Watercolor Painting with Polygons

Khan Mostafa

khan.mostafa@stonybrook.edu

Graduate Student, Computer Science

Stony Brook University

Based on

Painting with Polygons: A Procedural Watercolor Engine

by

Stephen DiVerdi Aravind Krishnaswamy

Radomı´r Mech Daichi Ito

IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 19, NO. 5, [p. 723-735] MAY 2013

Outline

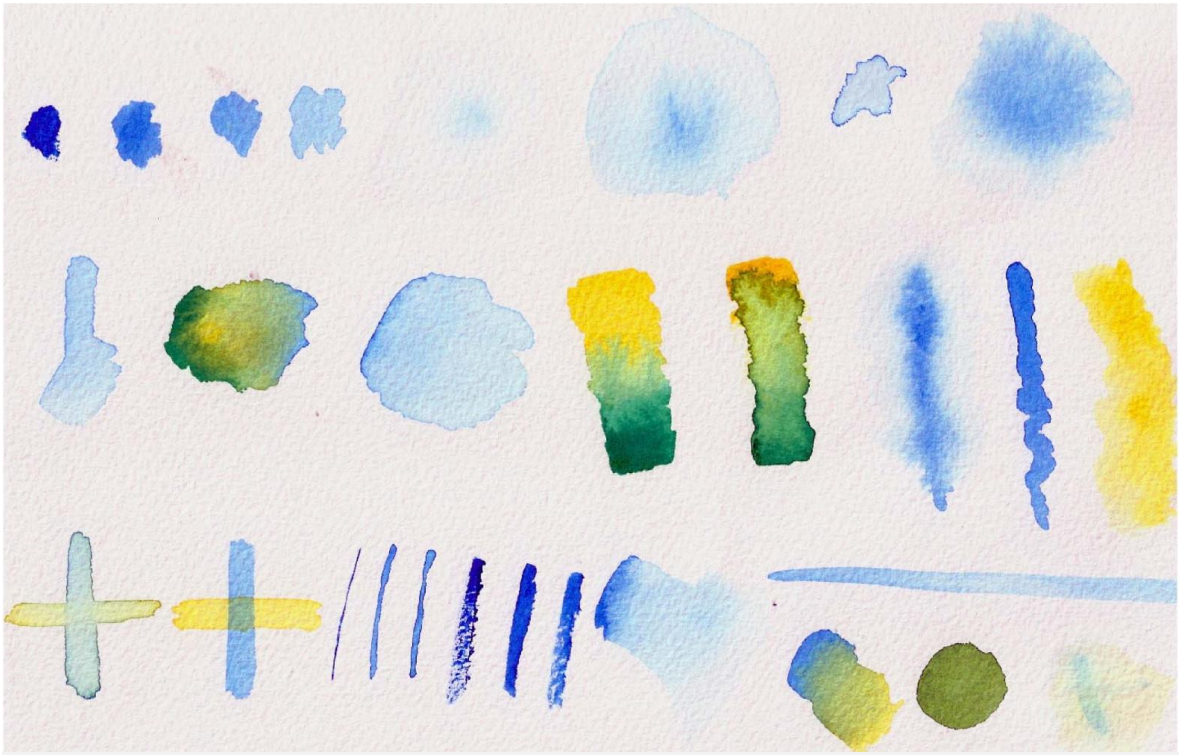
- Watercolor and its features
- Overview
- Project timeline
- Algorithm description
- Application issues
- Wrap up

"Gate in the Paris Ramparts"
A famous watercolor by Van Gogh

Features of Watercolor

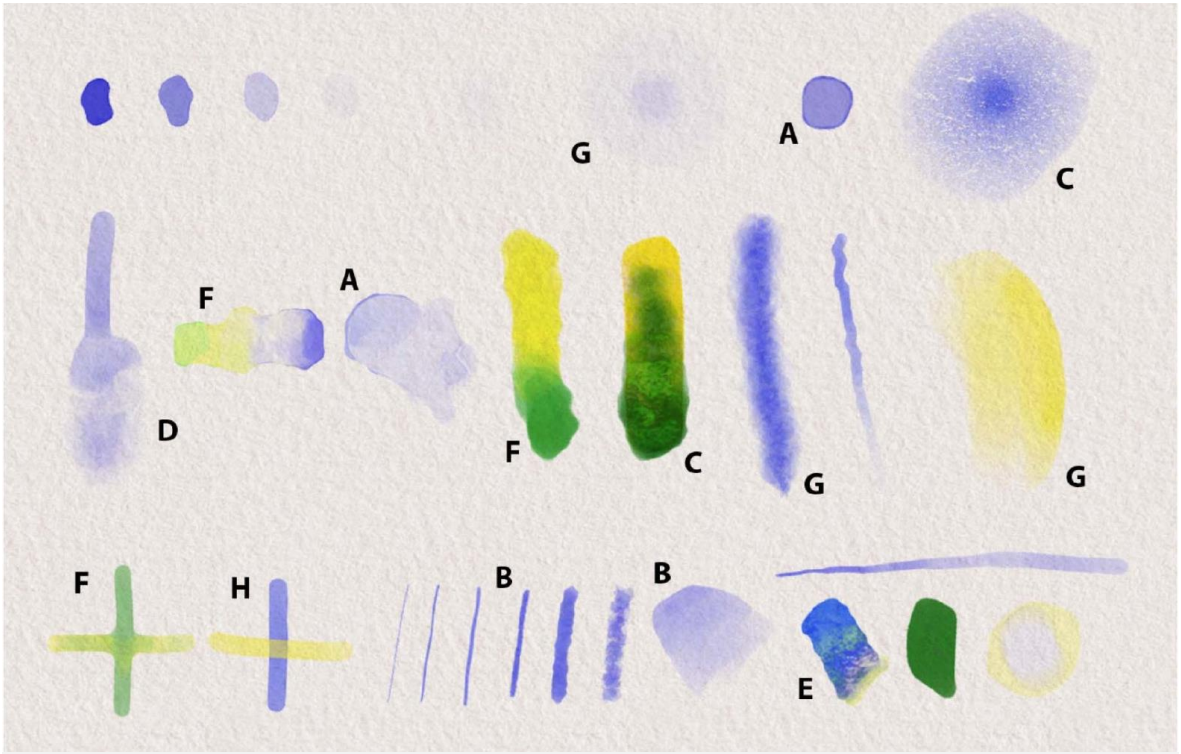
A. edge darkening
B. nonuniform pigment density
C. granulation
D. rewetting
E. back runs
F. color blending
G. feathering
H. glazing

Features of Watercolor



Real watercolor strokes

Features of Watercolor



Watercolor strokes in reference work

A. edge darkening
B. nonuniform pigment density
C. granulation
D. rewetting
E. back runs
F. color blending
G. feathering
H. glazing

Advertisement

	Others	DiVerdi *et al*	Advantage
Pigment flow	Gird based	Particle based	Reduced compute and space cost
Particle representation	Raster	Vector	Render at arbitrary resolution
Particle update		Procedural	Fast to compute

- high-resolution output on low-powered devices
- recreating interactive watercolor paint behaviors
  - edge darkening, nonuniform pigment density, granulation, back runs
  - Variety of brush types

Intuition

- pixel grid-based simulations are dense
- watercolor stroke effects are generally sparse

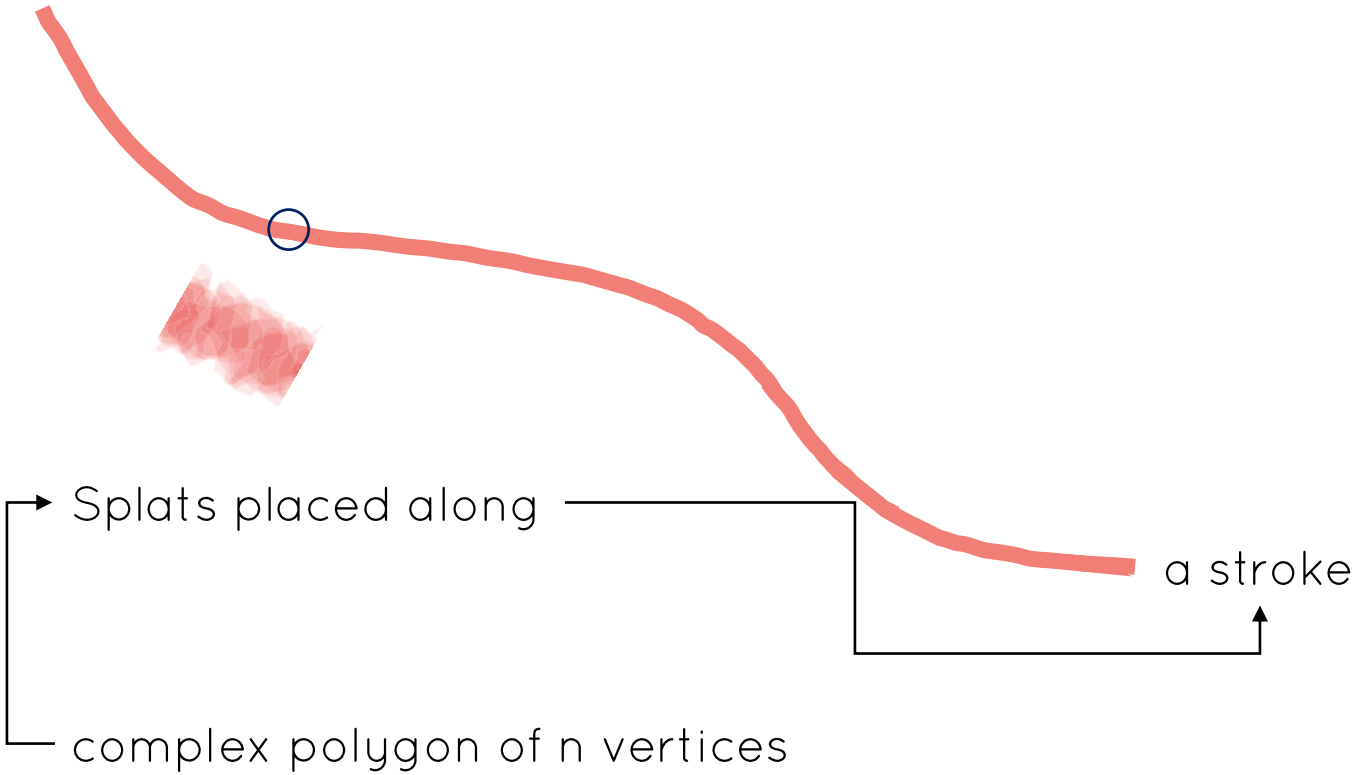
- each particle in real watercolor paint can be thought of as taking a random walk, and the aggregate behavior is that of watercolor paint

- sparse representation for paint pigment
- a random walk algorithm to update each time step

Algorithm

Painting with Polygons: A Procedural Watercolor Engine

Representation/Model



- has own opacity value
- rendered back to front
- standard transparency blending

Algorithm

- Paint Initialization
- Pigment Advection
- Sampling Management
- Lifetime Management
- Brush Types

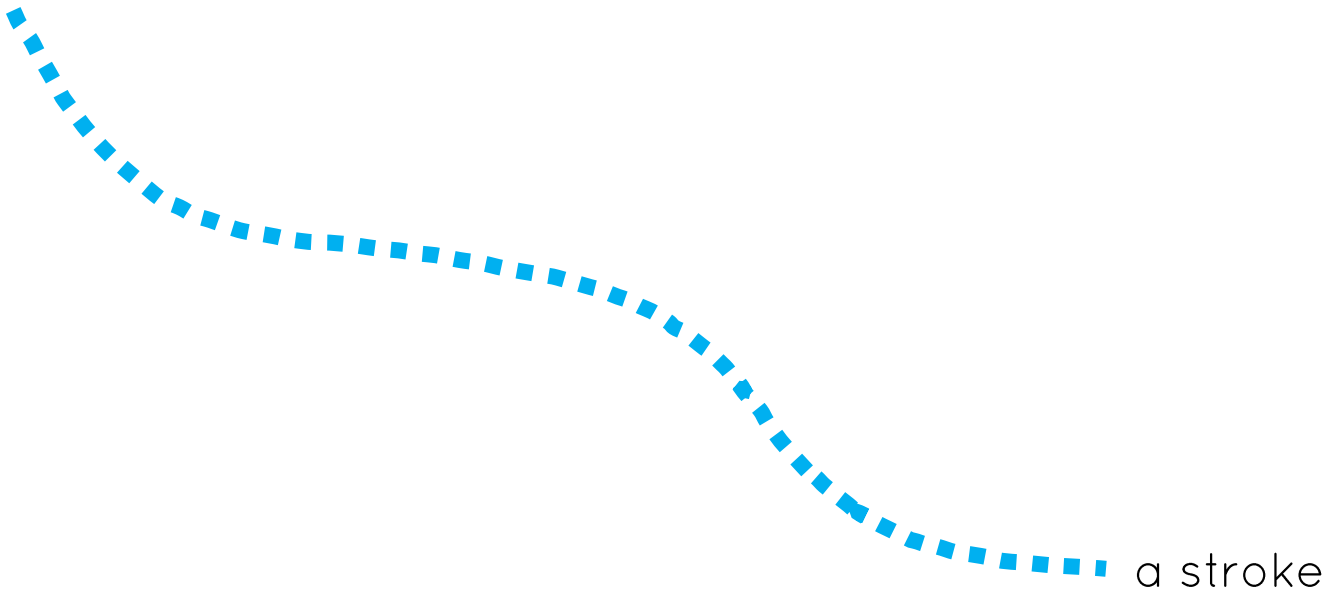
Project Timeline

Week	Dates	Task
1	09/23~09/29	Thorough reading and understanding of the paper
2,3	09/30~10/13	Implementation of *Paint Initialization* and basic user interface
4,5	10/14~10/27	Implementation of *Pigment Advection*
6	10/28~11/03	Preparation for mid-term demo
7	11/04	mid-term system demo
	11/5~11/10	Implementation of *Sampling Management*
8,9	11/11~11/24	Implementation of *Lifetime Management*
10	11/25~12/01	User interface enhancement
11	12/02~12/05	Submission preparation: report + software + presentation slide
	12/06	Submission due

Paint Initialization

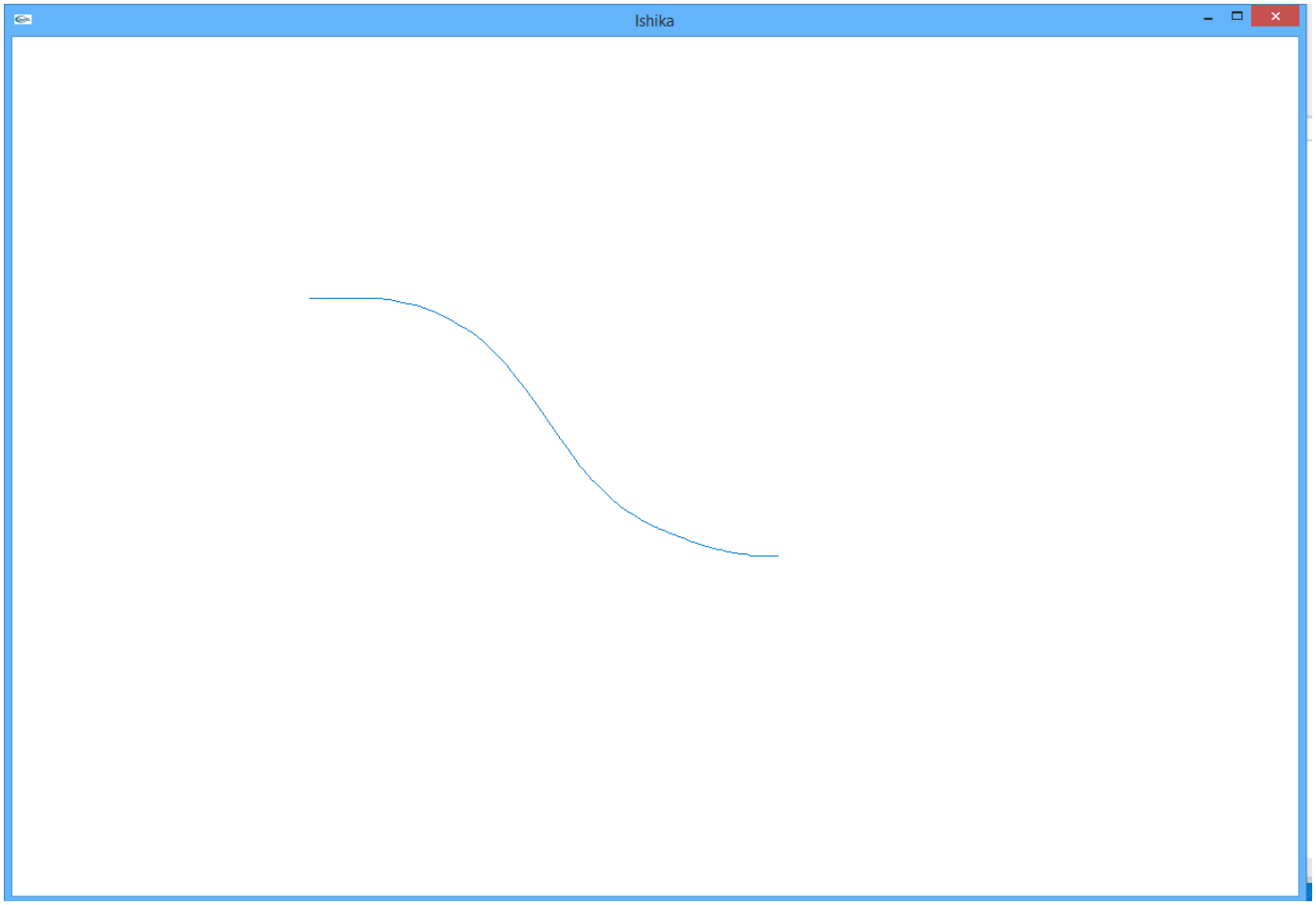
- Stroke Input
- Stroke to Collections of Stamps
- Stamps to Splats
- Splat rendering

Stroke Input

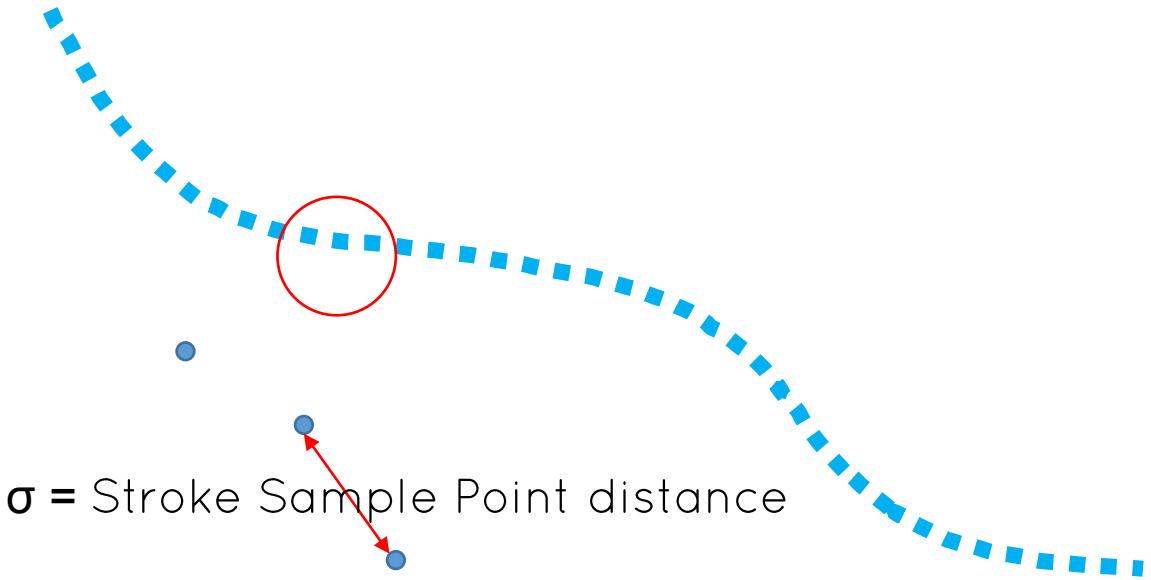



```
regStrokePoint(int x, int y){
        strokePt[strokePtIdx][0]=(float)(x-xmid)/RATIO;
        strokePt[strokePtIdx][1]=(float)(ymid-y)/RATIO;
        strokeCol[strokePtIdx] = currentCol;
```

Stroke Input



Stroke to Stamps



σ = Stroke Sample Point distance

If (**σ** < stamp distance threshold)
advance a min distance
Else
interpolate

Splats

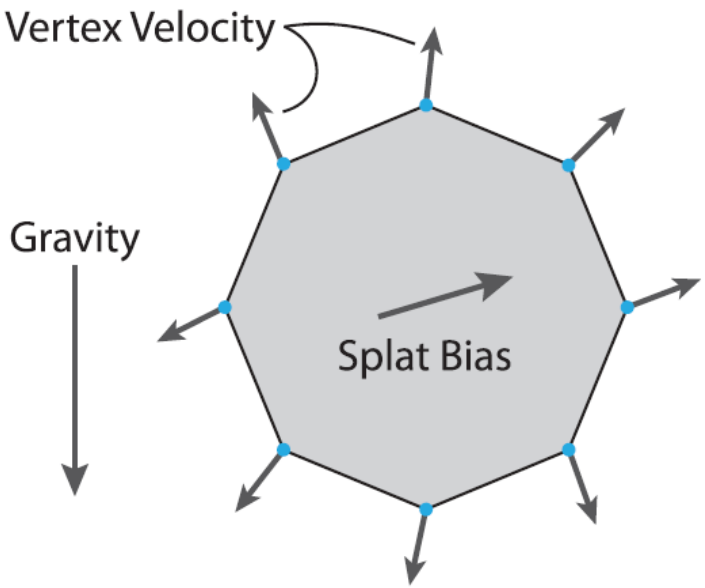
- Stamp = set of splats
- Splat = polygon with n vertices
  - motion bias vector b
  - per vertex velocity vector v
  - age a (in steps)
  - brush parameters
    - roughness r (in pixels)
    - flow f (percentage)
- Wet Map
  - Water added to canvas (as grid, screen res.)
  - Grids dry gradually in each time step

Splats

```
struct SplatParamSt{
GLushort bx; //motion bias x
GLushort by; //motion bias y
GLushort a;//age
GLubyte  r;//roughness [1~255px]
GLubyte  f;//flow percentage [0-100]
GLubyte  o;//opacity
}SplatParam[SPLATS];

GLfloat SplatVertex[SPLATS][N][DIM];
GLint SplactColor[SPLATS];
GLushort WetMap[WIDTH][HEIGHT];
```

Splats



Initial splat

Pigment Advection

$$\mathbf{d} = (1-\alpha)\mathbf{b} + \alpha \frac{1}{\mathbf{U}(1,1+r)}\mathbf{v}$$

Tuning parameter

Non-zero random <= 1+r

$$\mathbf{x}^* = \mathbf{x_t} + f\mathbf{d} + \mathbf{g} + \mathbf{U}(-r,r)$$

Global Gravity vector

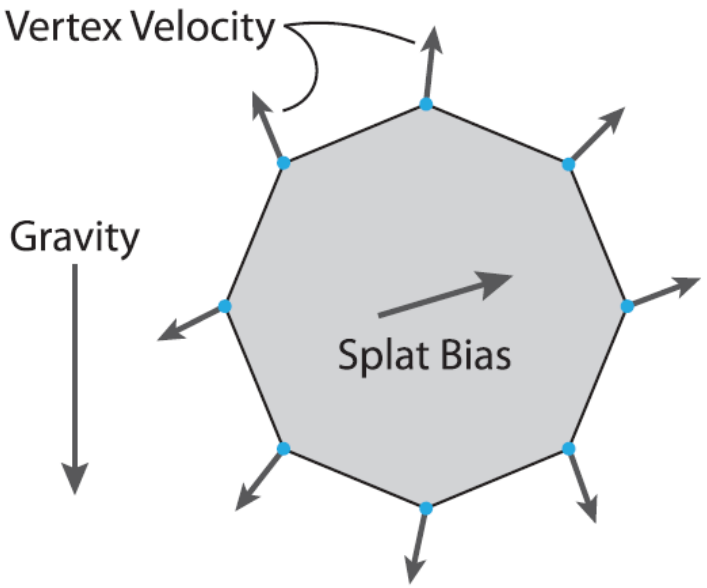
Flow percentage

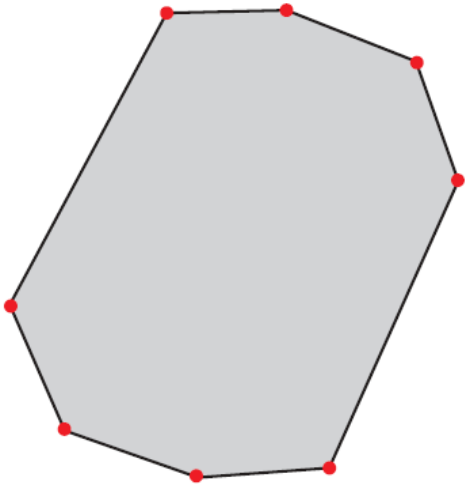
$$\mathbf{x_{t+1}} = \begin{cases} \mathbf{x}^* & \text{if } w(\mathbf{x}^*) > 0 \\ \mathbf{x_t} & \text{otherwise,} \end{cases}$$

Update vertex to new position if new position is wet

Opacity recomputed to conserve amount of pigment

Pigment Advection on a Splat



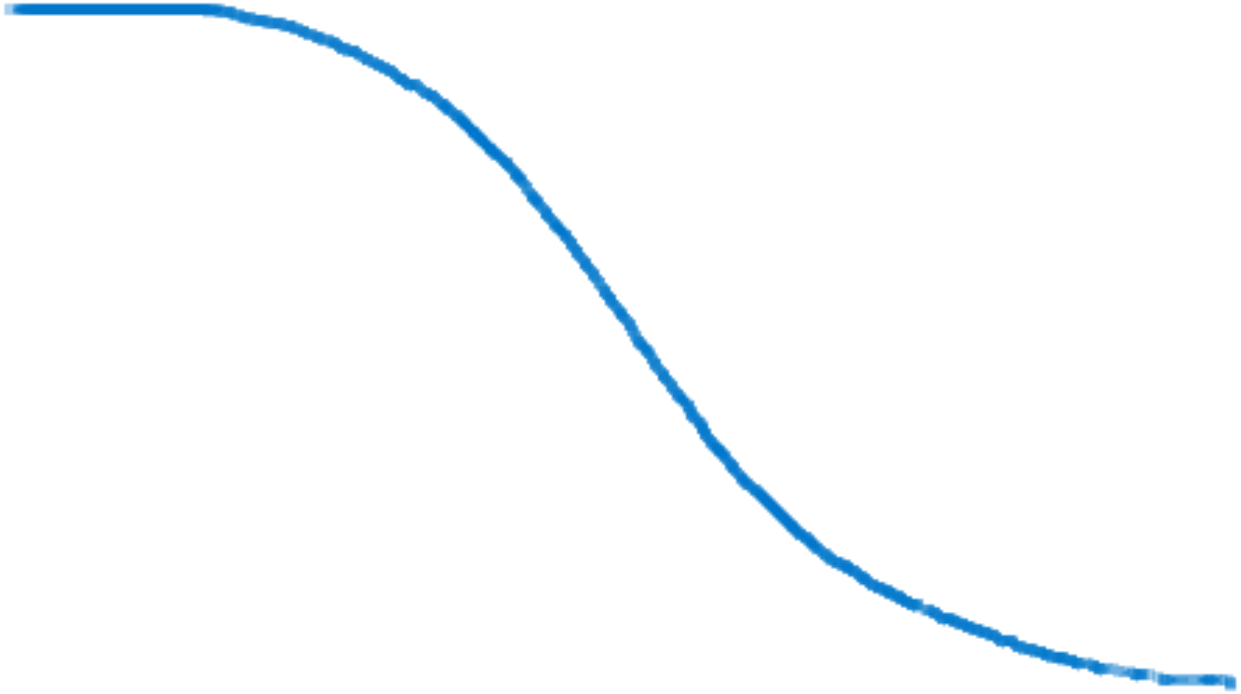


After advection

Pigment Advection

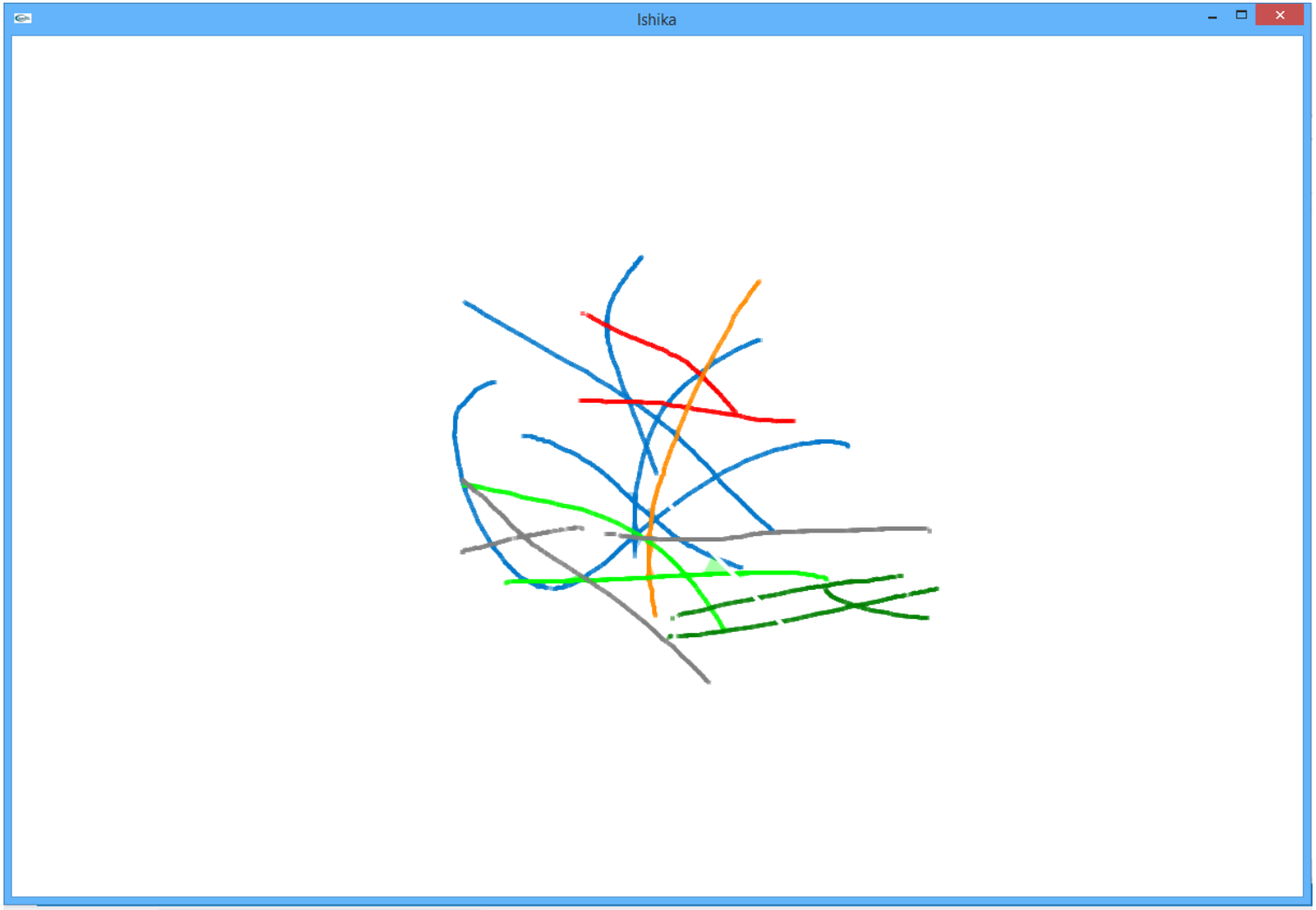
- Simulate biased random walk behavior of real water color
- Random parameter provides randomness

Stroke rendered as advected splats



(current implementation)

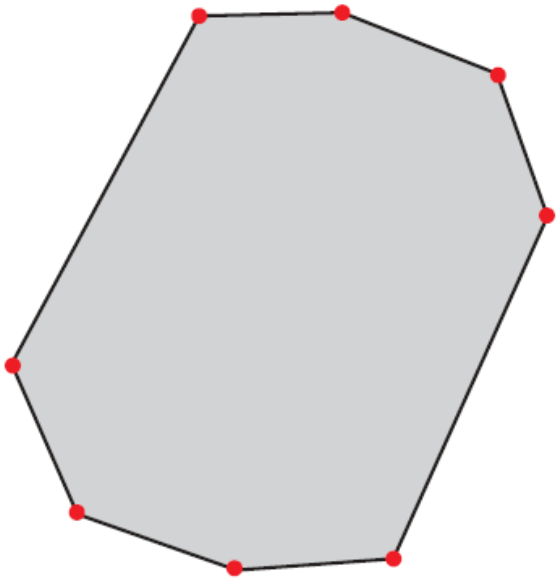
Several Strokes



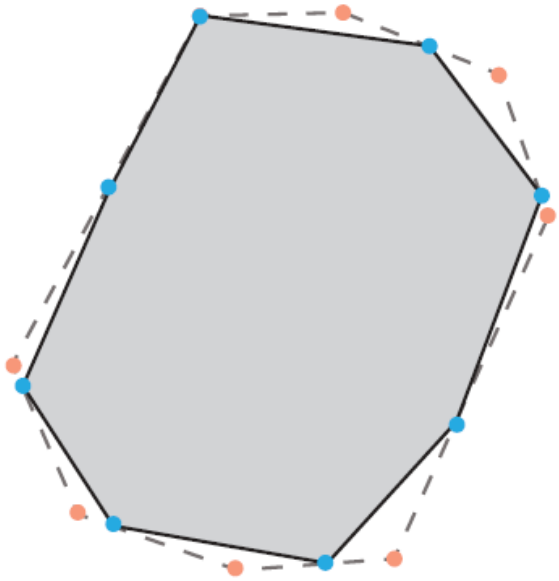
Sampling Management

- different advection directions
  - may create unrealistically straight hard edges
    - due to local undersampling
- Deal this artifact:
  - Constrained vertex motion
    - Vertex can not move too far from its neighbor vertices of same splat
  - Periodic resampling of splat boundary
    - Compute total perimeter
    - Arc length per vertex
    - Vertices moved to uniform arc length

Sampling Management



After advection



Boundary resampling
for uniform arc length

Sampling Management

- Smoothing of splat boundary
- Limits polygonal artifact
- Constrained motion is fast but limits paint propagation
- Boundary resampling is slower but achieve higher quality

Lifetime Management

Three stage of splat life

- Flowing
  - When first added
- Fixed
  - After a steps – splat stops being advected
  - Can be potentially rewetted to resume advection
- Dried
  - After a period of not moving, permanently sainted
  - Rasterized into dry pigment buffer
  - Removed from simulation (reduce cost)

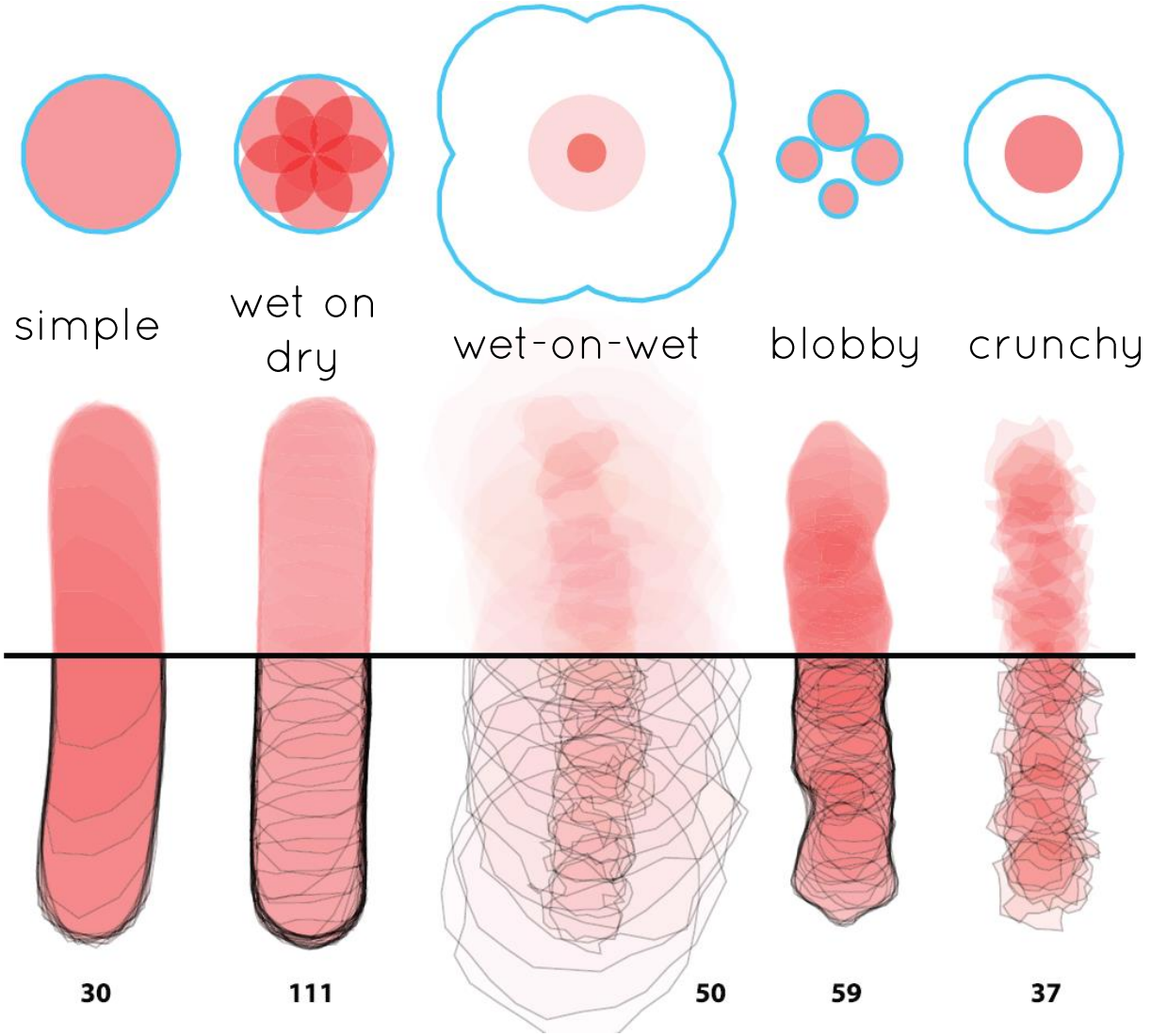
Lifetime Management

- Water addition may rewet fixed splats
  - Simulates back runs and feathered edges
  - Rewetted vertex motion is set by water
- As a splat dries granulation texture is applied

Brush types

- Different arrangement of splat per stamp
- Different brush parameter settings
  - Target width, w
  - Initial wet at wet map (=255)
  - Splat life (I= initial a)
  - Roughness, r
  - Flow, f

Sample Brushes



Initial splat configurations and resulting stroke for each brush type.

Cyan outlines indicate the water region per-stamp. Black outlines indicate final splat shapes.

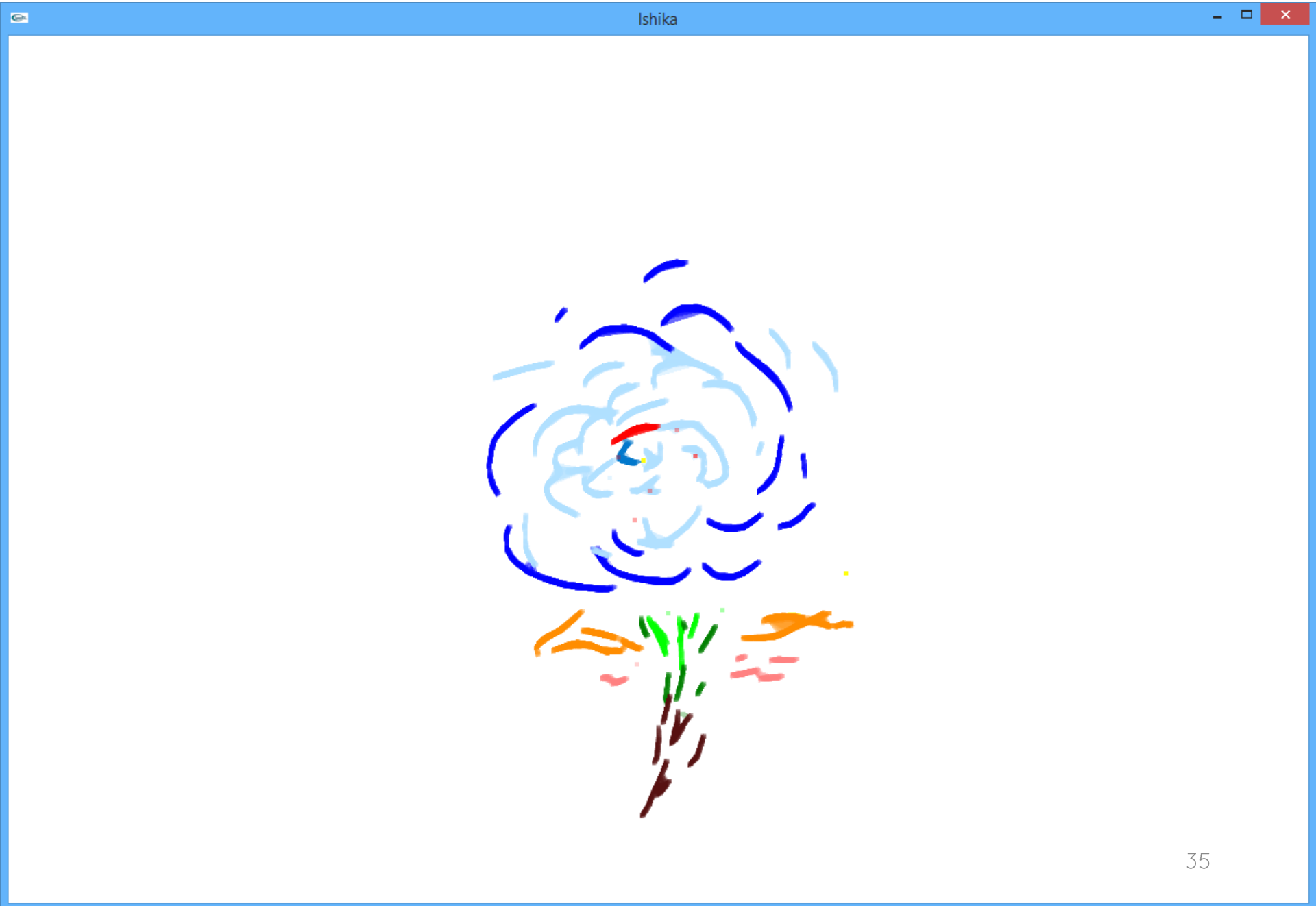
Total number of splats in each stroke is also indicated

Sample Brushes

- Simple
  - Single splat, d=w, b = ⟨0,0⟩
- Wet on dry
  - 7 splats, central splat bias = ⟨0,0⟩, d=w/2
  - Perimeter splat bias = $\langle \frac{d}{2}\cos\theta, \frac{d}{2}\sin\theta \rangle$
- Wet on wet
  - Small splat (d=w/2) inside large splat (3w/2)
  - r=5, l=15, b=⟨0,0⟩
- Blobby
  - Randomly sized 4 splats, l=15, b=⟨0,0⟩
- Crunchy
  - 1 splat, r=5, f=25, l=15

Application

Painting with Polygons: A Procedural Watercolor Engine


Ishika

Interactive rendering

- Two pass stencil buffer per splat
- Anti aliasing
  - Full screen can be expensive
  - Post processing filter: adaptive per pixel blur
- Darkened wet map

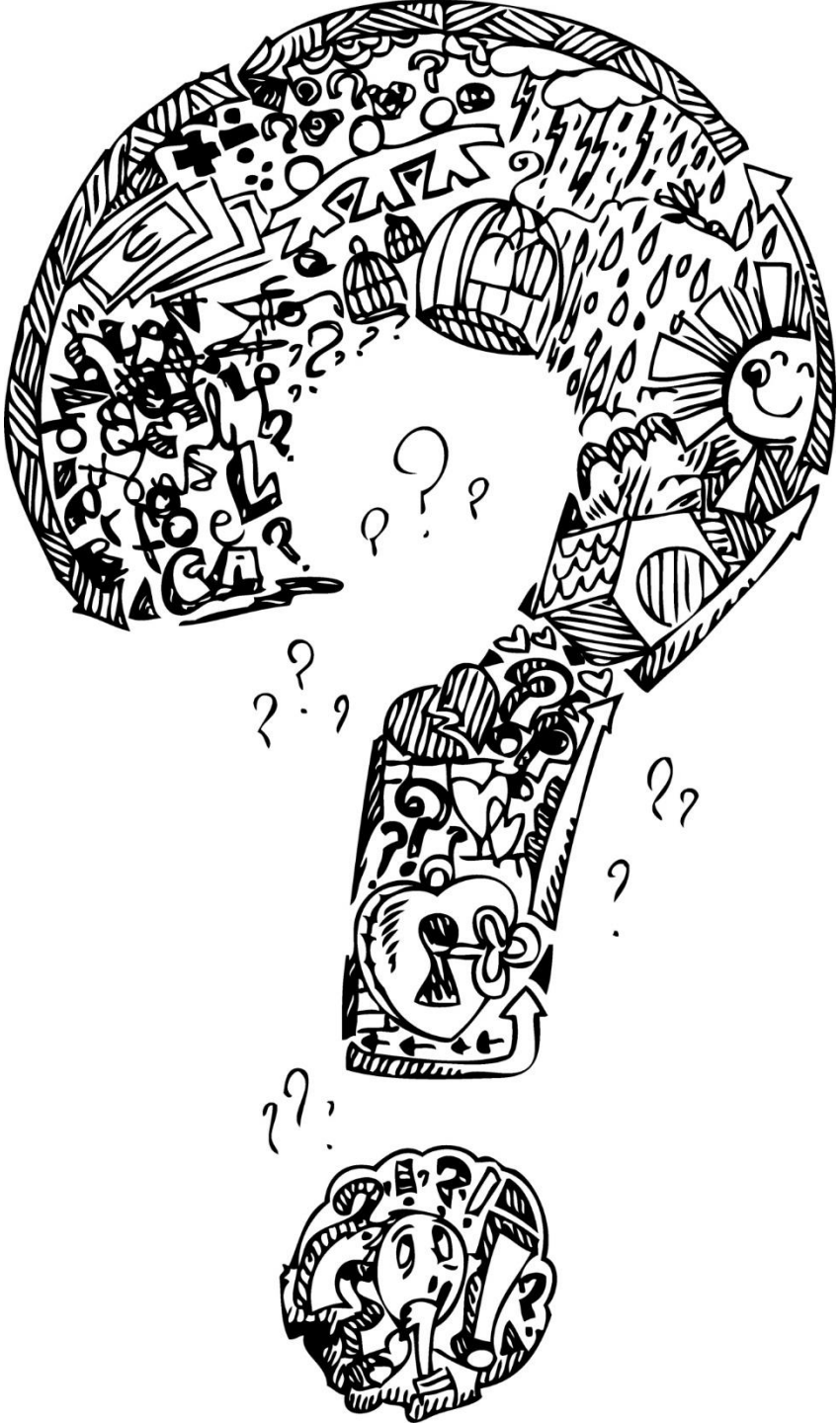
User Interface

- Mid term demo interface has naïve interface
  - Can chose colors
- Submission demo plans to include
  - Color palette
  - Brush selection
  - Brush sizes
  - Save and load drawn images

Comparison

Real watercolor drawing

Using reference app




thank
you