Nefotorealistično upodabljanje

Whither Graphics?

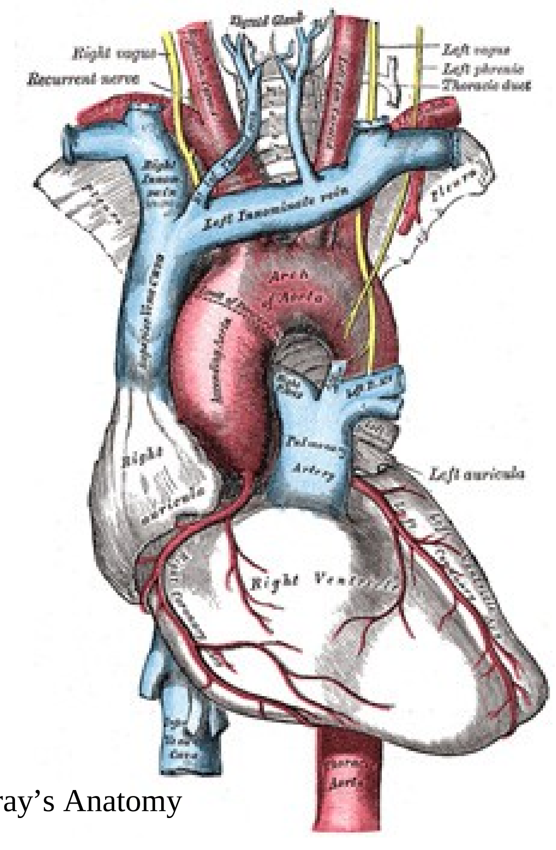
What is our ultimate goal in computer graphics?

Photorealism

- Makes synthesized pictures appear like photographs of real objects
- Includes distracting artifacts of the photographic process (e.g. depth of field, lens flare)
- Breeds dishonesty

Communication

- Graphics is a high-bandwidth medium for transmitting information into the brain



Gray's Anatomy

Goals of Computer Graphics

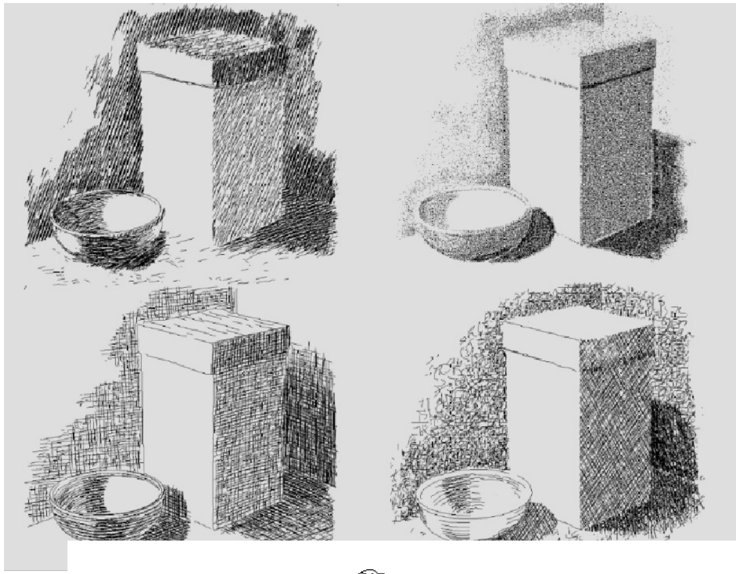
- Traditional: Photorealism
- Sometimes, we want more
  - Cartoons
  - Artistic expression in paint, pen-and-ink
  - Technical illustrations
  - Scientific visualization

- Each rendering has a different “feel”.
- Bottom 2 images would most likely be presented to customer as concept art.
- Top 2 images would most likely be presented to a customer as the final design.

FIGURE 6.9 Different stylistic variations achieved by using different drawing tools: photorealistic rendition (a), added environment (b), pen-and-ink style (c), and painted style (d).

Image source:
[Strothotte T., Schlechtweg, S. 2002]

Non-Photorealistic Rendering



- Departs from the limits of photorealism to better communicate visual information
- Uses concepts from art instead of physics
- Two fundamental visual cues
  - Silhouette – the visible edges of a surface
  - Hatching – the use of texture to indicate the local orientation (shading) of a surface

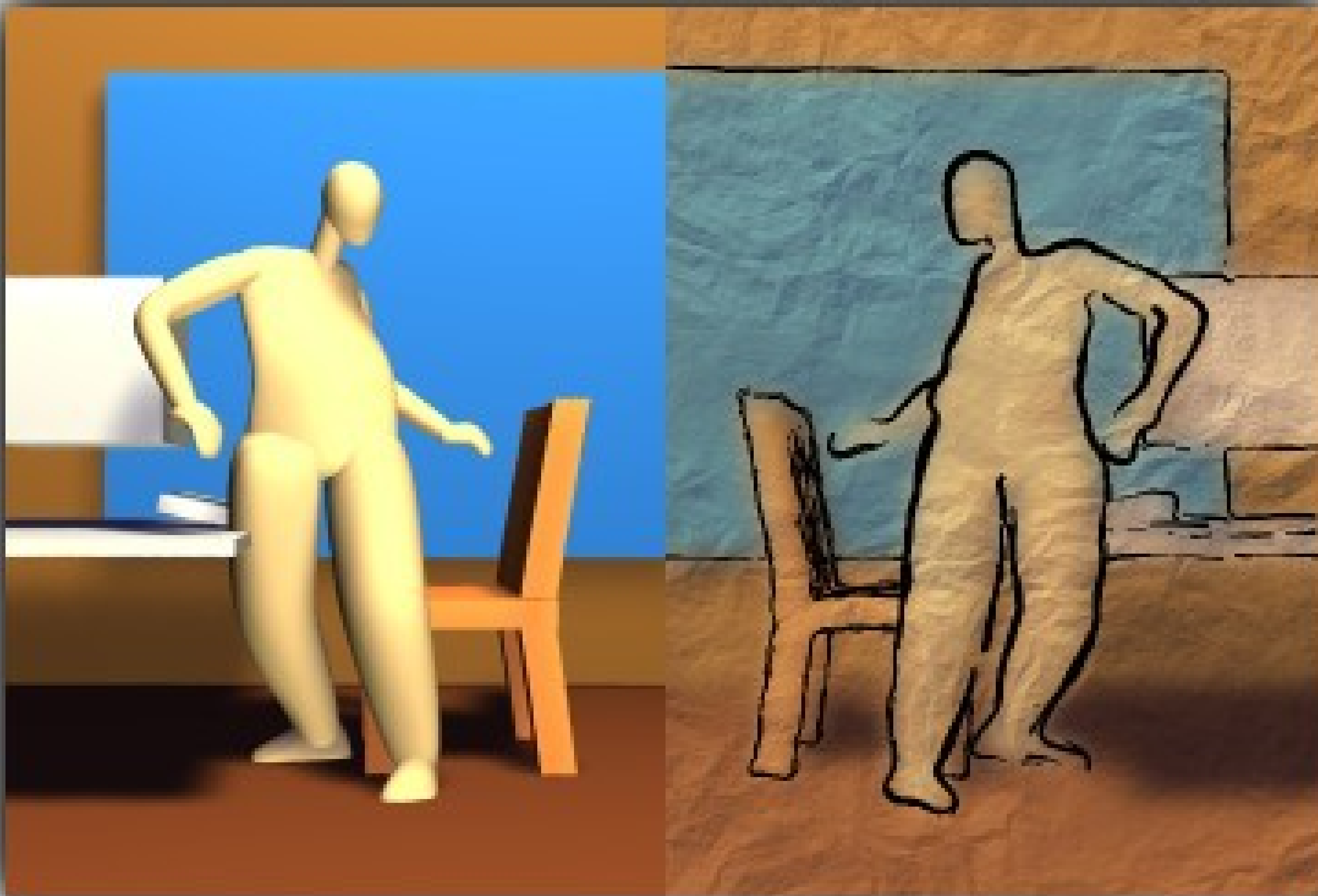
Non-Photorealistic Rendering

"A means of creating imagery that does not aspire to realism" - Stuart Green

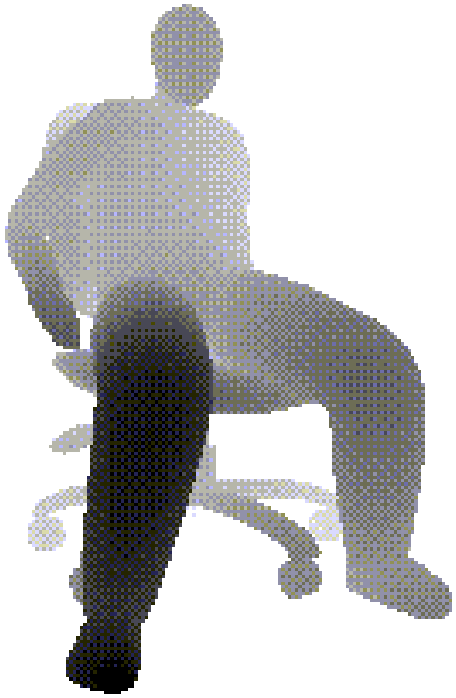
Cassidy Curtis 1998

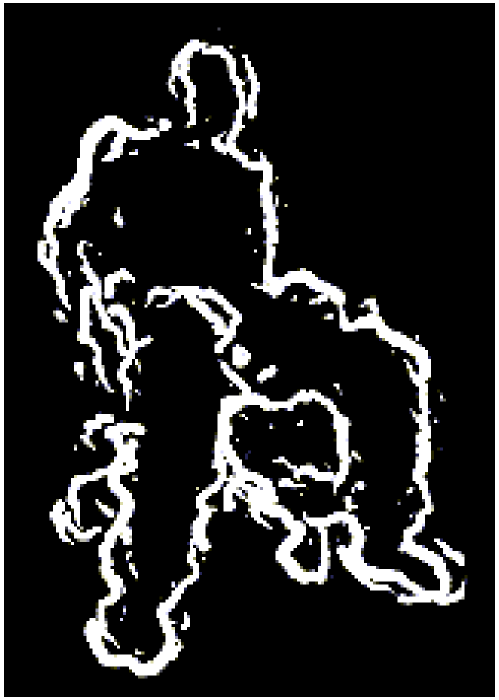
David Gainey

Dvojčka, Predelava silhuete

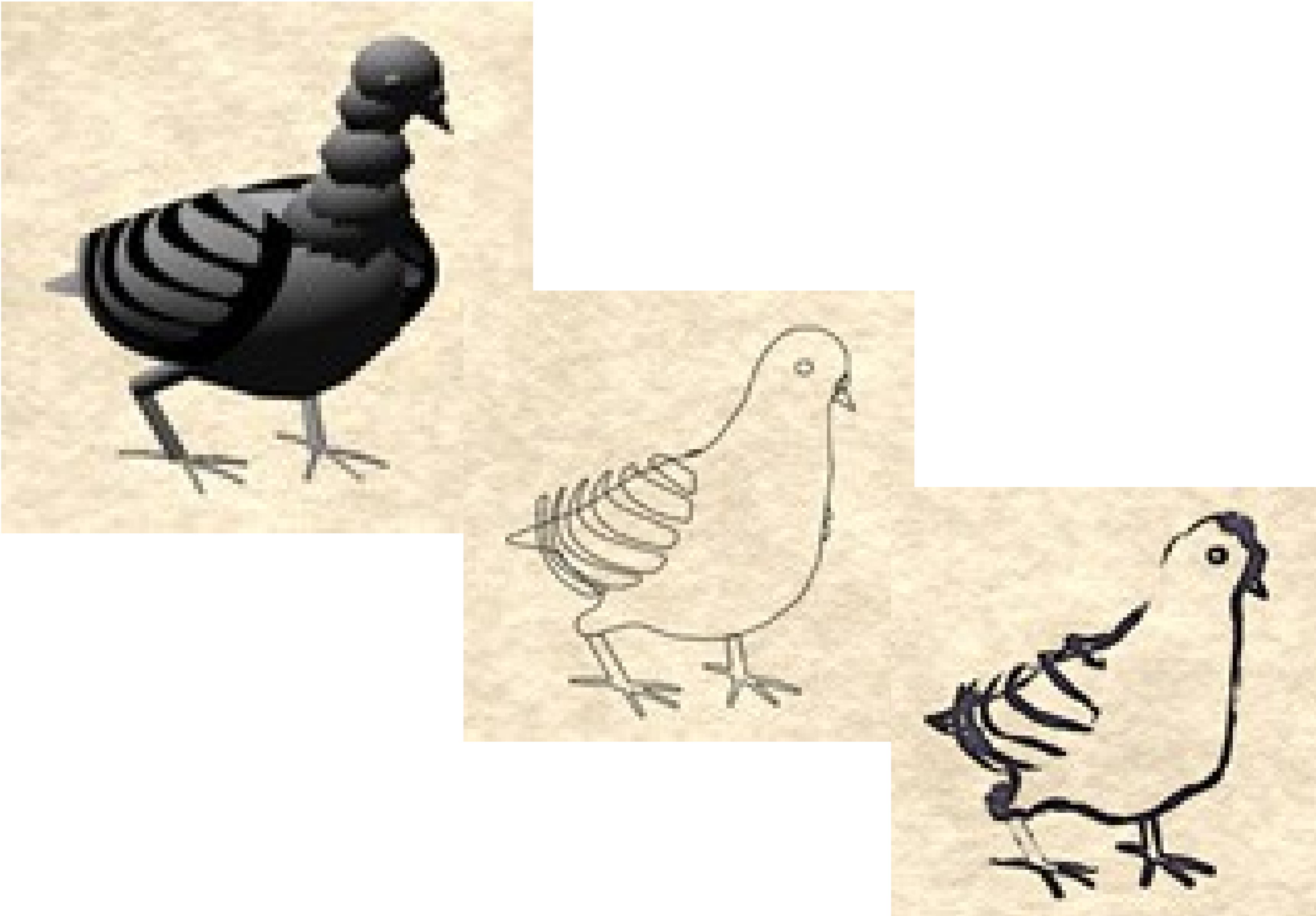


Princip obdelave





Predelava upodobljene kokoši



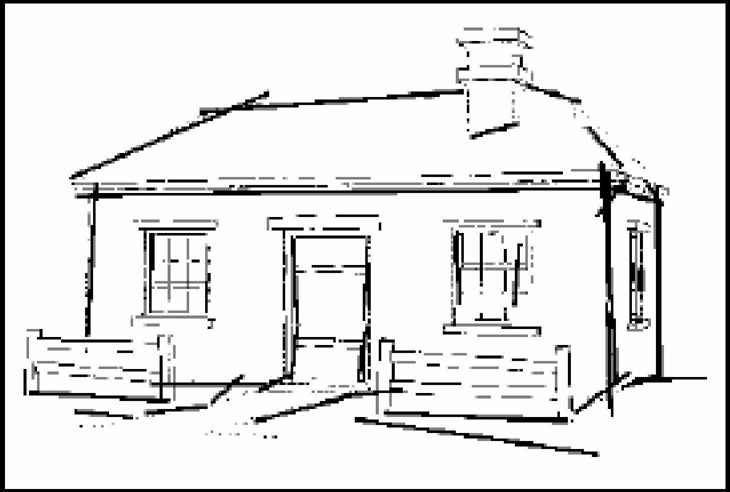
Some NPR Categories

- Pen-and-Ink illustration
  - Techniques: cross-hatching, outlines, line art,etc.
- Painterly rendering
  - Styles: impressionist, expressionist, pointilist, etc.
- Cartoons
  - Effects: cartoon shading, distortion, etc.
- Technical illustrations
  - Characteristics: Matte shading, edge lines, etc.
- Scientific visualization
  - Methods: splatting, line drawing etc.

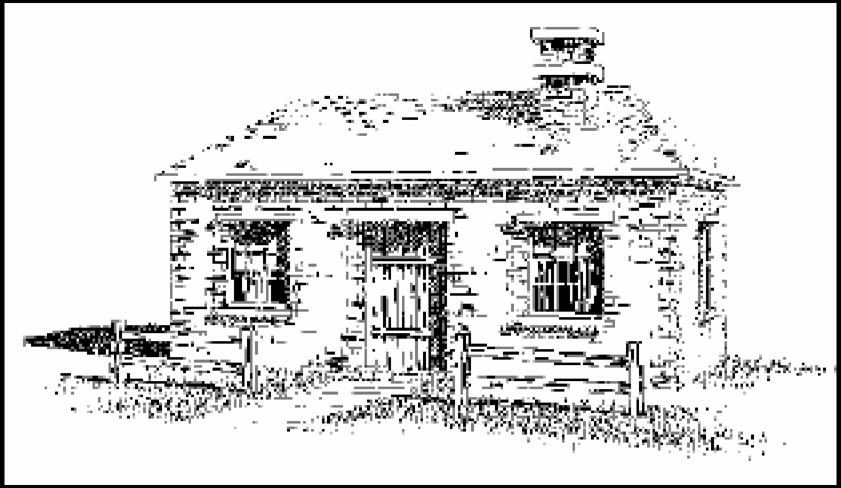
Pen-and-Ink Illustrations

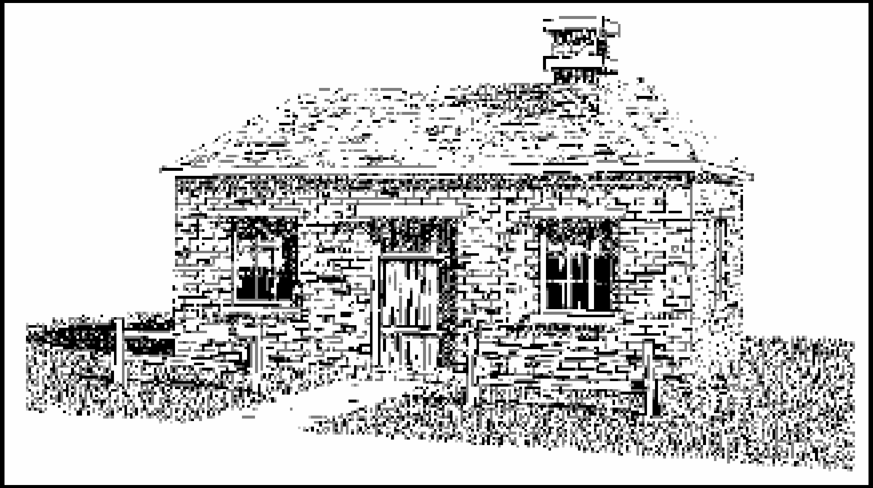
- Strokes
  - Curved lines of varying thickness and density
- Texture
  - Character conveyed by collection of strokes
- Tone
  - Perceived gray level across image or segment
- Outline
  - Boundary lines that disambiguate structure

Pen-and-Ink Example



Winkenbach and Salesin 1994

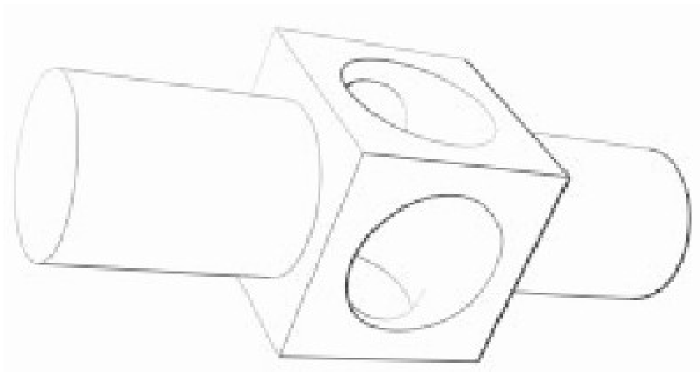


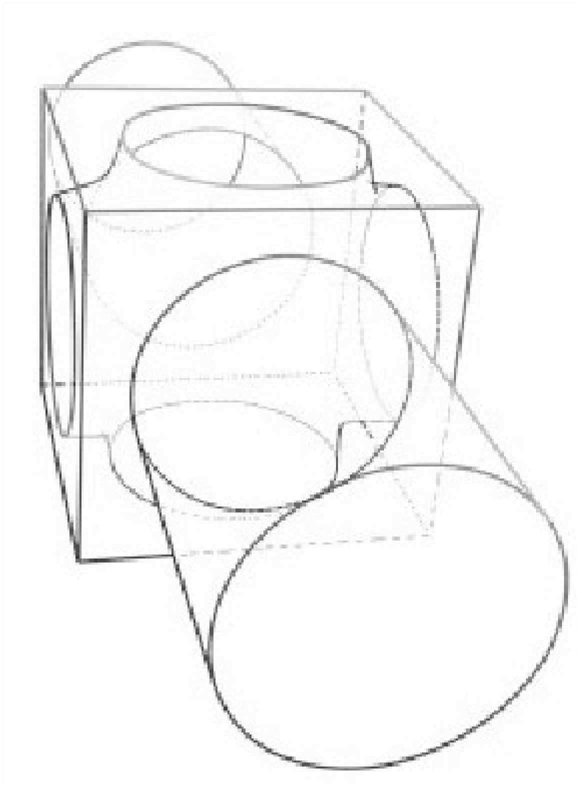


Drawing Strokes

- Stroke generated by moving along straight path
- Stroke perturbed by
  - Waviness function (straightness)
  - Pressure function (thickness)

Silhouette Curves





- Constructed from edges shared by both front-facing and back-facing mesh polygons
- Also include boundary edges
- Can be traced incrementally as a string of silhouette edges
- May not be visible, or not entirely visible
- Probability that an edge is a silhouette is proportional to $\pi - \theta$, where θ is the edge's dihedral angle

Edge Highlighting

Toon shading (and other NPR techniques based on drawing) requires some edges be drawn or highlighted:

- Silhouette edges
- Mesh boundaries (always on silhouette)
- Creases (ridge and valley)
- Material boundaries

Find first at run-time, precalculate the others (unless object is deformable)

Silhouette Edges

Surface angle silhouetting

- Calc N●V, if below threshold → draw black
  - Best as a per-pixel routine
  - The Cg program we looked at
  - Also can do with a spheremap, or use a mip-map with top-level textures dark
- Pros:
  - Uses the texture hardware → fast
  - Can antialias the resulting lines
- Cons:
  - Line width depends on curvature
  - Doesn't work for some models (e.g., a cube)

Silhouette Edges

- Procedural Geometry Silhouetting
  - Idea: render the geometry in such a way that the silhouettes "fall out", e.g.:
    - Draw frontfacing polygons
    - Draw backfacing polygons
      - But draw them in (possibly thick) wireframe
      - Or draw them z-biased forward a bit
      - Or "fatten" them
      - Or displace them along their normals ("halo" effect)
        - Flip normals
        - Amount of displacement varies w/ distance (*why?*)
        - Perfect task for vertex shader!
  - Pros: relatively robust, doesn't need connectivity info
  - Cons: Wastes some fill & some [illegible], needs antialiasing

Silhouette Edges

Image Processing Silhouetting

- Idea: analyze the image after it's rendered, and extract silhouettes (i.e., edge detection)
- Perfect for fragment program!
- Can help by rendering e.g. depth image, object-ID image, normal image

Silhouette Edges

- Silhouette Edge Detection
  - Idea: find silhouette edges geometrically on the CPU and render them explicitly
    - Brute force: test every edge to see if its adjoining polygons face opposite directions in eye space
      - Can speed this up with randomized coherent search
    - Most work, but gives the most flexibility in how silhouettes are drawn
  - GPU variant:
    - Draw degenerate quadrilateral at each edge
    - Use vertex shader to “fatten” quad into a “fin” when edge is on silhouette
      - Fin thickness based on distance to eyepoint

Highlighting Ridge Edges

- Clever related technique by Raskar:
  - Add “fins” to every edge at dihedral angle
  - Size fins according to distance to viewer
  - Again, perfect for vertex shader
- Similar but more complicated technique for highlighting valley edges

Drawing Lines: Outlining Polygons

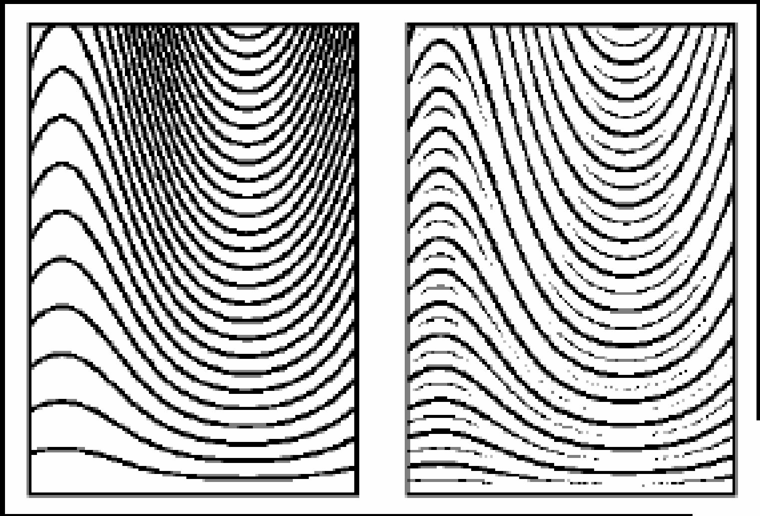
- Surprisingly hard to draw polys as filled outlines
  - Problem: depth buffer values of edge & polys same
  - 2-pass technique: draw polys, then draw edges
    - Z-bias edges forward or polygons back (**`glPolygonOffset`**)
    - Works okay, but has occasional problems
  - 3-pass technique:
    - Render filled polygon
      - Disable depth buffer writes (leave depth test on)
      - Enable color buffer writes
    - Render polygon edges polygon
      - Normal depth & color buffering
    - Render filled polygon again
      - Enable depth buffer writes
      - Disable color buffer writes

Drawing Lines:Hidden-Line Rendering

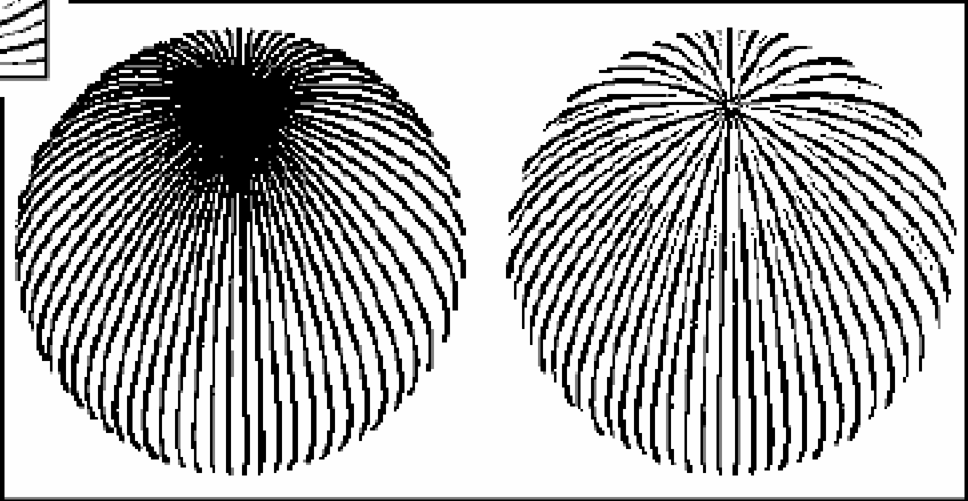
- Hidden-line vs. obscured line vs halos
  - Hidden-line
    - Draw polygons to depth buffer (not color buffer)
    - Draw edges using previous technique
  - Obscured (light, dotted, dashed) line
    - Draw all edges in obscured style
    - Draw polygons to depth buffer (not color buffer)
    - Draw edges using previous technique
  - Haloed line
    - Draw all edges as thick background-color lines
    - Draw edges using biasing, foreground-color

Stroke Width

- Adjust stroke width retain uniform tone

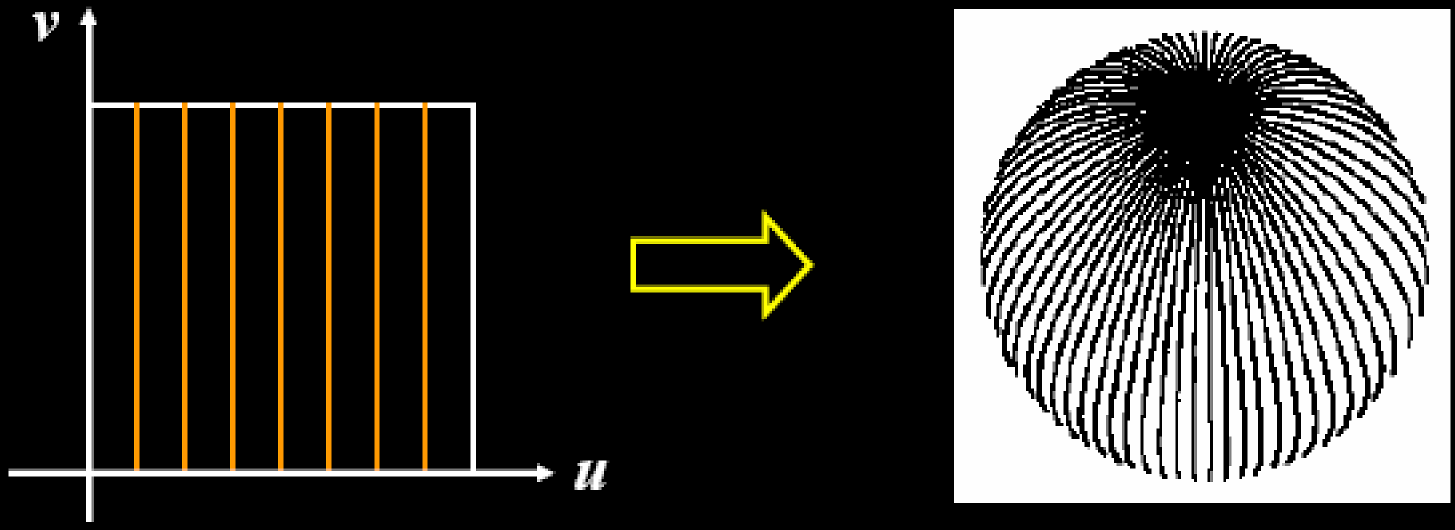


Winkenbach and Salesin 1996

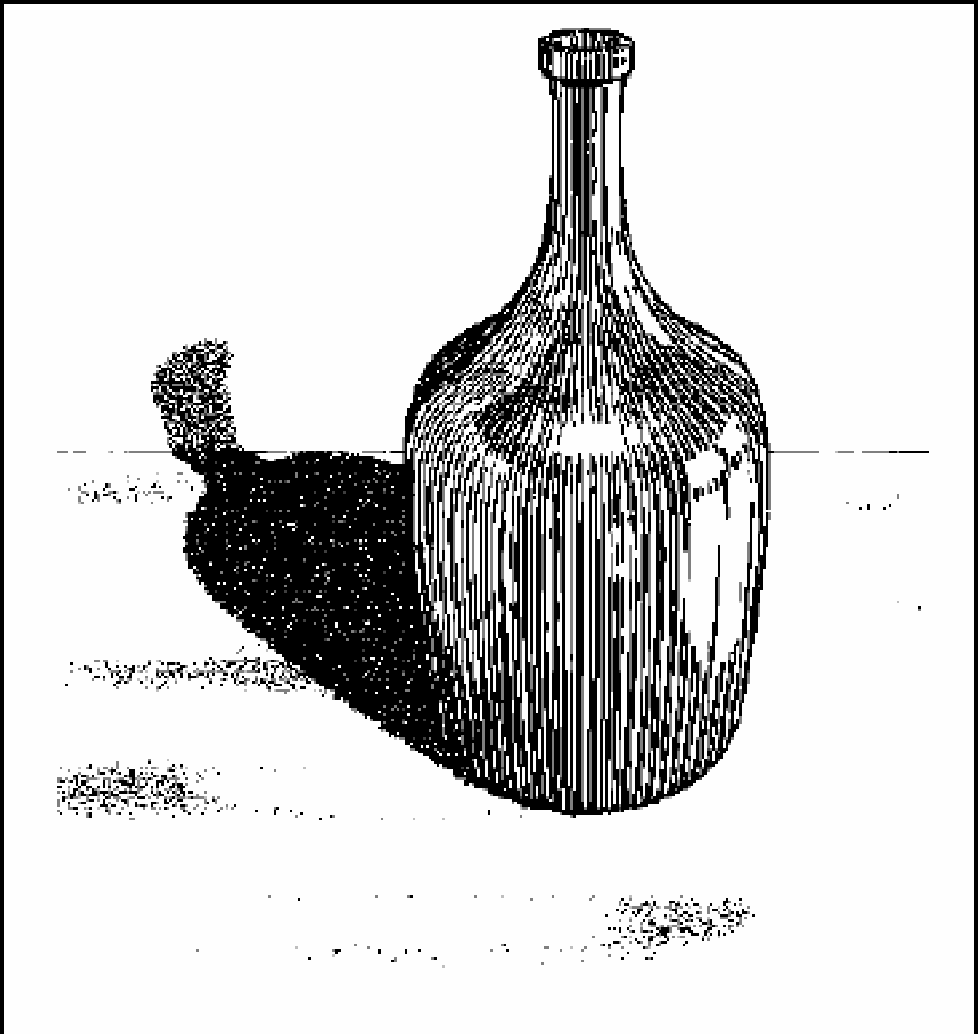


Rendering Parametric Surfaces

- Stroke orientation and density
  - Place strokes along isoparameter lines
  - Choose density for desired tone
  - tone = width / spacing



Parametric Surface Example

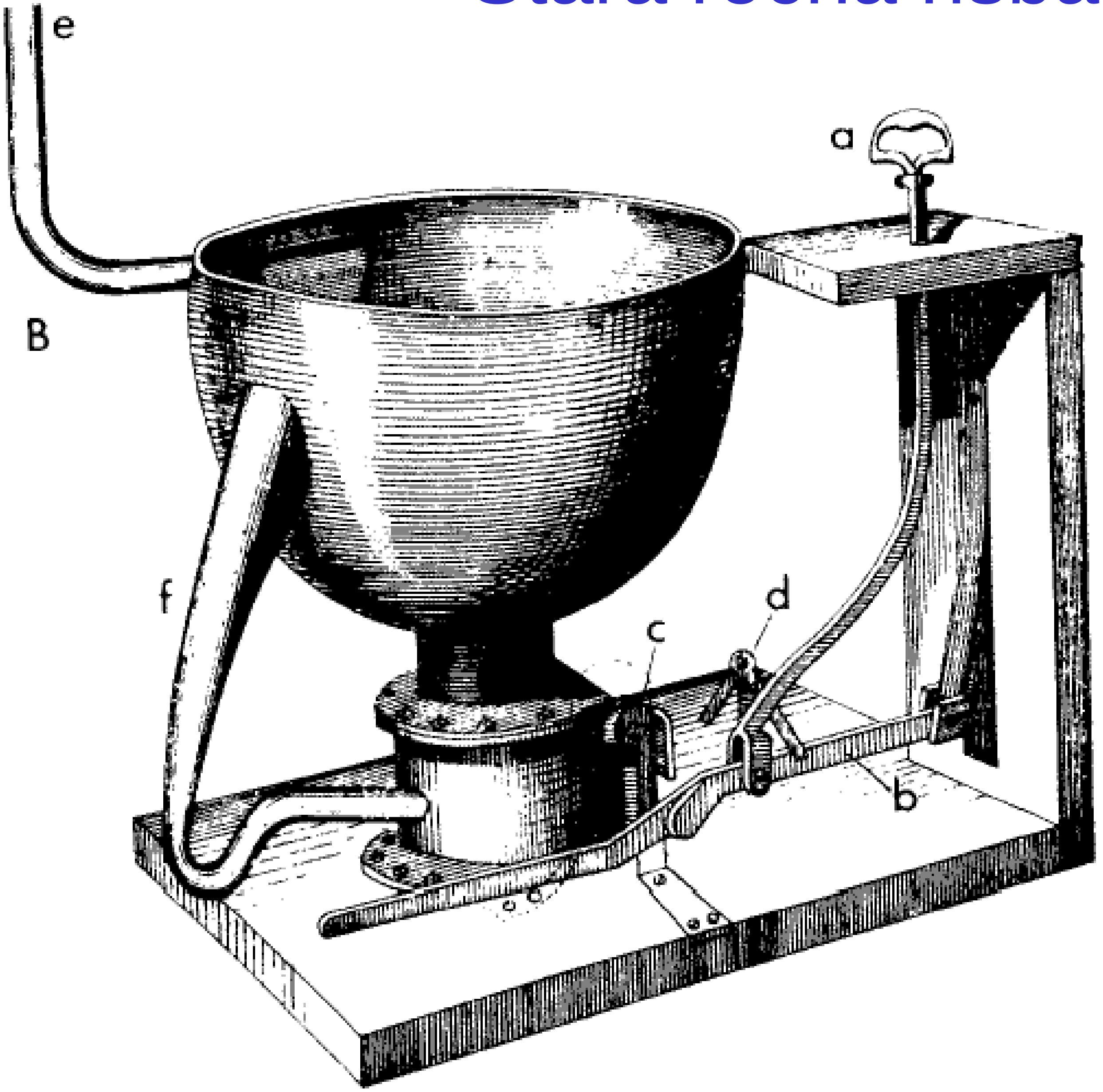


Winkenbach and Salesin 1996

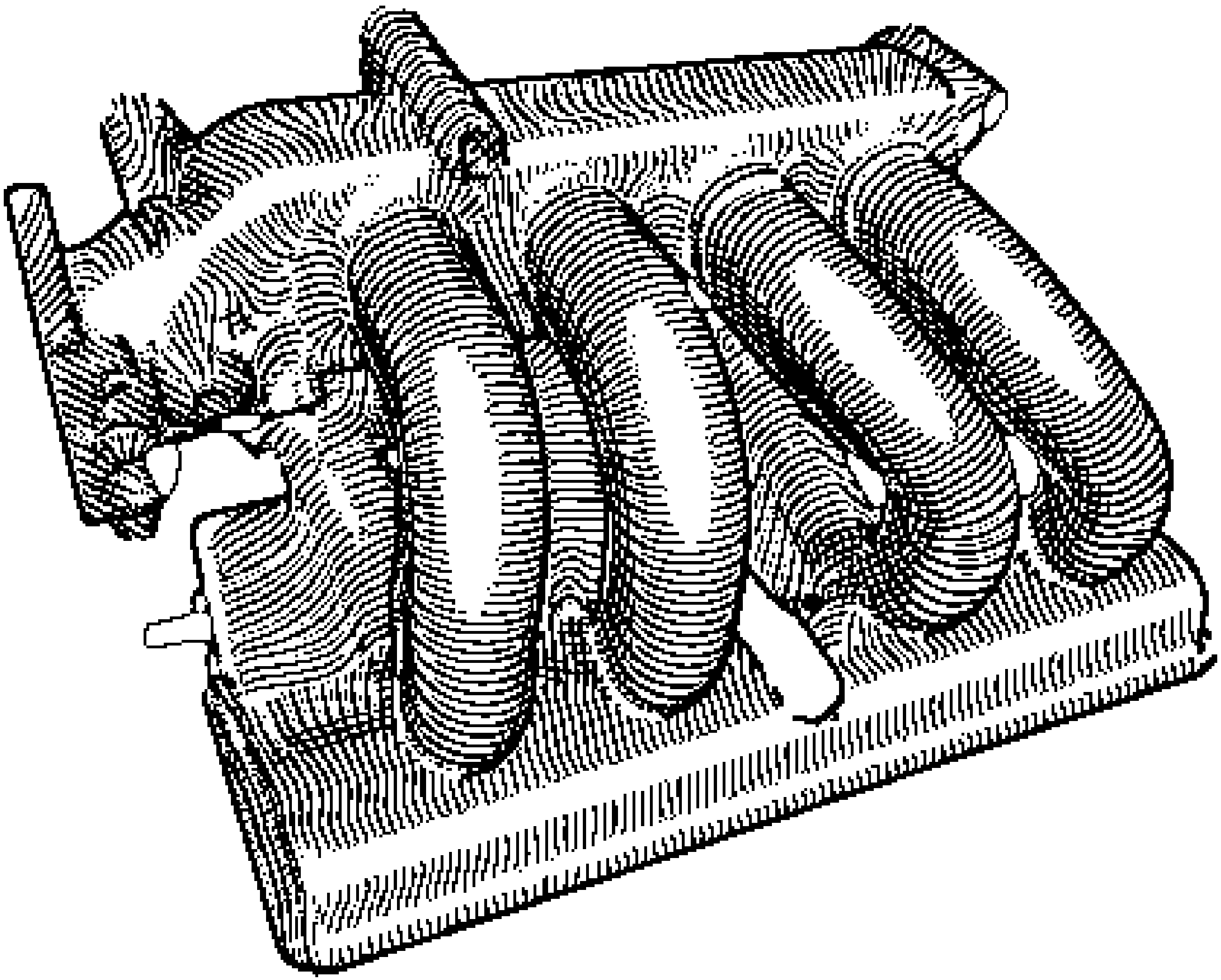
navidezno črtana grafika

Računalniška simulacija ročnega risanja

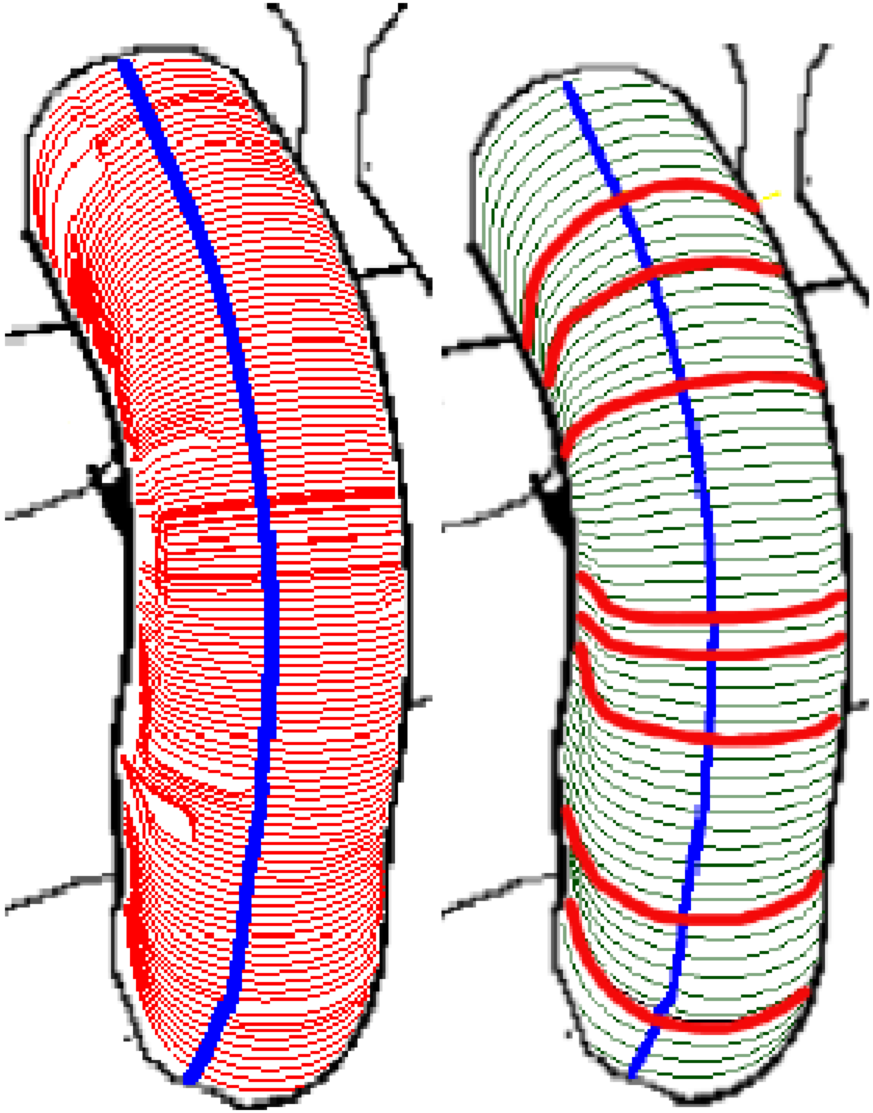
Stara ročna risba



Računalniško generirana “risba”



Koncept računalniške šrafure



Koncept “ribje kosti”

Najmanjša ukrivljenost – hrbtenica

Največja ukrivljenost – kosti

Painting

A painting can be seen as a collection of n brush stokes, with each stroke made up of several properties.

Painterly Rendering

- From strokes to brush strokes ...

- Automatic painting
  - User provides input image or 3D model
  - User specifies painting parameters
  - Computer generates all strokes
- Physical simulation
  - Computer simulates media
- Subject to controversy

Automatic Painting Example

Hertzmann 1998

Creative techniques

- Like real painting, render the scene in layers
  - Paint each object with multiple layers, each shrunk in more. Outside layers are painted sparsely, inner layers painted thicker.
  - Isolate highlights, shadows using image processing techniques and paint in a separate layer
  - Each object or group of objects in a scene can be given its own layer
- Painting parameters can be chosen per-layer
- Semi-transparent layers allow compositing of styles

Layered Painting

Blurring

Adding detail with smaller strokes

Painterly Rendering

Goals

- Avoid “shower-door” effect
- Provide for frame-to-frame coherence

Previous techniques achieved one or the other

Painterly Rendering

- How to achieve goals:
  - Use object geometry, color to decide where to place strokes
  - Distribute particles on object surface
  - Paint in screen space whereeever a particle is placed
- Randomness adds character
  - Store random seed in “particle”
  - Perturb color, orientation, scale based on user-selectable parameters

Study of painterly styles

Many painterly styles correspond closely to perceptual features that are detected by the human visual system.

Focus on Impressionism

Trying to pair each style with a corresponding visual feature that has been proved to be effective in a perceptual visualization environment.

Different painterly styles

Painterly styles can be identified by studying those paints:

- Path of the stroke
- Length
- Density
- Coarseness
- Weight

Some Styles

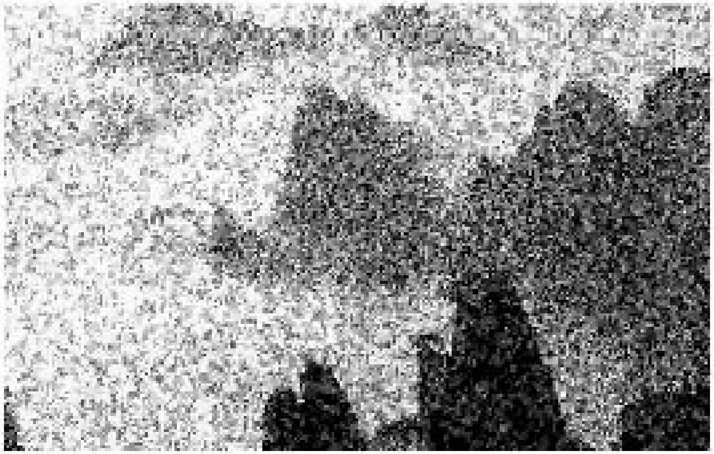
- "Impressionist"
  - No random color, 4 · stroke length · 16
  - Brush sizes 8, 4, 2; approximation threshold 100
- "Expressionist"
  - Random factor 0.5, 10 · stroke length · 16
  - Brush sizes 8, 4, 2; approximation threshold 50
- "Pointilist"
  - Random factor ~0.75, 0 · stroke length · 0
  - Brush sizes 4, 2; approximation threshold 100
- Not convincing to artists

Style Examples

Source image

"Impressionist"

"Expressionist"



"Pointillist"

Impressionism

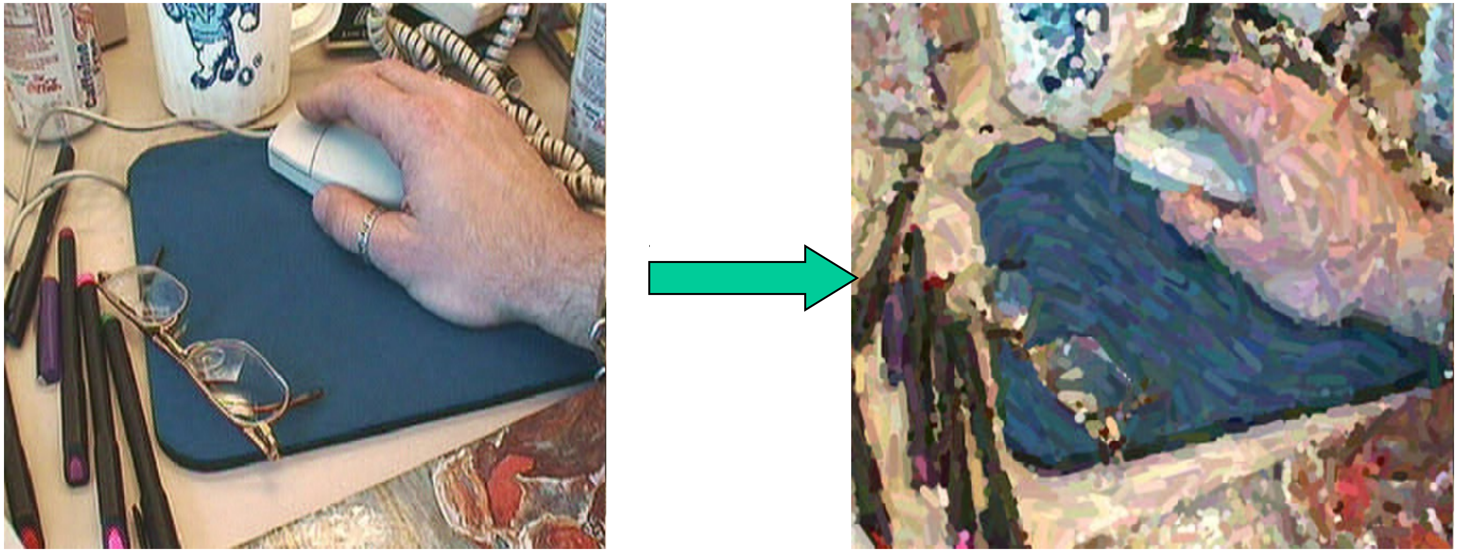
Attached to a small group of French artists, Monet, Van Gogh…who broken the traditional schools of that time to approach painting from a new perspective

Some underlying principles…

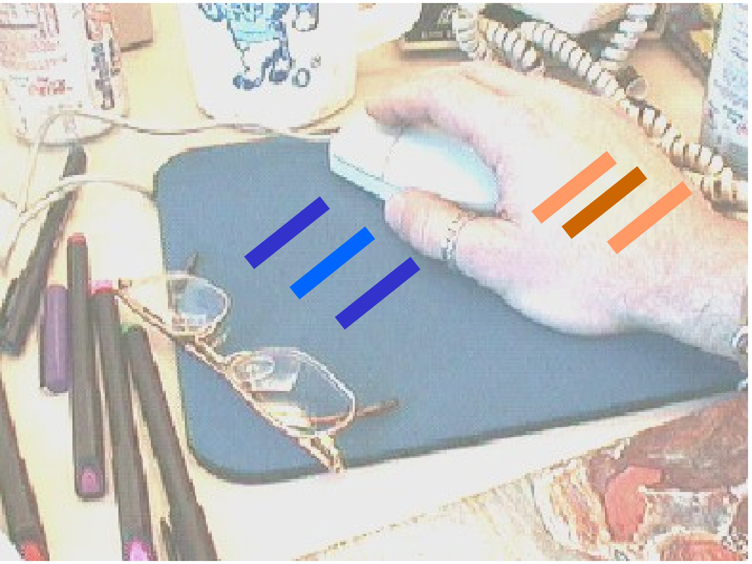
- Object and environment interpenetrate
- Color acquires independence
- Show a small section of nature
- Minimize perspective
- Solicit a viewer’s optics

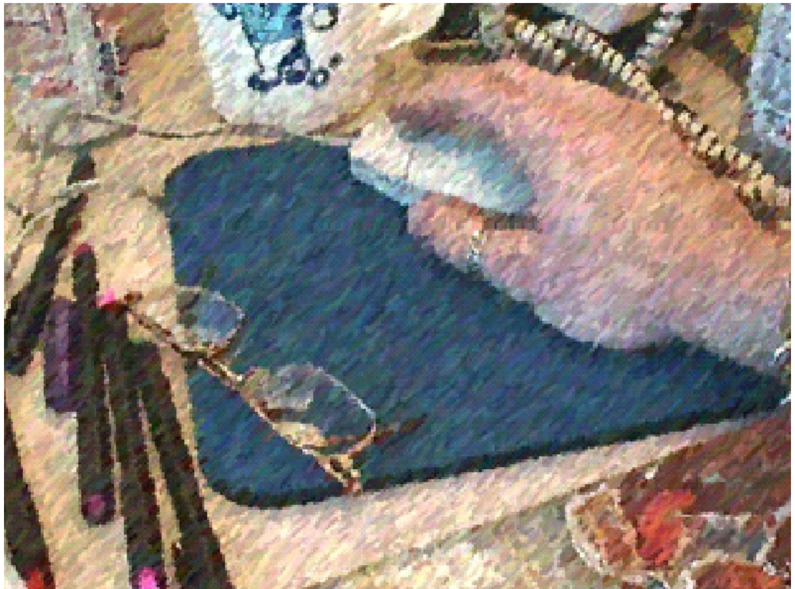
Processing Images and Video for an Impressionist Effect

Transform images/video into animation with Impressionist effect, particularly, with hand-painted style



Princip grafičnega impresionizma





Stroke rendering

Stroke generation

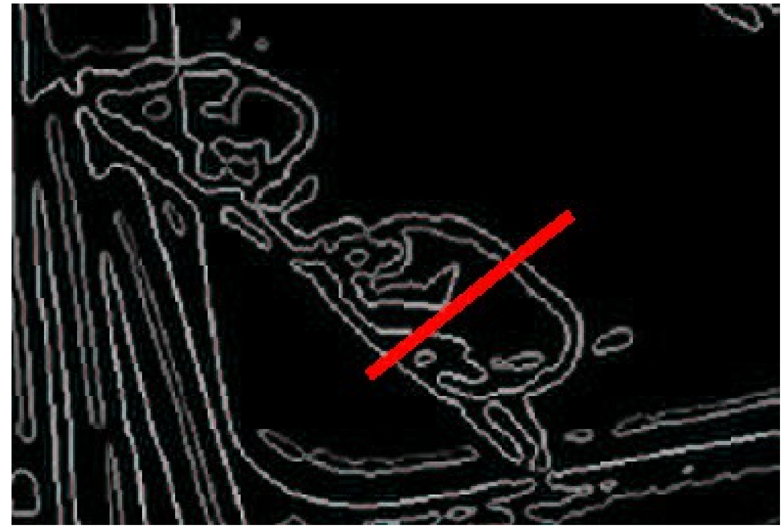
- Size, position, length
- color
- Orientation

Random perturbation

Clipping and rendering

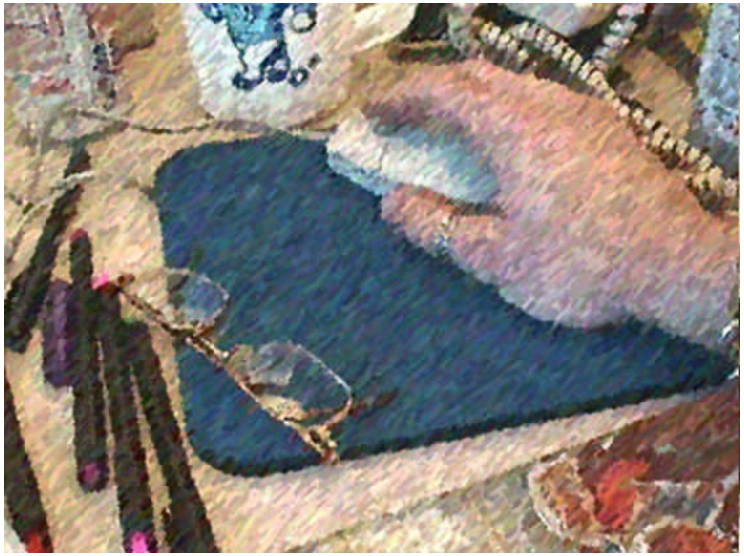
- Edge preservation

Using brush textures



Example

Without clipping



With clipping

Brush stroke orientation

Draw stroke in direction of constant color

the normal to the gradient direction

Area with small magnitude of gradient ?

Interpolate surrounding “good” gradient

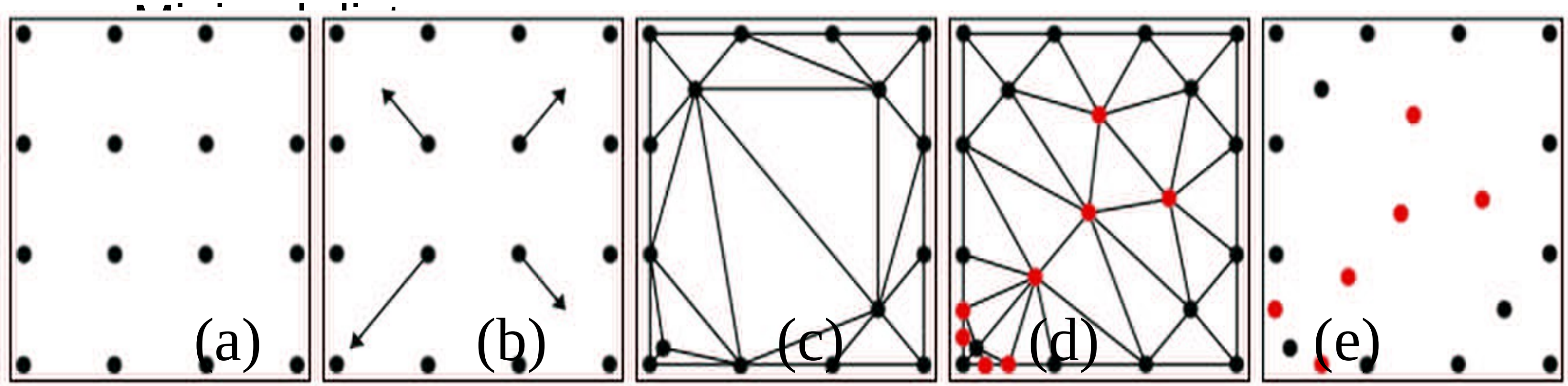
Frame-to-frame coherence

How to move strokes across frames

- Using Optical flow [Bergen 90] as stroke displacement

How to avoid over-sparse and over dense stroke distribution?

- Delaunay triangulation
- Maximal area



Conclusion

An algorithm for producing painterly animation from video

Highlights

- Use optical flow to move strokes across frames to keep temporal coherence
- Orient strokes using gradient-based methods
- Methods to redistribute strokes
- Edge preservation strategy

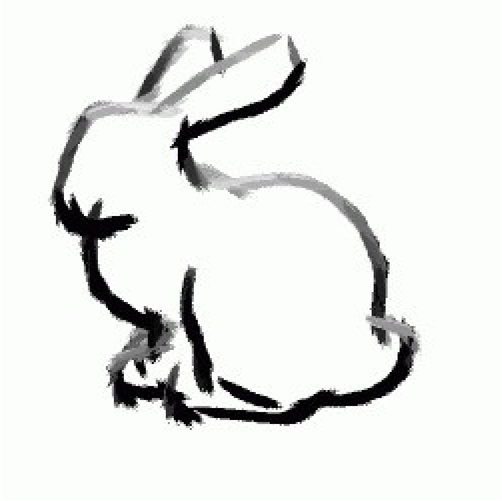
Drawback

- jittering

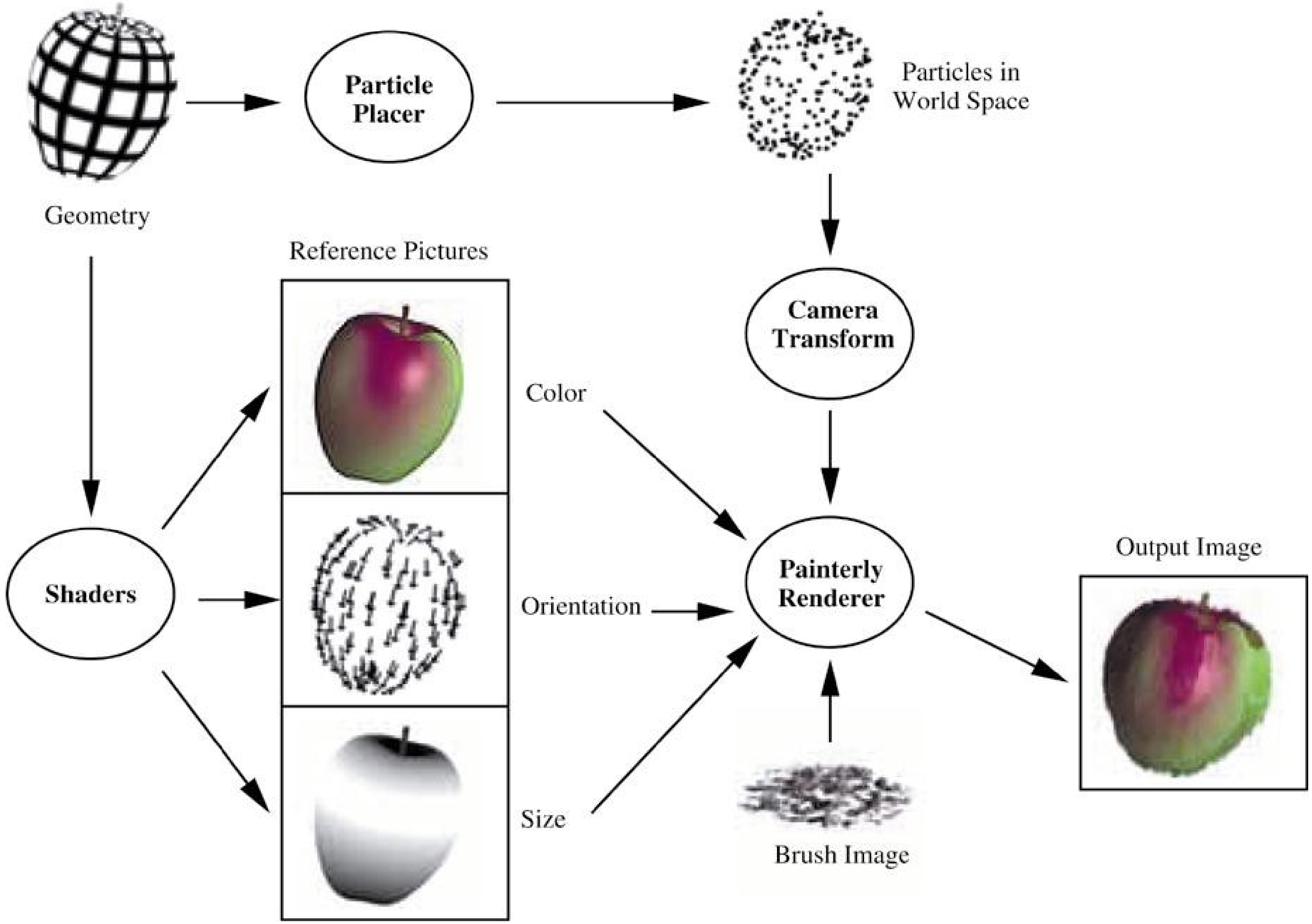
Other Styles

Impressionistic or “painterly” rendering:

- Sprinkle particles on object surface
- Draw particles as brushstrokes
- Can render images to encode normals, surface curvature, depth, color/tone info



Painterly Rendering



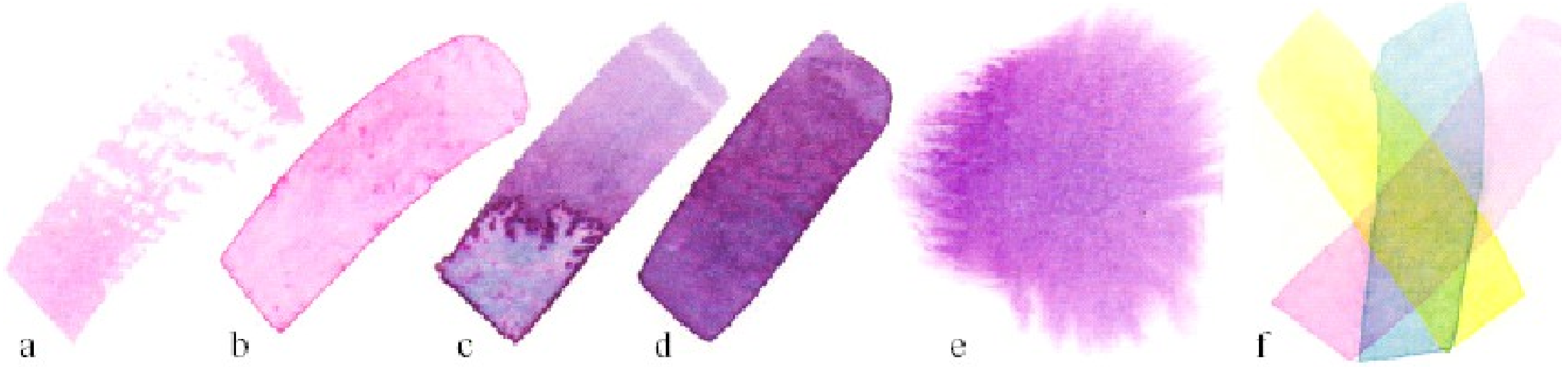
More info if time permits…

Physical Simulation Example

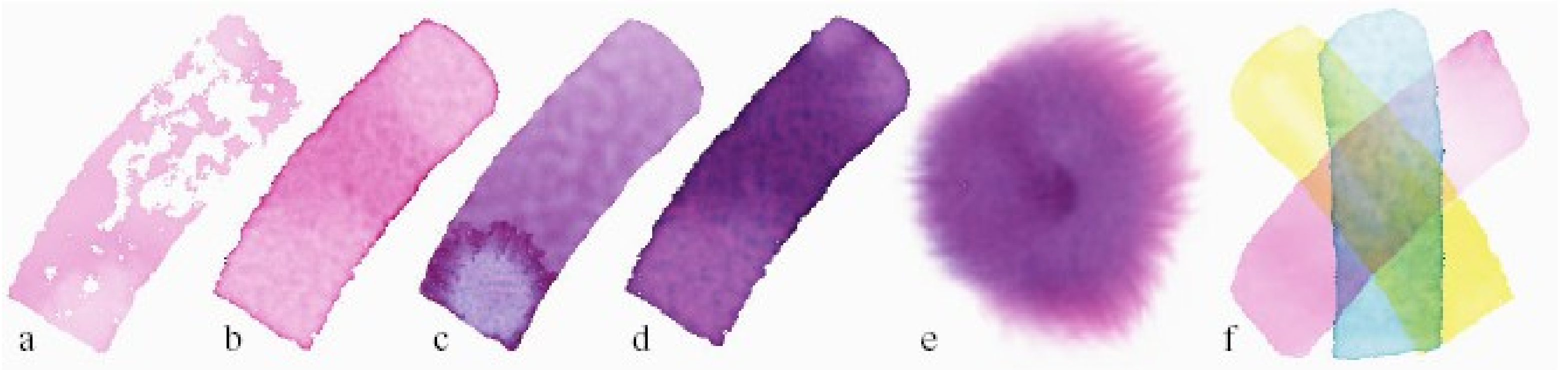
Curtis et al. 1997, *Computer Generated Watercolor*

Simulacija vodnih barvic

Resnične vodne barvice

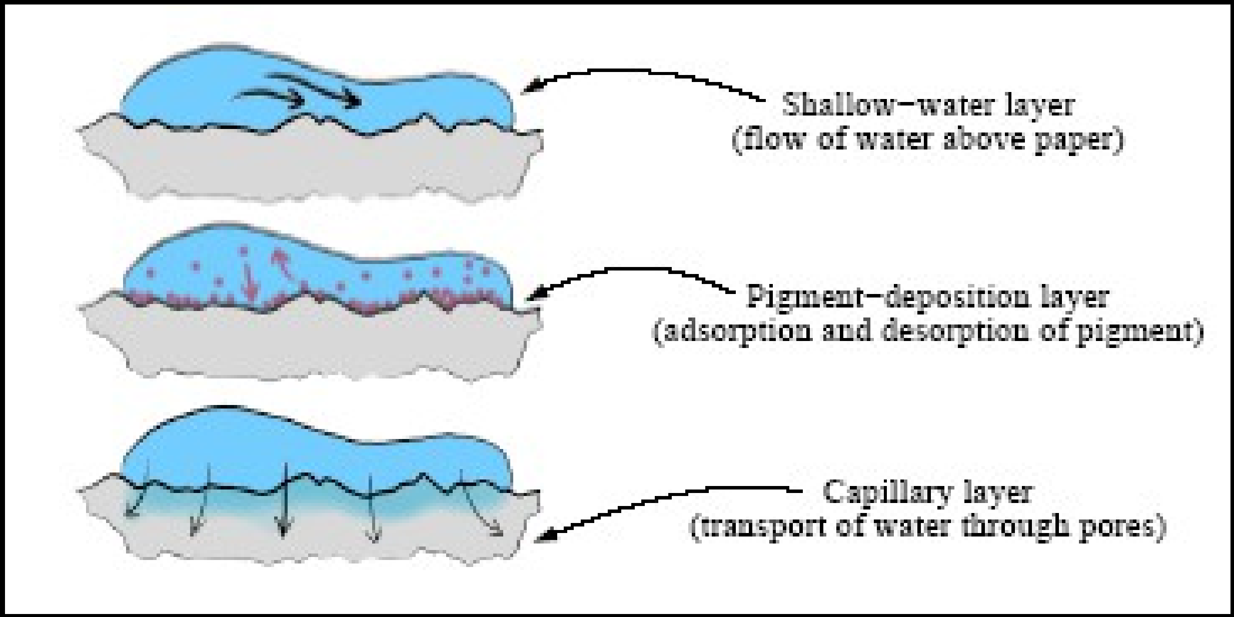


Računalniško generirane “vodne barvice”

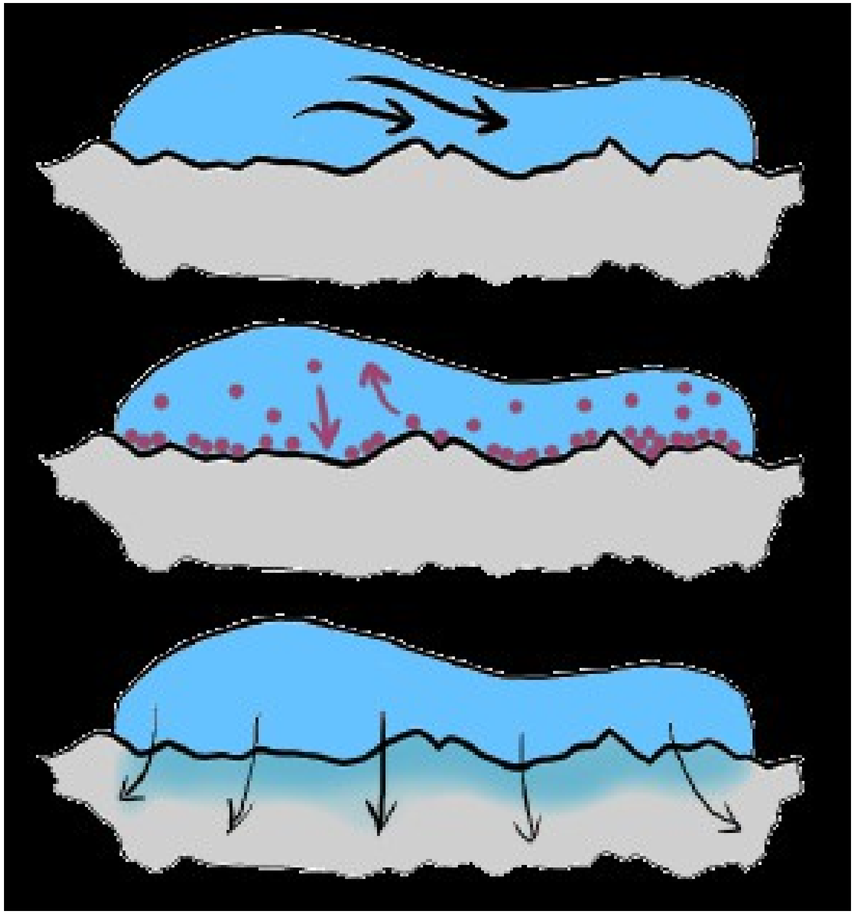


Fluid Simulation

- Use water velocity, viscosity, drag, pressure, pigment concentration, paper gradient
- Paper saturation and capacity



Princip simulacije vodnih barvic

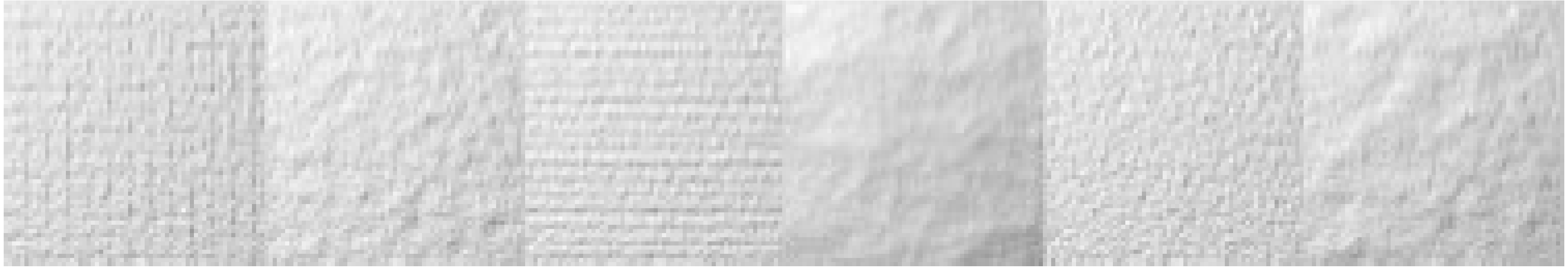


Preliv vode po papirju

Vpijanje in izplakovanje pigmenta

Kapilarna absorbcija vode v papir

Simulacija papirja



In real watercolor, the structure of the paper affects fluid flow, backruns, and granulation. The mechanics underlying these effects may be quite complex, and may depend on the precise connections among the individual fibers, as well as the exact slopes of the finescale peaks and valleys of the paper.

We use a much simpler model in our system. Paper texture is modeled as a height field and a fluid capacity field. The height field ***h*** is generated using one of

a selection of pseudo-random processes, and scaled so that $0 < h < 1$. Some examples of our synthetic paper textures can be seen in Figure. The slope of the height field is used to modify the fluid velocity ***u***, ***v*** in the dynamics simulation. In

addition, the fluid capacity ***c*** is computed from the height field ***h***, as

$c = h * (cmax - cmin) + cmin.$

Različni barvni pigmenti

Glavna zanka

proc *MainLoop*$(\boldsymbol{M}, u, v, p, g^1, \ldots, g^n, d^1, \ldots, d^n, s)$:
    for each time step **do**:
        MoveWater$(\boldsymbol{M}, u, v, p)$
        MovePigment$(\boldsymbol{M}, u, v, g^1, \ldots, g^n)$
        TransferPigment$(g^1, \ldots, g^n, d^1, \ldots, d^n)$
        SimulateCapillaryFlow$(\boldsymbol{M}, s)$
    end for
end proc

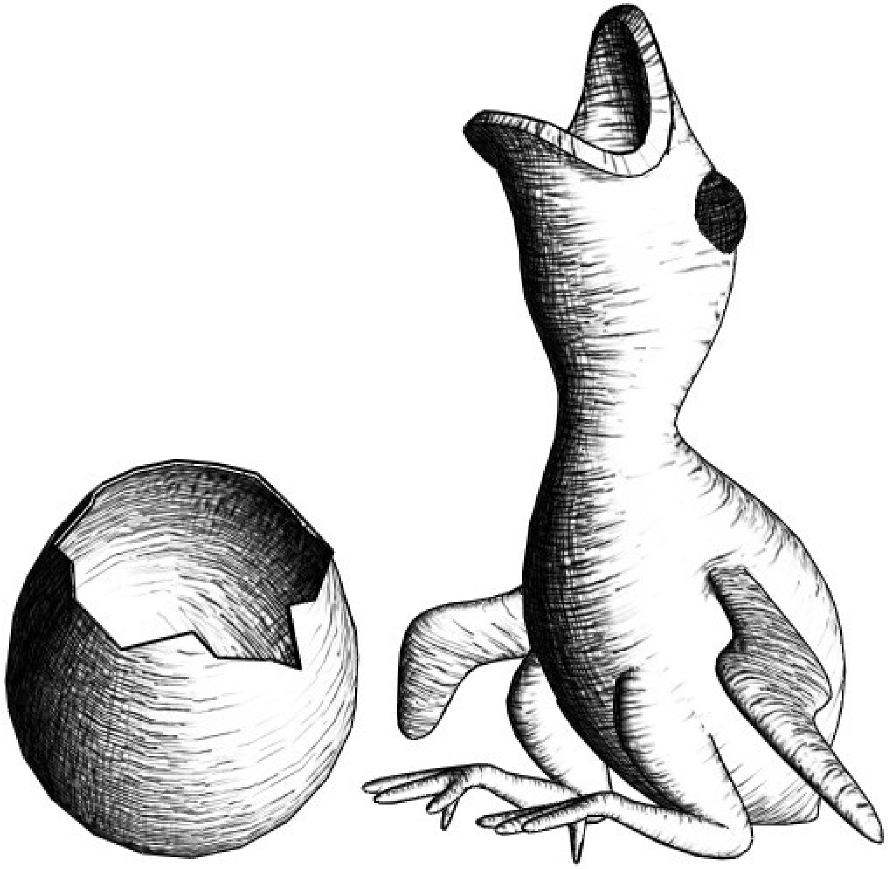
Članek

Drugi stili

Hatching:

- Store different cross-hatch patterns representing different tones as textures
- Clever ways to use texture hardware to blend between tones at run-time

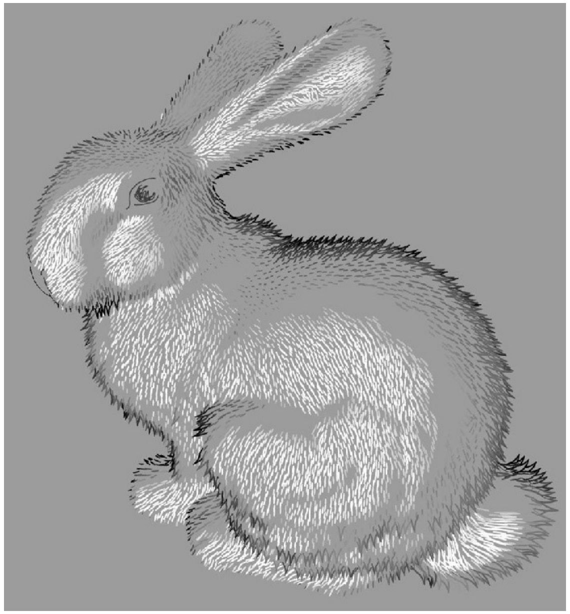
More info if time permits…

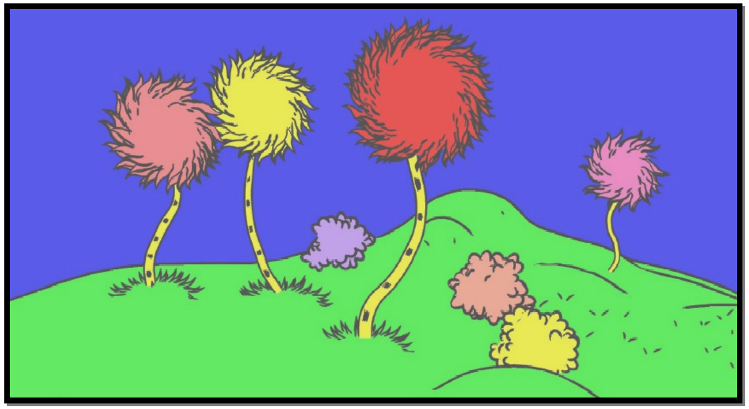


Other Styles

“Graftals” are a general term used for strokes, decals, little bits of geometry

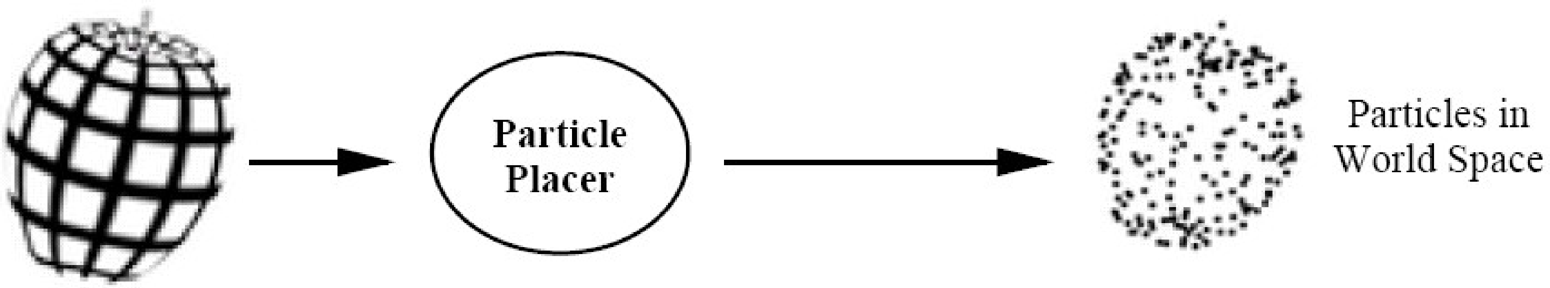
Dynamic placement of graftals to achieve certain effects/styles:





The Algorithm in Detail

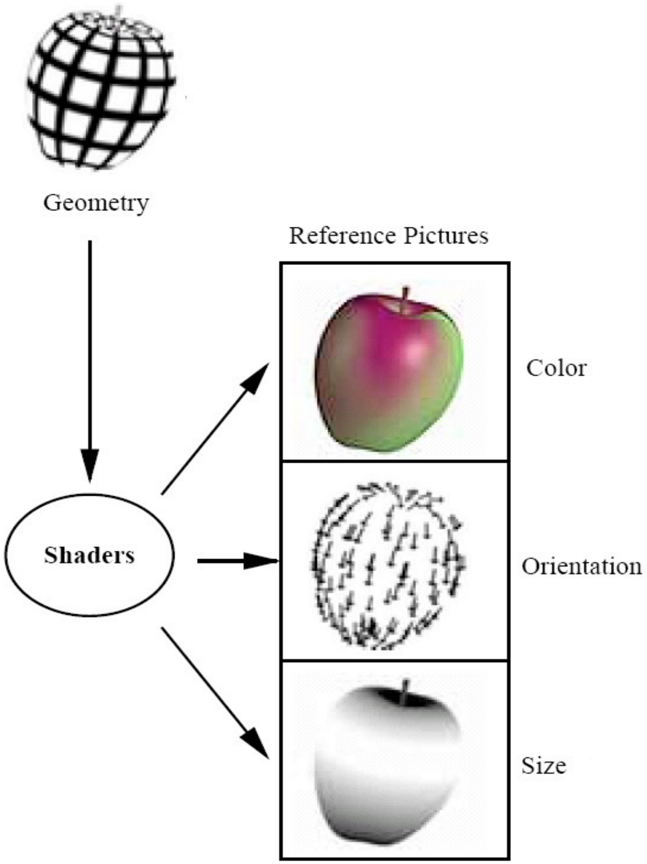
Step 1: Create particles to represent geometry



The Algorithm cont...

Step 2: For each frame of animation...

create reference pictures using geometry, surface attributes, and lighting



The Algorithm cont...

Step 3: Also for each frame of the animation...

transform particles based on animation parameters

sort particles by distance from viewpoint

for each particle, starting with furthest from viewpoint

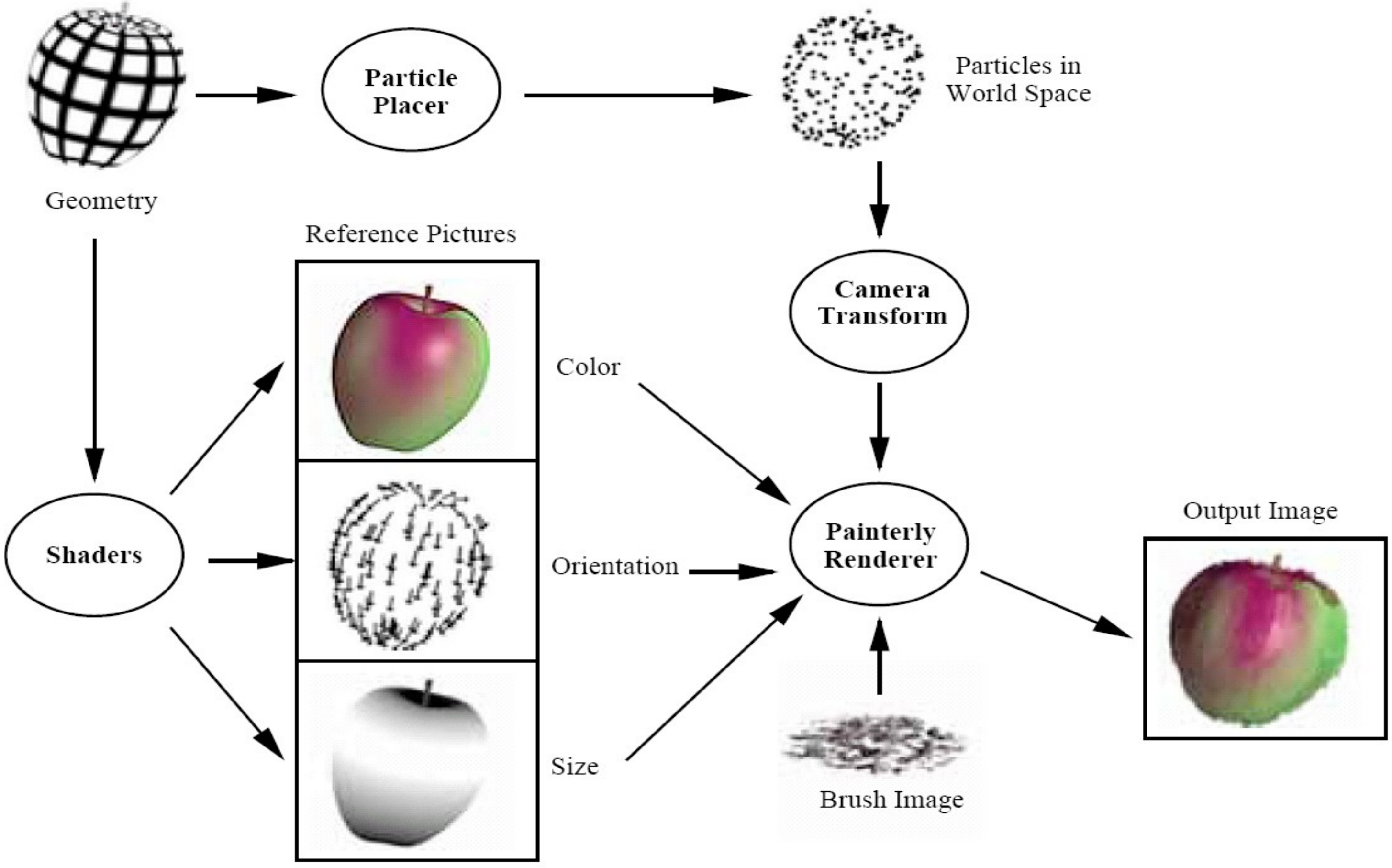
transform particle to screen space

determine brush stroke attributes from reference pictures or particles and randomly perturb them based on user-selected parameters

composite brush stroke into paint buffer

end (for each particle)

Putting it all together



Technical considerations

- Brush strokes may jitter in size and orientation slightly between frames
  - So blur the size and orientation reference images before sampling
- Rendering of back-facing particles
  - Useful so previously obscured strokes don't pop in when animating
  - Can cause visual problems when layering
  - Their solution culls back-facing particles, but fades them in as they get close to front-facing

Future Directions

Combining painterly look with traditional renderer

Automatically handling changing object size

Improving particle-placement algorithm to cover geometric surfcace and screen space more evenly

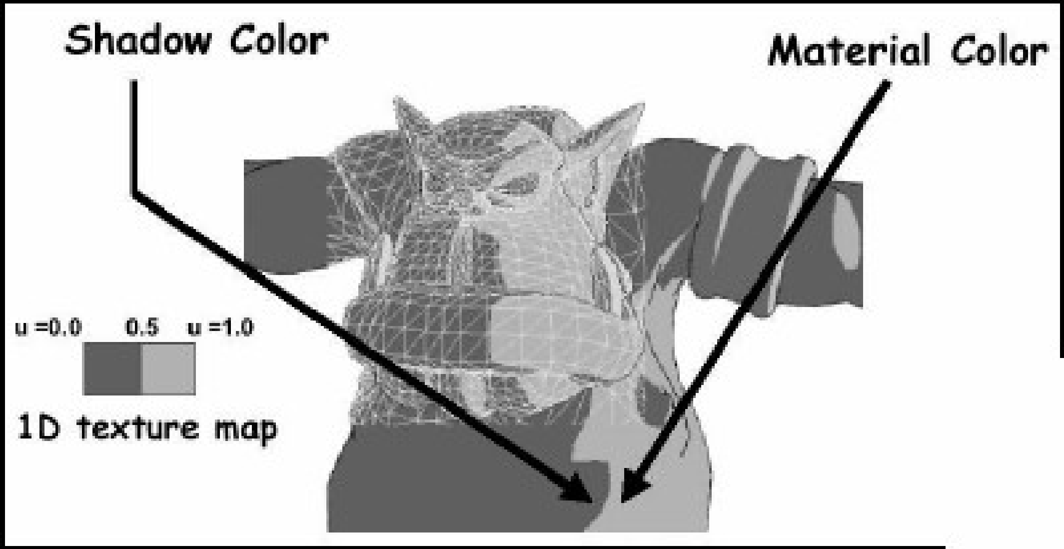
Implementing longer, deformable brushes that can follow curves on a surface

Cartoon Shading

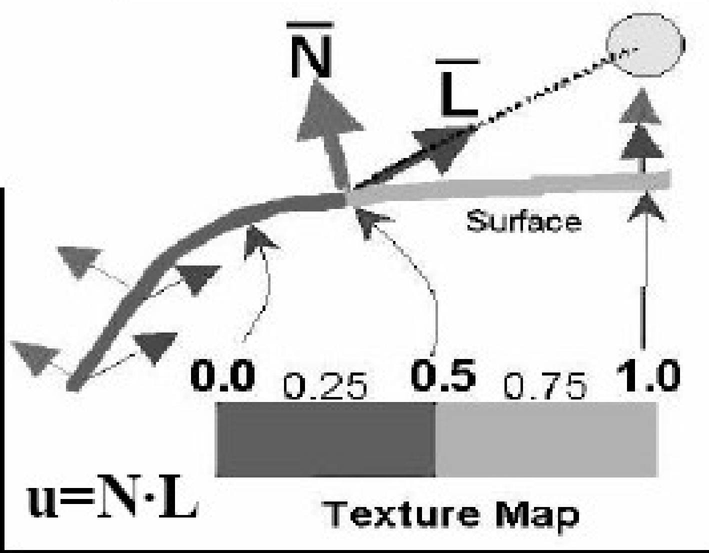
- Shading model in 2D cartoon
  - Use material color and shadow color
  - Present lighting cues, shape, and context
- Stylistic
- Used in many animated movies
- Developing real-time techniques for games

Cartoon Shading as Texture Map

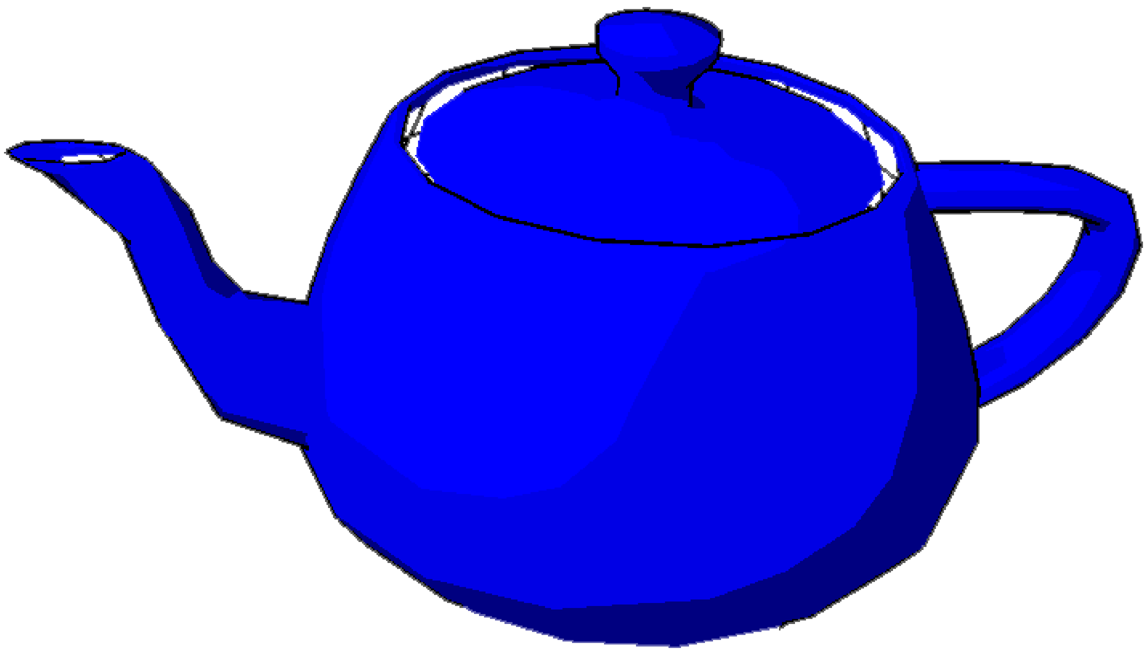
- Apply shading as 1D texture map



Carl Marshall 2000



Cel-shading Concepts



Cel Shading

Also called Cartoon Shading, or Hard Shading.

Named after the process of inking and coloring cels (clear plastic sheets) in hand-drawn animation.

3D objects look like a 2D cartoon.

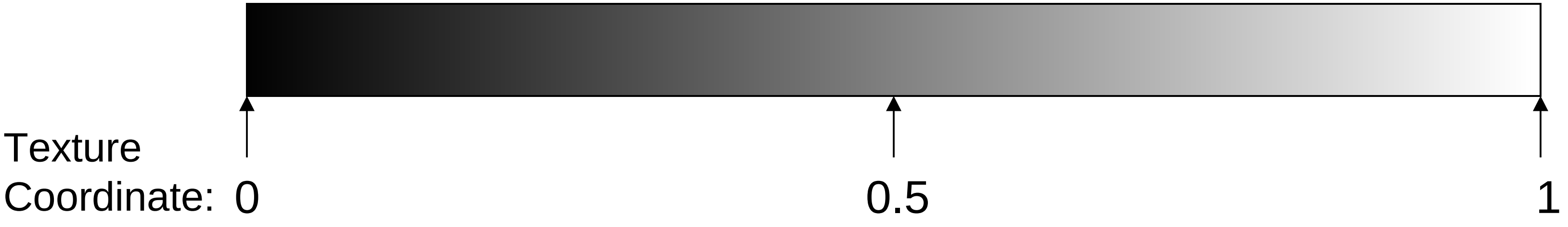
Two steps: Shading, Outline Drawing

Cel-Shading Example

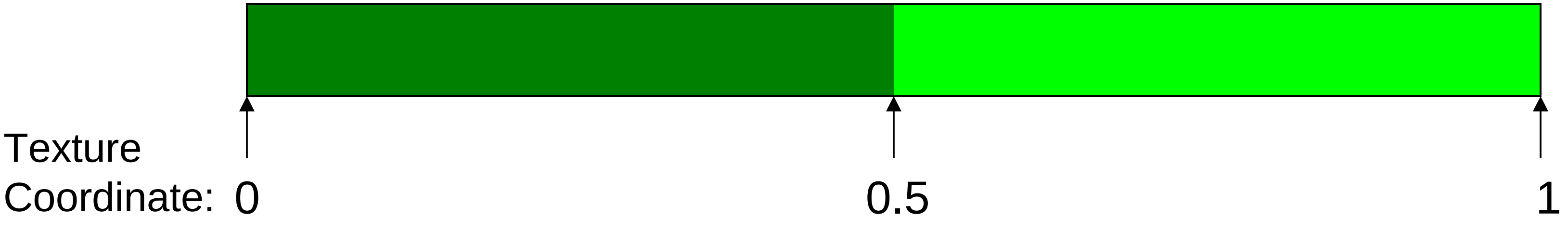
Image source: Intel Corporation, http://www.intel.com/labs/media/3dsoftware/nonphoto.htm

Shading

1D Textures – a 1x*n* texture.



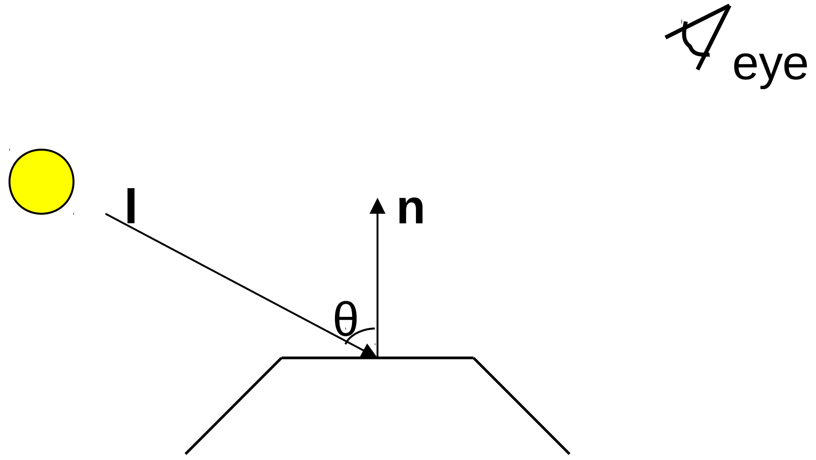
Consider this texture:



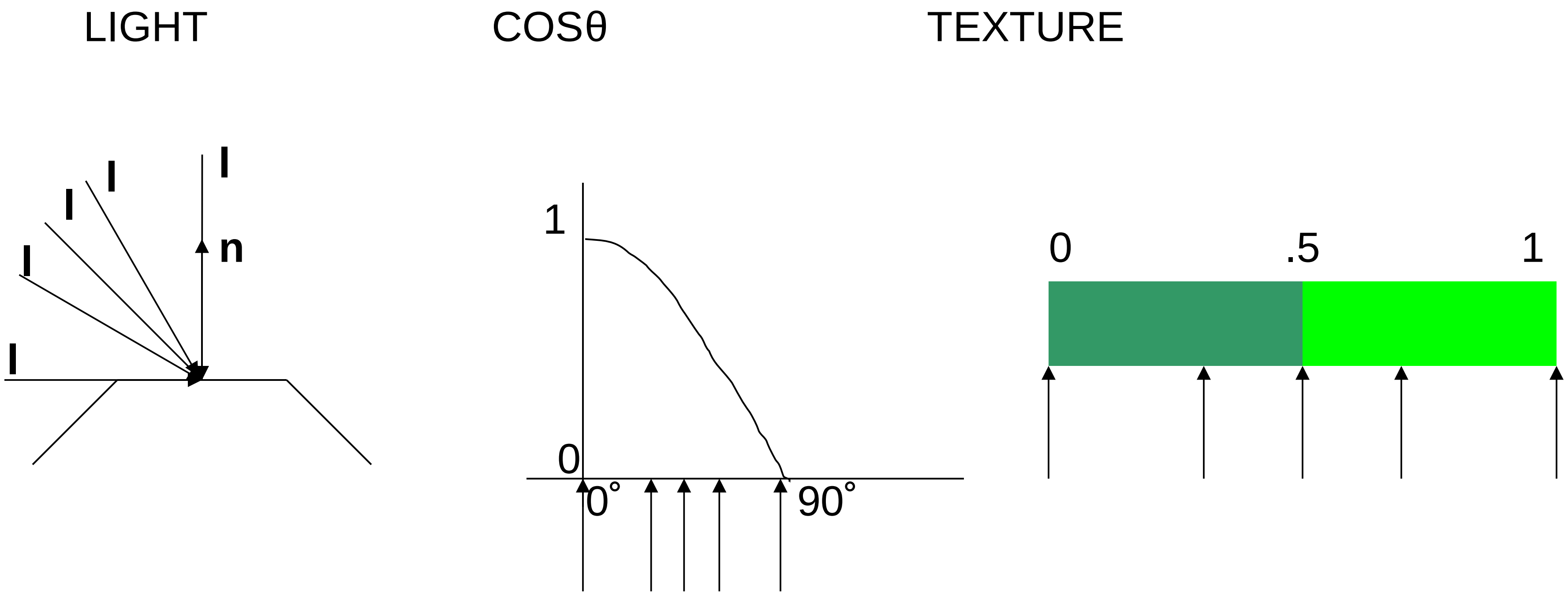
Choosing Texture Coordinate

Use light equation to pick a texture coordinate.

Recall the lighting model:



Choosing Texture Coordinate

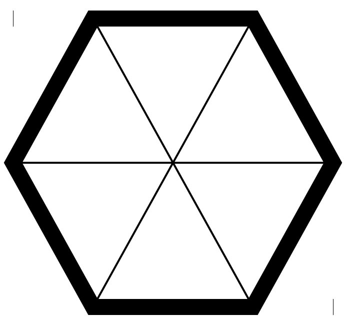


Outlining

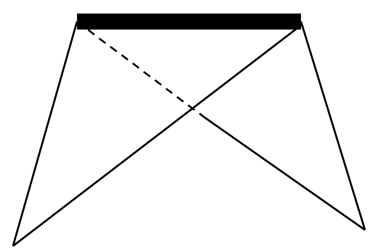
When is an edge part of the object outline?

- Border – The edge is not shared.
- Silhouette – An edge is shared between a front-facing and back-facing polygon.
- Hard edges – An edge shared by polygons that meet within some angle threshold (0° through 180°).

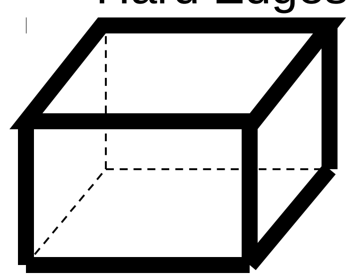
Border



Silhouette



Hard Edges



Algorithms

Software: (after drawing shading)

Build an edge list.
for each edge in edge list
  if edge not shared **then**
    draw edge.
  if edge belongs to front-facing and back-facing polygon **then**
    draw edge
  if edge belongs to two front-facing polygons that meet within some threshold **then**
    draw edge

Hardware: (after drawing shading)

Use Z-Buffer and hardware support for drawing front-facing or back-facing polygons.

Using the OpenGL pipeline

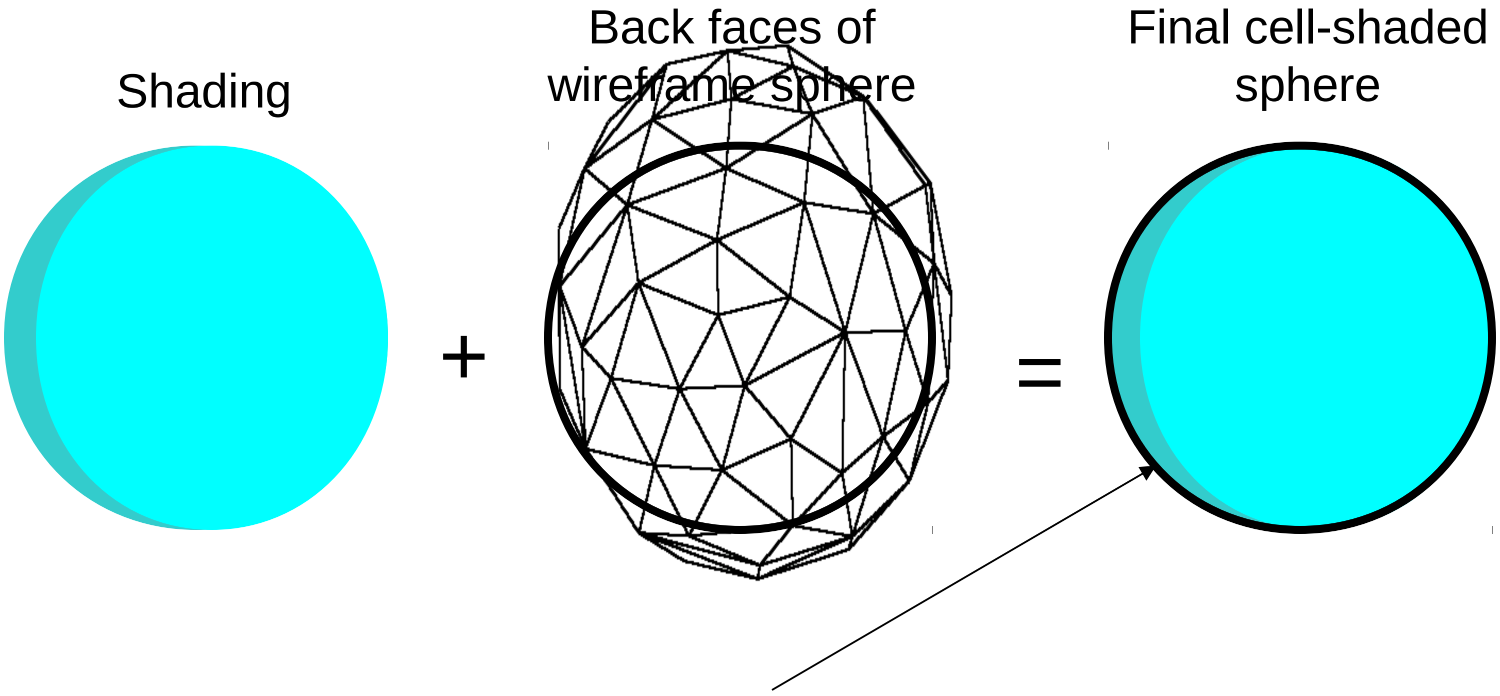
Set Z-Buffer comparison to less than or equal.
::glDepthFunc(GL_LEQUAL)

Tell the hardware to draw only back-facing polygons, and to draw them in wireframe mode.
::glCullFace(GL_FRONT)
::glPolygonMode(GL_BACK, GL_LINE)

Set line width ≥ 1.0 for outlines
::glLineWidth(*w*)

Draw model.

Using the Hardware (continued)



The border comes from the Z-Buffer test. The only edges of the back-facing wireframe that are drawn are the edges that are at the same depth as the shaded front-facing polygons (i.e., silhouette edges).

Trade Offs

Software	Hardware
Have to maintain an edge list.	No additional data structures.
Outline can be drawn in the same rendering pass as the shading.	Outline requires another rendering pass.
Allows borders, outlines, and hard edges.	Draws only silhouettes and certain borders.
More difficult to implement.	Very easy to implement.

Examples

Left: Screenshot from the game XIII (UbiSoft), an example of real-time cel shading.

Right: Image from Hotaru, a short animation created using a cel-shading plugin (unReal) for Lightwave3D.

Related Resources

NeHe’s cel-shading tutorial: http://nehe.gamedev.net/, lesson 37.

Many resources for cel shading: http://www.celshader.com/links.html

BESM – Freeware (source code available) plugin for Lightwave 3D:
http://www.celshaded.com/

Hotaru animation - http://ikkyuu-an.homeip.net/%7Eshishi/hotaru/

XIII game - http://www.whoisxiii.com/community/index.php

Academic Resources

Lake, A., Marshall, C., Harris, M., and Blackstein, M. Stylized rendering techniques for scalable real-time 3d animation. Proceedings of NPAR 2000, 13--20. http://citeseer.nj.nec.com/lake00stylized.html

Ramesh Raskar. Hardware support for non-photorealistic rendering. Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware, pages 41--46, August 2001. http://citeseer.nj.nec.com/raskar01hardware.html

L. Markosian, M. Kowalski, S. Trychin, and J. Hughes. Real-Time NonPhotorealistic Rendering. In SIGGRAPH 97 Conference Proceedings, August 1997. http://citeseer.nj.nec.com/markosian97realtime.html

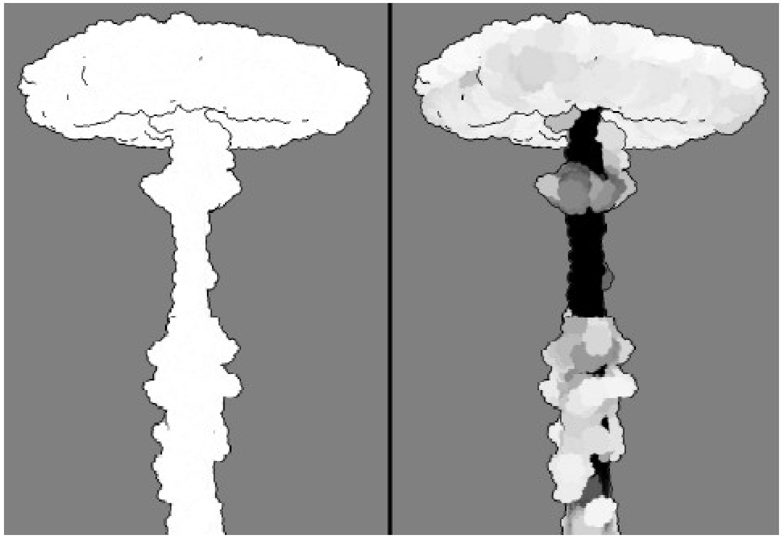
Gooch B., Gooch A. Non-Photorealistic Rendering. A.K. Peters, 2001.
(available in Noble Library)

Strothotte T., Schlechtweg, S. Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation. Morgan Kaufmann Publishers, 2002.
(available in Noble Library)

Cartoon Rendering of Smoke Animations

Cartoon Smoke

- Uses physically-based simulation to drive nonphotorealistic rendering
- Draws silhouette edges based on depth differences technique
- Maintains temporal coherence

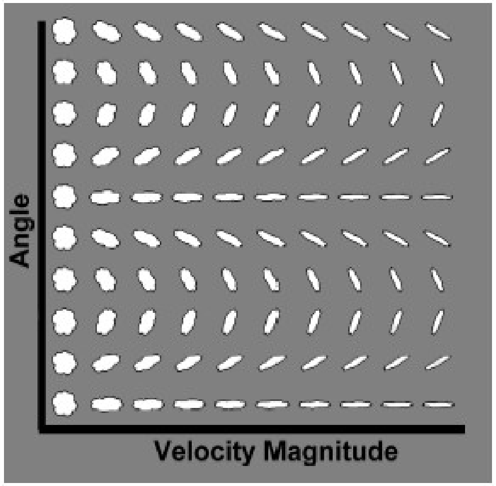


Simulation-Rendering Interface

Size determined by density around particle

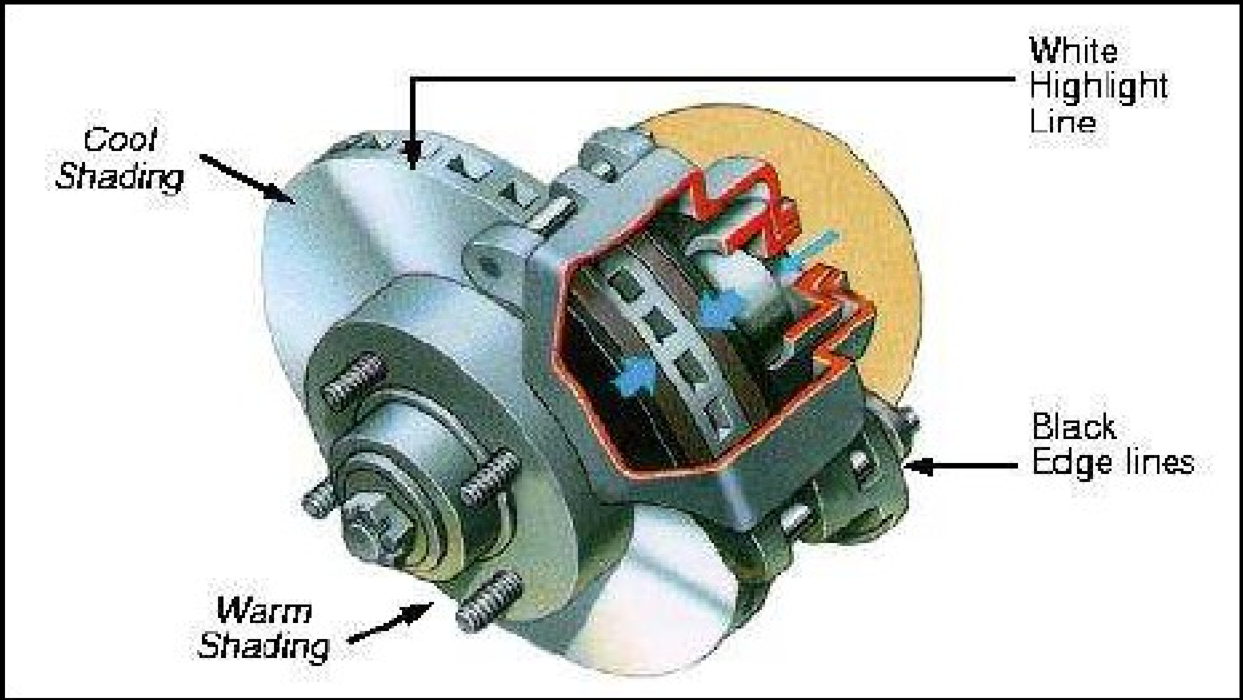
Color determined by temperature or density

Rotation and amount of stretch determined by particle's velocity



Conventions in Technical Illustrations

- Black edge lines
- Cool to warm shading colors
- Single light source; shadows rarely used



Summary

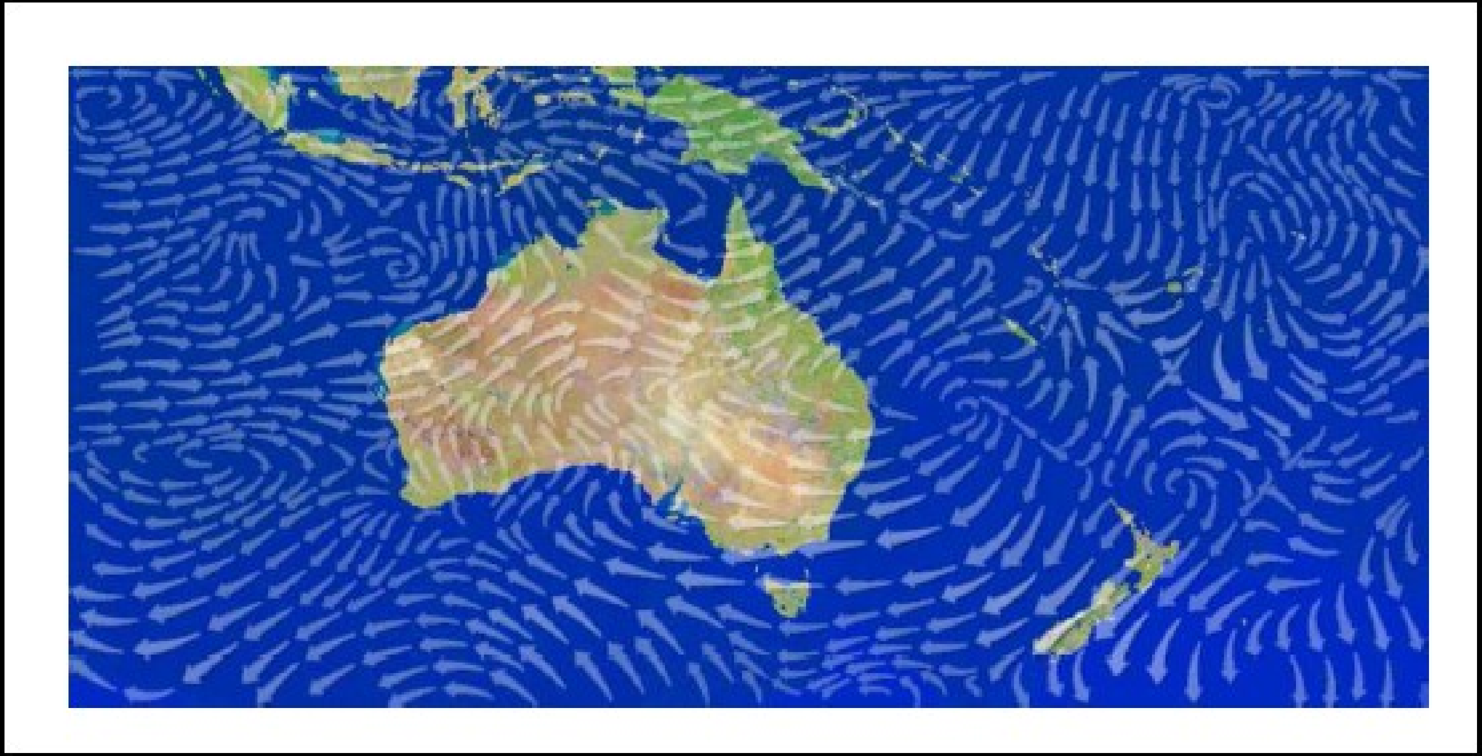
What is NPR?

“A means of creating a work of art that appeals to human perception”

— Carl Marshall

Scientific Visualization

– **Effective visualization of large, multidimensional datasets**



Turk & Banks, “Image-Guided Streamline Placement,” SIGGRAPH 96

Combining Perception and Impressionist Techniques for Nonphotorealistic Rendering of Multidimensional Data

Nonphotorealistic rendering in Sci Viz

- Art and perceptual psychology's inspiration for scientific visualization
  - Art is a natural source for visual inspiration
  - Perceptual psychology attempts to understand how the human visual system sees.

Presentation sequences

Today …

“Visualizing multivalued data from 2D incompressible flows using concepts from painting”

“Line direction matters: an argument for the use of principal directions in 3D line drawing”

Multidimensional visualization

A multidimensional dataset D consists of n sample points, each of which is associated with multiple data attributes.

Establishment of a data-feature mapping that converts the raw data into images

The visualization process should be rapid, accurate and effortless.

Methods

Applying results from human perception to create images that harness the strengths of our low-level visual system

Using artistic techniques from the Impressionist movement to produce painterly renditions that are both beautiful and engaging.

Relations

These definitions provide an effective way to relate the visualization process to a painted image.

- Match many of the painterly styles to visual feature used in visualization
- Data elements in a dataset are analogous to a brush stroke in a painting. Attribute value could be used to select specific value for each style

Perceptual characteristic

The goal of visualization is to explore and analyze the data rapidly, accurately and effortlessly.

Perceptual psychology identifies a limited set of visual features that can detected by low-level visual system rapidly, accurately and effortlessly---preattentives

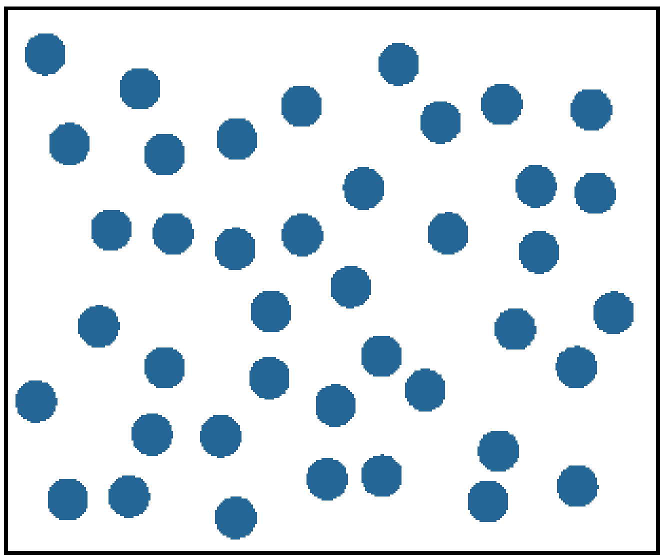
Preattentives

Analysis is rapid and accurate, often requiring no more than 200ms.

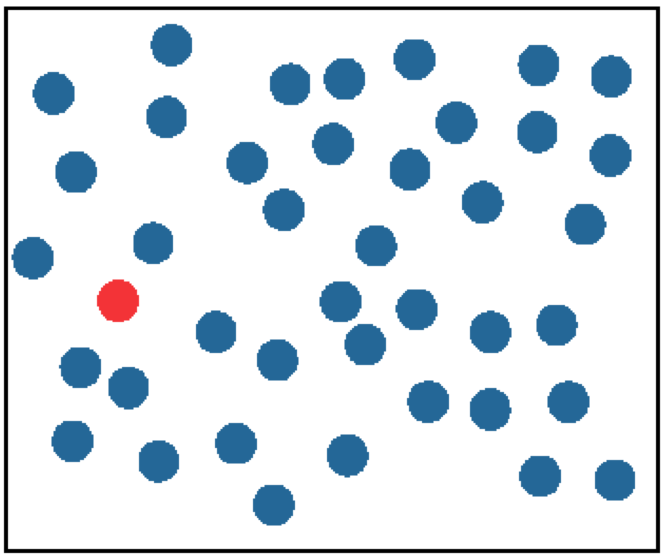
Task completion time is constant and independent of the number of elements in a display

When combining PROPERLY, preattentive features can be used to perform different types of high-speed exploratory analysis of large, multivariated datasets.

Preattentives

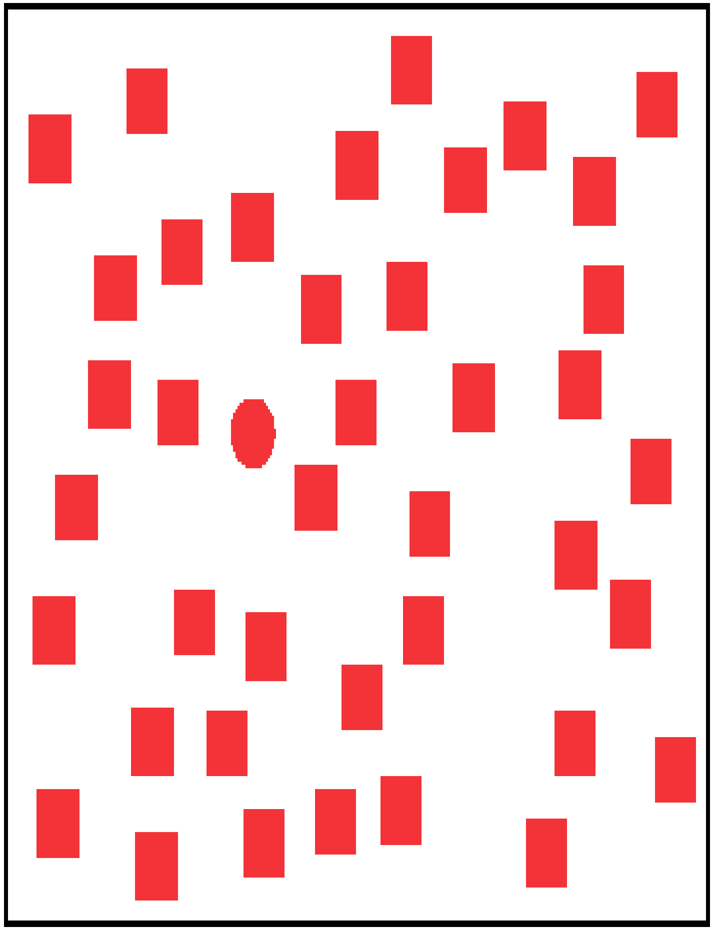


(a)

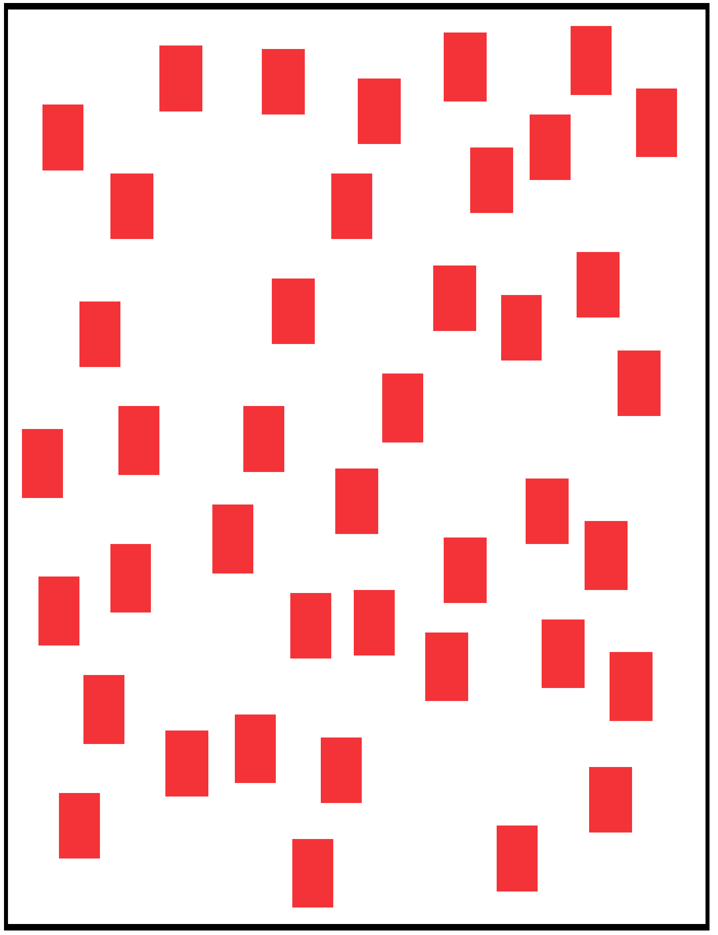


(b)

Preattentives

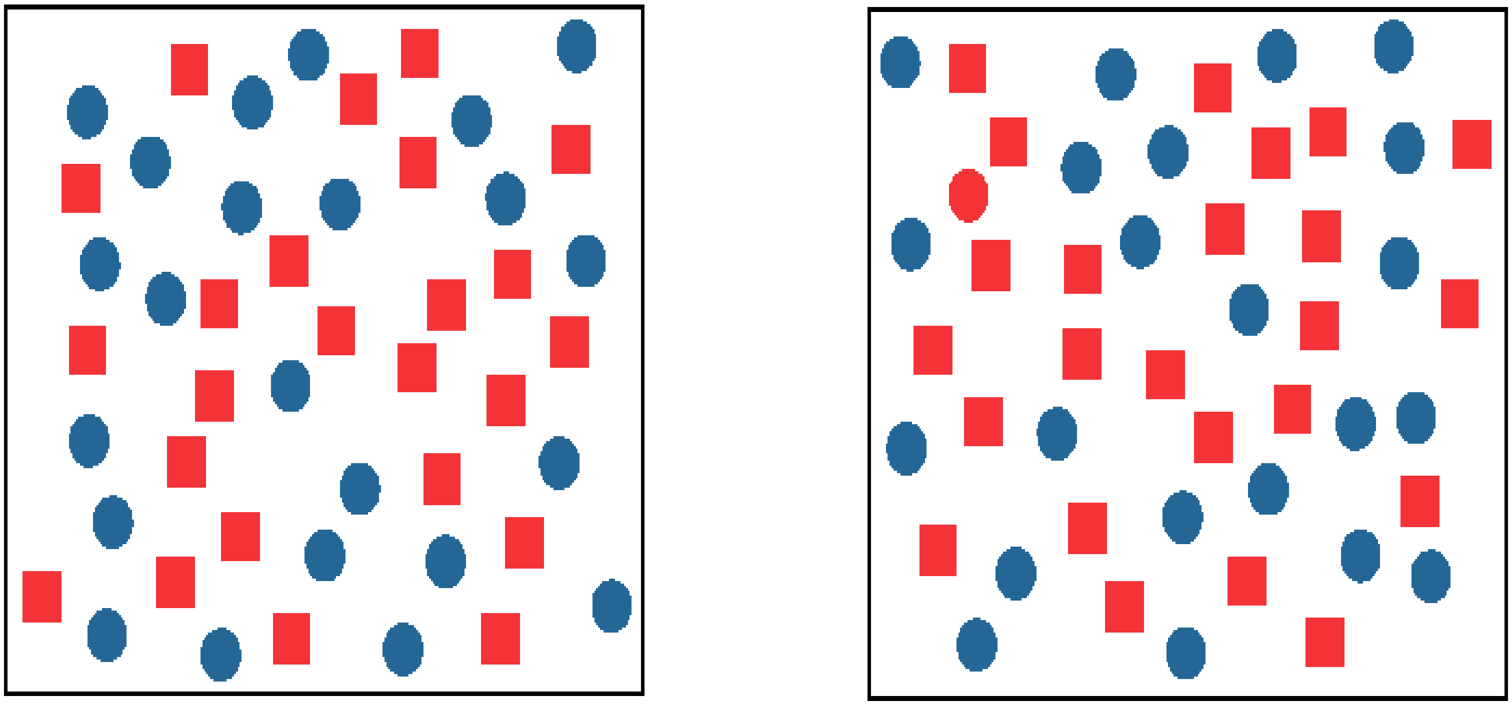


(c)



(d)

Preattentives



(e) (f)

Fig. 2. Examples of target search: (a, b) identifying a red target in a sea of blue distracters is rapid and accurate, target absent in (a), present in (b); (c, d) identifying a red circular target in a sea of red square distracters is rapid and accurate, target present in (c), absent in (d); (e, f) identifying the same red circle target in a combined sea of blue circular distracters and red square distracters is significantly more difficult, target absent in (e), present in (f)

Colors and textures

The paper focuses on the combined use of color and texture.

- Color selection
- Texture selection
- Feature hierarchies

Color selection

A set of colors should be selected such that:

- Any color can be detected preattentively, even in the presence of all the others.
- The colors are equally distinguishable from one another. Every color is equally to identify.

Three criteria

Background research and their experiment prove that three factors should be considered to achieve the goal:

- Color Distance
- Linear Separation
- Color Category

Color Distance

Perceptually balanced color models are often used to measure perceived color difference between pairs of colors.

CLE LUV are used in the paper.

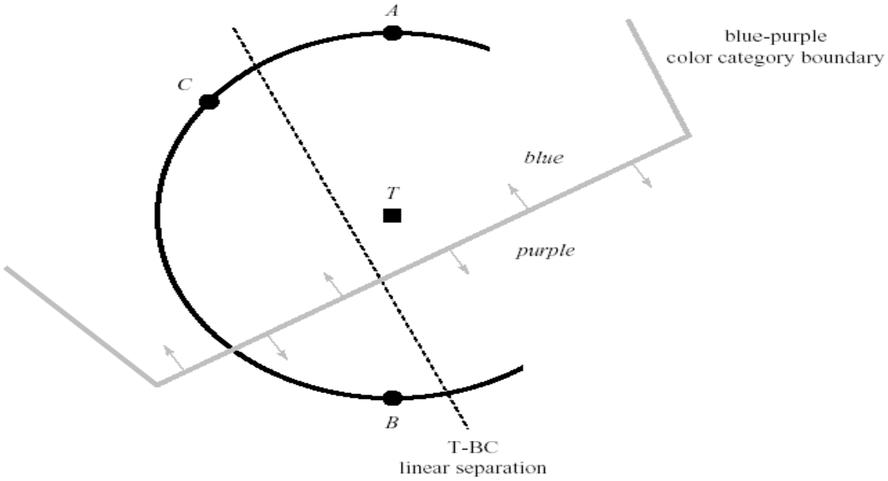
L: luminance UV:chromaticity

The Euclidean distance responds roughly to their perceived color difference.

$$\Delta E^*_{xy} = \sqrt{(\Delta L^*_{xy})^2 + (\Delta u^*_{xy})^2 + (\Delta v^*_{xy})^2} \qquad (5)$$

Linear Separation

Colors that are linearly separable are significantly easier to distinguish from one another.



Color Category

Colors from different named categories have a large perceived color difference.

In Munsell color model, the hue axis is specified using the ten color names R, YR, Y, GY, G, BG, B, PB, P, RP.

One color selection techniques

First the class space is subdivided into r named color regions.

N colors are then selected by choosing n/r colors uniformly spaced along each of the r color region.

Colors are chosen such that color distance and linear separation are constant in each named color region.

Texture selection

Textures can be decomposed into fundamental perceptual dimensions such as regularity, directionality, etc

The paper designed a set of perceptual texture elements, or pexels, that supports the variation of three separate texture dimension: density, regularity, height.

Examples

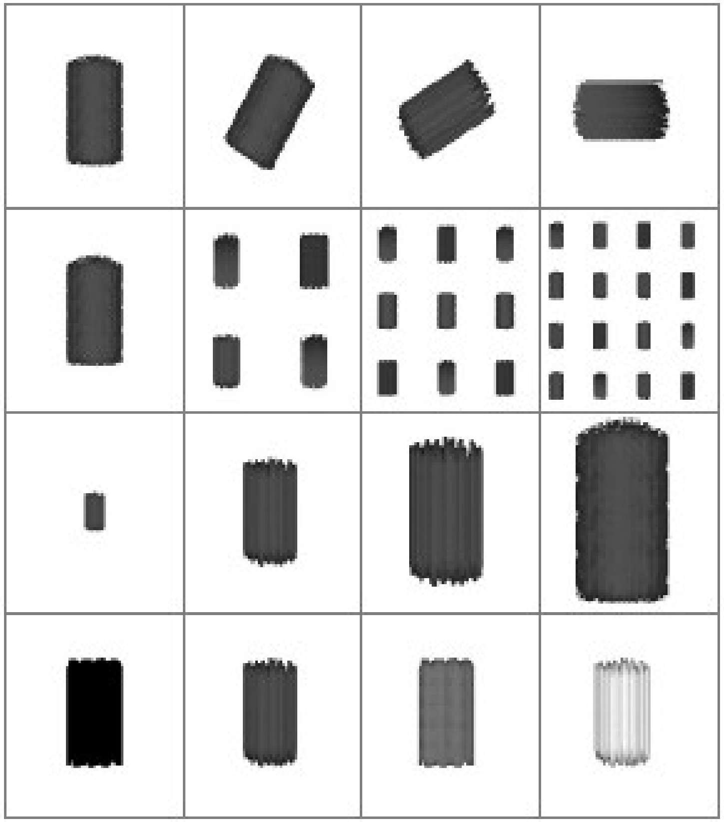


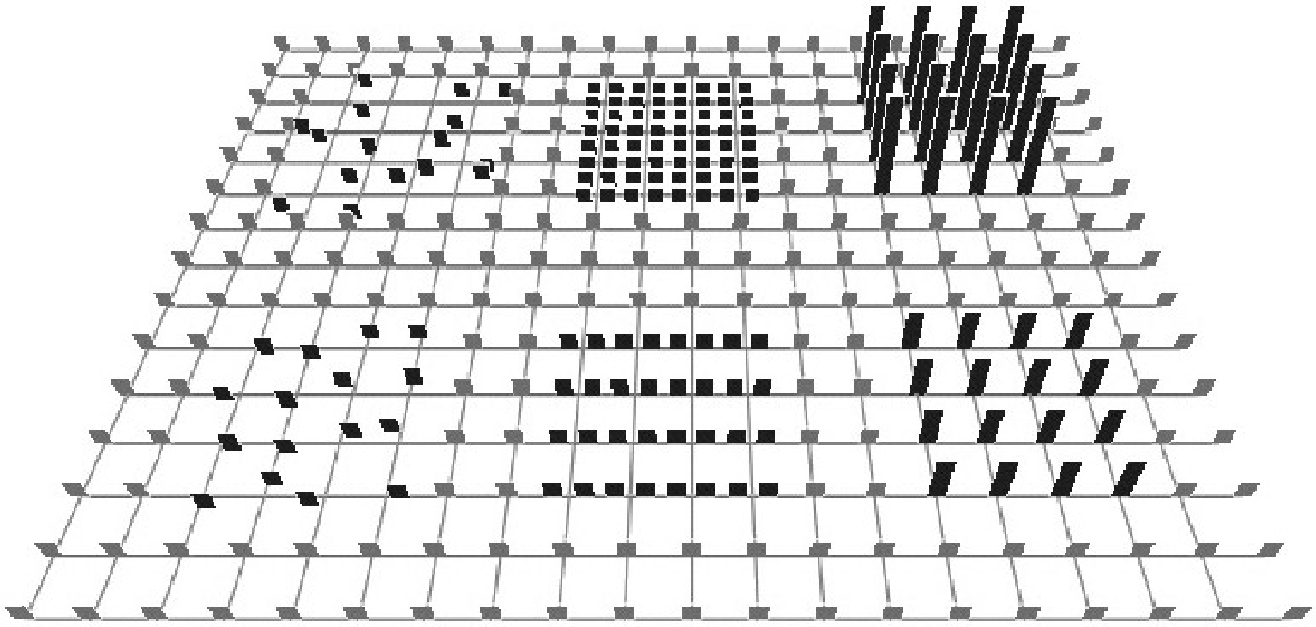
Figure 1: Examples of texture mapped brush strokes with different orientations (top row), densities (second row), sizes (third row), and greyscales (fourth row)

Pexel

Pexels look like a collection of one or more upright paper strips.

The attribute value for a particular element can control the appearance of its pexel, by mapping attributes to density, height and regularity.

Pexel example



(a)

Figure 6: Pexel examples: (a) a background array of short, sparse, regular pexels; the lower and upper groups on the left contain irregular and random pexels, respectively; the lower and upper groups in the center contain dense and very dense pexels; the lower and upper groups to the right contain medium and tall pexels; (b) Color, height, and density used to visualize open-ocean plankton density, ocean current strength, and sea surface temperature, respectively; low to high plankton densities represented with blue, green, brown, red, and purple, stronger currents represented with taller pexels, and warmer temperatures represented with denser pexels

Feature hierarchy

One visual feature may mask another, which causes visual interference.

The ranking of each brush stoke style's perceptual strength is critical for effectively visualization design.

The most important attribute should be displayed using the most salient features.

Low-level visual system hierarchy

A luminance-hue-texture interference pattern.

- Variation is luminance can slow a viewer's ability to identify the presence of individual hues. But not vice-versa.

Texture hierarchy

Experiments show a height-density-regularity pattern.

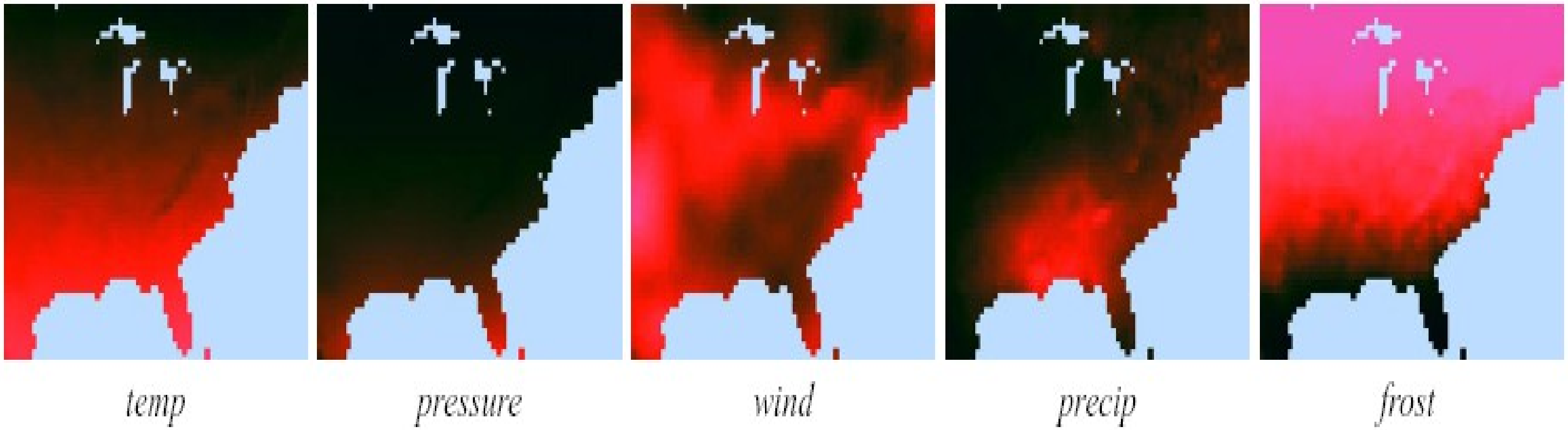
Visualization process

One or more computer generated brush strokes are attached to each data element in the dataset.

The brush stroke has style properties that we can vary to modify its visual appearance.

Data value in the data element are used to select specific states for the different properties.

Visualizing environmental weather data



Feature hierarchy

Color > orientation > density > regularity

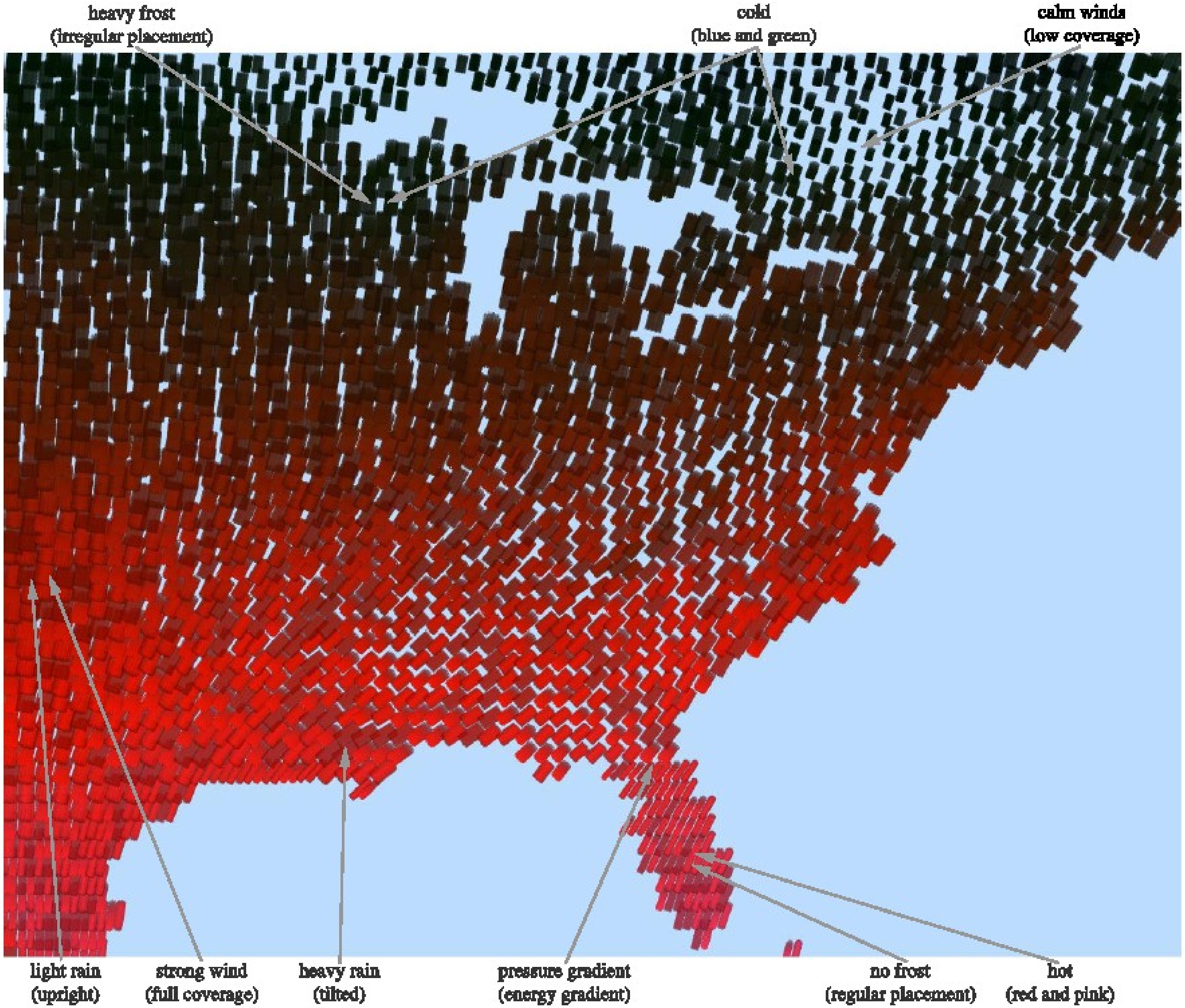
Density is divided into two separate parts:

- Energy: the number of strokes to represent a data element
- Coverage: the percentage of a data element’s screen space covered by its stroke

Mapping

- *temp* → *color:* dark blue for low *temp* to bright pink for high *temp*,
- *wind* → *coverage:* low coverage for weak *wind* to full coverage for strong *wind*,
- *pressure* → *energy:* a single stroke, a 1×2 array of strokes, or a 2×2 array of strokes for low to high *pressure*,
- *precip* → *orientation:* upright (90° rotation) for light *precip* to flat (0° rotation) for heavy *precip*, and
- *frost* → *regularity:* regular for low *frost* frequency to irregular for high *frost* frequency.

Results


heavy frost
(irregular placement)
cold
(blue and green)
calm winds
(low coverage)
light rain
(upright)
strong wind
(full coverage)
heavy rain
(tilted)
pressure gradient
(energy gradient)
no frost
(regular placement)
hot
(red and pink)