Hairy Brushes

Steve Strassmann

Computer Graphics and Animation Group,
MIT Media Laboratory

SIGGRAPH 1986

Presented by: Maria Pace
November 8, 2007
CS 536

Problem: Create a more realistic painting model

- Current painting simulations are oversimplified
- Most basic model is static or simple pattern repeated like a “rubber stamp”
- Airbrush model: simply fills in circular areas of pixels

Problem: Create a more realistic painting model

- Other systems allow for some variation in brush pattern (position, pressure, brush shape), but still do not adequately represent unique strokes
- “Drawing prism” model (Greene): uses image of real brush, but still only a pixel level abstraction

Motivation

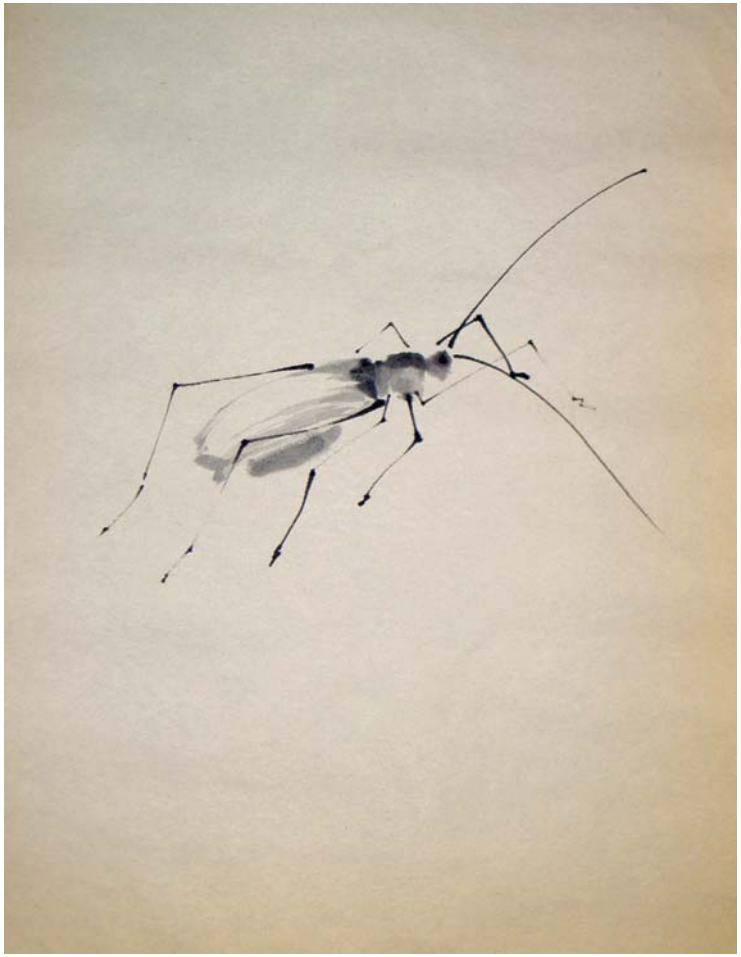
- More realistic model: paint brush as a set of bristles with independent properties
- Bristles can evolve in the process of creating a stroke
- Simulate traditional Japanese art: Sumi-e painting.
- Animated reproducible brushstrokes.

Sumi-e painting

- Sumi-e is a style of traditional Japanese painting.
- Uses minimal brush strokes on a light background, in black and white

Sumi-e painting

- Emphasis is on the quality of each stroke
- Process and materials are varied to create different strokes.



Example of computer-generated sumi-e

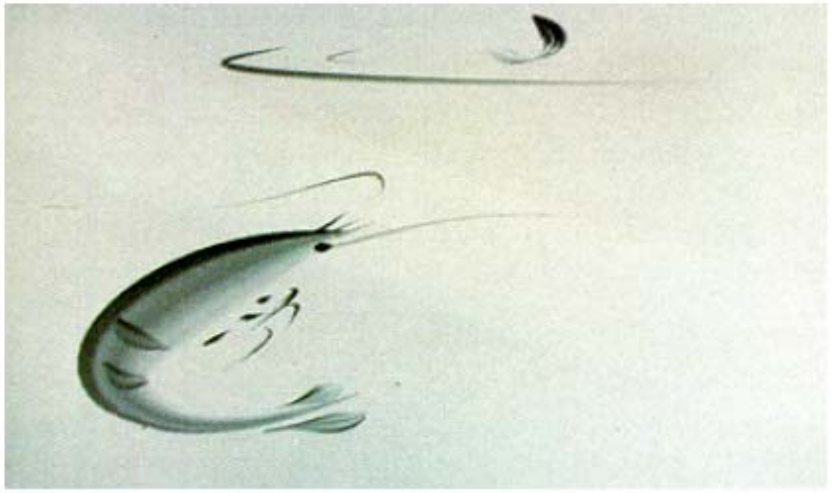


Figure 1: An example of *Sumi-e*: "Shrimp and Leaf"

- 17 strokes total, defined as splines with 3-8 control points
- Drawn interactively using mouse
- 8-bit 640x480 frame buffer

Solution: a modular abstraction for sumi-e brush strokes

- Brush: composed of individual bristles
- Stroke: defined by position and pressure
- Dip: how the paint is applied to the brush/bristles
- Paper: a mapping to the display device

Brush

- One-dimensional array of bristle objects
- Properties:
  - Position relative to handle
  - Ink supply

Brush

- Motion:
  - Perpendicular to path of stroke
  - State changes are periodically recomputed
    - Bristle states updated using coded “rules” to define properties (color, ink quantity, position)
    - Properties change over time, unlike “rubber stamp”
    - Determine bristle’s mark on paper based on pressure, position and ink quantity

Stroke

- Defined by list of pressure and position samples
- Parameters (pressure and position) change as a function of time (or distance)
- Path represented by spline
  - Control points specified by user mouse clicks
  - Pressure values assigned to each point

Dip

- Colors and distribution of ink on brush
- Same brush can be used with different dips to create different effects

Dip

- Repeatability:
  - Ink quantity & position of bristles can change over the course of a stroke
  - Dip stores info about initial state so it can be reused
- Dip parameters:
  - Blotchiness
  - Smoothness
  - Access to brush’s parameters
  - Randomness

Paper

- Renders ink as it comes off the brush
- Brush sends message to paper
  - Position
  - Ink
- Paper renders single dot based on these parameters

Reasons for this representation

- Modular representation:
  - Brush: composed of individual bristles
  - Stroke: defined by position and pressure
  - Dip: how the paint is applied to the brush/bristles
  - Paper: a mapping to the display device

1. Adaptable: can represent wide range of paint/brushes
2. Reusability: stroke can be saved and used with different brush/dip/paper
3. Modular: easy to change level of complexity by turning on/off effects

Implementation

1. Use cubic spline to interpolate series of nodes based on position and pressure samples (stroke)
2. Compute width of stroke at each node as function of pressure
3. Approximate filled areas between nodes by quadrilaterals

Implementation

Figure 2: A stroke defined by 4 control points with intervening nodes generated by a cubic spline. The area covered by the stroke is approximated by quadrilaterals.

- Use position/time to sort the pixels in chronological order
- Use brush to determine nearest bristle to each pixel

Details: Stroke Path

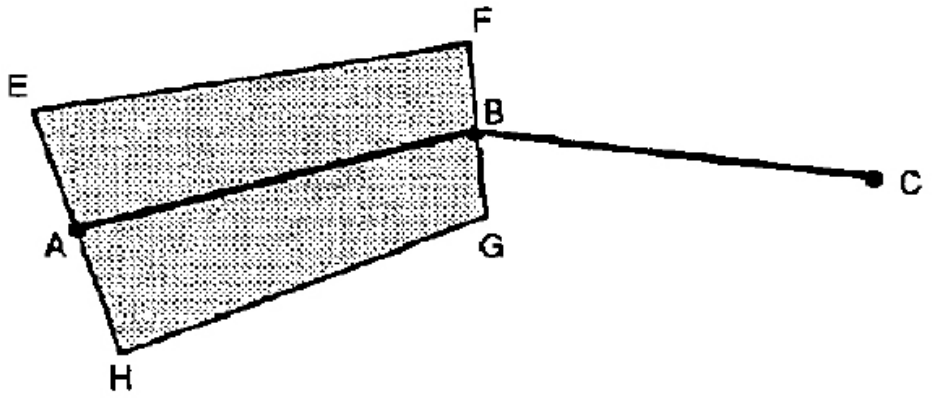
- Represent stroke path as N nodes $(X,Y,P,S)_i$ for i = 0,…,(N – 1)
  - X, Y are position coordinates
  - P is pressure
  - S is distance along the curve, where $S_0 = 0$
- Brush center moves along line segments connected by consecutive nodes using Bresenham line drawing algorithm

Using cubic splines to calculate distance

- User inputs $(x,y,p)_j$ for jth control point
- Calculate distance s_j :

$$s_0 = 0, s_j = \sum_{k=1}^{j} \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}$$

Approximating the quadrilateral



- *A* bisects *EH*
- *B* bisects *FG*
- |*EH*| is the width computed from the pressure at *A*
- |*FG*| is the width computed from the pressure at *B*
- *FG* bisects <*ABC*

Generating the Pixels

- Position (x,y) on frame buffer – generated during interpolation
- Position along stroke (S) – found by interpolating (S_A, S_B, S_B, S_A) on polygon *EFGH*
- Position on brush (B) – found by interpolating (1, 1, 0, 0) on polygon *EFGH*
  - Brush position B is a value between 0 and 1
  - Used to select closest bristle(s)

Anti-aliasing

- Can’t just anti-alias edges of polygons, since since brush could change over time
- Use supersampling:
  - Stroke rendered first on patch of “virtual paper” at higher resolution than frame buffer
  - Patch is then sampled and copied back to frame buffer

Efficiency

- Two main parts of algorithm:
  - Serial: computation of stroke geometry (polygons)
  - Parallel:
    - Bristle uses evolution rules to calculate next state
    - Pixel gets color info from brush
- Parallel part uses 90% - 99% of total computation time on average

Effects: Ink Quantity

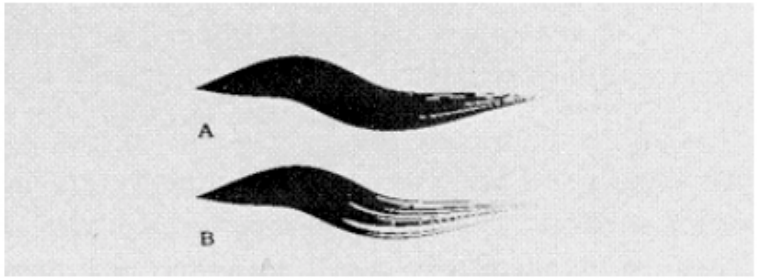


Figure 5: Different quantities: (A) 50% of the bristles are approx. 33% dry. (B) 75% of the bristles are approx. 50% dry.

- Ink supply decreases as brush moves through stroke
- Scratchiness effect : dip puts smaller amount of ink on brush
  - Some percentage of bristles is set to run out of ink early
  - Can determine this qty based on known stroke length for desired effect

Effects: Ink Color

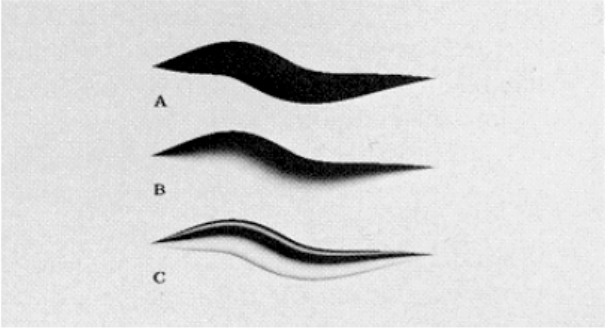
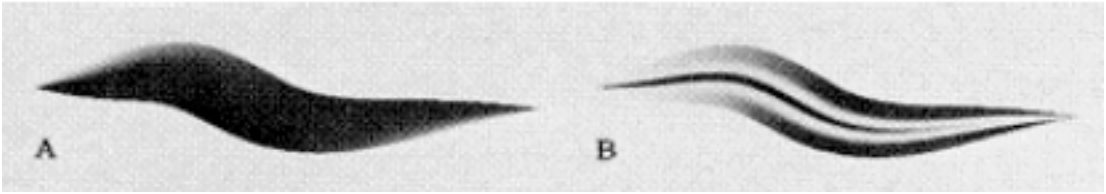


Figure 6: Different colors: (A) Constant (B) Linear (C) User-specified

- Color for each bristle is represented by fraction between 0 and 1: shade of grey
- Distribution on brush can be constant, linear progression, or arbitrary values

Methods for evolving color over course of stroke

- Specify distribution for start and end of stroke
  - For points in middle of stroke, color is interpolated linearly based on start and end value



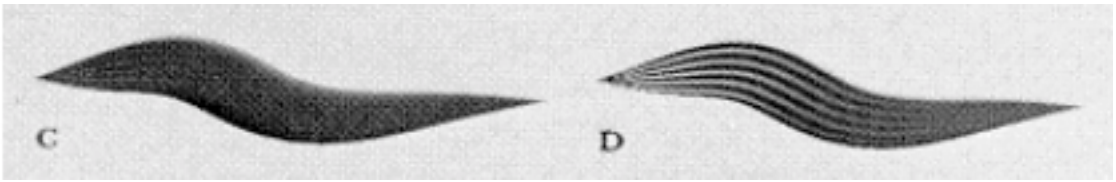
Methods for evolving color

- Diffusion: smoothing colors of neighboring bristles

  – Example: C_{i_t} = color of *i*th bristle at time *t*

  D is speed-of-diffusion parameter between 0 and 1

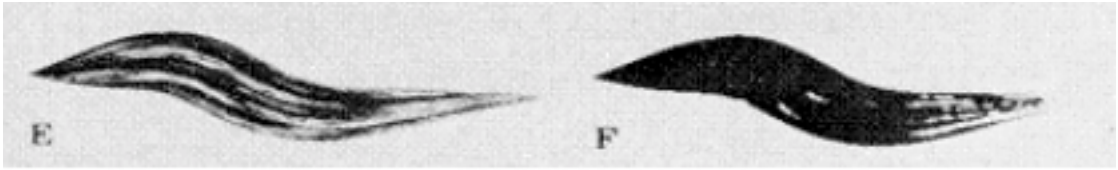
  Then $C_{i_{t+1}} = C_{i_t}(1\text{-}D) + 1/2(C_{i\text{-}1_t} + C_{i+1_t})D$



Left: Fast color diffusion (D = .5) Right: Slow diffusion (D = .1)

Methods for evolving color

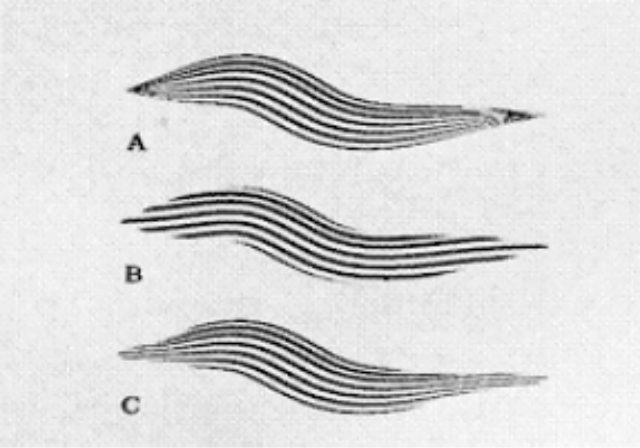
- Generalized evolution algorithm
  - color may be function of pressure, distance, or ink remaining
  - “ink stealing”: transfer of ink between neighboring bristles



Left: random (Brownian) evolution of color Right: “Ink Stealing”

Effects of Pressure

- Spreading: more pressure spreads bristles further apart
- Contact: more pressure brings more bristles in contact with paper



(A) More pressure spreads bristles

(B) More pressure bring more bristles in contact

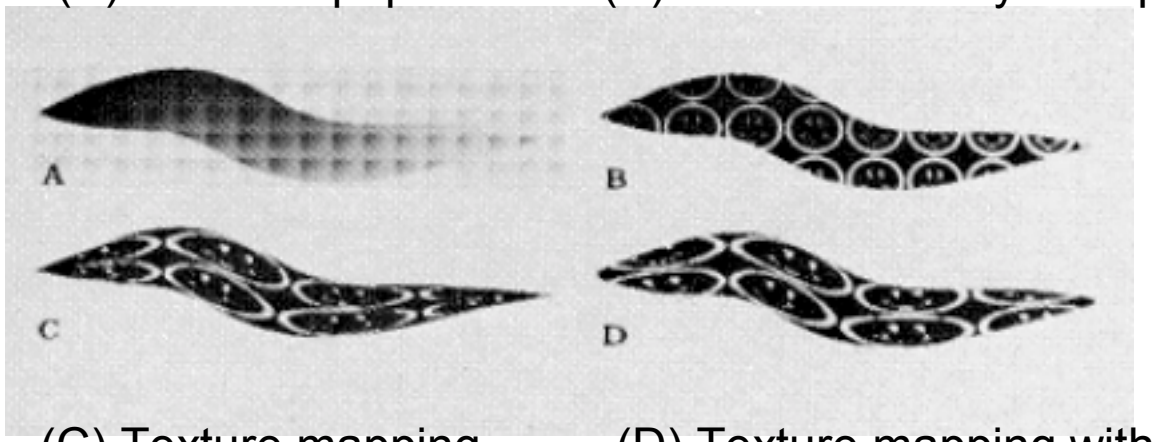
(C) Combination of spreading and contact effects

Texture Mapping

- Two ways to map texture to stroke image:
  1. Flat tiling: Use rectangular array to represent paper's texture.
  2. Map array along long axis of stroke
- Multiply texture array value by ink color (between 0 and 1) to apply texture.

(A) Textured paper

(B) Textured smiley-face paper



(C) Texture mapping with spreading bristles

(D) Texture mapping with pressure threshold bristles

Animation

- Basic animation code
- Two-dimensional keyframing system
  - User specifies key shapes of brush strokes
  - Position and pressure is interpolated between key frames
  - Same brush and dip used throughout

Animation: “Shrimp and Leaf”

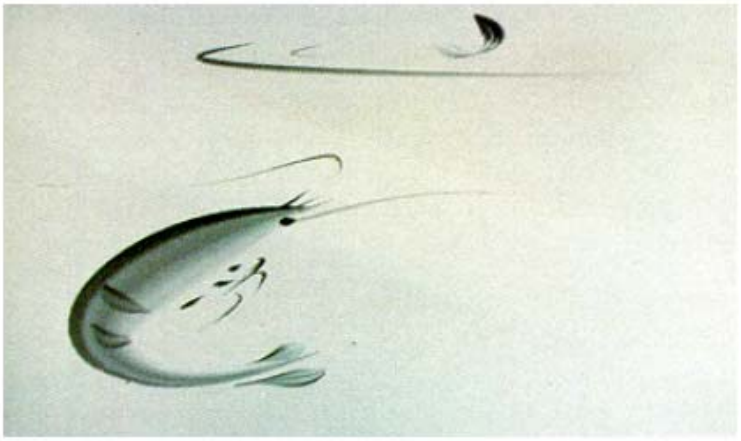


Figure 1: An example of *Sumi-e*: “Shrimp and Leaf”

- 92 frame sequence animated from 4 keyframes
- Each frame took 1 minute to render
- “very lifelike” – antenna and legs move, tail kicks, water ripples

Further Work

- Better input methods
  - Mouse input not as expressive as real brush
  - Explore other input devices, including force-sensitive touch-screens, LED-based body trackers, magnetic pointing devices
- Better rendering hardware
  - Use parallel computers to render strokes faster, possibly in real-time

Further Work

- Expanding Rules
  - Larger library of rules
  - Include rules for more subjective qualities: “blothiness”, “dryness”, “clumpiness”
- Real Color
  - Experiment with full range of colors supported by 24-bit color frame buffer
  - Color change as function of paint thickness, reaction with brush & paper

Further Work

- Paper Effects
  - Specify absorption/wetness of paper and redistribute ink accordingly during anti-aliasing
  - Can use simple asymmetrical fractal to simulate bleeding effects on dry paper
- Splatter
  - Simulate splatter effects based on brush velocity/acceleration
  - Use fractal distribution of scattered dots on paper

Further Work

- Music and Painting
  - Stroke analogous to contour of musical note over time
  - Strokes create recognizable images, while notes create recognizable chords
- 3D Strokes – simulated sculpture
  - Would require 3D input device – moving through air or another medium
  - Rendered with stereoscopic displays or holograms