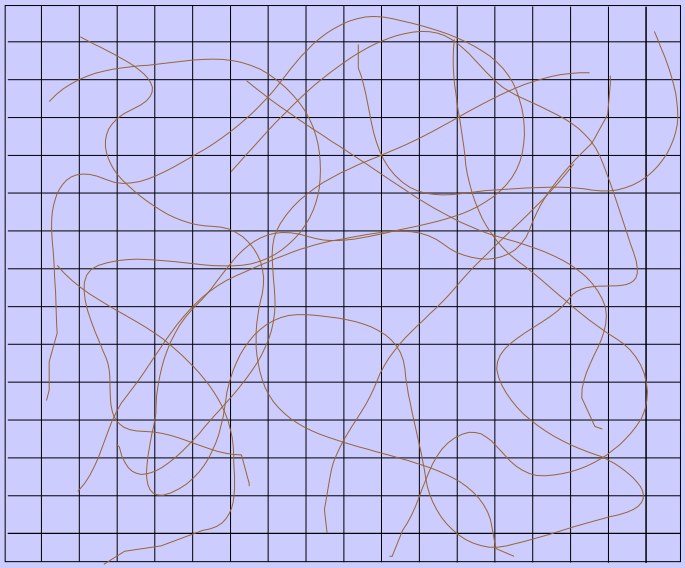
Simulating Chinese Brush Painting

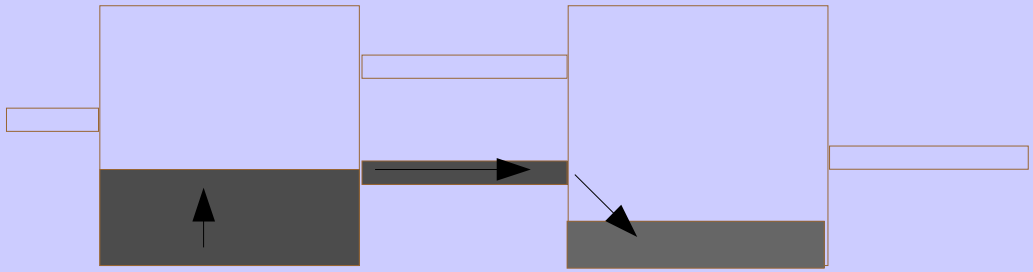
Three Problems

- Modeling ink and water diffusion in paper
- Modeling the soft brush
- Brush stroke input

Ink and Water Diffusion

- The focus of most research
- Can now achieve realistic simulation
- One interesting approach: “Fiber Mesh” + “Tanks and Pipes”





Tank 1 spills over into tank 2

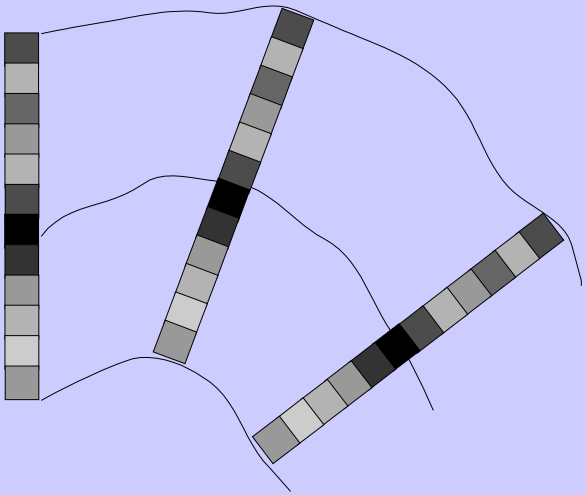
Ink Diffusion is not Enough

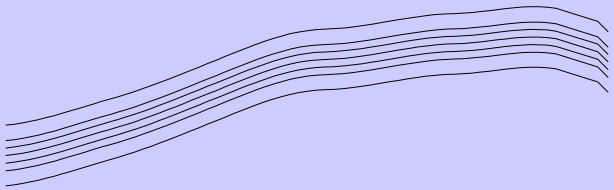
- The brush is largely responsible for the unique look of Chinese brush painting
- Before ink diffusion can be simulated, the ink must first be applied to the paper – we need a brush footprint

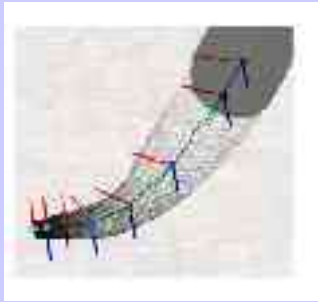
Brush Models

Various brush models have been proposed

- A linear array of bristles (older work)
- Interval splines (a quite different approach)
- 3D spine and mesh with physics (recent work)







3D Brush Models

- The linear array and spline based approaches don't produce very good results
- Recent 3D Brush Models are better, but miss some key characteristics
  - “Flying White”
  - Dry brush hair separation
- The spine + mesh model doesn't provide enough detail for these effects

3D Brush Models Continued

- Why not use an individual hair based model?
  - The physical simulation is too expensive
  - Therefore an optimization is needed…

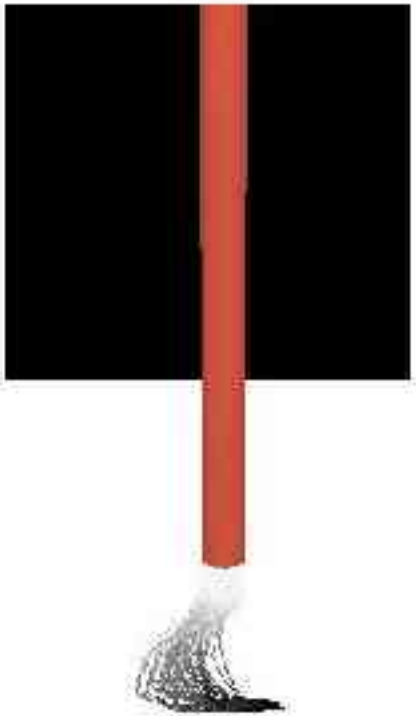
A Geometric Approximation

- If the real physics are too costly to simulate, why not approximate them?
  - Individual hairs are simple enough to approximate geometrically
  - Use intuition to solve the constrained energy minimization problem for one hair

medium friction
medium pressure

medium friction
low pressure

medium friction
brush lifted off paper



high friction
medium pressure

Implementation in OpenGL

- Each bristle is a collection of line segments
  - Detect when each bristle tries to penetrate the paper, then use the geometric approximation to deform it
- The paper is a dynamic texture
  - Render the brush and paper from an orthographic projection viewing down the negative y-axis
  - Use the new orthographic rendering as the texture map for the next frame

Implementation in OpenGL

- Use shadows as a height cue for the brush
  - http://www.opengl.org/developers/code/glut_examples/advanced/projshadow.c
  - findgroundplane(retPlane*, p0*, p1*, p2*)
  - shadowmatrix(retMatrix*, groundplane*, lightpos*)
  - Render ground plane and push the matrix stack
  - Use glMultMatrixf() with the shadow matrix
  - Turn glDisable(GL_DEPTH_TEST)
  - Draw shadow casting objects
  - Pop off the shadow matrix, re-enable GL_DEPTH_TEST, and render the objects again

The End

Except for a small* demonstration

*Small because my old, non-OpenGL accelerated laptop cannot render this in realtime at a decent resolution :`(.