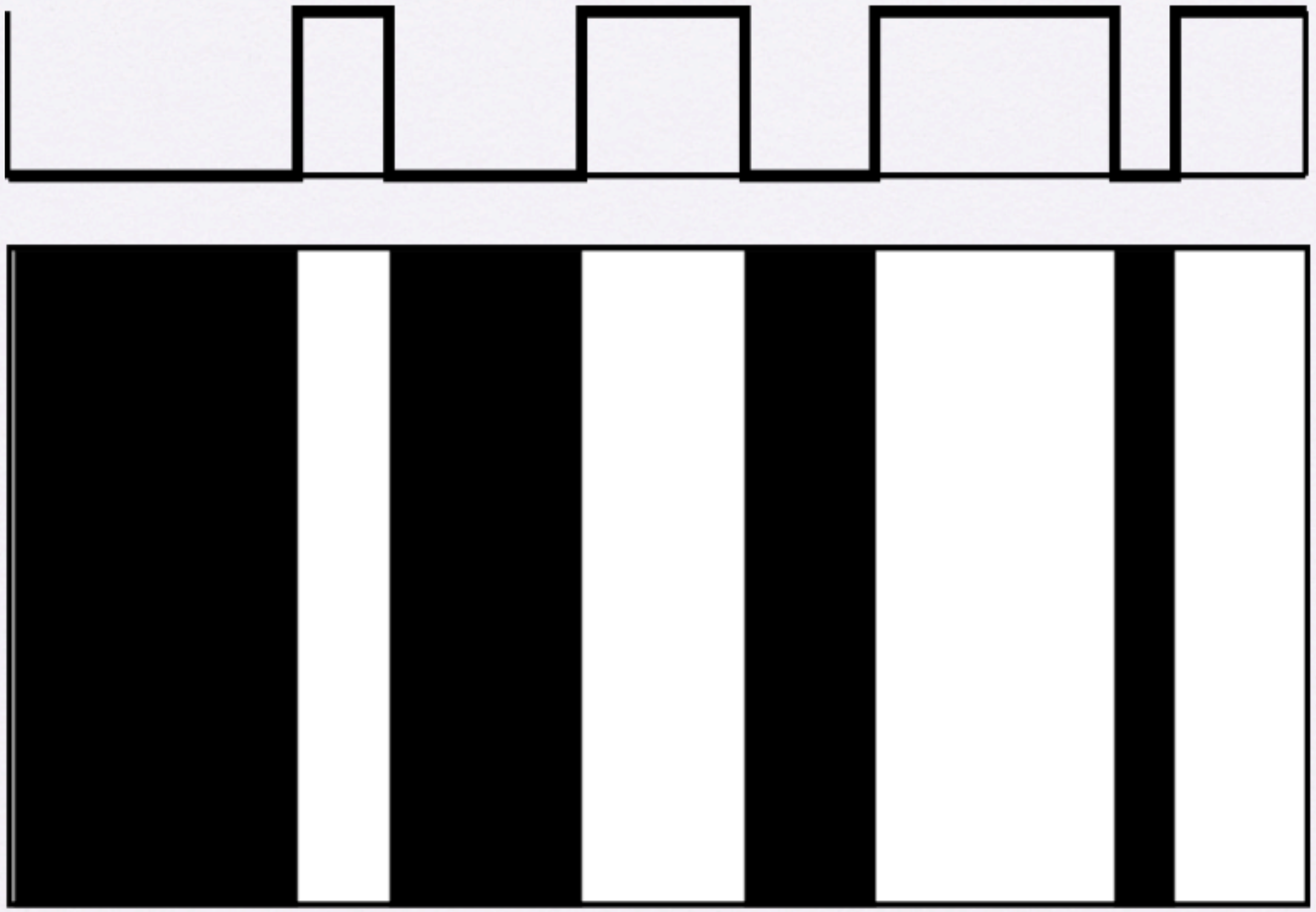
Intensity

Intensity-Based Stroke Width



Halftone Screen

Intensity-Based Stroke Width

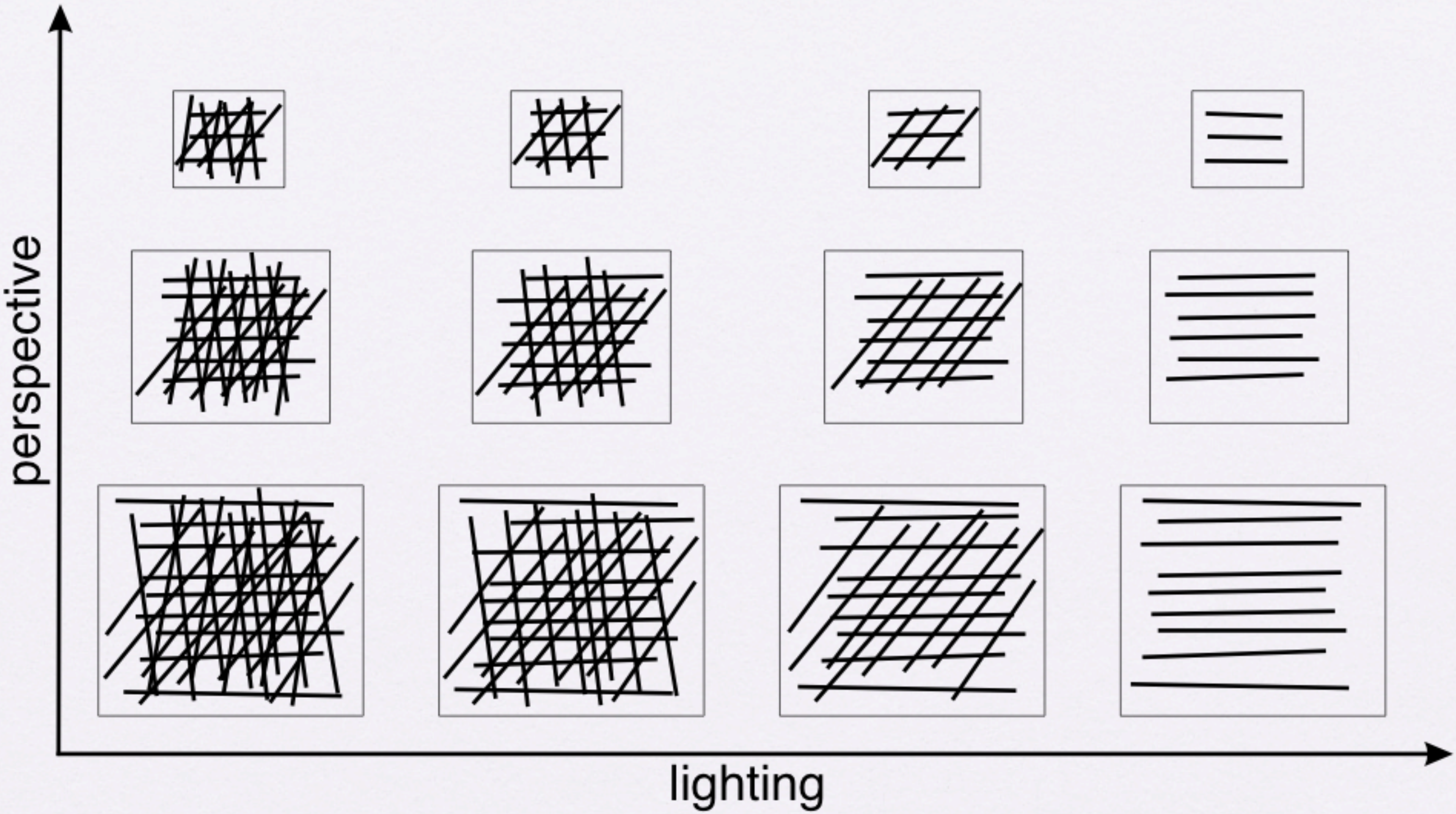


Result

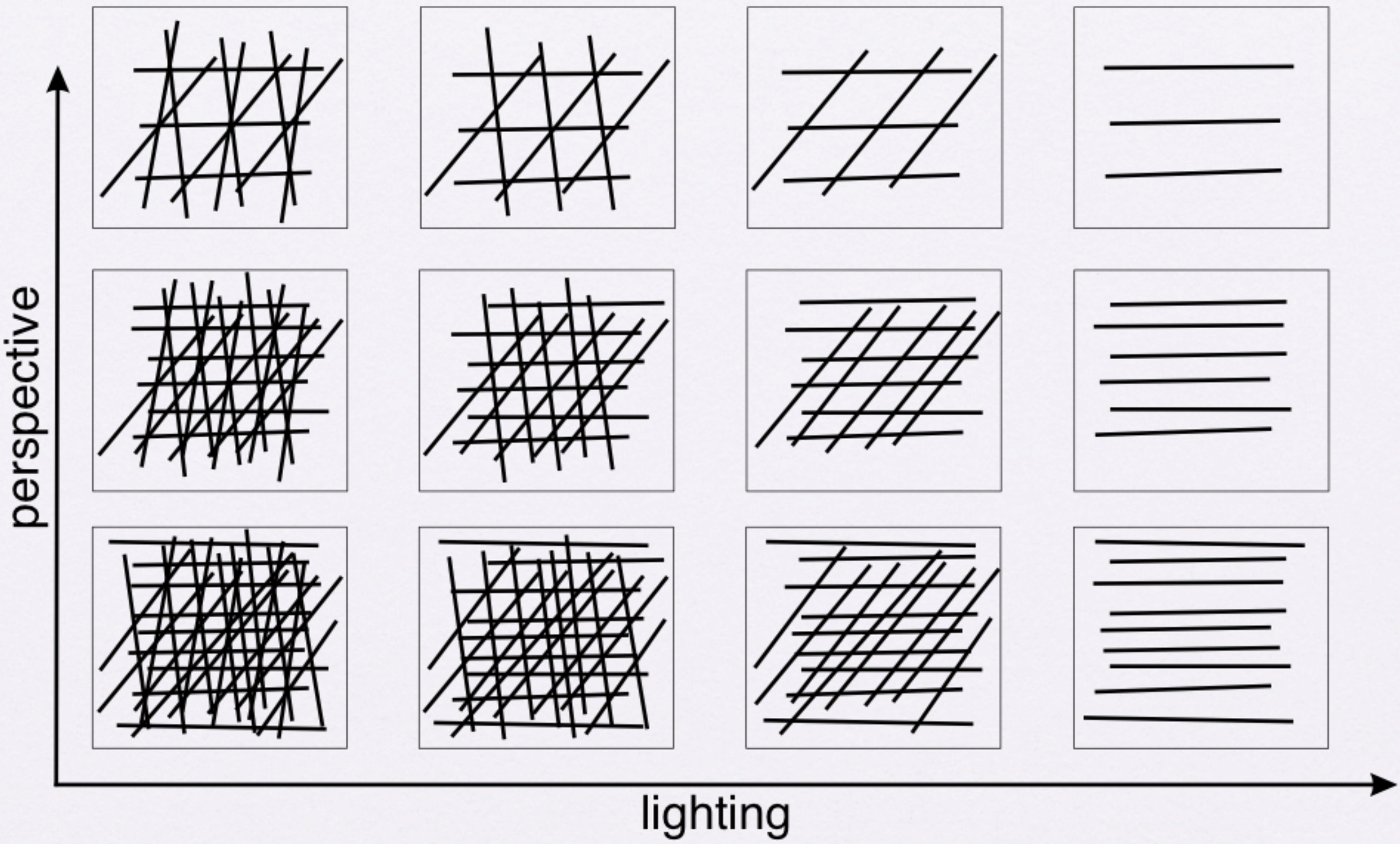
Stroke Density

- maintain density even if perspective changes
- adjust density according to lighting

Stroke Density



Stroke Density



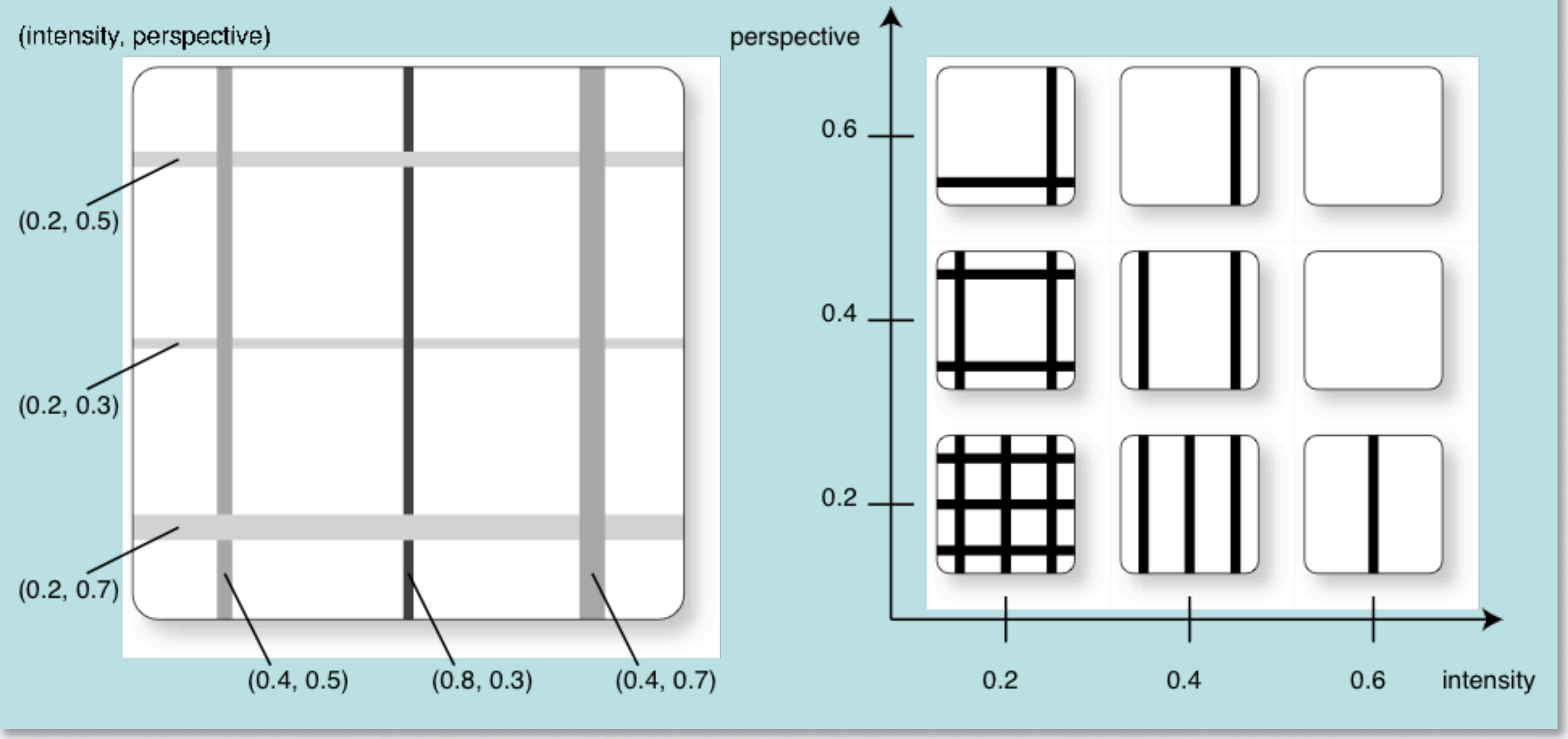
Dual Threshold

- disassociate density control:
  - to reflect lighting
  - to depict perspective
- separate thresholds

Dual Threshold

- intensity
  - lighting
  - material
- perspective size
  - distance
  - slope

Dual Threshold



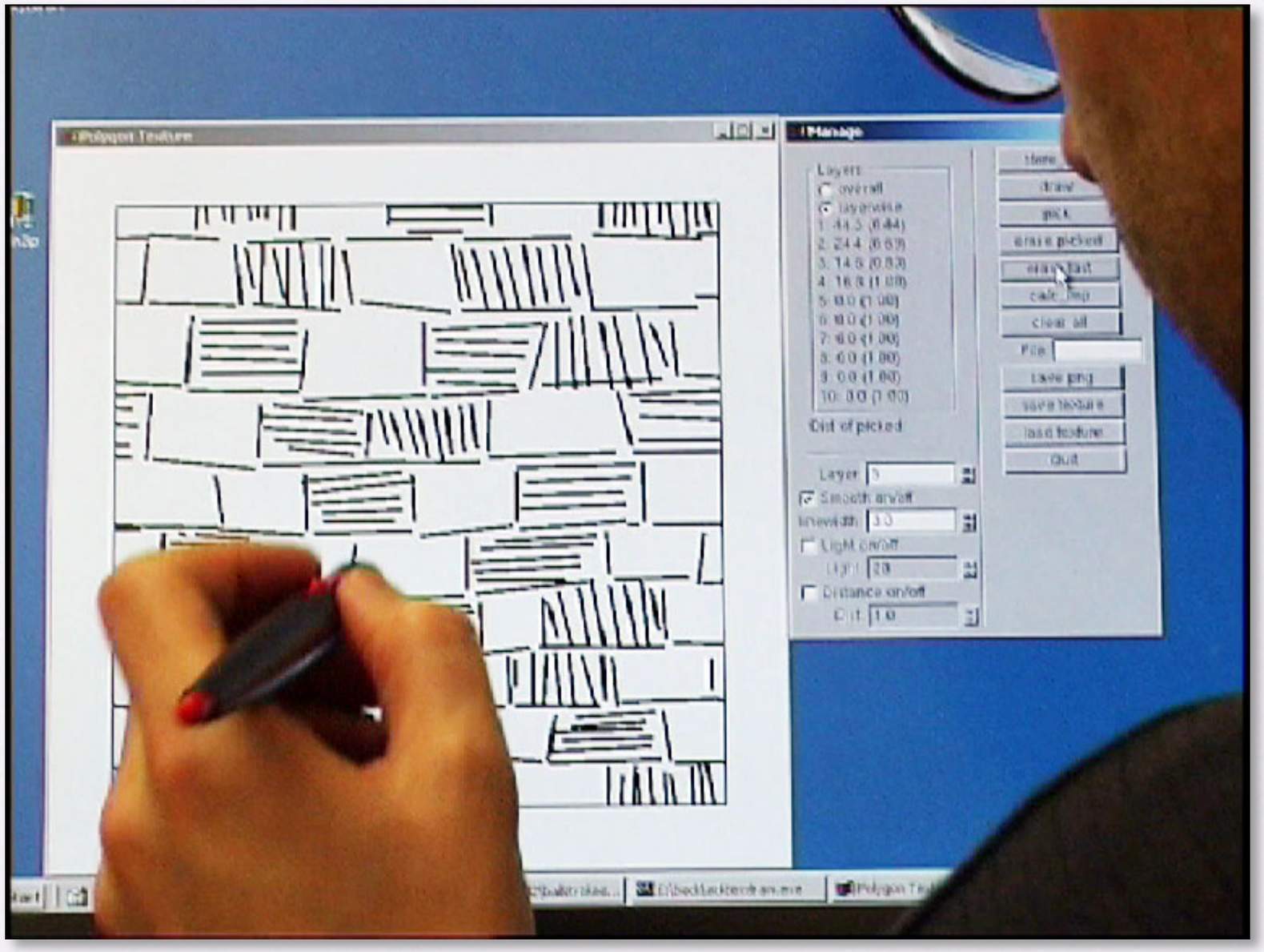
Explicit Strokes

- geometrically defined strokes
- directly implement dual threshold scheme
- vertex program

Creating Stroke Textures

- interactive drawing tool
- adjust for distance and lighting
  - creates dual stroke thresholds
- stored as both, vector data and bitmaps

Creating Stroke Textures



interactive drawing tool

Surface Parametrization

- bitmap textures applied to surface
  - conventional modeller (3ds max)
  - helps in adjusting texture coordinates
- exported for stroke application

Stroke Application

- strokes placed on surface according to texture coordinates
- clipped to polygon boundaries

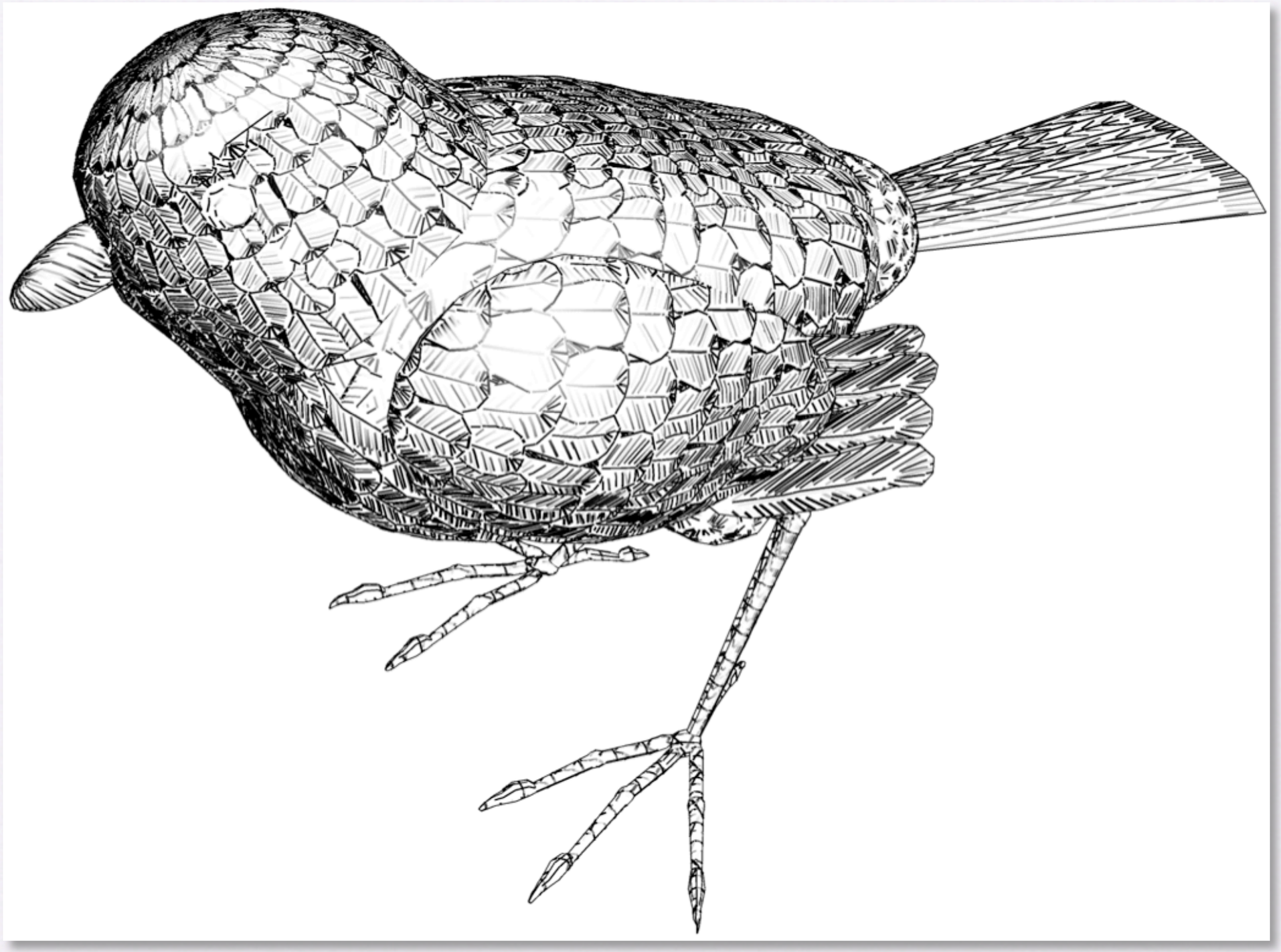
Stroke Rendering

- vertex program
  - calculate lighting and perspective values
  - compare to respective thresholds
  - possibly discard stroke

Discarding Strokes

- discard primitive
  - not possible in a vertex progra
- set alpha to zero
  - alpha test discards fragments
- make line width zero
  - degenerated polygon creates no fragments

Explicit Strokes



ca. 80.000 strokes

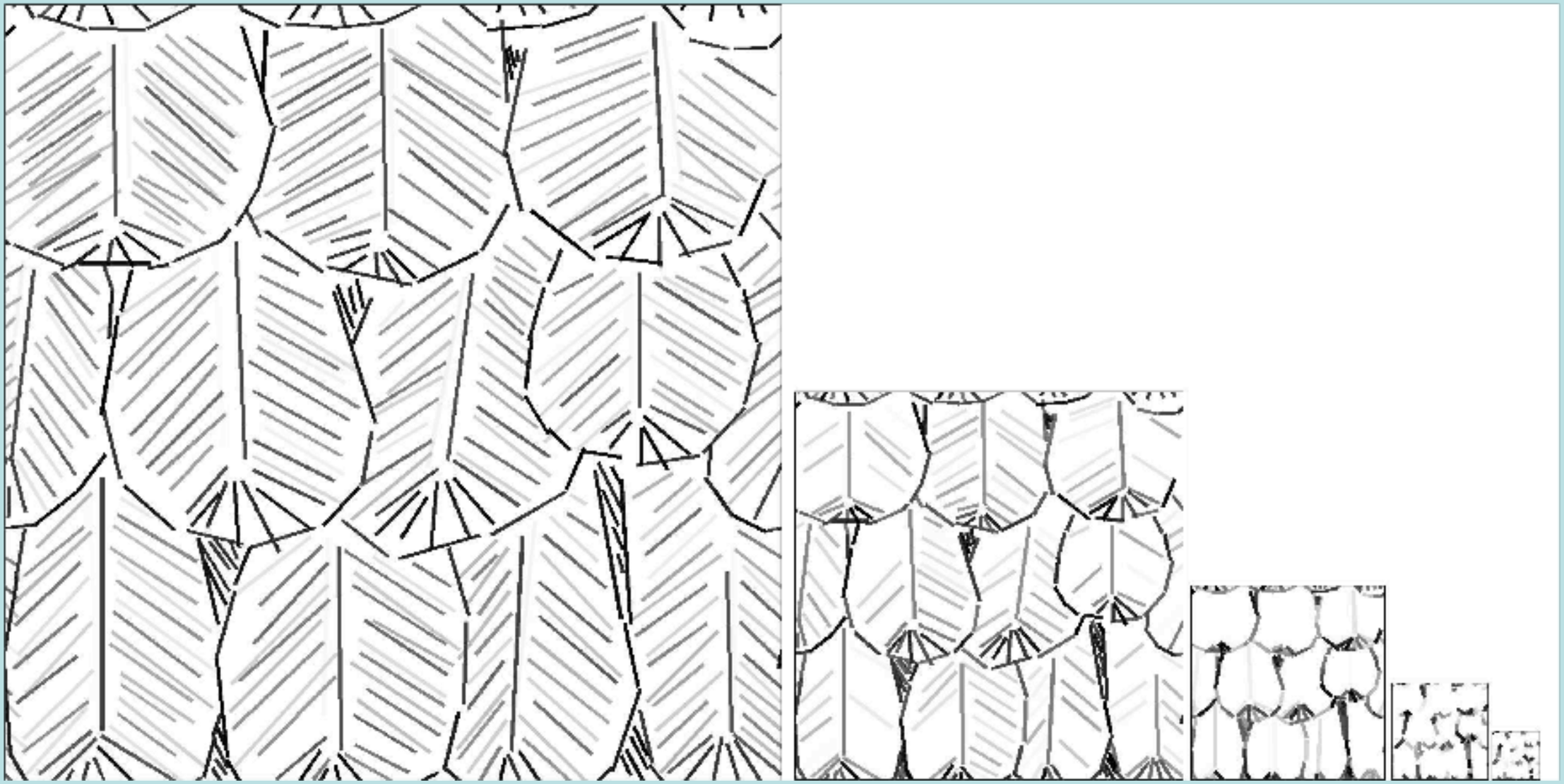
Explicit Strokes

- huge number of strokes
- individually drawn
- flexible, but inefficient

Implicit Strokes

- strokes implicitly encoded in halftone screen
- threshold operation in texture stage
- maintain density and width by mipmapping
- well suited for common hardware

Constant Density and Width

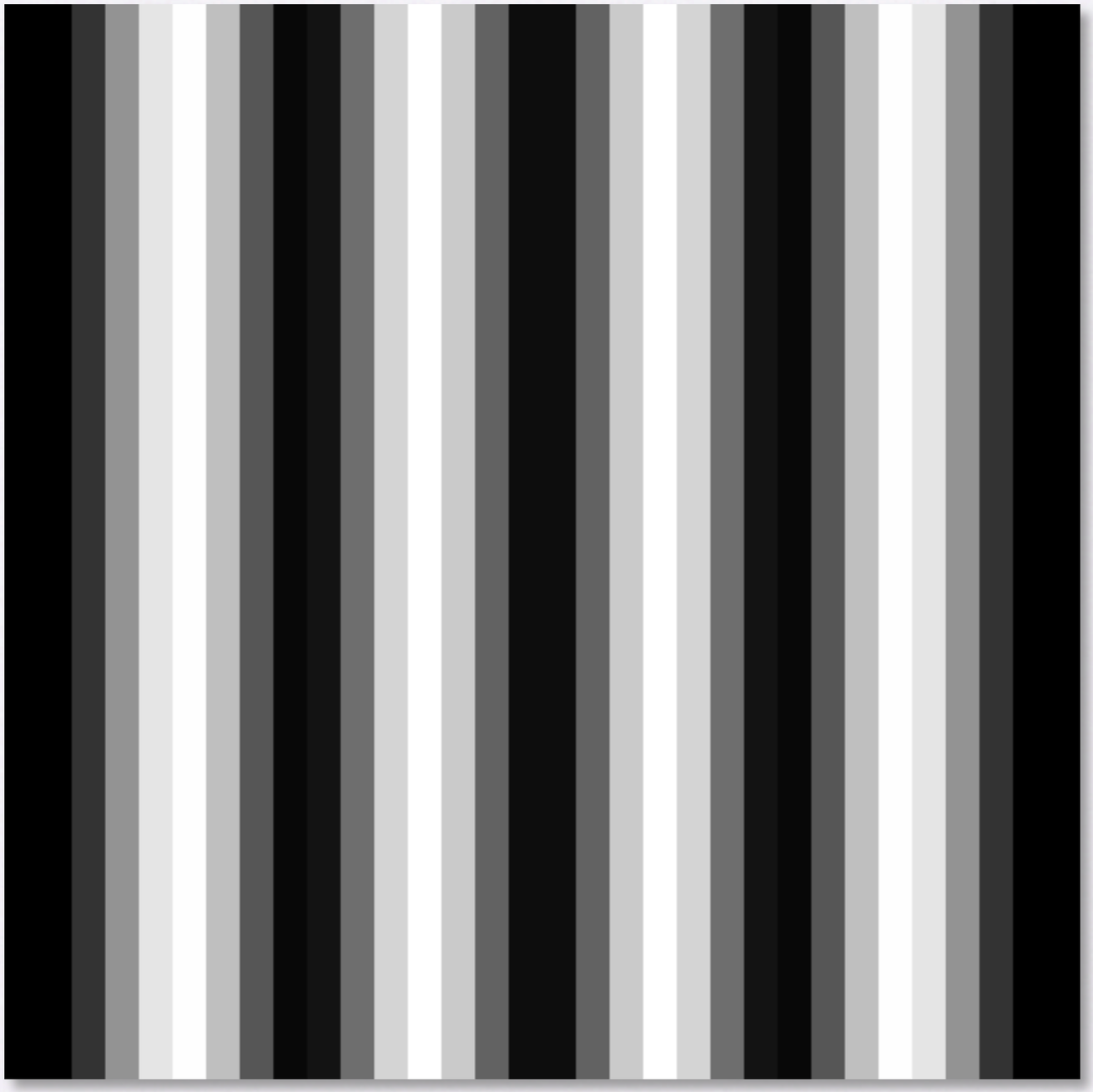


Series of Mipmaps

Stroke Width

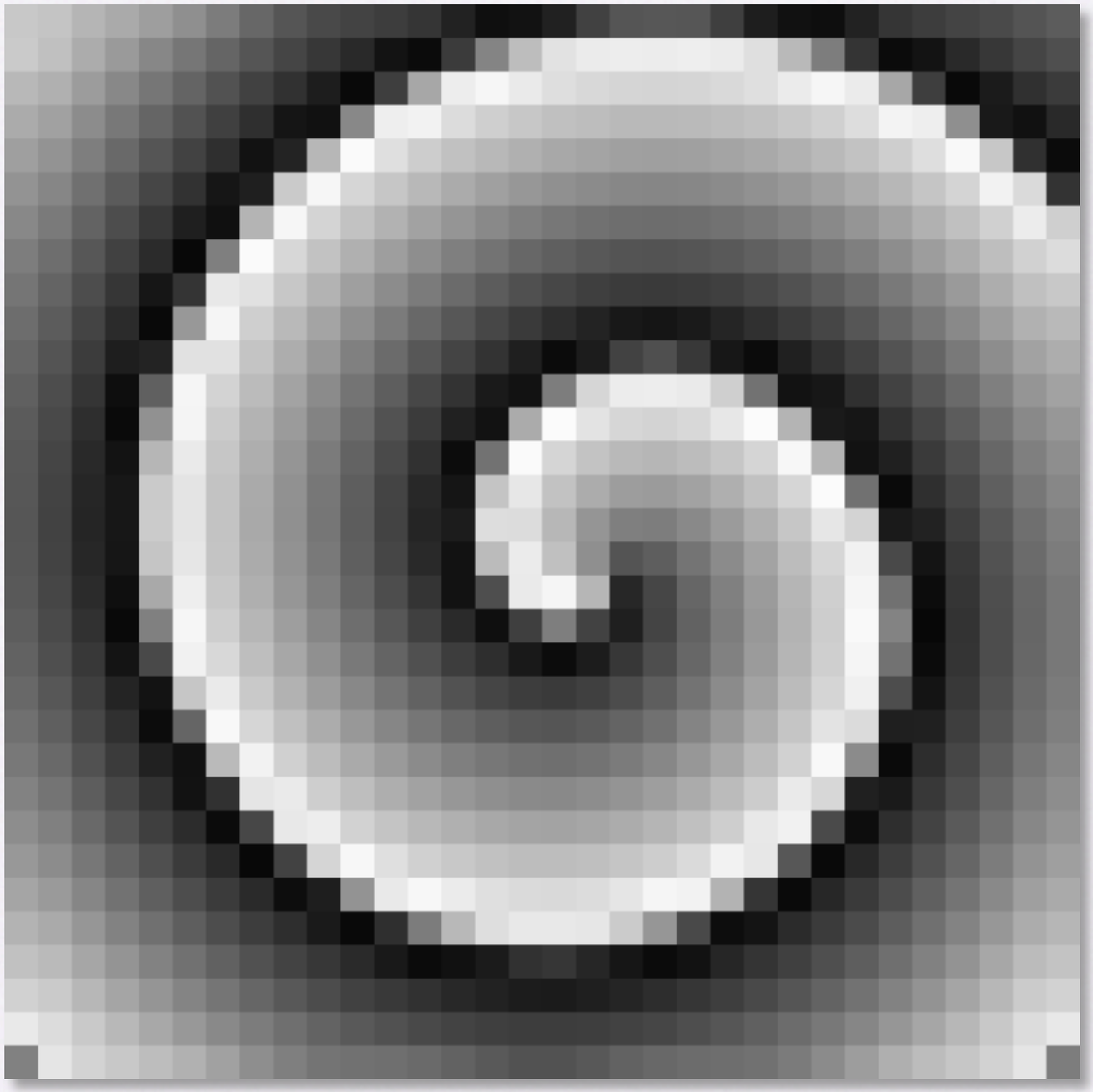
- special halftone screen yields varying width

Stroke Width



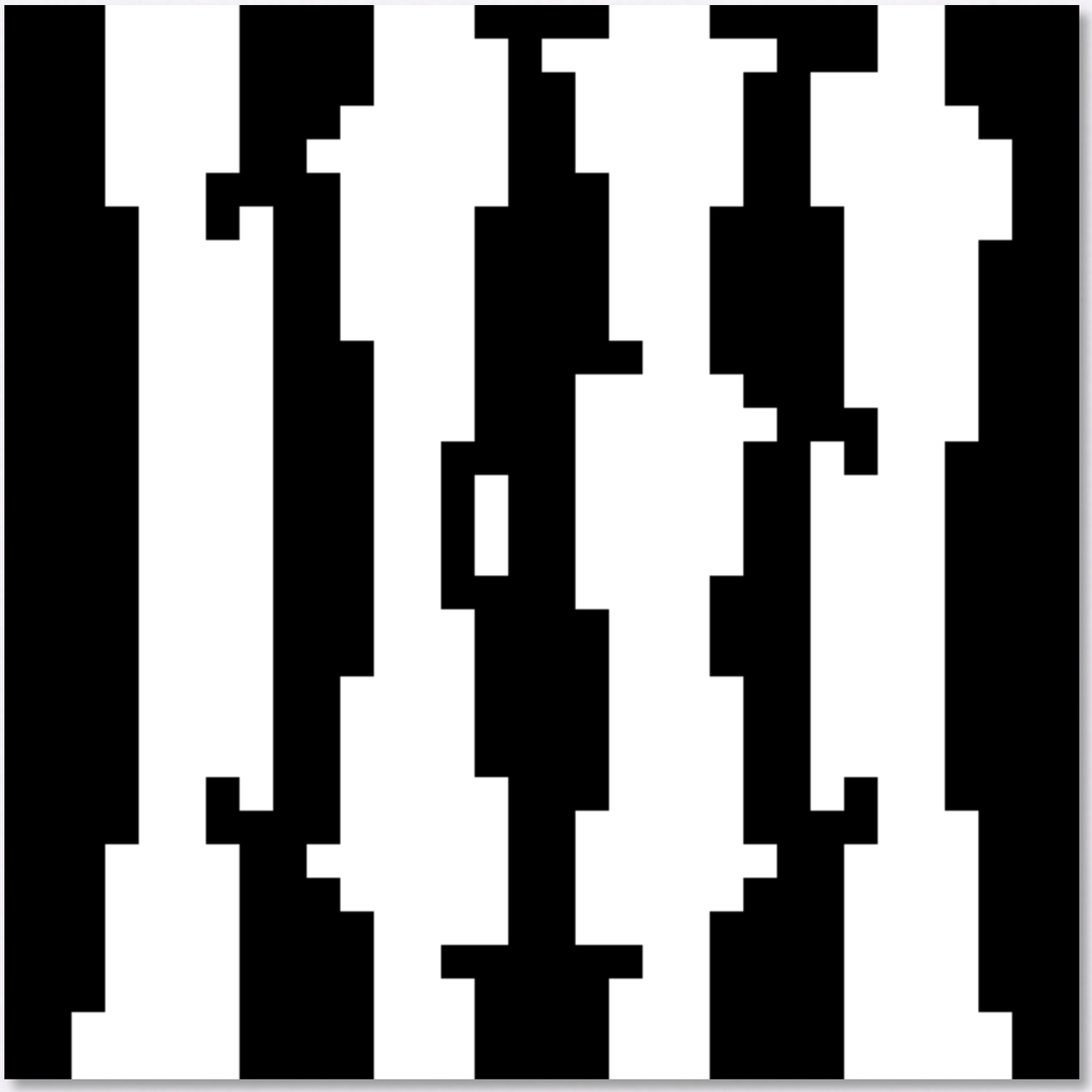
Halftone Screen

Stroke Width



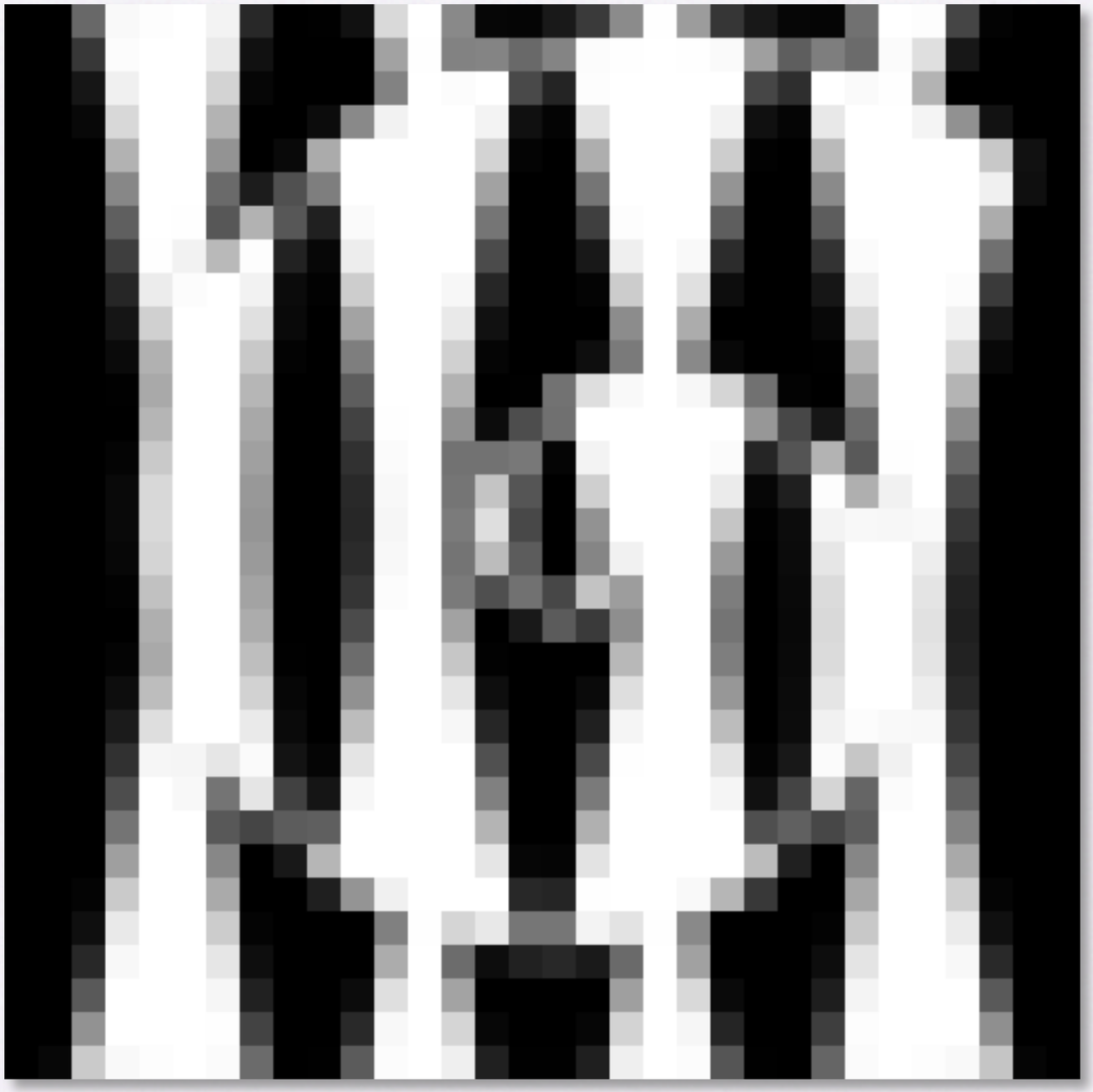
Intensity

Stroke Width



Halftoned Result

Stroke Width

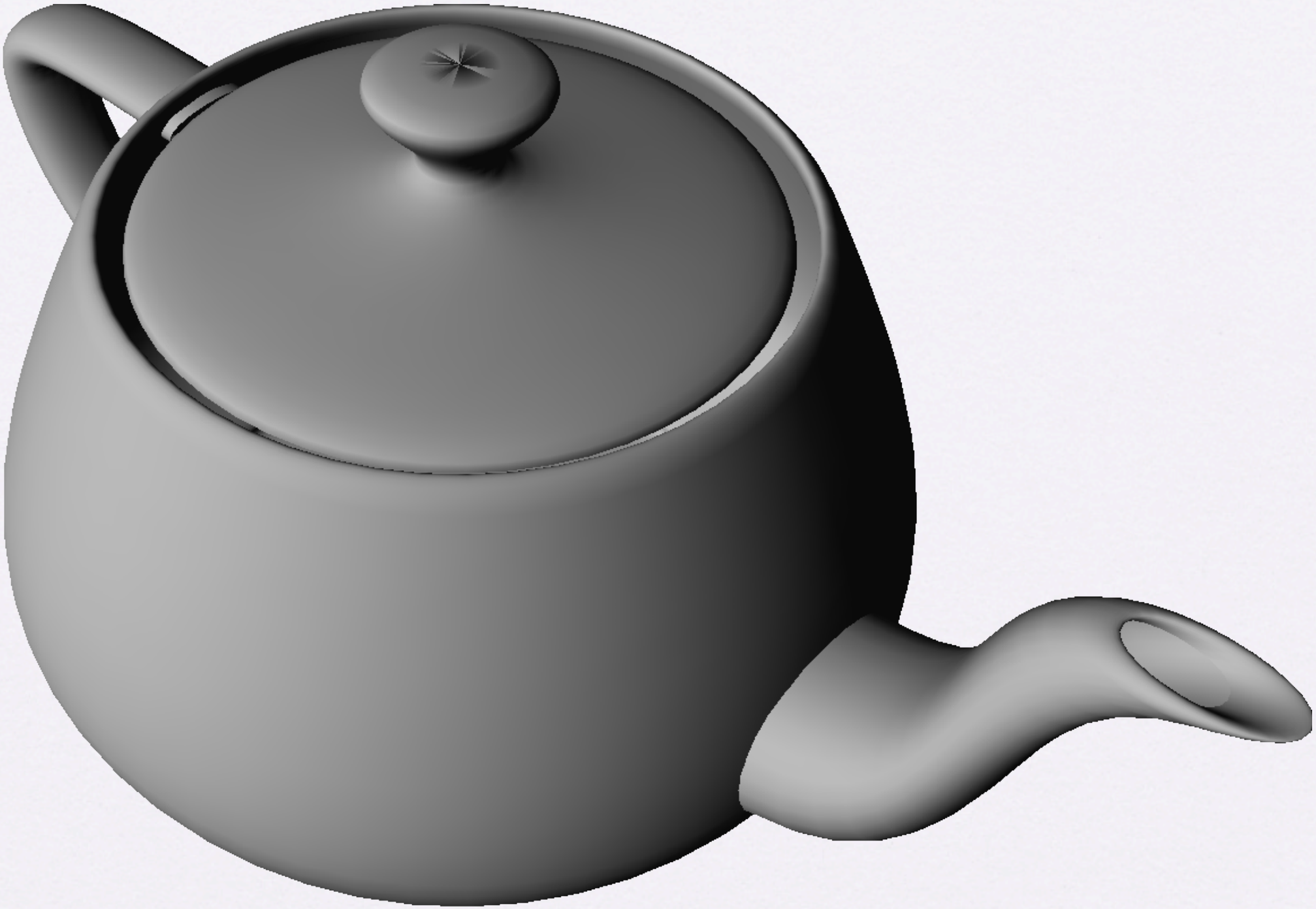


Smooth Threshold Operation

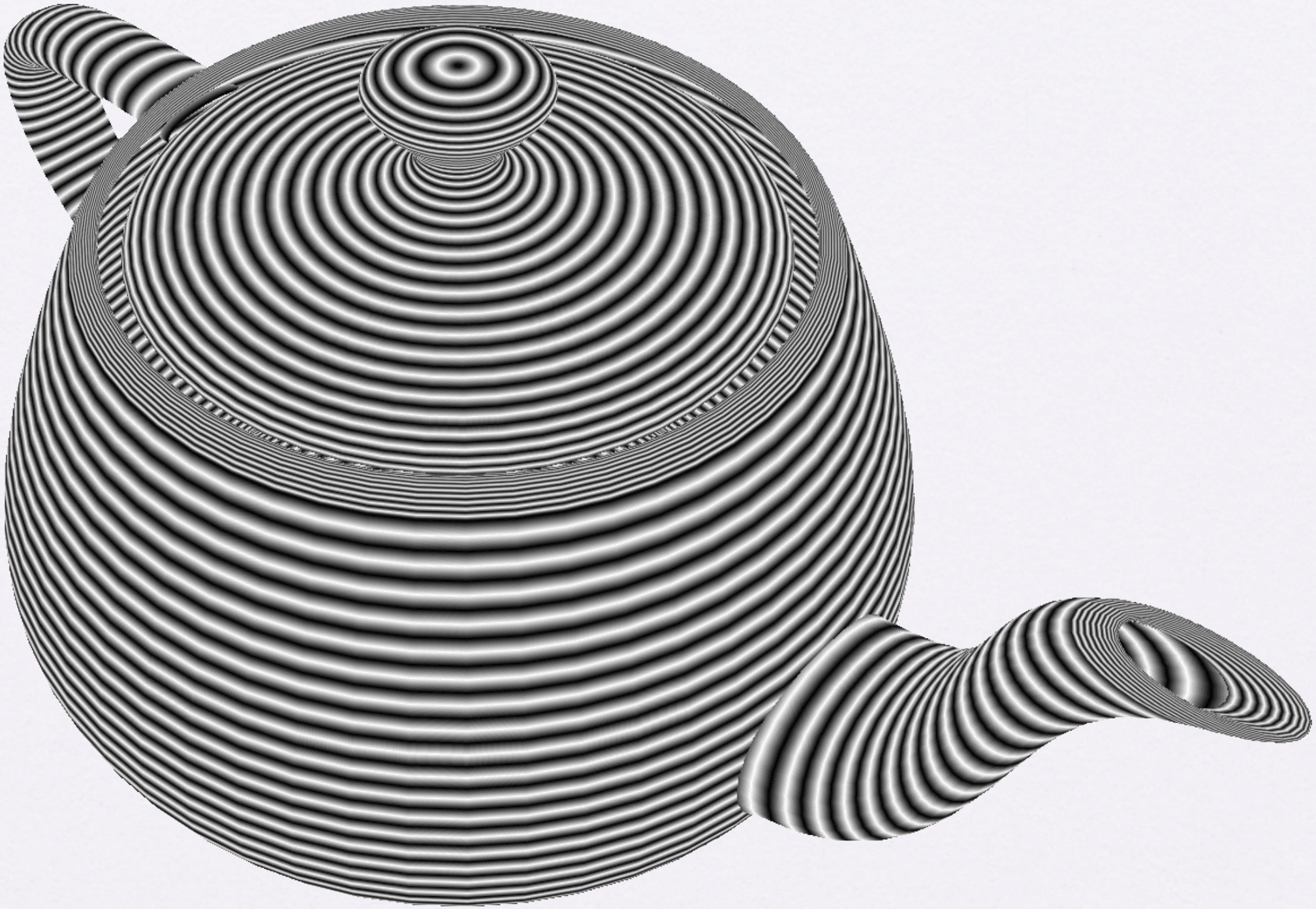
Implicit Halftoning

- simple threshold:
  - aliasing
- smooth threshold
  - not only black/white

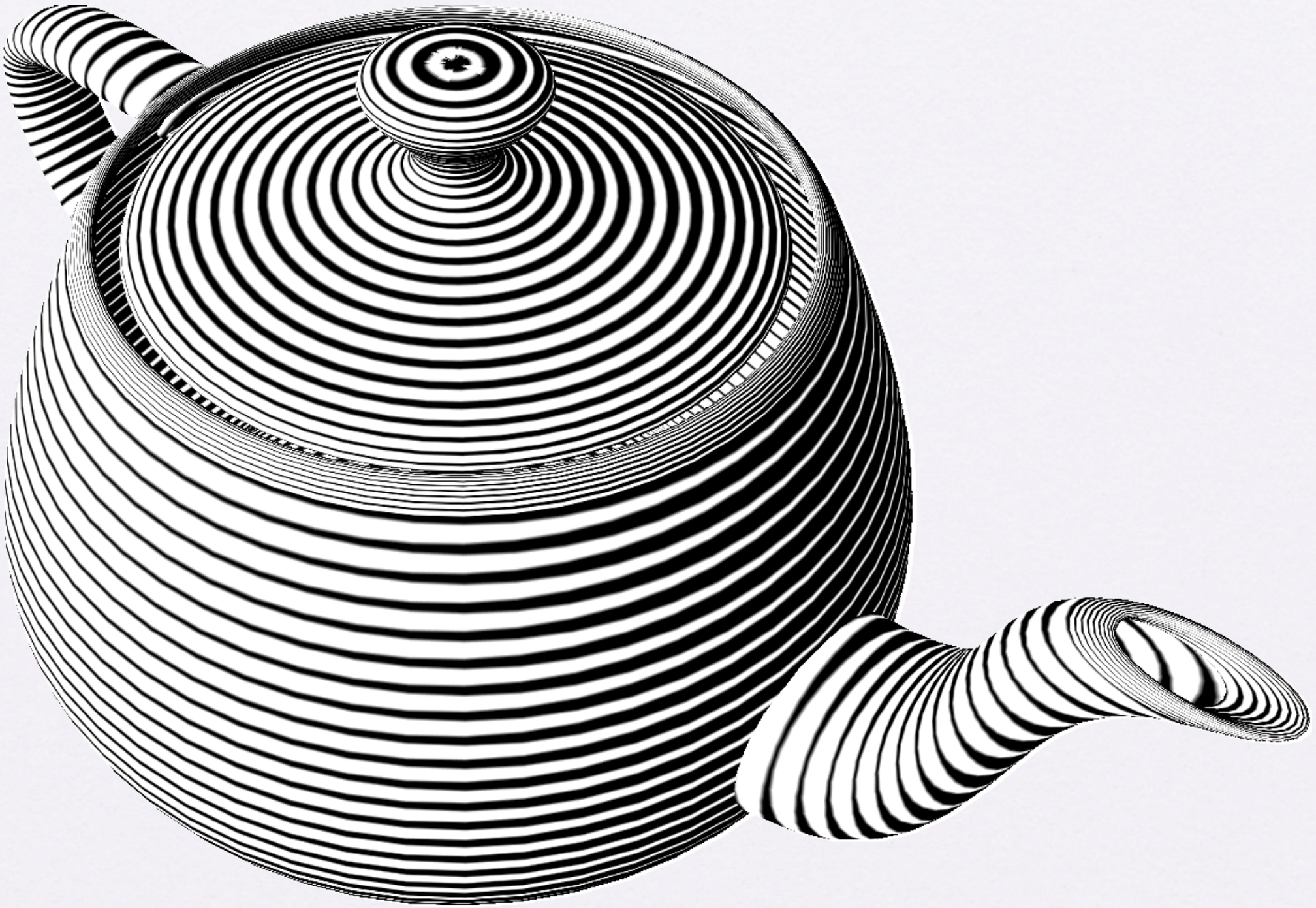
Implicit Halftoning



Implicit Halftoning



Implicit Halftoning



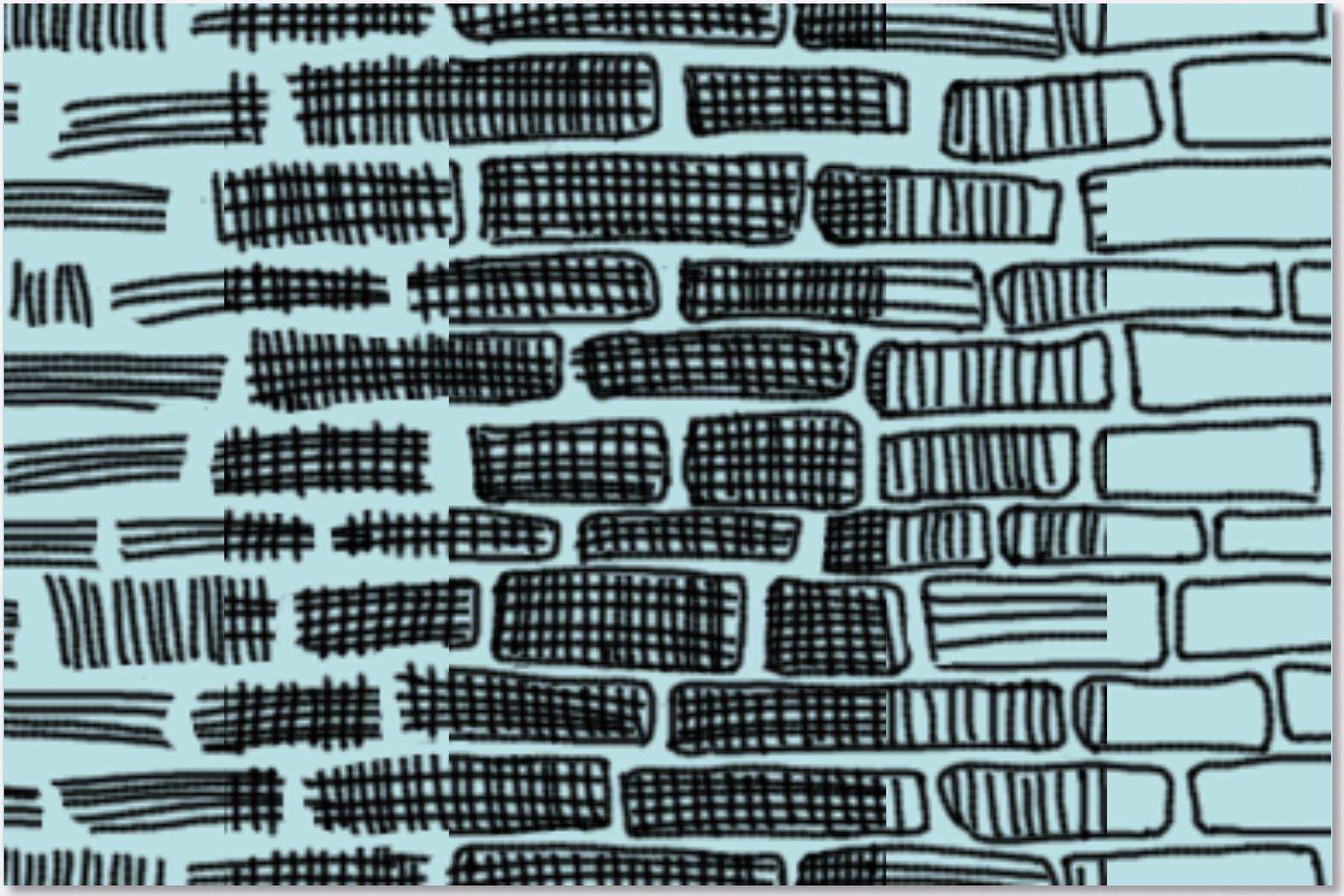
Smooth Threshold

- simple formula
  - **tex** = half-res screen
  - **out** = **tmp**, **tex**, **col**) << 2;
- only two texture stages
- works on virtually any board sold since 1999

Stroke Density

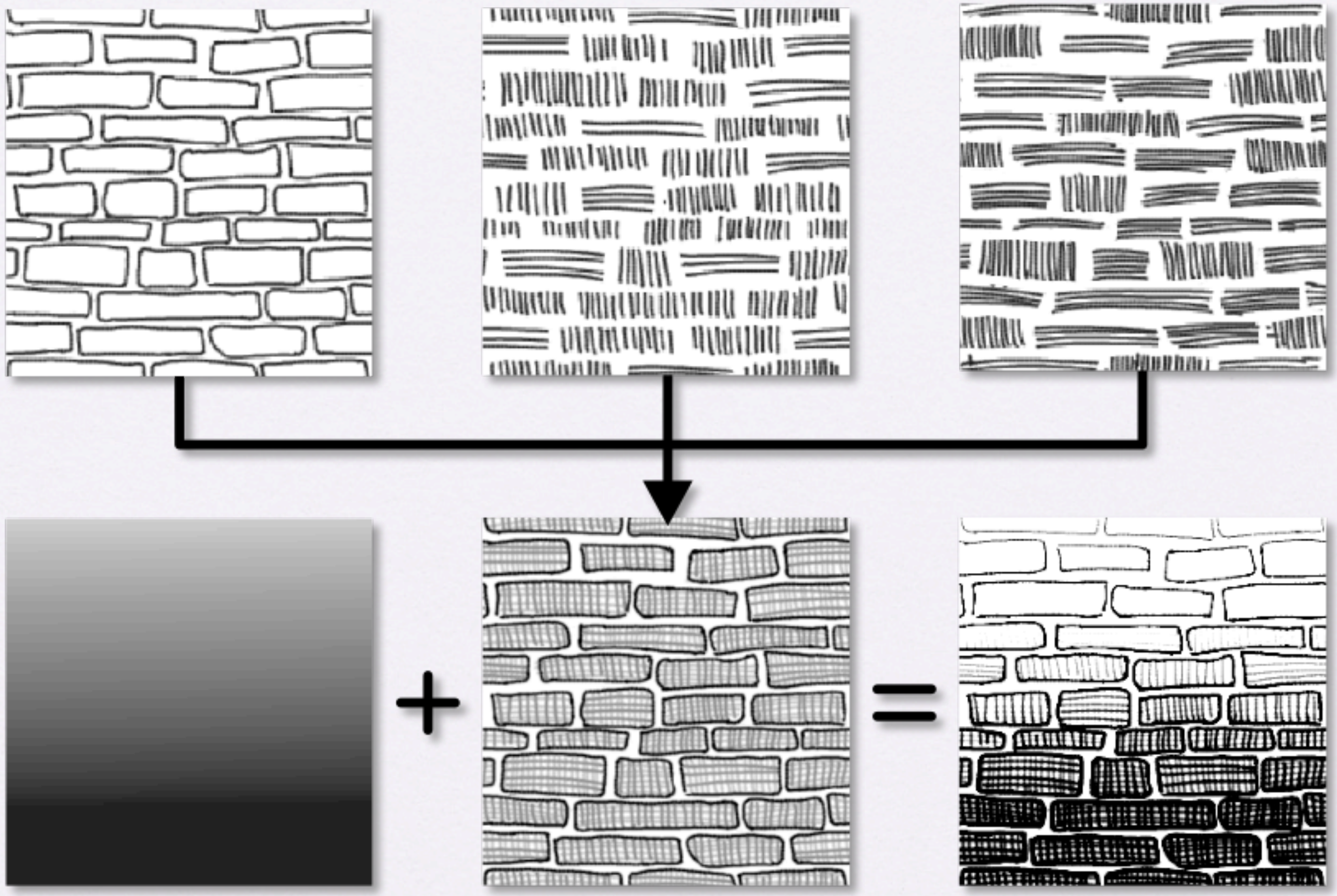
- construct halftone screen with increading density
- use same smooth threshold operate

Stroke Density



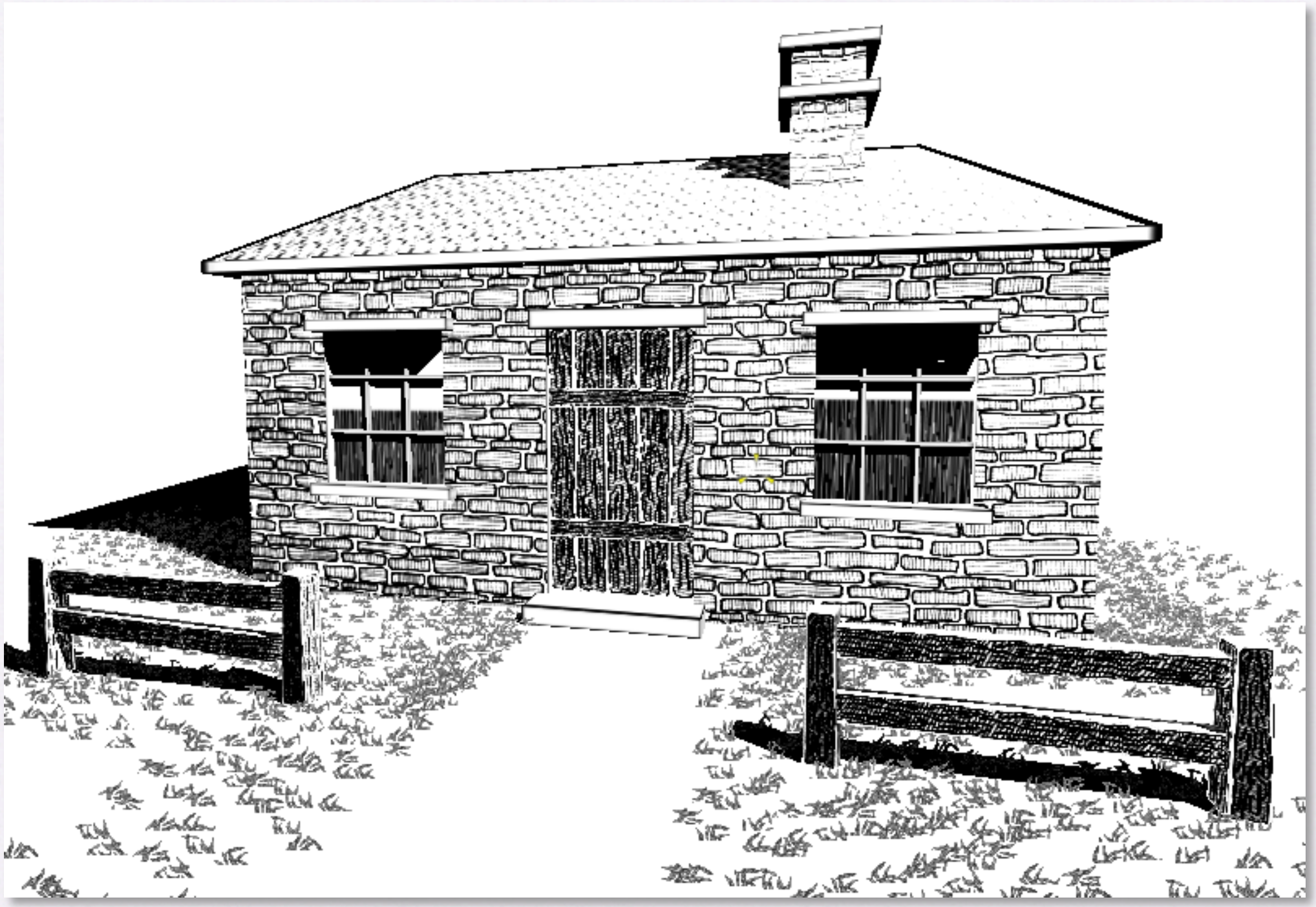
multiple layers

Stroke Density



construct halftone screen

Stroke Density



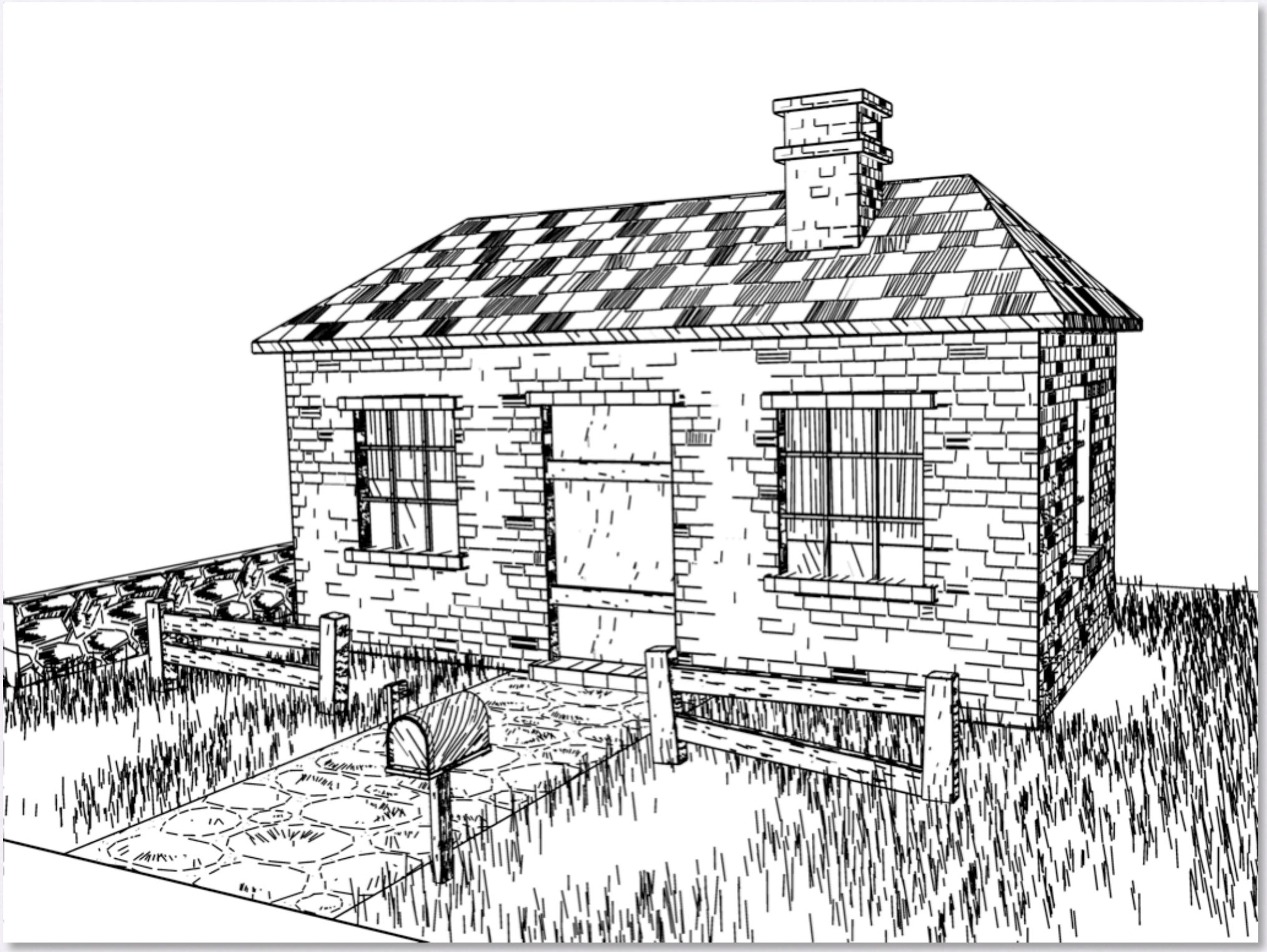
Visual Effects

- indication mapping
- individual stroke lighting
- warp map

Shading with Strokes

- both implicit and explicit techniques
- also in combination

Shading with Strokes



Overview

- Shading with Strokes
- **Outline Rendering**
- Applications
- Conclusion

Outline Rendering

- determine feature lines
- determine silhouettes
  - view-dependent
- draw both

Outline Rendering

- explicit silhouettes
  - vertex-program
- implicit
  - G-Buffer based

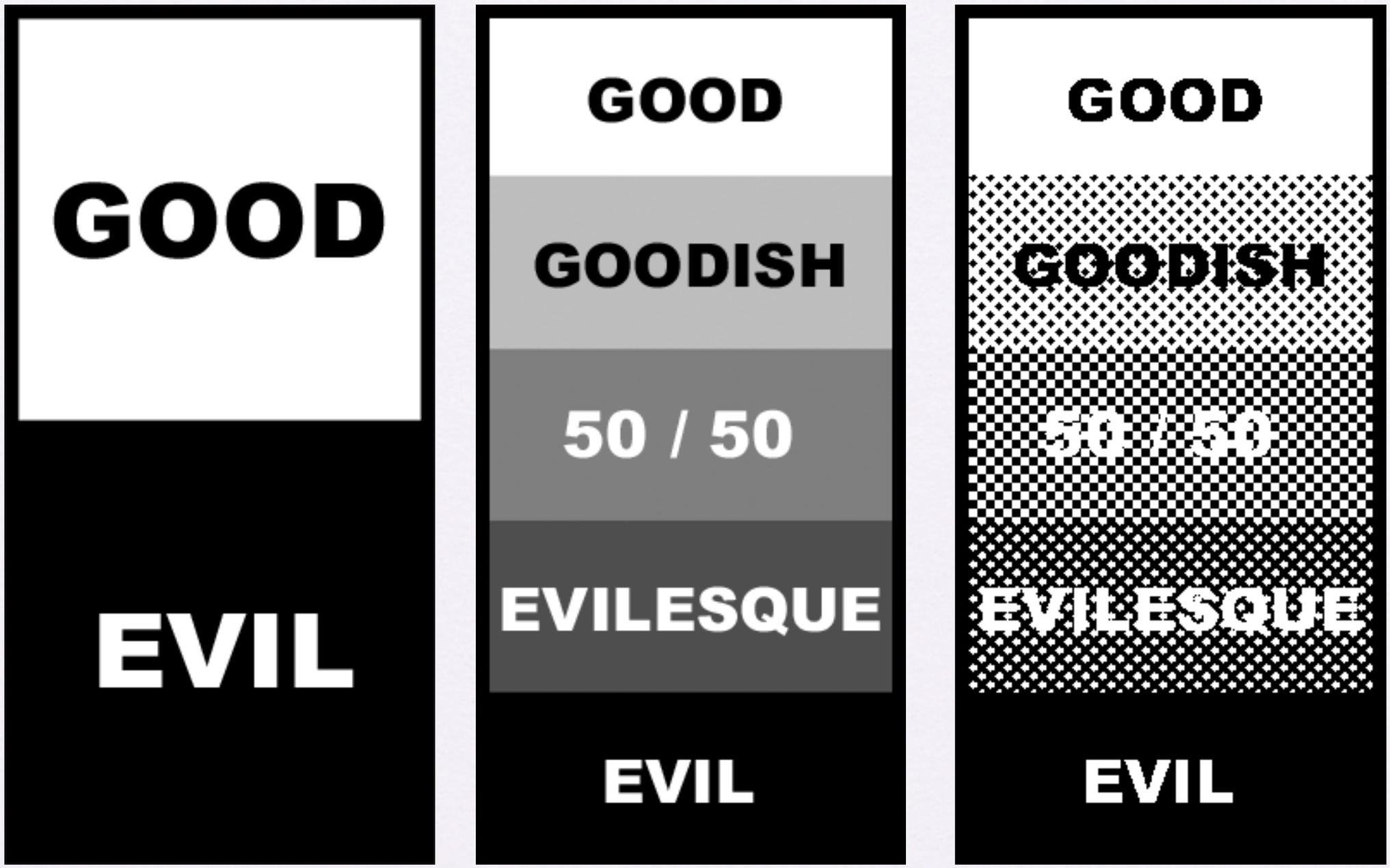
Explicit Outlines

- store potential silhouette edges
- determine facingness of adjacent faces
  - vertex program
- discard line segments
  - similar to stroke-based shading

Overview

- Shading with Strokes
- Outline Rendering
- **Applications**
- Conclusion

Halftoning

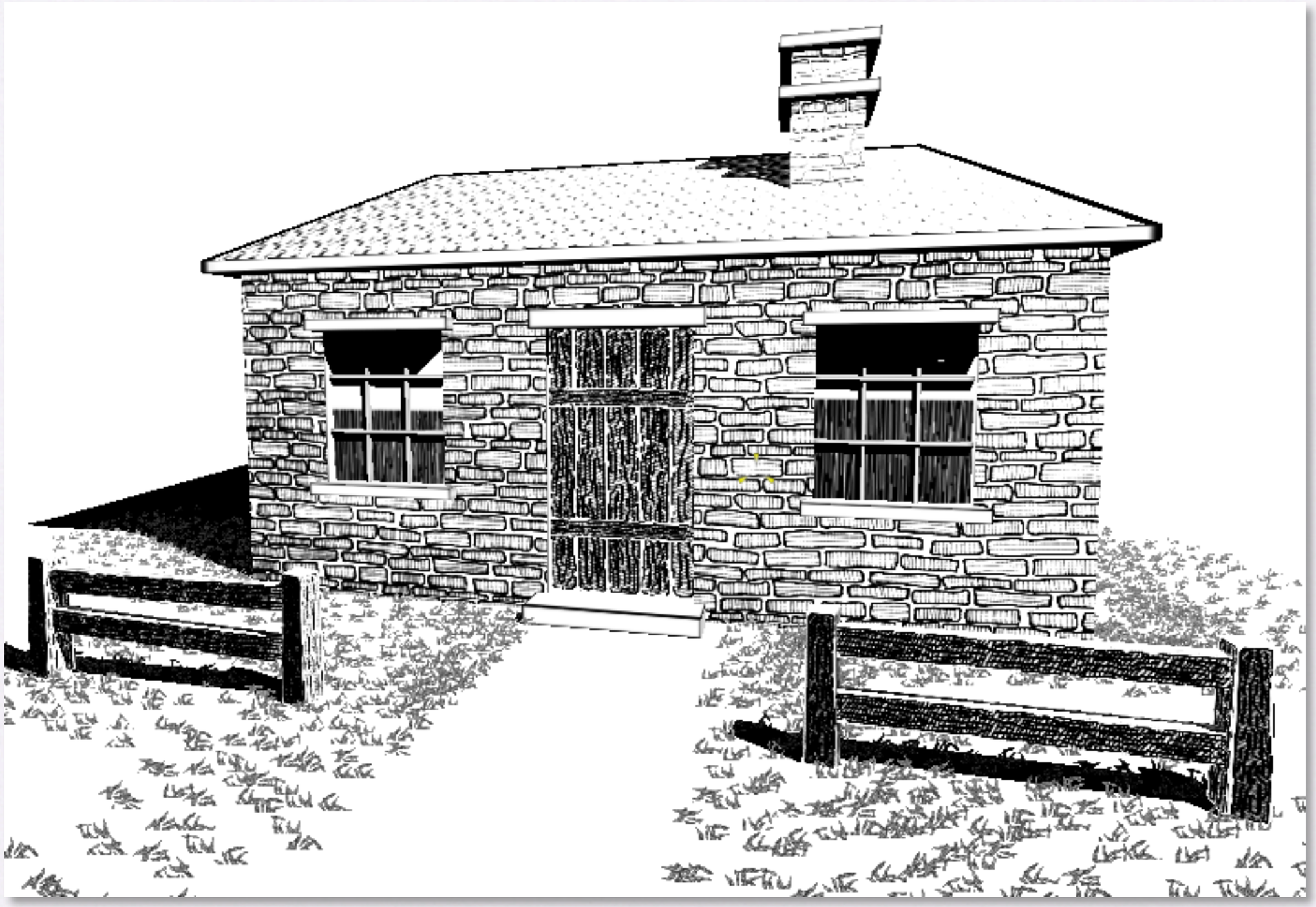

GOOD
EVIL
GOOD
GOODISH
50 / 50
EVILESQUE
EVIL
GOOD
GOODISH
50 / 50
EVILESQUE
EVIL

Slide Trash

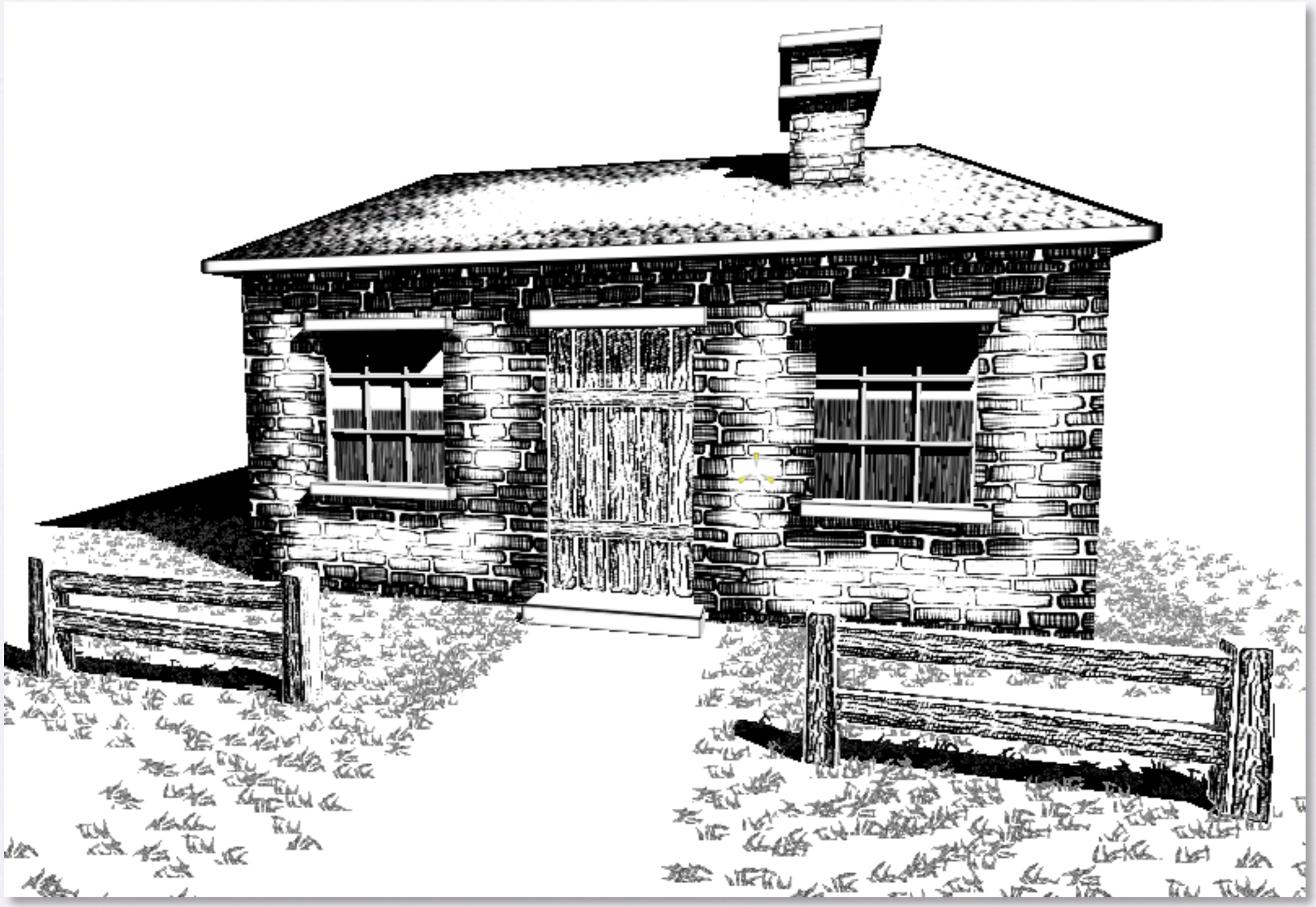
Indication Mapping

- show detail only where necessary

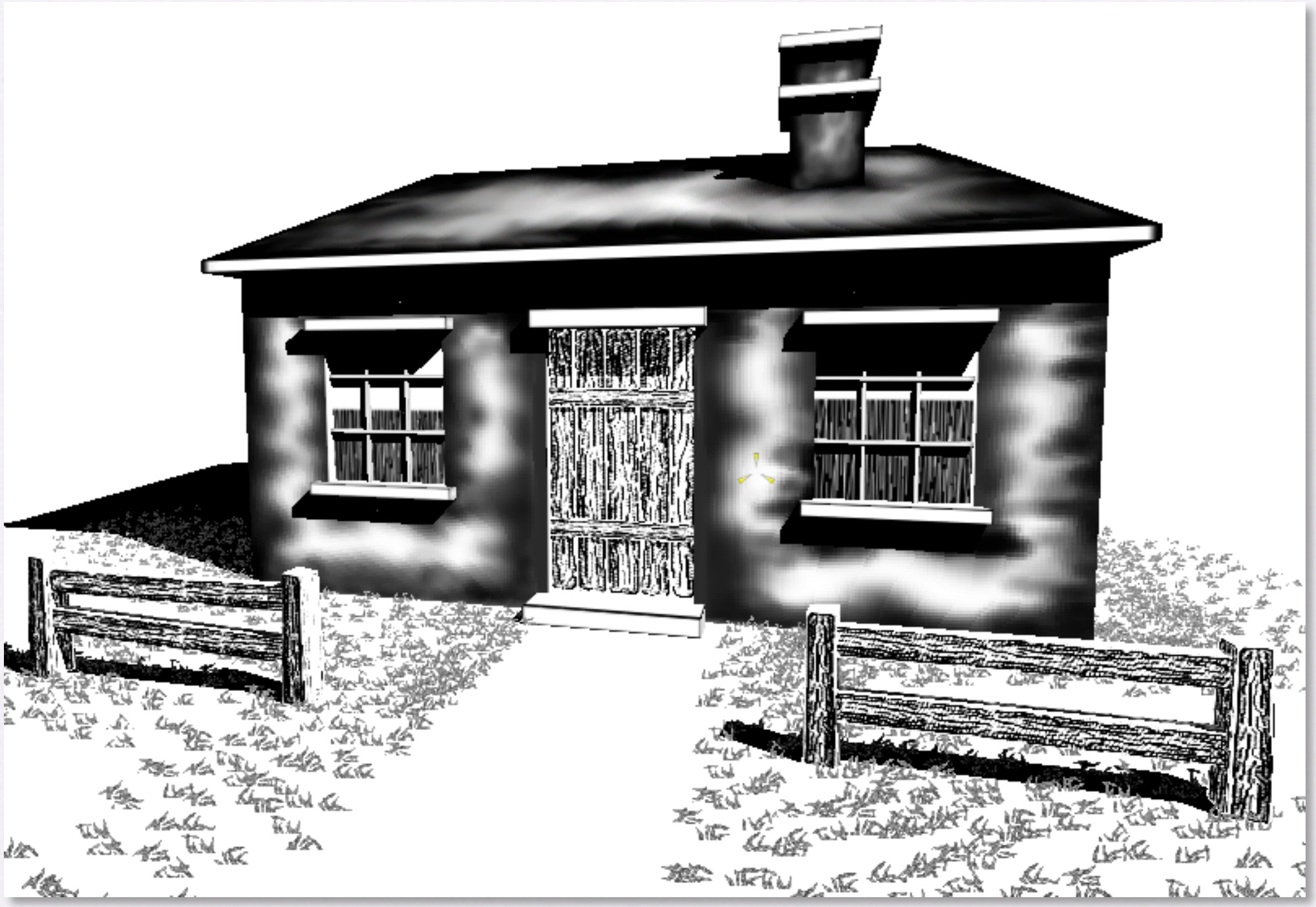
Indication Mapping



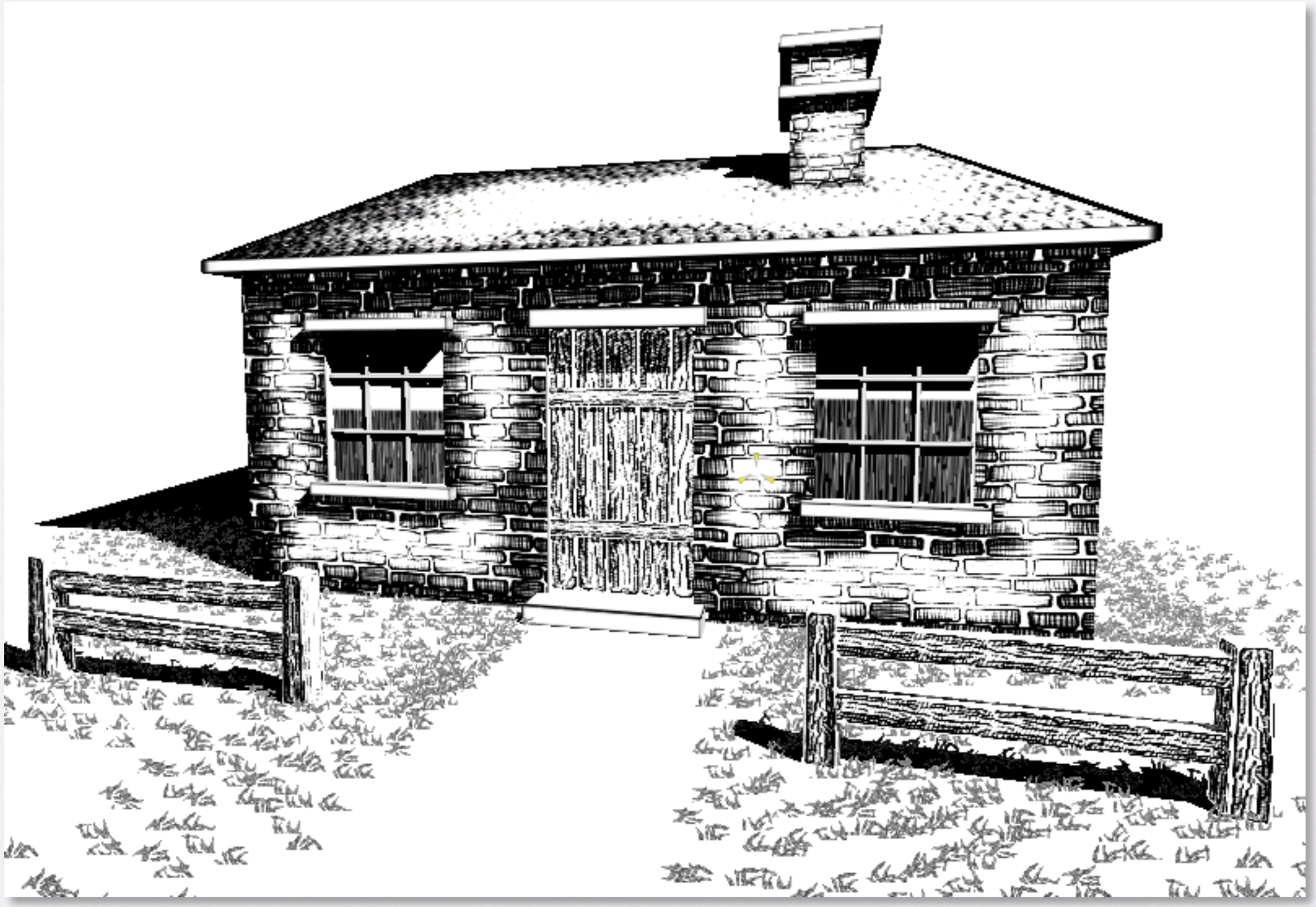
Indication Mapping



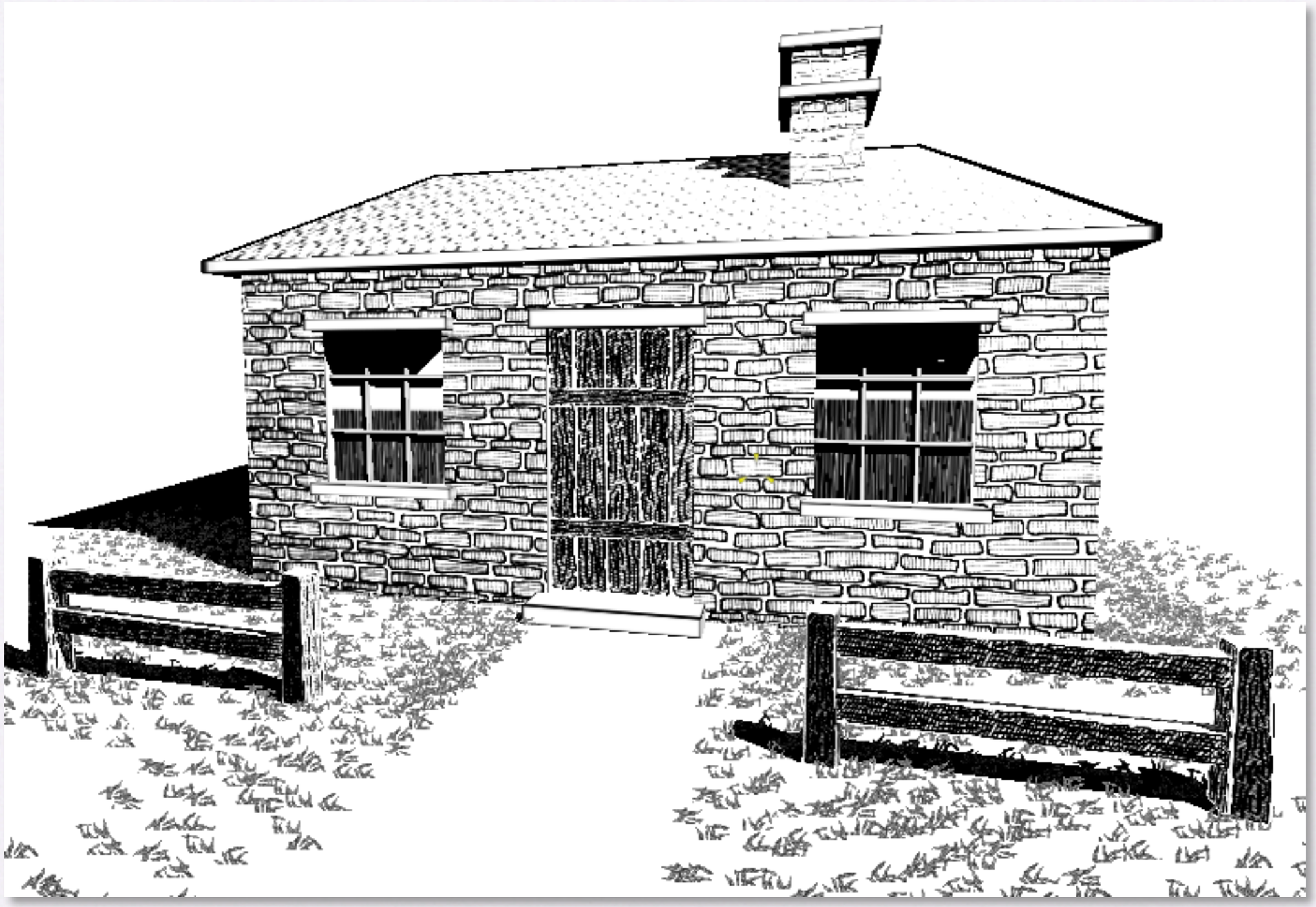
Indication Mapping



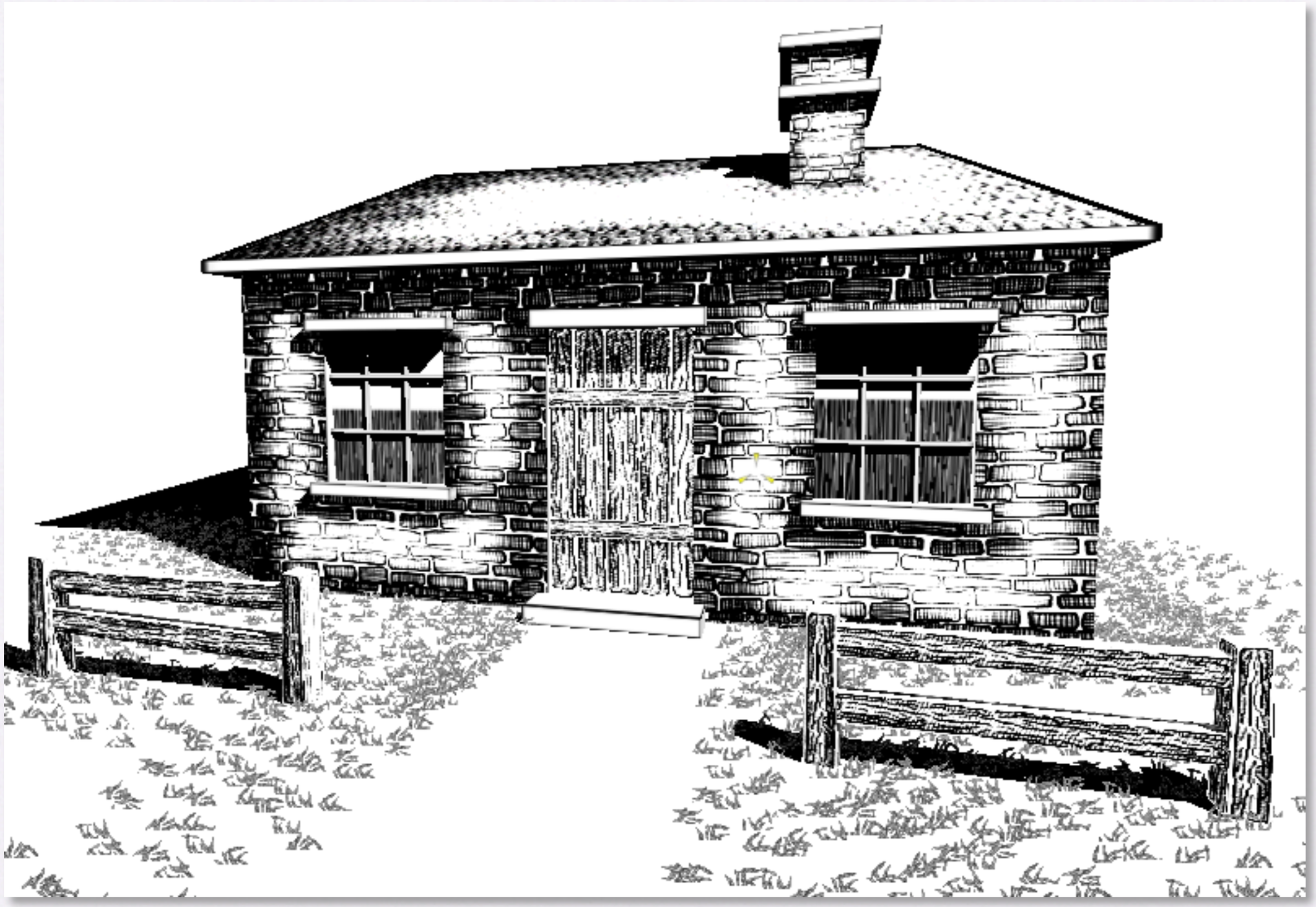
Indication Mapping



Indication Mapping



Indication Mapping



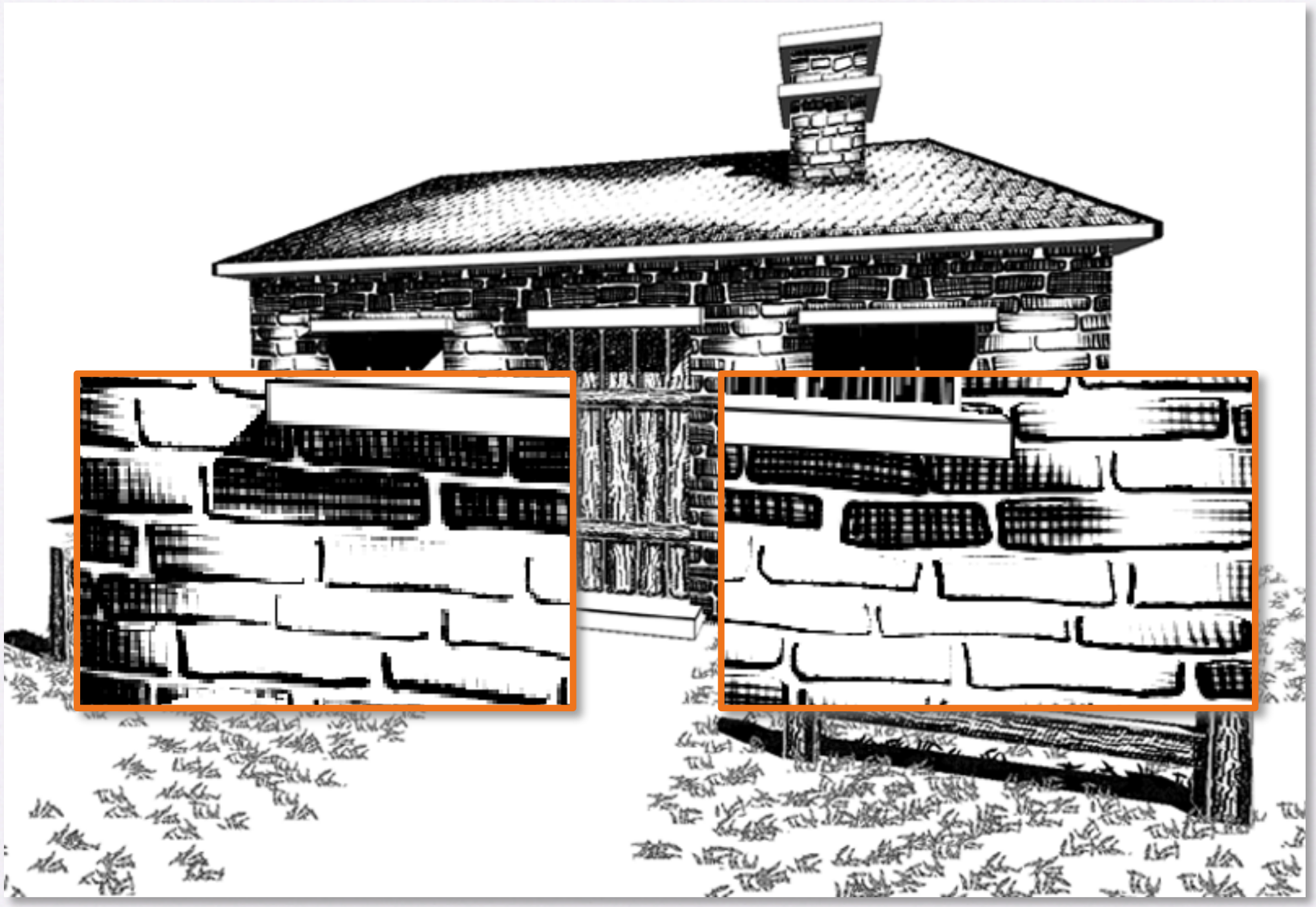
Indication Mapping

- indication map stores signed values
  - fewer lines if > 0
  - more lines if < 0
  - unchanged if 0
- bias intensity by indication map

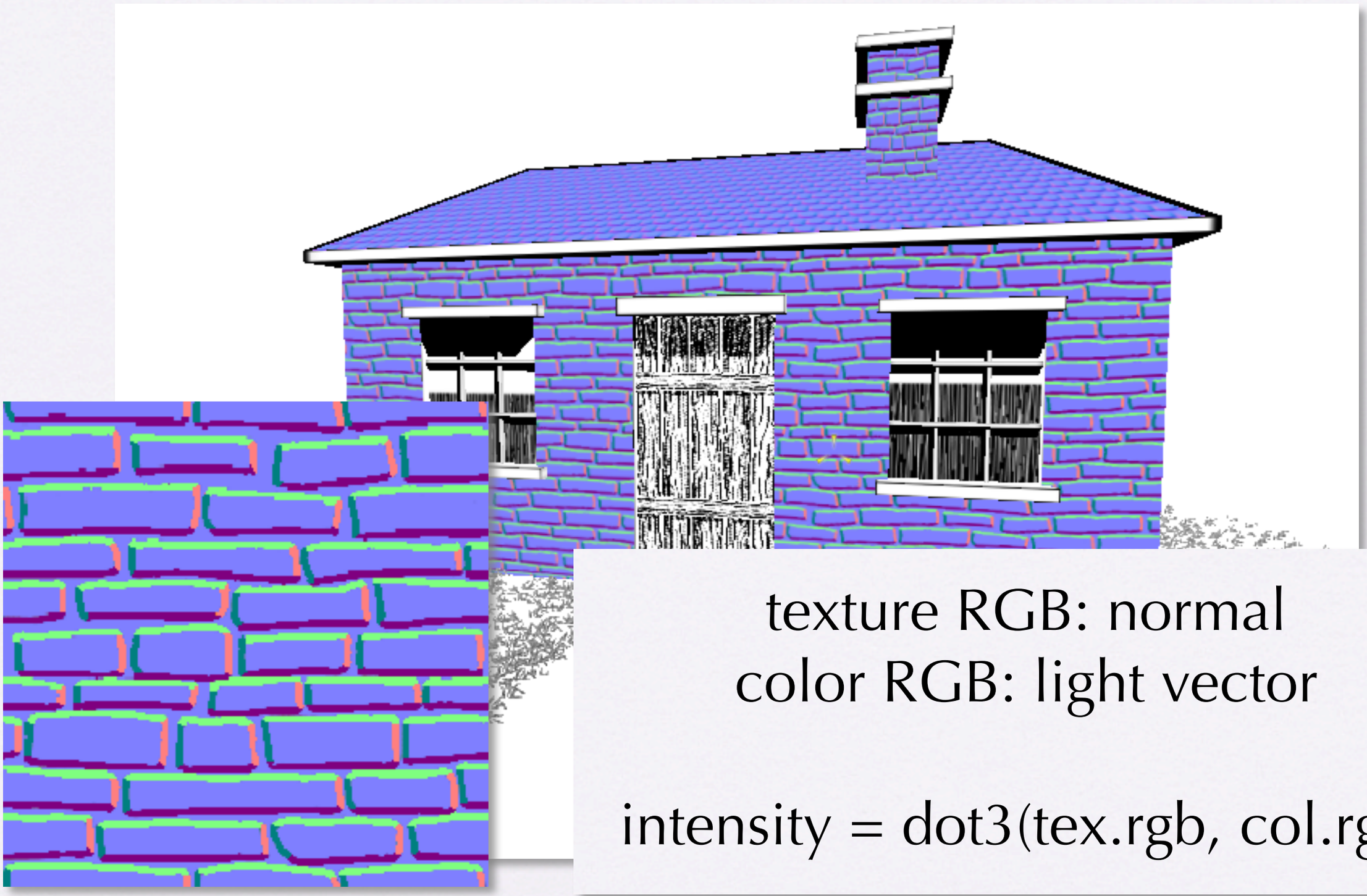
Lighting Individual Strokes

- make strokes respond to light

Individual Stroke Lighting



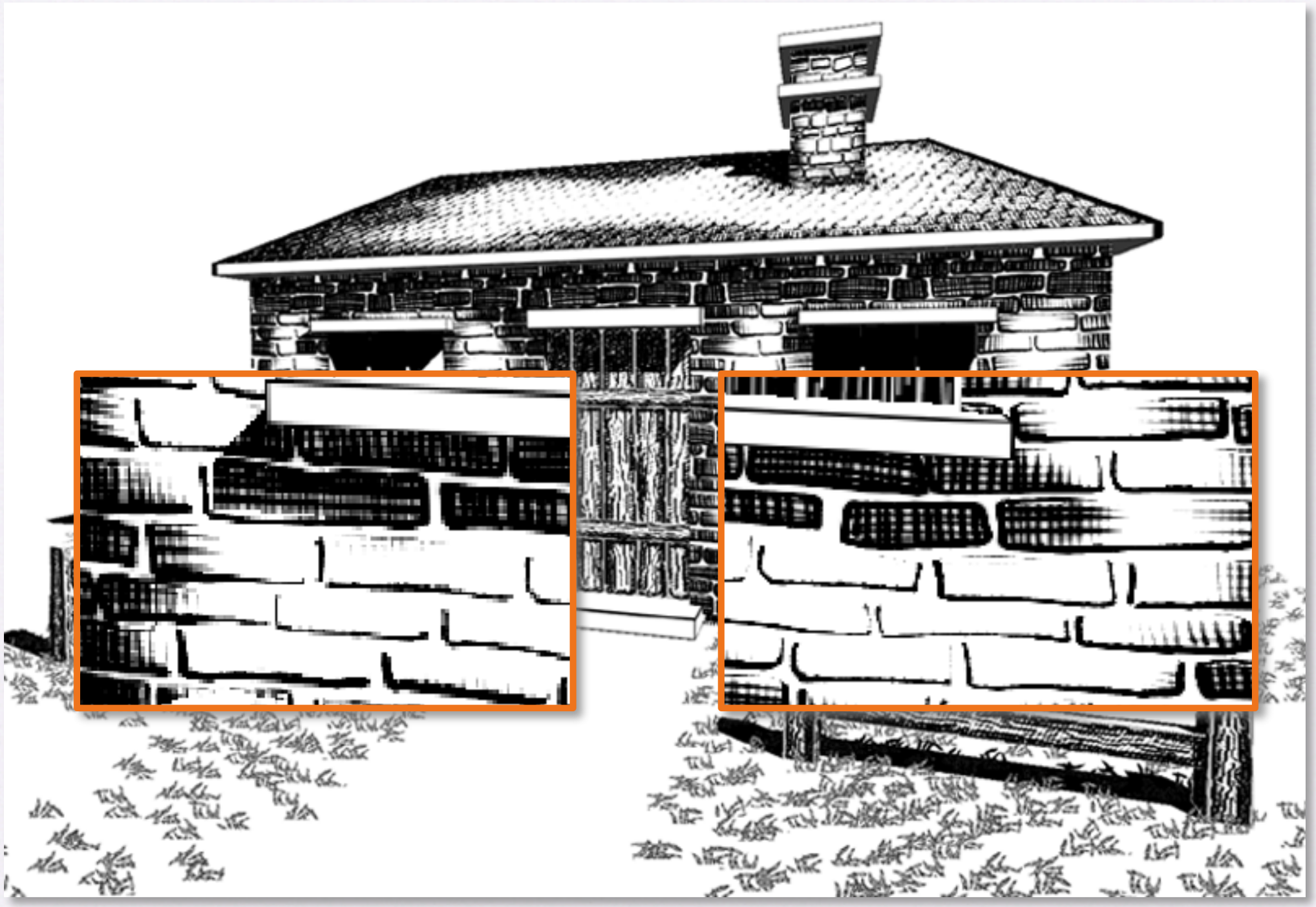
Individual Stroke Lighting



texture RGB: normal
color RGB: light vector

intensity = dot3(tex.rgb, col.rgb)

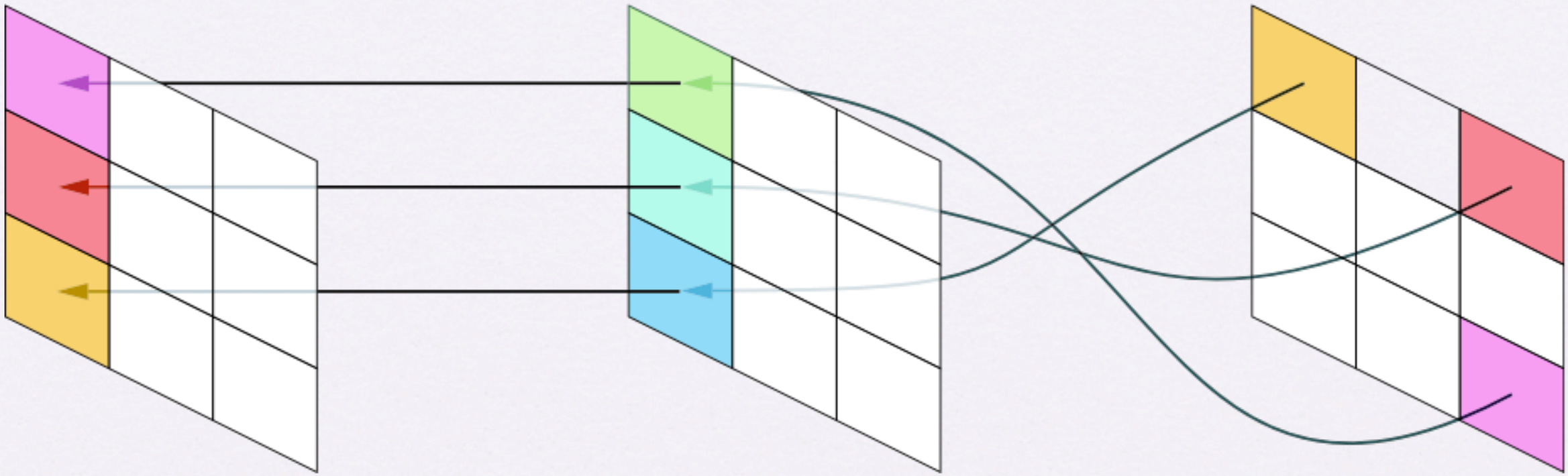
Individual Stroke Lighting



The Warp-Map

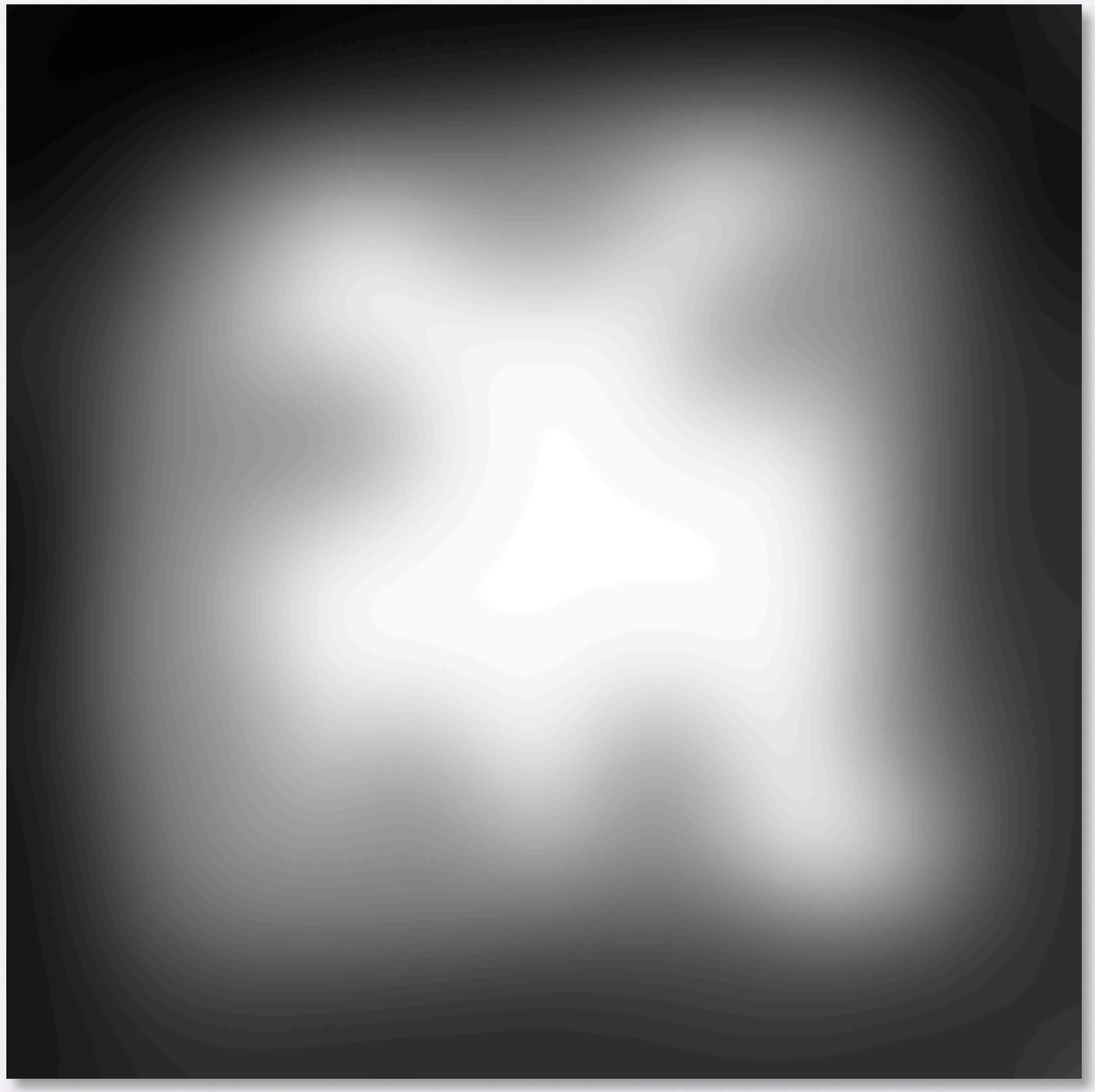
- new technique for avoiding partial strokes

The Warp-Map



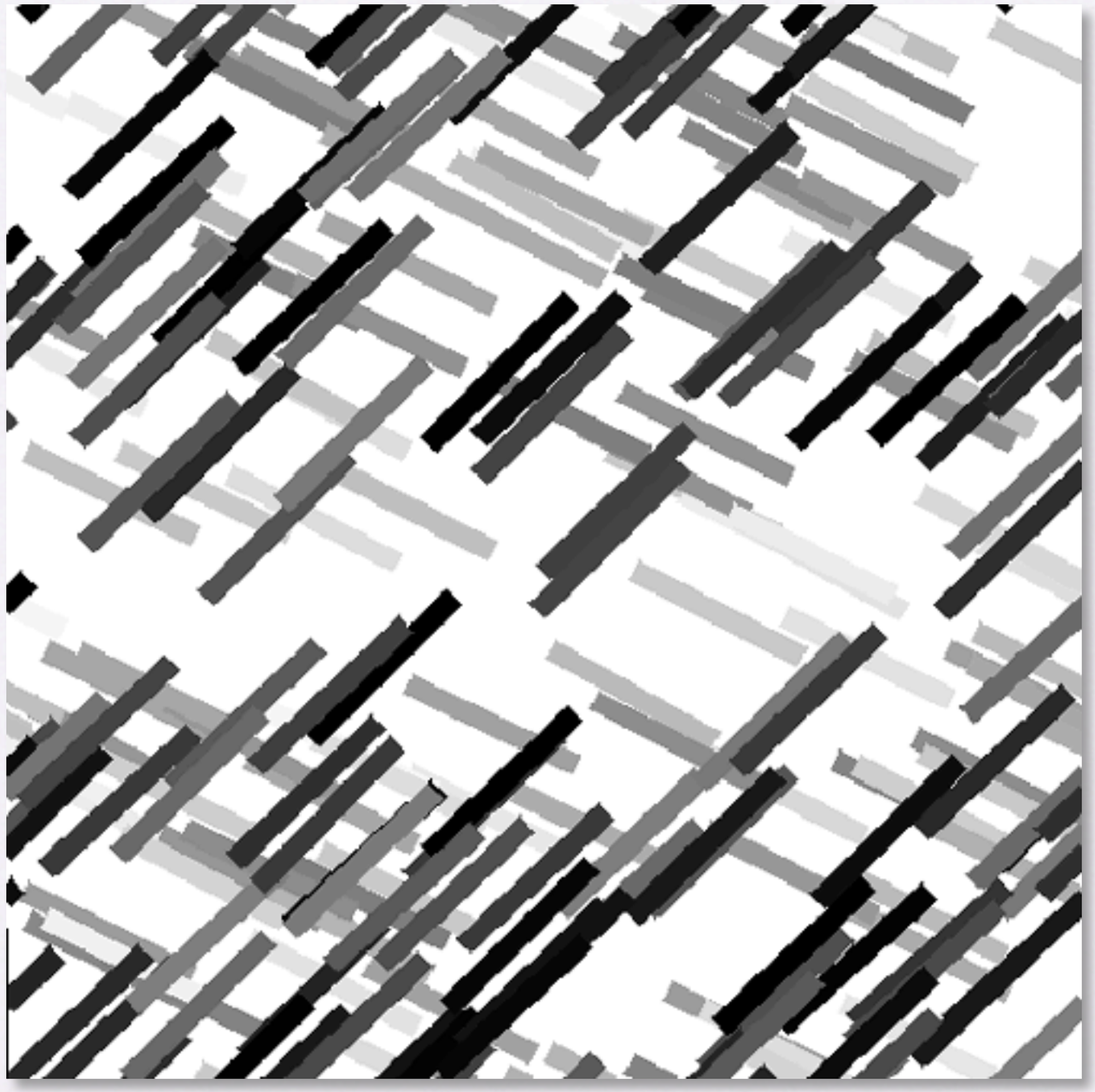
dependent texturing

The Warp-Map



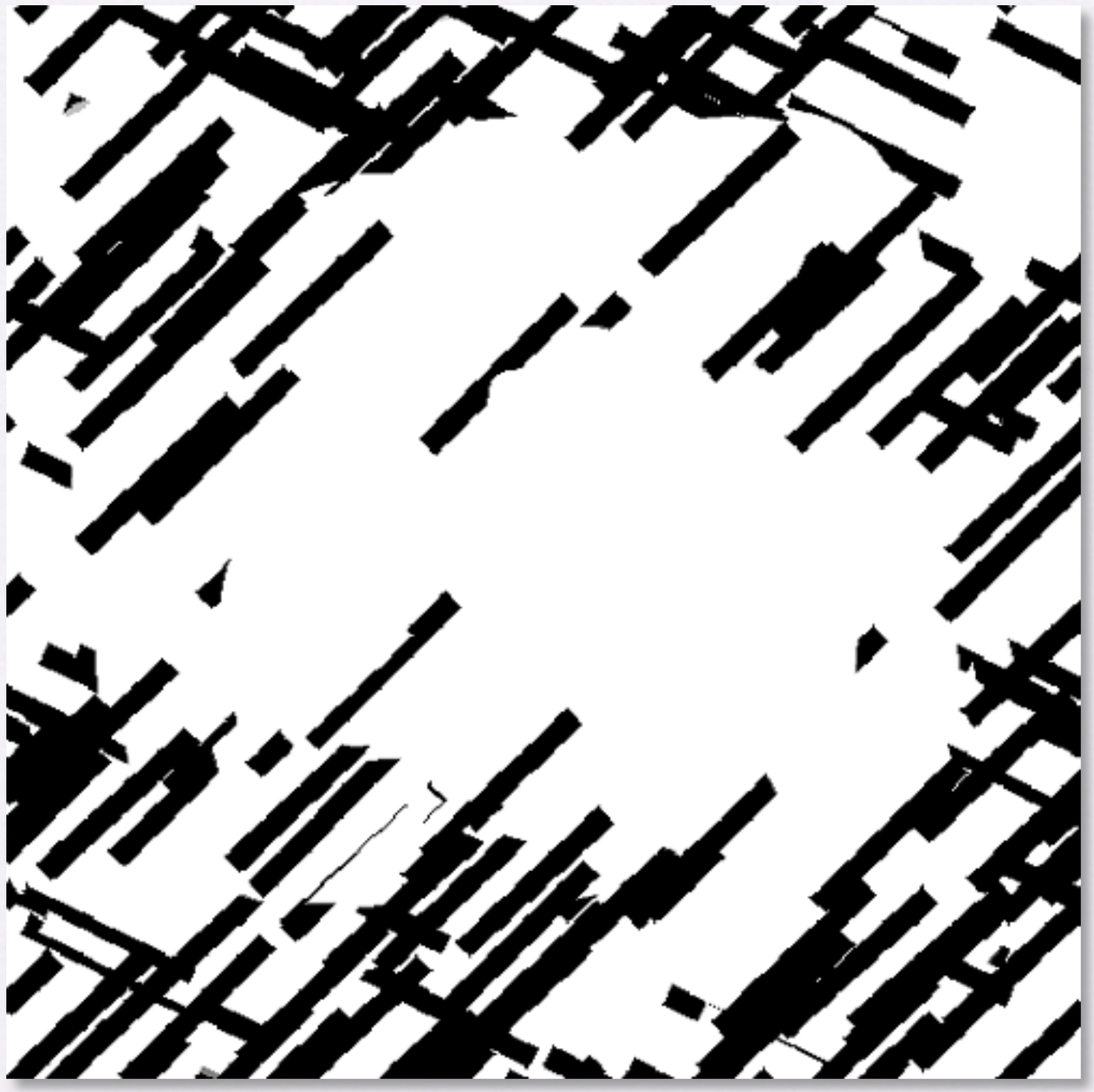
Light-Map

The Warp-Map



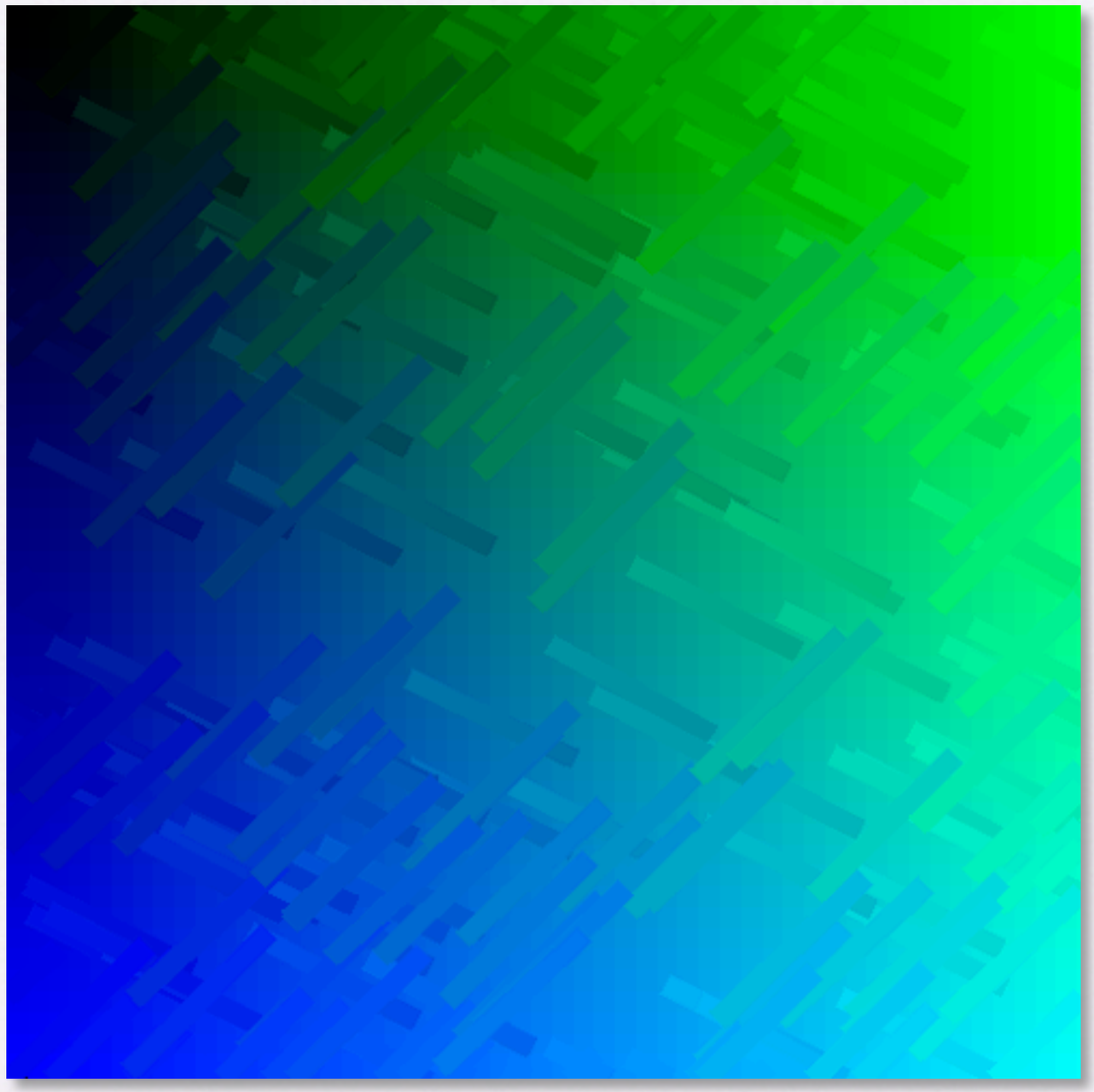
Halftone-Map

The Warp-Map



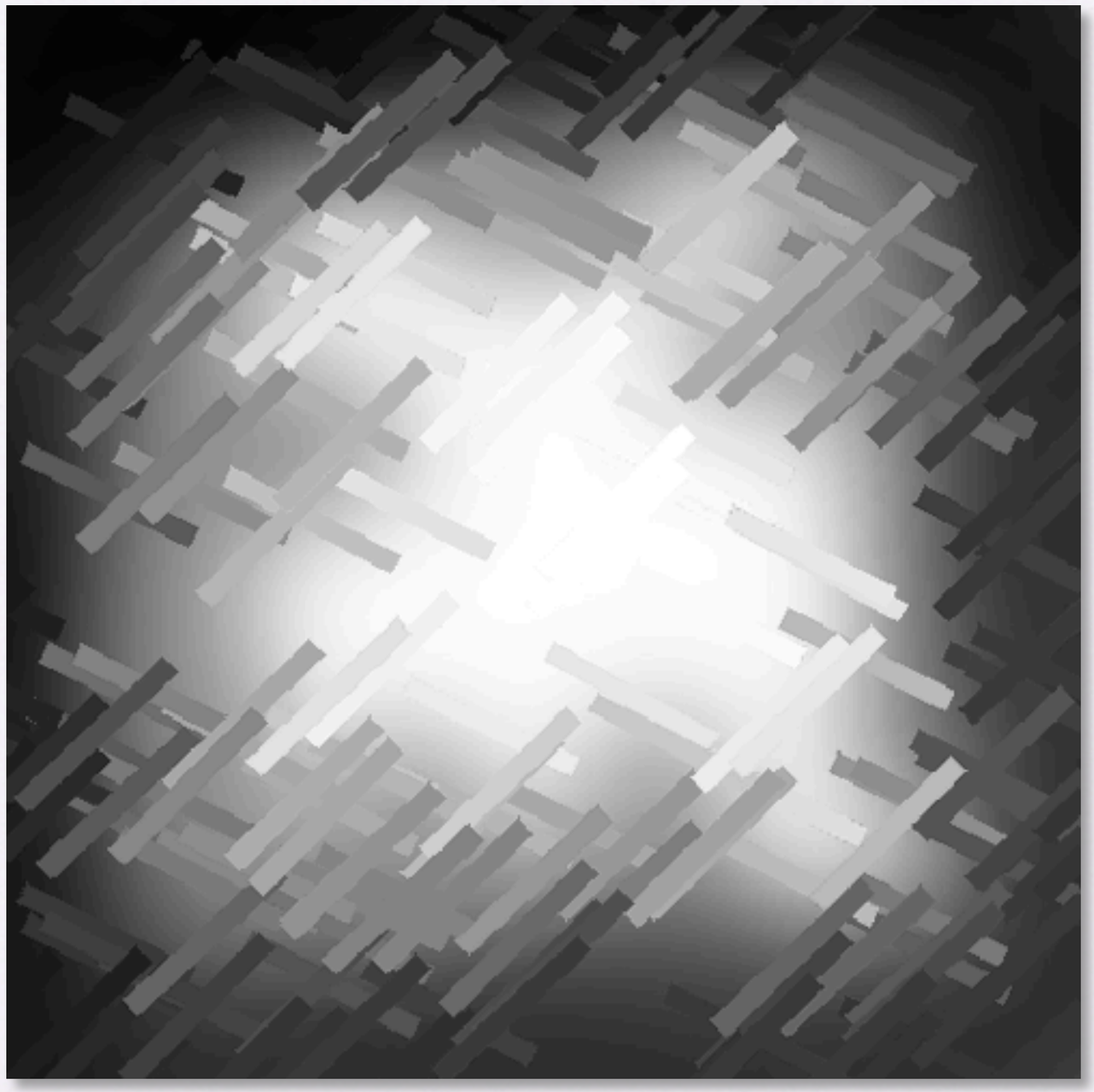
Rendered Image – Partial Strokes

The Warp-Map



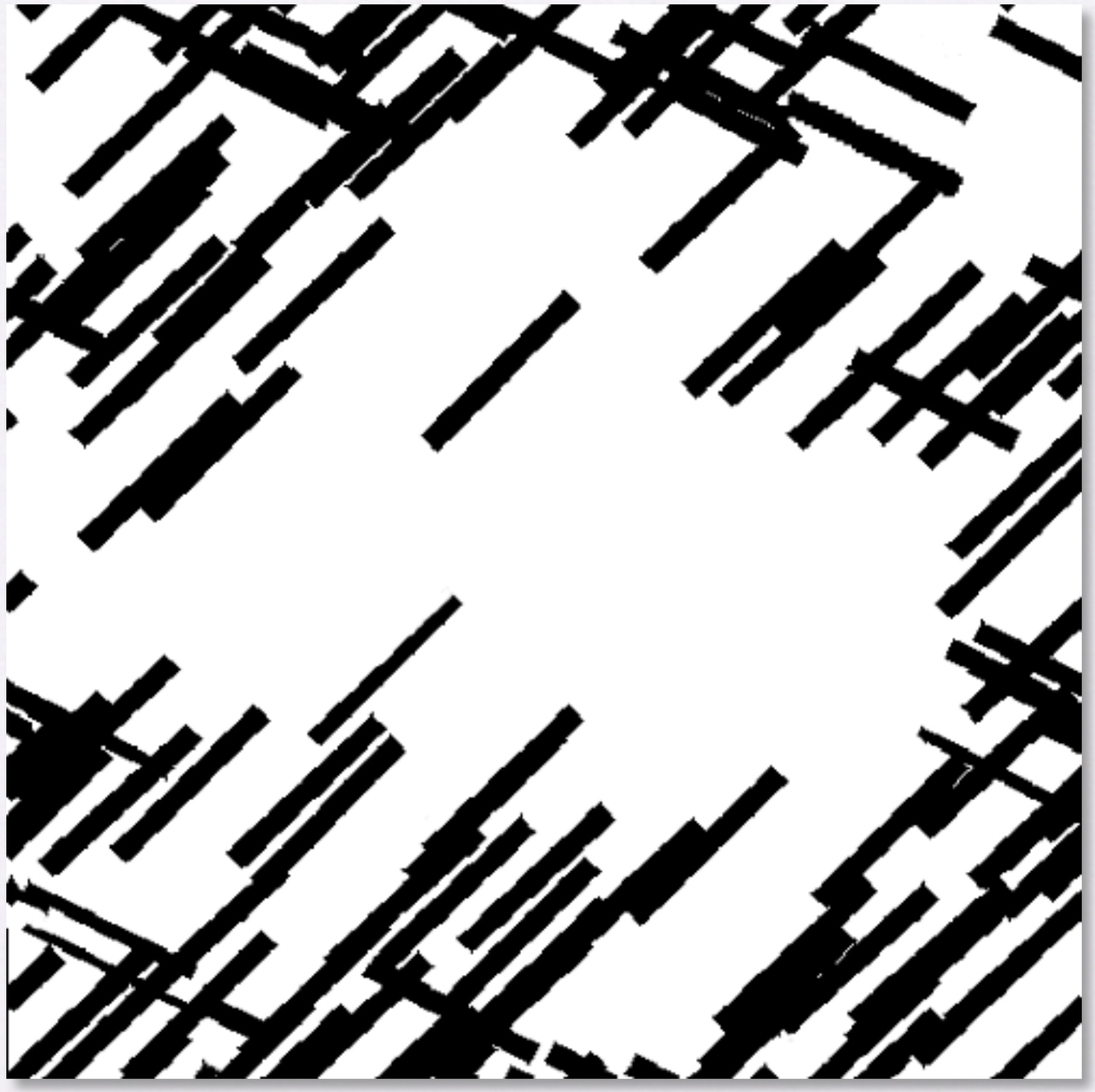
Warp-Map

The Warp-Map



Warped Light-Map

The Warp-Map



Rendered Image – Complete Strokes

Implicit Outlines

- render G-Buffers
- run edge-detection filter

Sobel Filter

-1	0	1
-2	0	2
-1	0	1

6 non-zero samples needed

Sobel Filter

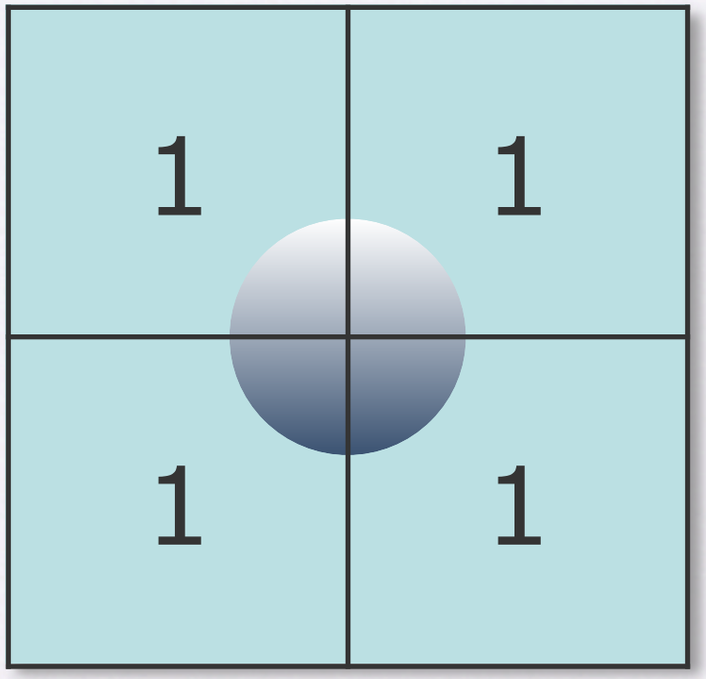
-1	0	1
-2	0	2
-1	0	1

only 4 samples needed

Denorm Filter

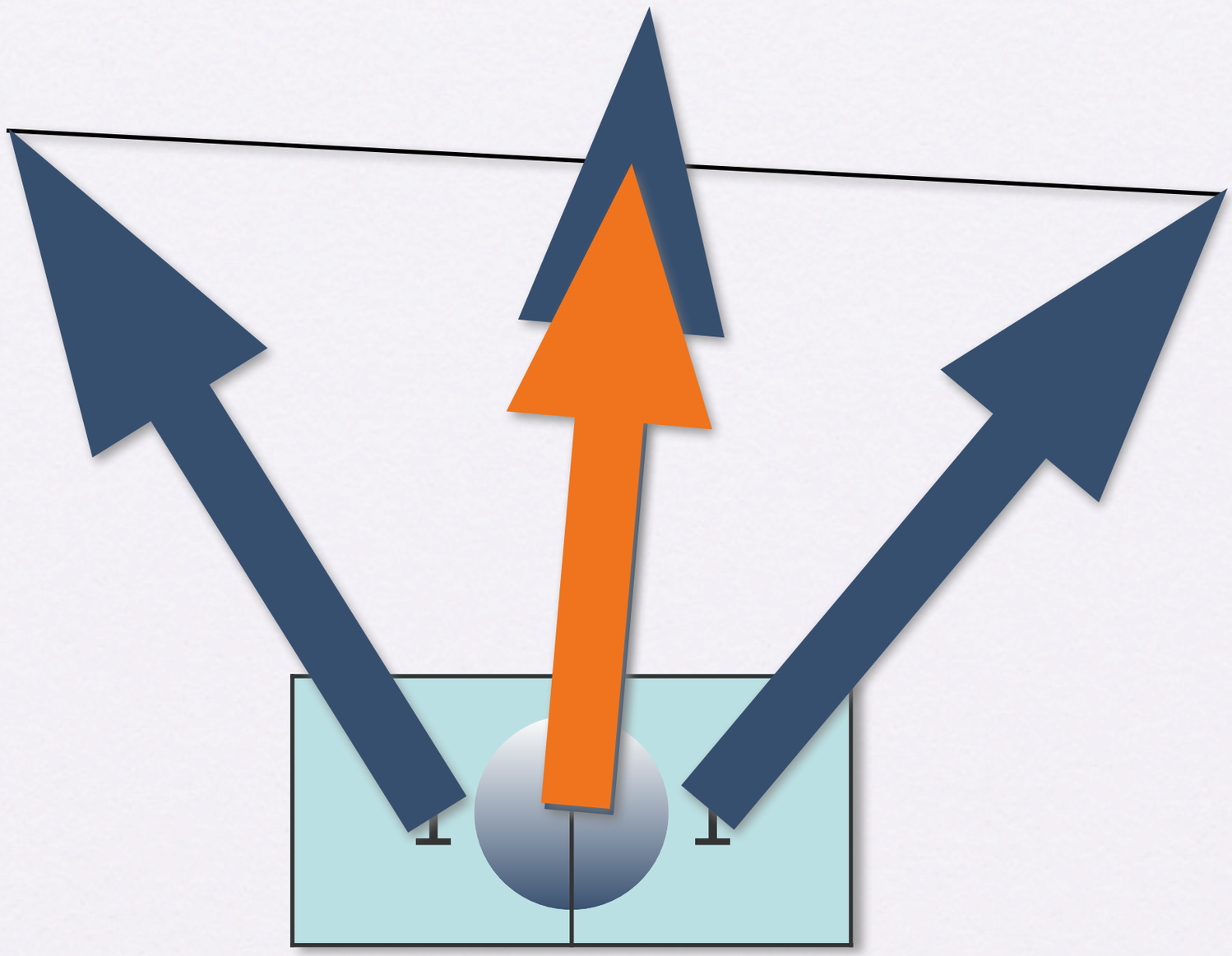
- for Normal-buffers
- averages neighboring normals
- detects de-normalization

Denorm Filter



1 sample = average of 4 normals

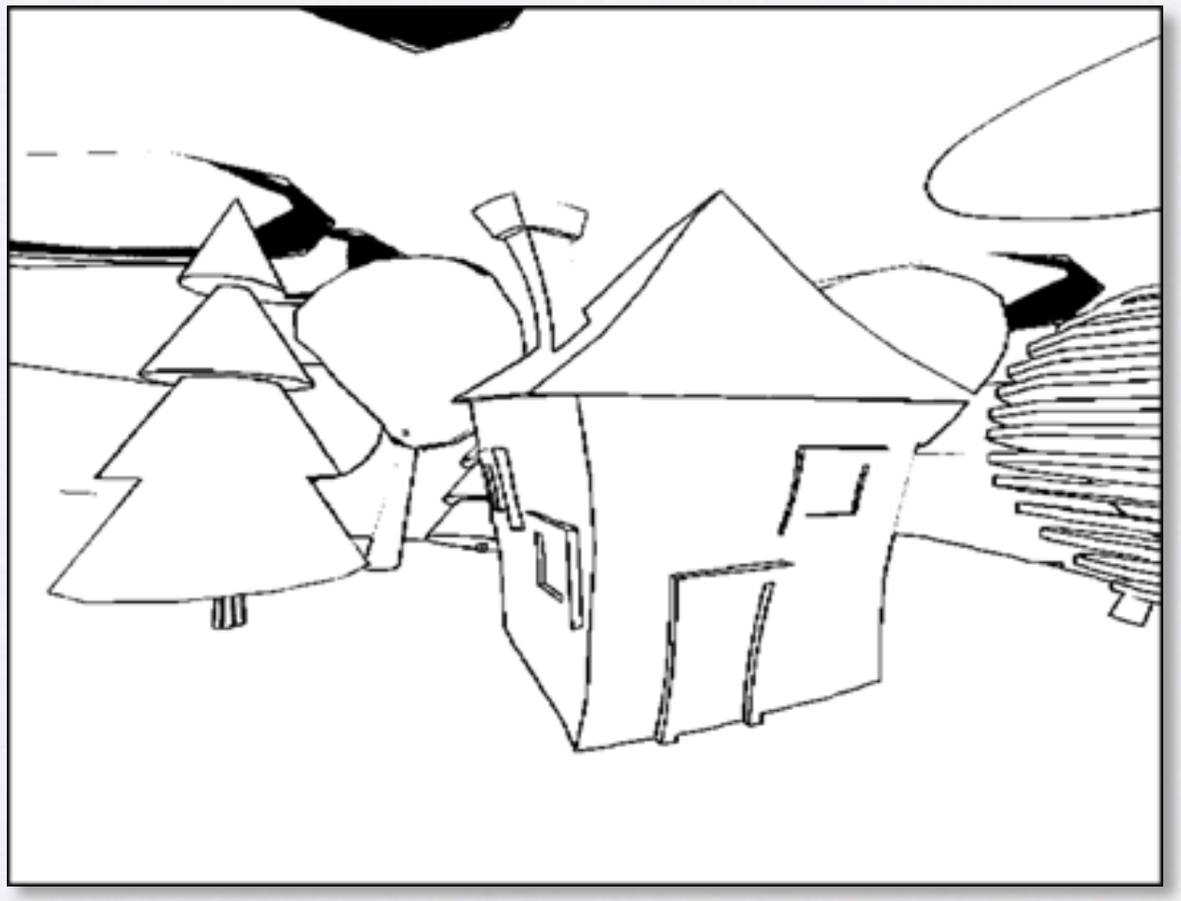
Denorm Filter



linear interpolation causes de-normalization

Comparison

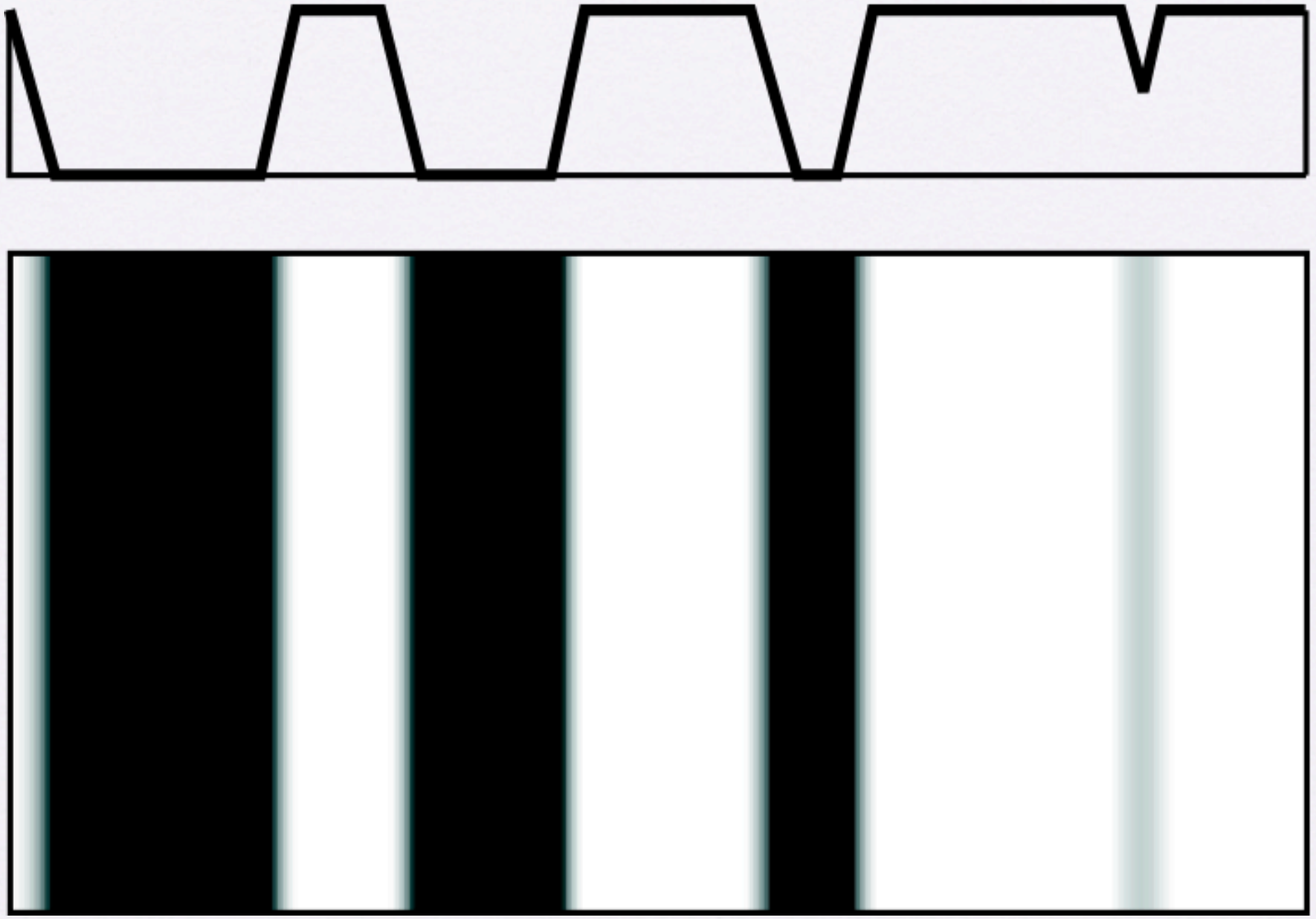
Sobel



Denorm

Comparison

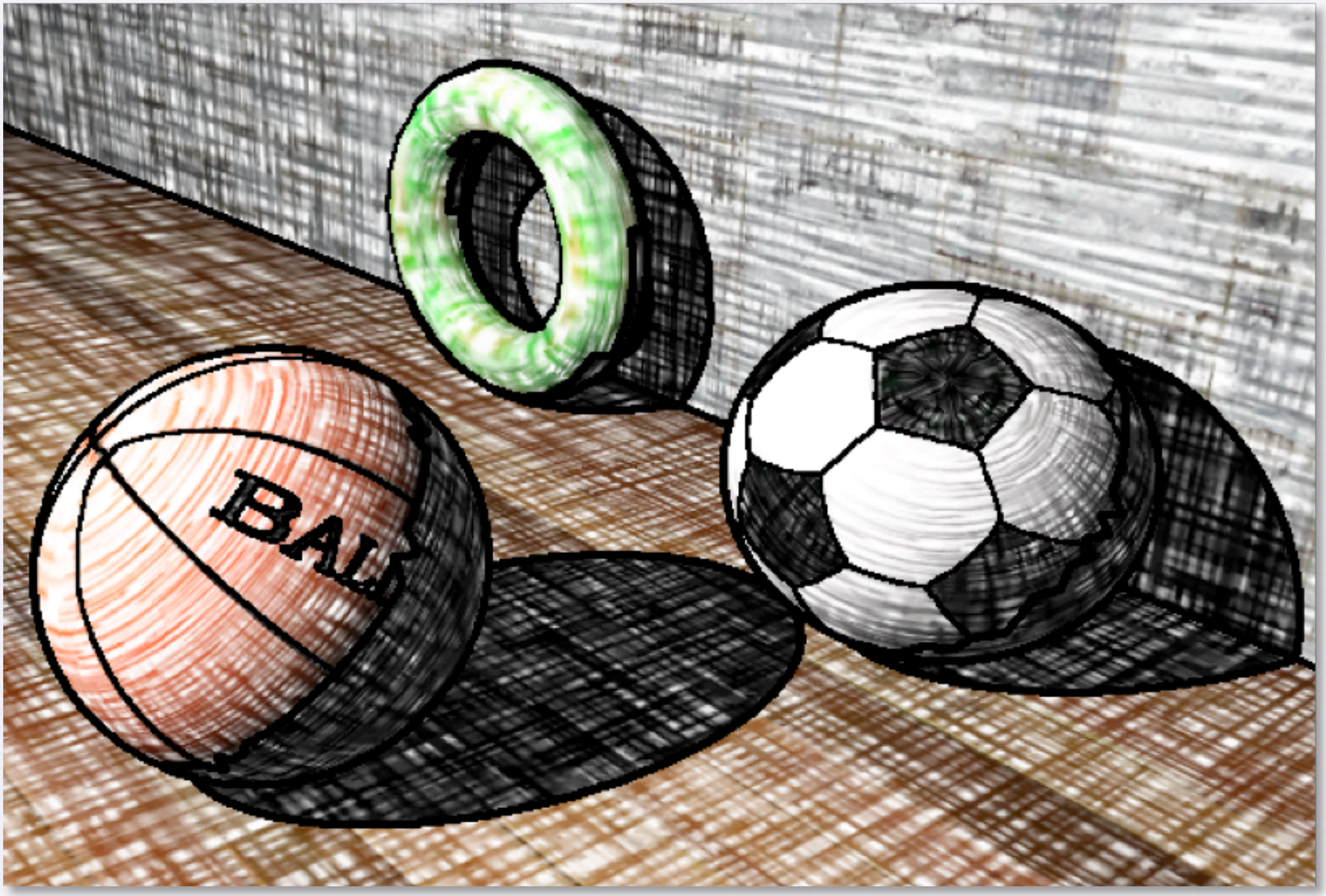
	Sobel	**Denorm**
Quality	high	low
Samples	4	1



Acknowledgements

- Bernd Eckardt: Geometriebasiertes Echtzeit-Halbton-Rendering.
- Christian Mantei: Texturbasierte Echtzeit-Cartoons
- Bert Vehmeier: Qualität in Liniendarstellungen durch lokale Informationen.
- Ragnar Bade, Bernd Eckardt, Niklas Röber, Ingo Thieme

Related Work

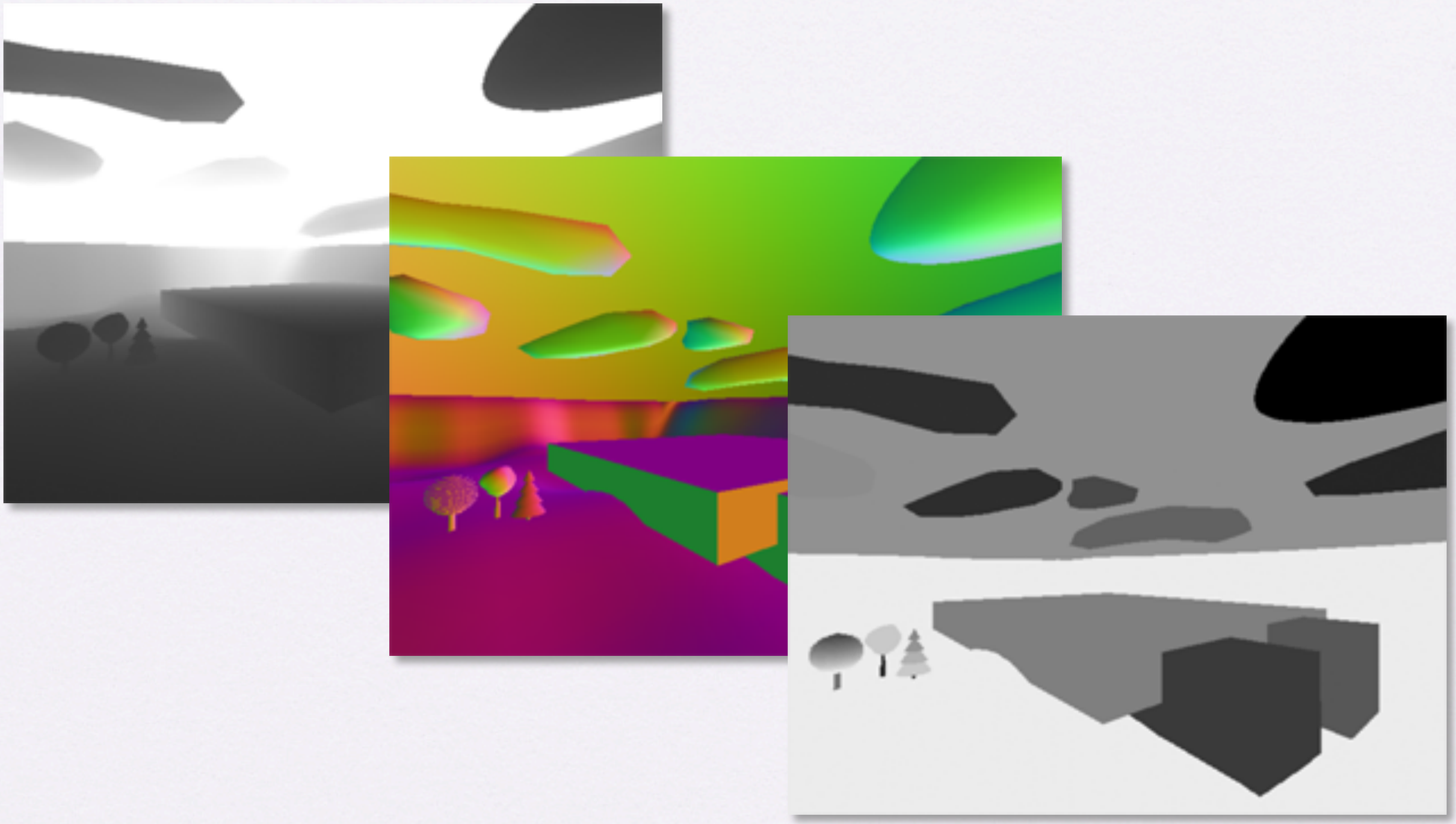


Mitchell et al. (2002)

Stroke Animation

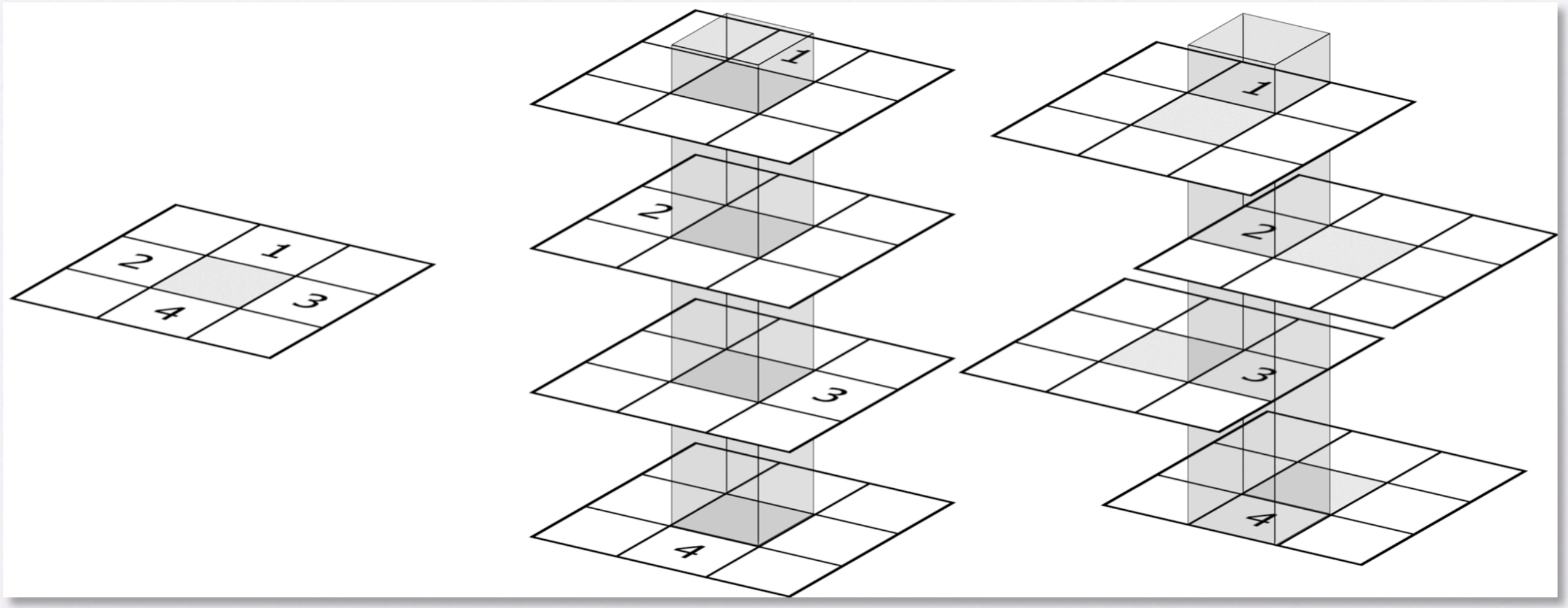
noise function in fragment shader

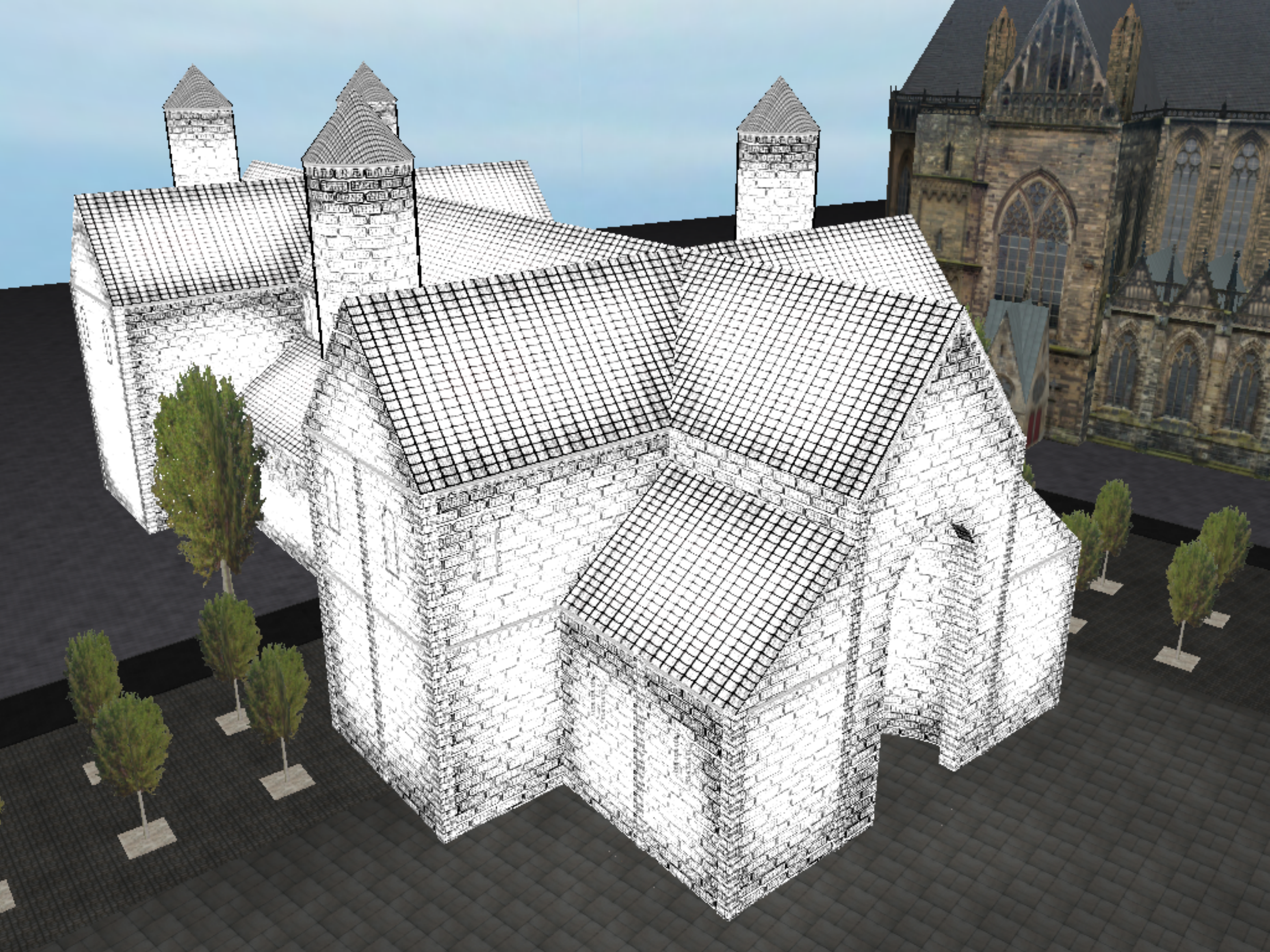
G-Buffers



depth – normal – material

Neighbor Sampling

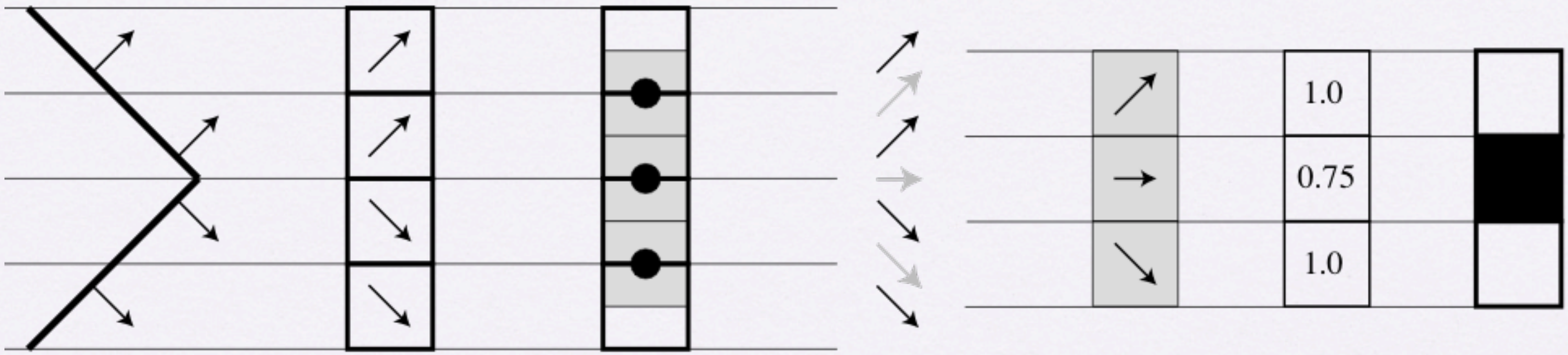


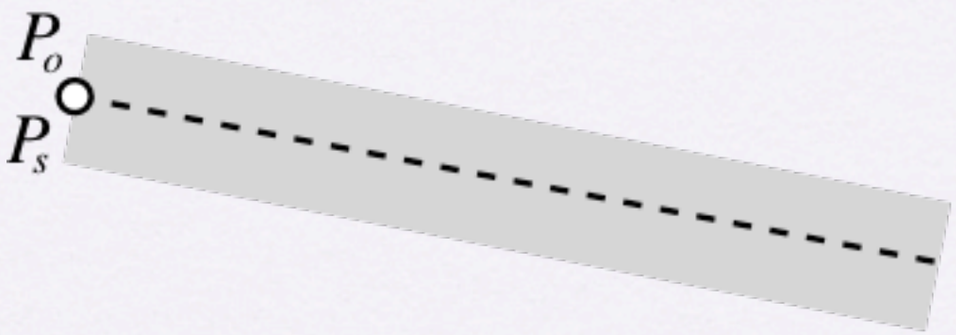


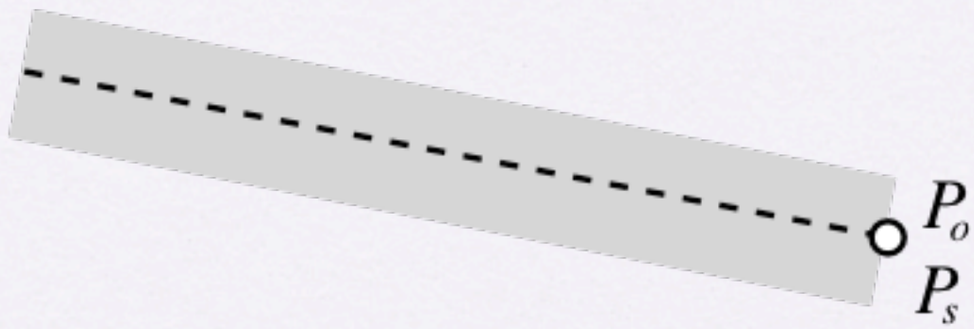
Game

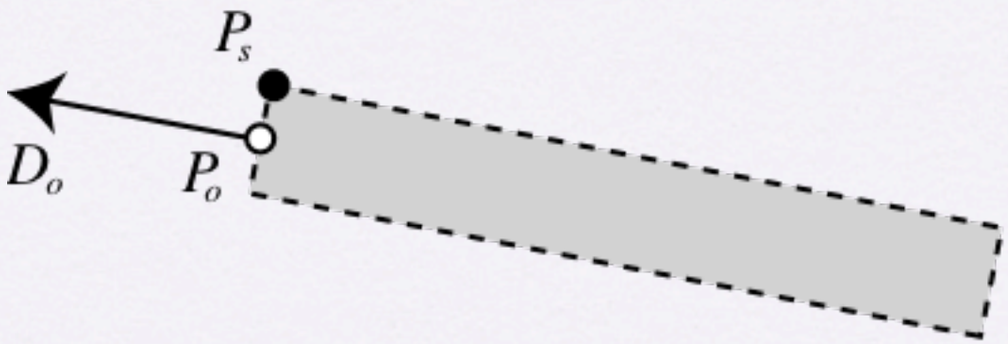
- photorealism adapted to comic style

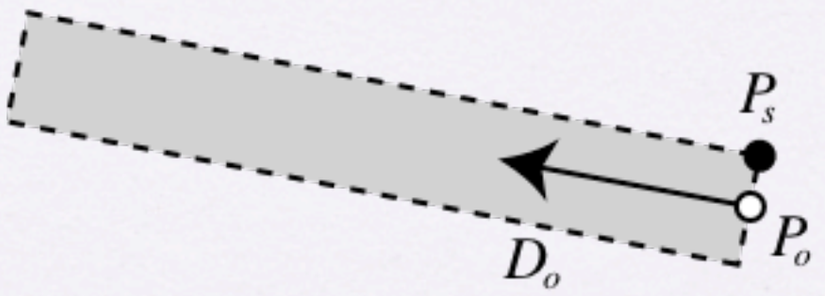
Denorm Filter

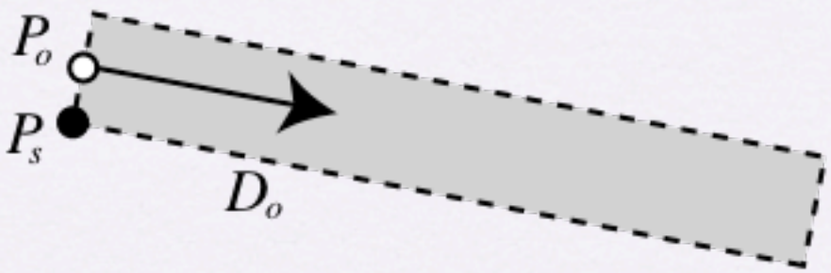


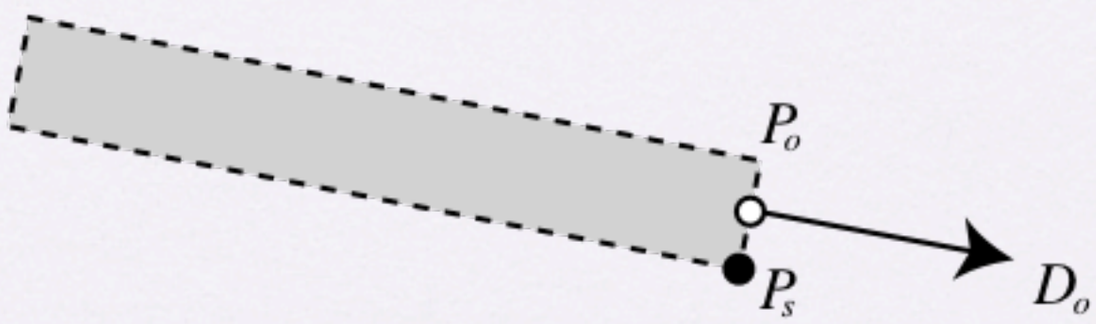

P_o
P_s


P_o
P_s


P_s
D_o
P_o

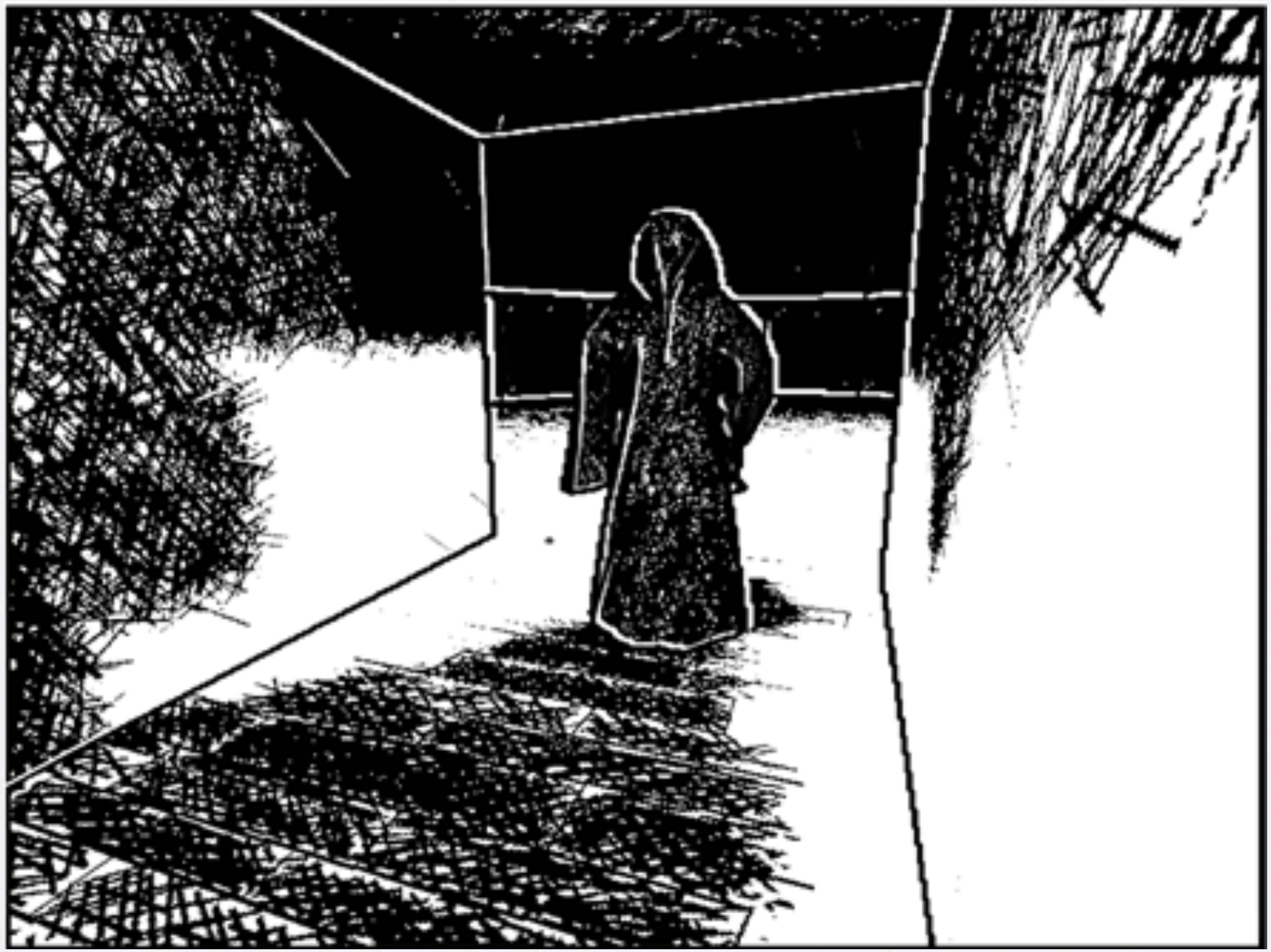

P_s
P_o
D_o


P_o
P_s
D_o


P_o
P_s
D_o

Colored Strokes

Contrasting Edges



Things

- hardware 1999–2004, games in 2004
- Halftoning = thresholds for both, bitmap and vector textures
- Two-valued thresholds (p. 56)
- Explicit vs. implicit
- width vs. density
- Suitability for games
- Artist-controlled
- Tone-reproduction curve (p.83)
- Why inverted halftone screen (p. 88)
- Hard vs. smooth
- indication map p 90
- stroke lighting p 91
- Warp Map
- procedural hatching, noise, animated strokes
- how to discard primitives
- denorm filter