

Supplementary material to the paper:

MoXi: Real-Time Ink Dispersion in Absorbent Paper

GPU Implementation Details and Issues

Data Packing

All the data fields (like water density, velocity) needed for the ink simulation are stored as RGBA textures. For the LBE flow simulation, we use 4 textures listed in Table 1. We call these *simulation textures*.

Texture	Contents	Symbol Descriptions	
		Symbol	Description
1. VelDen	[u, v, wf, seep]	u, v	Water velocity
		wf	Water density in flow layer
		lwf	Water density in flow layer in the last iteration
		ws	Water amount on surface layer
2. Misc	[blk, f0, lwf, ws]	seep	Amount of water seeping from surface layer to flow layer
		blk	Blocking factor
3. Dist1	f[N, E, W, S]	f0	Distribution function for stationary particles
		f[N, E, W, S]	Dist. functions towards nearest neighbors
4. Dist2	f[NE, SE, NW, SW]	f[NE, SE, NW, SW]	Dist. functions towards next nearest neighbors

Table 1. Texture data packing.

Texture Updates

Each of the above simulation textures is updated by rendering the paper geometry to a pixel buffer using a fragment program that perform the needed LBE operations. The content of the pixel buffer is then copied to the destination texture. The six texture updates for the LBE simulation are listed in Table 2.

Texture Update	Output Texture	Operations
1.	Misc	Derive f0, blk Deposit or update ws Save wf to lwf
2.	Dist1	Collide f[N, E, W, S]
3.	Dist2	Collide f[NE, SE, NW, SW]
4.	Dist1	Stream f[N, E, W, S]
5.	Dist2	Stream f[NE, SE, NW, SW]
6.	VelDen	Derive u, v, wf, seep Receive water from surface

Table 2. Texture updates for the LBE simulation.

For ink deposition, we apply fragment programs that perform the necessary operations on the brush tuft geometry. During the deposition, a small part of the brush tuft penetrates the virtual paper, giving the

brush footprint [Chu and Tai 2004]. The brush tuft geometry is stretched as shown in Figure 1 [Wloka and Zeleznik 1996] to give a continuous stroke (rather than instances of brush geometry at discrete times given by the brush dynamics simulator). Both the penetrating part of the stretched tuft and the paper geometry are rendered to the pixel buffer simultaneously so that deposition does not need extra rendering passes.

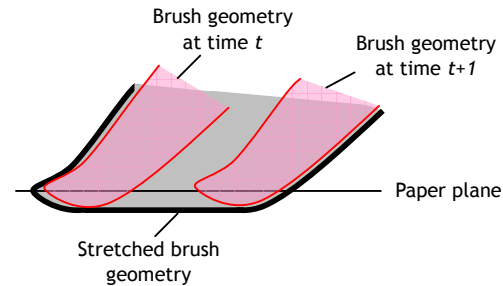


Figure 1. Stretching brush geometry for deposition.

Hardware Precision

We use the 16-bit float data type for our simulation textures, but perform the computation in the fragment programs with 32-bit float precision. There are two reasons for using 16-bit float texture instead of 32-bit: (1) it is currently the only float type with hardware bilinear texture interpolation, and (2) it lowers the memory consumption and the overheads in copying data.

We encountered a few precision issues when using the 16-bit float textures. The first issue lies in the transfer of pigments from the flow layer to the fixture layer (Section 5.4.3). We found that pigments are lost during the transfer, with the destination texture not getting the correct amount. One likely cause is the hardware round-off/truncation procedure for the 16-bit float texture writes. We solve the issue by manually quantizing the transferred pigment amount with a quantization step of 1/2048 (determined empirically).

Another issue is in the hardware interpolation of 16-bit float textures as needed in the generation of high quality output (Section 6.1). Some artifacts appear in the generated higher resolution output I' : the pixels in I' that are supposed to be interpolated from the pixels in I were lighter than what they should be. We suspect this is caused by the hardware interpolation giving values a bit smaller than they should be. These small errors would not be noticeable if the interpolation is done only once, but become visible when accumulated. We solve this problem by reducing the number of updates to I' . In our implementation, we only update I' every 200-400 time steps.

References

- CHU, N. S., AND TAI, C.-L., 2004. Real-Time Painting with an Expressive Virtual Chinese Brush, *IEEE Computer Graphics and Applications* 24, 5, 76-85.
- WLOKA, M. M. AND ZELEZNIK, R. C., 1996. Interactive real-time motion blur, *The Visual Computer* 12, 6, 283-295.