http://dx.doi.org/10.5392/IJoC.2012.8.4.036

Efficient Watercolor Painting on Mobile Devices

Junkyu Oh
Raysoft, Korea

SeungRol Maeng
Kongju National University, Korea

Jinho Park
Multimedia Department, Namseoul University, Korea

ABSTRACT

We present a novel watercolor painting drawing system which can work even on low powered computing machine such as tablet PC. Most digital watercolor systems are generated to perform on desktop, not low powered mobile computing system such as iPad. Our system can be utilized for art education besides professional painters. Our system is not a naïve imitation of real watercolor painting, but handles with properties of watercolor such as diffusion, boundary salience, and mixing of water and pigment.

Keywords*: watercolor painting system, fluid dynamics, lattice Boltzmann method, mobile device.*

1. INTRODUCTION

We present a novel watercolor painting drawing system which can work even on low powered computing machine such as mobile devices. Our system reflects properties of watercolor painting such as diffusion and mixing of pigment. Our aim includes the system brings realistic watercolor painting results to digital artists. Artists can draw their works in outdoor with iPad where our system performs interactively. Our system can be utilized for art education besides professional painters. UI of the system allows users to select suitable drawing paper, change their brushes and dry their art work to get final works.

Digital painting has many good points compared to real painting. Painting requires specific tools including canvas and paints such as watercolor, oil, or acrylic. However, such artistic tools are not cheap even for professional artists. Although real painting has its own characteristics which is hard to be expressed by digital manner, digital painting does not require wasting paints or canvas which is used up. Unlike real painting, restriction for working space is little for digital painting. User can draw while reclining on sofa. Digital painting allows easy save in digital media. Undo and fast dry are specific functions in digital painting which cannot be imagined in real painting.

Tablet PCs such as iPad are widely spread in nowadays. Among Business, education and entertainment, many applications are published in specific manner to tablets. Tablets are also suitable for drawing since its touch screen with user finger or digital pen brings intuitive and primitive input for drawing. Drawing apps can be used for preview tool for professional artists.

Most digital watercolor systems are generated to perform on desktop, not low powered mobile computing system such as iPad. Processor and memory of tablet are relatively low powered. Watercolor system must fluid evolution to guarantee the output quality. Accurate fluid evolution computation includes in general physics based simulation which requires heavy computing power. Due to computational load of fluid simulation, some works adopt procedural approaches rather than physics based simulation. For enhancing realism and efficiency, our system solves the Navier-Stokes equations for fluid flow in Lattice Boltzmann framework. Moreover, in fluid simulation, our system deals only with regions where pigments lie, not whole drawing paper. Commercially implemented application of our system should possess small amount of memory as possible. Our system is designed for such memory saving framework. Interactive manipulation is achieved so that user is not required to wait for the end of brush strokes. Our system is not a naïve imitation of real watercolor painting, but handles with properties of watercolor such as diffusion, boundary salience, and mixing of water and pigment.

* *Corresponding author, Email: c2alpha@gmail.com*
Manuscript received Sep. 12, 2012; revised Nov 23, 2012; accepted Dec 03, 2012

2. RELATED WORK

Many researchers worked on painterly rendering whose objective is to generate paintings drawn by artists. Majority of such works converts an input real image to painterly image with analysis of the input image contents and stroke generation [1]. These conversion methods are not suitable for generating painting system since they focus on the transformation of color.

Curtis et al. model the various artistic effects of watercolor and an ordered set of translucent glazes, which are created independently using a shallow-water fluid simulation [2]. They achieved impressions of real watercolor behaviors, but due to the complexity of the algorithm were unable to achieve a fully interactive system.

Van Laerhoven and Van Reeth [3] presented a physically-based system for creating images with watery paint, where goal is to design a system that runs in real-time, and yet able to recreate the various effects specific to this medium. There has not been demonstrated that the algorithm can handle high resolutions on commonly available hardware since the system requires high computational loads and resources.

Chu and Tai [4] presents a physically-based method for simulating ink dispersion in absorbent paper for art creation purposes. They similar to us devise the lattice Boltzmann equation based fluid flow model. Although their framework enhances the performance through GPU programming, it is far from the interactive manipulation on tablet computers.

Baxter et al. [5] present an interactive painting system that captures styles of oils or acrylics. The system includes both a numerical simulation to recreate the physical flow of paint and an optical model to mimic paint appearance. Chu et al. [6] demonstrates that both repeated exchanges and direct mapping of paint onto brush surfaces are sub-optimal choices, leading to excessive loss of color detail and computational inefficiencies. However, oil painting is fundamentally different from watercolor painting. Unlike to watercolor, oil paints do not tend to flow arbitrarily across the canvas.

DiVerdi et al. [7] presents a procedural vector based algorithm for generating watercolor-like paint behaviors in a low end system. As the authors mentioned in the paper, the method is not sufficient to describe realistic watercolor painting. Moreover, it is hard to guarantee that the method can also show interactive performance on recent slate devices with high resolution.

Physics based fluid animation techniques [8]-[10] require heavy computations. Various approaches have been introduced for enhancing the efficiency of fluid simulation. Losasso et al. [11] presented a method for simulating water and smoke with an octree data structure, exploiting mesh refinement techniques to capture small-scale visual details. Their approximation to the Poisson equation is a little tricky while the resulting linear system is symmetric and positive definite. Irving et al. [12] presented a hybrid method of 2D and 3D simulation to describe large bodies of water. The bulk of the water volume is represented with tall cells and the surface layer of water is simulated with a full 3D Navier-Stokes free surface solver.

Lattice Boltzmann Methods(LBM) have been used to simulate quasi-incompressible viscous flows in computational fluid dynamics. Bhatnagar et al. [13] approximated the collision operator in the Boltzmann equation with the Bhatnagar-Gross-Krook (BGK) model. The BGK model is easy to implement and makes simulation much efficient. He et al. [14] analytically solved the lattice Boltzmann BGK equation for several simple flows. Guo et al. [15] showed that their lattice Boltzmann formulation can be exactly reduced to the Navier-Stokes equations with body forces. The LBM becomes an emerging simulation technique for fluid animation in computer graphics: gaseous phenomena [16], ink dispersion [4], free surface animation [17], and miscible fluids [18].

3. PROPOSED METHOD

3.1 Overall Structure of System

Our painting system provides realistic watercolor effects with interactive performance on mobile devices. Problems of implementing watercolor LBE on mobile devices can be itemized as follows:

- High resolution leads heavy computational cost.
- Memory shortage: our system needs frame and flow buffers with full system resolution.
- Synchronization issue of simulation and drawing thread
- Only main thread can use OpenGL. GPU simulation cannot work on working thread. Heavy calculation of GPU simulation decreases the response time of user painting.
- The GPU of current mobile devices has shader 2.0 model. Complicated algorithm is hard to be implemented due to restrictions of fragment shader.

Our system is designed to make the maximum use of hardware resources including CPU and GPU. The system is divided by UI thread and simulation thread. UI thread handles user input, rendering, simulation thread control, and display update through OpenGL. Simulation thread performs LBE based fluid simulation with user input from UI thread.

Our method decompose whole simulation domain into blocks. Because region of drawing stroke is usually local, we don't need to simulate whole simulation region. We simulate only wet block to accelerate the simulation calculation. Beside acceleration, our block based approach has following properties:

- Good for multithread framework. The number of CPU cores tends to increase.
- SIMD(Single Instruction Multiple Data) can be applied. Our system achieves 2-8 times performance enhancement with ARM® NEON™, a general-purpose 128-bit SIMD architecture.

- GPU utilization for preview display update. Combination of background and watercolor simulation textures

The procedures by our system are decomposed by drawing and simulation modes. Fluid simulation halts when a user draws brushes on screen. Information on brushes drawn by user is delivered to simulation thread. Simulation speed is dependent on hardware performance like CPU and GPU.

Simulation resolution is the major factor for computational costs and memory occupation which are proportional to the resolution. Watercolor painting does not need detailed fluid behaviors. Diffusion is the main fluid behavior for describing watercolor painting. Since target device such as new iPas has very high resolution, it is hard to use the resolution. Also the diffusion effect of watercolor does not have high frequence feature, the resolution of the simulation can be smaller. Our system performs fluid simulation on one third of display resolution. Such resolution reduction allows real-time performance on mobile devices as iPad and Galaxy Tab. Figure 4 shows our system generates realistic watercolor painting. While the resolution reduction works well for inside lattices (fluid lattices), it evolves problems mentioned in the following section for boundary lattices.

3.2 Flow Simulation

We adopt lattice Boltzmann method (LBM) for calculating time varying fluid flow. LBM has benefits over general fluid simulation methods as grid and Smoothed Particle Hydrodynamics. An important benefit of LBM is all computations are localized which means LBM does not require time consuming global operations. Drawback of LBM is numerical dissipation, but we don't need accurate volume preserving fluid simulation. The object of our fluid simulation is 2D water on paper. Main feature of our fluid is diffusion. LBM is therefore the best tool for our watercolor painting system.

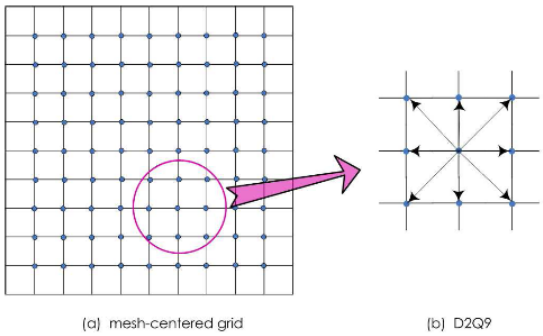

Fig. 1. Space discretization in lattice Boltzmann framework

Lattice Boltmann method models fluid dynamics using a simplified particle kinetic model. The simulation is divided into a regular lattice. At each lattice point $\mathbf{x}$ and time t, the fluid particles moving at arbitrary velocities are modeled by a small set of particle distribution functions $f_i(\mathbf{x}, t)$, each of which is the expected number of particles moving along a lattice vector $\mathbf{e}_i$. We use D2Q9 model for 2D fluid simulation in Fig. 1. Lattice Boltzmann equations are formulated with equilibrium distribution functions $f_i^{(\mathrm{eq})}$:

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = (1 - \omega) f_i(\mathbf{x}, t) + \omega f_i^{(\mathrm{eq})}(\mathbf{x}, t)$$

Where w is the relaxation parameter. $f_i^{(\mathrm{eq})}$ is modeled as a polynomial of lattice vector $\mathbf{e}_i$ and velocity. Refer [4] to details of LBM.

We adopt grain and pinning textures for describing paper properties from [4]. Grain textures given as normailized thickness images serve as patterns for the paper grain or empty space in the fibers. For the toe patterns found in real ink marks made with concentrated ink, texture maps called pinning texture is utilized to model the effect of paper disorder. The map is generated by sprinkling light line segments on a dark background.

Resolution of new iPad is 2048*1536 while iPad 1 and iPad 2 is 1024*768. Drawing region is equal to full region of iPad. We decompose whole region by blocks with 32*32. We don't need accurate impressibility which requires global operation such Poisson equation solving. Our object is just 2D flow conveying pigments. Main feature is diffusion of pigments. LBM has only local operations. So, we don't need to simulate whole region. For our fluid simulation, one third of full drawing resolution serves as simulation buffer. For new iPad, the whole region has 2048/3/32 * 1536/3/32 blocks.

3.3 Dry

Watercolor behaviors depend on whether paper is dry or wet in Fig. 2. When a brush is drawn on wet region, water in the drawn brush is mixed with water on wet paper region. For dry region, the added brush generates a semitransparent pigment layer like cellophane. That's why wetness of paper is important in drawing. We add user controlled 'dry' button. After manipulating wet brushes, user can order 'dry' command to show such effects. Dry makes water in canvas evaporate. In our system, f_i should be zero. Current distribution of pigments is stored as an image: background image.

Real watercolor painting has deep boundary. To describe such effect, 'dry' makes color in boundary lattice deep where pigment is increased by multiplying user given constant. Pigment tends to gather into valley of bending paper. Our system uses a user given texture while Moxi has complicated process.

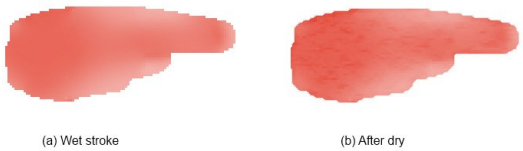

Fig. 2. Dry function (a) wet stroke (b) after dry

3.4 Implementation

Our system works in 2 threads independently. UI(drawing)

thread makes a role of drawing and call for simulation. Pigments are displayed in the following manner. Simulation threads are charge of flow calculation and pigment advection. Simulation performs unceasingly if 'dry' command does not interrupt or a user start to drawing. Updated pigment information is delivered to update image buffer.

The ARM® NEON™, a general-purpose SIMD (Single Instruction, Multiple Data) engine, efficiently processes current and future multimedia formats, enhancing the user experience. NEON technology is a 128-bit SIMD architecture extension for the ARM Cortex™-A series processors, designed to provide flexible and powerful acceleration for consumer multimedia applications, delivering a significantly enhanced user experience. Since lattice Boltzmann method consists of only local computations, SIMD can be easily adopted into our framework. Vector registers are utilized for velocity calculation with 4 lattices and pigment advection with RGB channel. Our SIMD implementation achieves 3-4 times efficiency improvement.

Our simulation is implemented only on CPU. Drawing region is local due to block based simulation. OpenGL Shading Language (GLSL) has many limitations; restricted instruction number of fragment shader, restriction to flow control, inaccurate floating number precision. Commercial painting applications often need diverse brushes whose implementation is suitable on CPU by technical issues including antialiasing. For interactive manipulation and display update, our system utilizes GPU acceleration for rendering of UI, watercolor texture, and background texture.

Pigment advection is implemented by two different methods: channel advection and common semi-Lagrangian. In channel advection, each pigment component moves along basis direction of LBE. For our case, channel advection yields more realistic advection behavior than semi-Lagrangian which has advantages on efficiency due to GPU acceleration.

In our system, human finger is main input media and make a role of brush in Fig. 3. User can select brush type where our system provides more than 20 types of brushes. Dry command saves snapshots of background images after simulation. While undo is a useful function of painting system, it is hard to include undo function into simulation based system. Undo implies that a system should simulate again from previous step. Moreover, undo requires tremendous memory resources since simulation results by every brush should be stored.

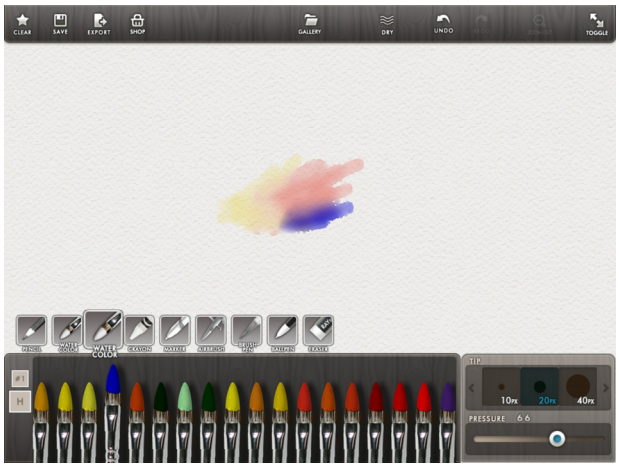

Fig. 3. User Interface of our system

4. EXPERIMENTAL RESULTS

For verifying interactive performance, we tested our system on an New iPad with Dual-core Apple A5X, quad-core PowerVR SGX543MP4 GPU SoC(System on a Chip). The implementation shows real-time performance and does not bring any waits to users. The performance was achieved on the iPad with high resolution of 2056 by 1536 due to our block based fluid computation. Table 1 shows computational costs on various resolutions under PC environment with intel Core i7 and 8GB RAM. The time in Table 1 means the simulation time per timestep. Computational costs including memory resource and simulation time are proportional to the resolution.

Table 1. Computational costs by resolution

Resolution	400*300	512*384	1024*768
Memory (MB)	52.072	73.104	129.760
Time (ms)	0.691	1.136	4.514

To enhance the computational efficiency, our method is implemeted by utilizing NEON, which is SIMD architecture of ARM processor. Table 2 shows computational perforamnce enhancement by NEON on a new iPad. Major improvement is achieved in LBE simulation those without NEON requires much time more than 1.5 times compared to NEON implementation. Other steps such as pigment advection and OpenGL update are not be affected by NEON.

Table 2. Computational performance enhancement with NEON

	LBE calculation	Pigment advection	Update to OpenGL	Total (ms)
With NEON	79.8	9.7	5.6	95.1
Without NEON	149	6.2	5.6	160.8

Our algorithm was used by many artists and general users of many different experience levels and aesthetic styles, which has resulted in a wide variety of different compositions in Fig. 4. It is difficult to objectively measure the superiority of digital painting system. Many testers commented that our system

respects well dominant properties of watercolor painting such as diffusion and mixing.

Fig. 4. Watercolor painting by our system

5. CONCLUSION

We have presented a novel watercolor painting engine that reproduces many of the important behaviors of watercolor paint. Lattice Boltzmann framework was utilized for describing fluid flow to enhance the realism of watercolor painting. Our well organized system overcomes limited resources of mobile devices to guarantee interactive performance. We will commercially release an iPad and Android application that implements full functionality.

For future work, we will unify other types of paints such as acrylics or oil paints into our watercolor painting system. Common digital painting techniques deal only with specific painting media since the unification of simulation methods for diverse media is challeging. Physical parameter configuration for representing the characteristics of diverse painting media can be a starting point of the unified painting system.

ACKNOWLEDGEMENTS

Funding for this paper was provided by Namseoul university.

REFERENCES

[1] A. Hertzmann, “A survey of stroke-based rendering”, IEEE Computer Graphics and Applications, vol. 23, 2003, pp. 70–81.
[2] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, “Computer-generated watercolor”, *Proc. ACM SIGGRAPH*, 1997, pp. 421-430.
[3] T. V. Laerhoven and F. V. Reeth, “Real-time simulation of watery paint”, Computer Animation and Virtual Worlds, vol. 16, 2005(July), pp. 429–439.
[4] N. Chu and C. Tai, “MoXi: real-time ink dispersion in absorbent paper”, *Proc. ACM SIGGRAPH*, 2005, pp. 504–511.
[5] W. Baxter, J. Wendt, and M. C. Lin, “IMPaSTo: a realistic, interactive model for paint”, *Proc. international symposium on Non-photorealistic animation and rendering*, 2004, pp. 45–57.
[6] N. Chu, W. Baxter, L. Wei, and N. Govindaraju, “Detail-preserving paint modeling for 3D brushes”, *Proc. International Symposium on Non-Photorealistic Animation and Rendering*, 2010, pp. 27–34.
[7] S. DiVerdi, A. Krishnaswamy, R. Mech, and D. Ito, “A lightweight, procedural, vector watercolor painting engine”, *Proc. ACM Symposium on Interactive 3D Graphics and Games*, 2007, pp. 63-70.
[8] J. Stam, “Stable fluids”, *Proc. ACM SIGGRAPH*, 1999, pp. 121-128.
[9] N. Foster and R. Fedkiw, “Practical animation of liquids”, *Proc. ACM SIGGRAPH*, 2001, pp. 23-30.
[10] D. Enright, S. Marschner, and R. Fedkiw, “Animation and rendering of complex water surfaces”, *Proc. ACM SIGGRAPH*, 2002, pp. 736-744.
[11] F. Losasso, F. Gibou, and R. Fedkiw, “Simulating water and smoke with an octree data structure”, *Proc. ACM SIGGRAPH*, 2004, pp.457-462.
[12] G. Irving and E. Guendelman, F. Losasso, and R. Fedkiw, “Efficient simulation of large bodies of water by coupling two and three dimensional techniques”, *Proc. ACM SIGGRAPH*, 2006, pp. 805-811.
[13] P. L. Bhatnagar, E. P. Grossa, and M. Krook, “A model for collision processes in gases. I. small amplitude processes in charged and neutral one-component systems”, Physical Review, vol. 94, 1954, pp.511–525.
[14] X. He, Q. Zou, L. S. Luo, and M. Dembo, “Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model”, Journal of Statistical Physics, vol. 87, 1997, pp.115–136.
[15] Z. Guo, C. Zheng, and B. Shi, “Discrete lattice effects on the forcing term in the lattice Boltzmann method”, Physical Review E, vol. 65, 2002.
[16] X. Wei, W. Li, K. Mueller, and A. E. Kaufmann, “The lattice-Boltzmann method for simulating gaseous

phenomena", IEEE Transactions on Visualization and Computer Graphics, vol. 10, 2004, pp. 164-176.
[17] N. Thurey and U. Rude, "Free surface lattice-Boltzmann fluid simulations with and without level sets", *Workshop on Vision, Modeling, Visualization (VMV)*, 2004, pp. 199–208.
[18] H. Zhu, K. Bao, E. Wu, and X. Liu, "Stable and efficient miscible liquid-liquid interactions", *Virtual reality software and technology*, 2007, pp. 55–64

Junkyu Oh
He received his B.S. degrees in computer science in 1997 from Chungang University and M.S. degrees in computer science in 2002 from Korea Advanced Institute of Science and Technology. He is CEO of Raysoft. Co. He recently interested in realtime drawing apps, especially watercolor.

SeungRol Maeng
He received his B.S. and M.S. degree in Computer Science in 1984 and 1986, respectively, and Ph.D. in Computer Science in 2004 from Korea Advanced Institute of Science and Technology. He is a professor in the Department of Computer Science and Engineering at Kongju National University, South Korea. His research interests include computational geometry and computer graphics.

Jinho Park
He received his B.S. and M.S. degrees in applied mathematics in 1999 and 2001, respectively, and Ph.D. in Computer Science in 2007 from Korea Advanced Institute of Science and Technology. He is an assistant professor in the Department of Multimedia at Namseoul University, South Korea. His research interests include fluid animation and scientific visualization.