A Computational Method for Interactive Design of Marbling Patterns

Mario Gazziro, João Paulo Gois
Federal University of ABC
Santo André, Brazil
mario.gazziro@ufabc.edu.br
joao.gois@ufabc.edu.br

Candy Tenorio Gonzales, José F. Rodrigues Jr
University of São Paulo
São Paulo, Brazil
candytg@ime.usp.br
junio@icmc.usp.br

***Abstract*—Paper marbling is a painting process where the artist makes use of special tools to carefully interact with paints deposited on an aqueous surface to produce marble-like paintings transferred to an absorbent paper. In this work, we present an interactive and intuitive application that simulates the marbling process digitally in real-time. To this end, we first map the artist tools into a simple and intuitive user interface. Secondly, we employ a Navier-Stokes equations solver on the GPU with a multi-layer approach to handling multiple colored paints with support to lighting and paints undulations. Our system accomplishes interactive frame-rates while manipulating tens of distinct colored paints, a requisite to applications like digital games. Results show the effectiveness of our real-time digital marbling system, providing to the artists an intuitive work interface.**

***Keywords*-Digital marbling ; Shader Programs ; Fluid Flow Simulation ; Electronic tools.**

I. Introduction

Paper marbling is a painting process where the artist employs special tools - as combs and brushes - to carefully interact with paints deposited on an aqueous surface to produce marble-like paintings (of abstract or even real objects) that are transferred to an absorbent paper [1], [2]. The dynamical nature of the marbling process has been receiving attention not only from artists but also from the computer graphics community and digital artists, creating the *digital marbling systems* field of research.

In general, digital marbling systems require two principal features:

- The first is the fluid flow simulation. Most of the works are based on 2D Navier-Stokes Equations since the dynamic of the desired marbling effects is determined by the artist interaction in a fluid-like representation. Moreover, one must ensure specific properties for the fluid flow simulation, *e.g.*, the immiscibility of the paints.
- The second is the user interface. It is fundamental to design an interface application offering a computational methodology process similar to the original artist process.

Accordingly, any proposal for a possible digital marbling system must reflect the traditional marbling context and also be straightforward to learn and interact like the traditional one. Besides, it is expected the system must be prepared to receive current electronic tools (single or multi-touch screens, graphics tablets or even mouses).

Objectives

Our primary objective relies on the development of an interactive and intuitive system for real-time marbling simulation. It provides access to the main tools used by marbling artist. Since our simulation system has support for multi-layer paint rendering, it also has enabled changing a color paint along the simulation. Also, our multi-layer paint rendering simulates 3D lighting and paint undulations that provide a 3D aspect to the paintings. Concerning the user interface, our system is implemented to support multiple interaction input devices. Thus, the user can interact using a mouse, multi-touch screen or a graphics tablet to manipulate tens of color paints at real-time frame rates.

II. Related Work

To the best of our knowledge, *AtelierM* [1] was the first study about a digital marbling system. Like to most of its successors, *AtelierM* employs computational fluid dynamics (CFD) to simulate the paints flowing. According to the results presented by their authors, *AtelierM* took up to 32 seconds for creating a single marbling pattern (running on an Intel Pentium III 933MHz processor), in 2003.

For improving the time-processing, following techniques employed the use of GPUs to perform the CFD. Jin *et al.* [2] achieved interactive frame-rates by proposing a multi-grid solver implemented for the Cg shader language. The approach proposed by Ando & Tsuruno [3] employs both CPU and GPU processor with CUDA, in which the paint is represented and rendered as vector graphics.

Wen [4] simulated digital marbling on the GPU using the GLSL shader language. The author used pre-computed velocity field, which produced faster simulations, but this system worked for one density color.

Further, instead of using CFD approaches, other systems are using procedural techniques [5], [6]. Lu *et al.* presented the Mathematical Marbling [5], where mathematical formulas were proposed to mimic the paint transportations, avoiding the computational cost of fluid simulation.

The most significant difference between the procedural techniques and those implemented by CFD is the computational cost. Furthermore, for CFD-based approaches, hardware-accelerated algorithms are essential to ensure interactivity frame-rates.

Stam [7] presented a CFD solver very popular for games engine developers, named Stable Fluids. This approach provides effects like swirling smoke trough emphasizing the fluid's stability and speed. This study was developed for both 2D and 3D environments. Jin *et al.* [2], mentioned above, thus employed Stable Fluids for the swirly effect in their marbling digital. Besides, recent studies present that a game approach to art appreciation for children stimulates learning through sensitivity, intuition, and creativity; obtaining as a result natural learning and association with other cultures. Therefore marbling painting would have great potential being used as a game-oriented learning tool.

Recently even the VR technologies are exploring the marbling techniques [8] allowing full immersion during pattern development.

III. The Marbling System

We assemble our marbling system aiming at the trade-off between real-time fluid flow simulation and user-interactivity. In the following sections, we present our user-interface and the structure of our system, as well as how we explore CPU and GPU to pursue our requirements.

A. Interface Design System

The interface contains the main marbling options in one single widget. Namely, the system is composed of two areas (Fig. 1): the *design area*, where the artist creates the digital marbling patterns, and the *painting toolbar*, which contains the different tools employed by the users such as colors, size of the paint drops and the color selection. Especially, the painting toolbar provides:

- Eyedropper: creates a single drop of paint;
- Brush: creates multiple drops of paints simultaneously. Mainly, when using a multi-touch screen, the system produces multiple drops by multiple touches (one for each touch);
- Needle: performs a single spreading of the paint;
- Comb: performs multiple spreadings of the paint;
- Color Palette: sets the colors of the paints on the output device;
- Layers: allows selecting the layers to change the colors;
- Size: sets the size of the paints to draw on the output device.

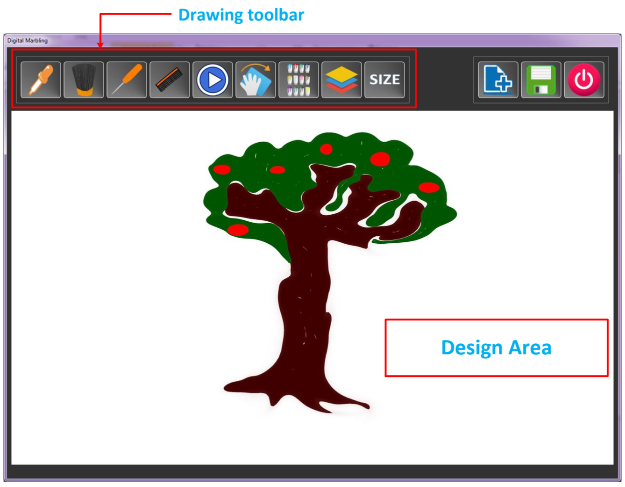

Figure 1. The interface of the application is composed of two main areas: the *painting toolbar*, which has the artist resources, and the *design area* where the artist makes the art.

B. Method Pipelines

The processes of our marbling system perform massive communications and data exchanges between the user interface (paint rendering and processing input devices call) and the fluid flow simulation. Each expected process leads to a distinct sequence of tasks. Fig. 2 – through a collaboration diagram – depicts one of this first process, which is the insertion of the first drop of paint. Concisely, the tasks are:

Preparing the system to receive a drop of paint:: The process begins with the user selecting the Eyedropper Tool (Item 1). This information triggers all the sub-process to simulate the drop of paint at the fluid flow solver as well as to render it (Items 2 and 3).

Fluid Simulation:: At this moment, the fluid flow simulation begins. Item 4 refers to the part of the solver that calls in a loop Items 5 and 6 to compute the Navier-Stokes Equations (Sec. III-D).

Domain Transformation and Density Computation:: The fluid flow solver is prepared to receive the user inputs, which are transformed to the domain simulation coordinates (Item 7). In this case, the system is prepared to receive the drop of paint, which means to compute the density values and store them in a *frame buffer object* [9] (Item 8).

Graphics Objects Configuration:: The texture objects and rendering shaders [9] are created (Items 9 and 10) for rendering the fluids.

Fluid to Shader Texture:: For rendering the drop of paint, our OpenGL manager classes require the frame buffer object that stores the density from the fluid flow solver (Item 11). The solver then sends the required buffer in a texture (Item 12). Notice that the OpenGL manager must specify the layer it will render (paintID).

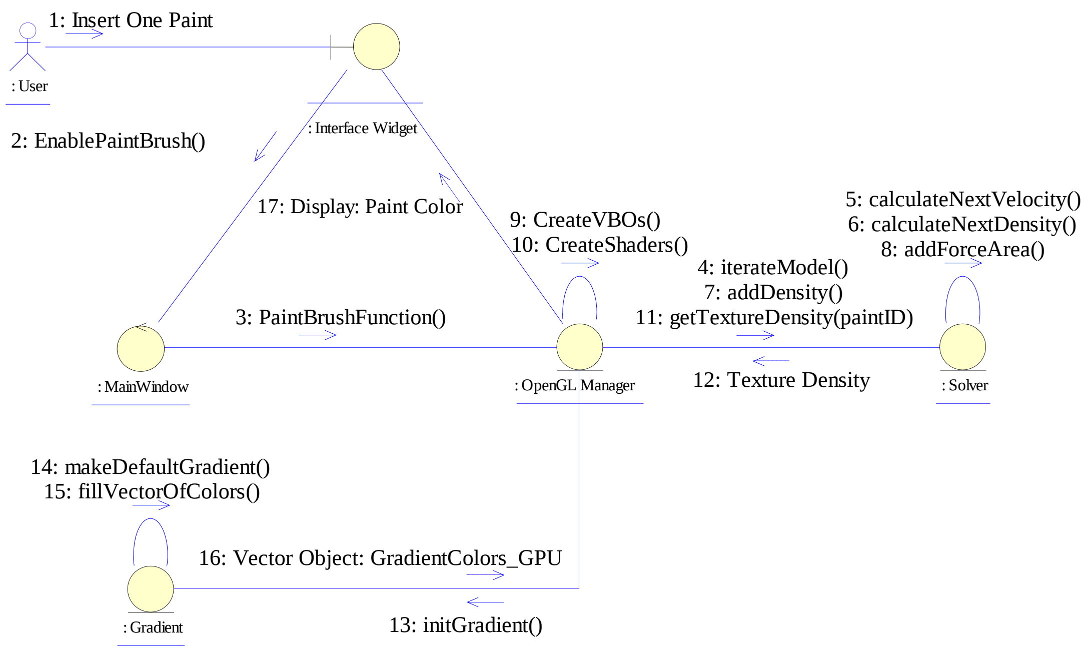

Figure 2. Collaboration Diagram: Example of inserting the first drop of paint.

Gradient Color Paint:: In parallel, to enhance the rendering quality, a gradient lookup table is created to simulate fluid undulations and light interactions (Items 13-15). This lookup table is sent to the GPU as a texture (Item 16).

Display Result:: This last stage is responsible for combining into the GPU both the texture density and the texture of gradient to display the paint color (Item 17).

C. Fluid Simulation

The simulation of the paint flow of our digital marbling system is based on 2D incompressible fluid flows, modeled by the Navier-Stokes Equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + F \qquad (1)$$

$$\nabla\cdot\mathbf{u} = 0 \qquad (2)$$

where ρ is the fluid density (constant), ν is the kinematic viscosity, and F represents any external forces that act on the fluid [10]. We employ and adapt to our system the *syfluid* library [11], which explores shader programs to the fluid flow simulation. This approach is based on the Stable Fluids technique [7]. This library allows not the only real-time simulation but ensures favorable results for marbling. Individually, tasks are distributed to render the fluid properties to textures in real-time. While the CPU manages the input/output routines for the user's interaction and starts the computation of paint undulations and lighting (Sec III-D); the GPU manages the GLSL algorithms, simulating the fluid flows, and combining the textures to render the painting.

D. The System Architecture

The domain simulation is discretized into a two-dimensional grid, in which both the velocity field and density field are represented as textures. Henceforth, we will name *density texture* to define the representation of the digital paints (fluids) and *velocity texture* to define the vector field in which the paints will be transported.

Since fluid flow simulation is executed on the GPU, there is an interaction among all the shader programs (shaders for rendering and the fluid simulation) used to provide real-time feedback to the user. The whole process is systematized in Fig. 3. The light blue objects represent routines of the fluid flow library (*syfluid* [11]) that we explored and adapted to our system, the yellow objects are responsible for rendering the paints with undulations and light interactions. To these purposes, we employed and adapted the routines provided at [12]. The red objects correspond to our communications and rendering approaches.

The CPU is responsible mainly for two tasks: the domain transformations from the user position on the electronic input devices to the fluid flow domain simulation, and the generation of the gradient lookup tables used for simulating paint undulations and lighting. The GPU processor is responsible for the three core process, each one represented by a block in Fig. 3.

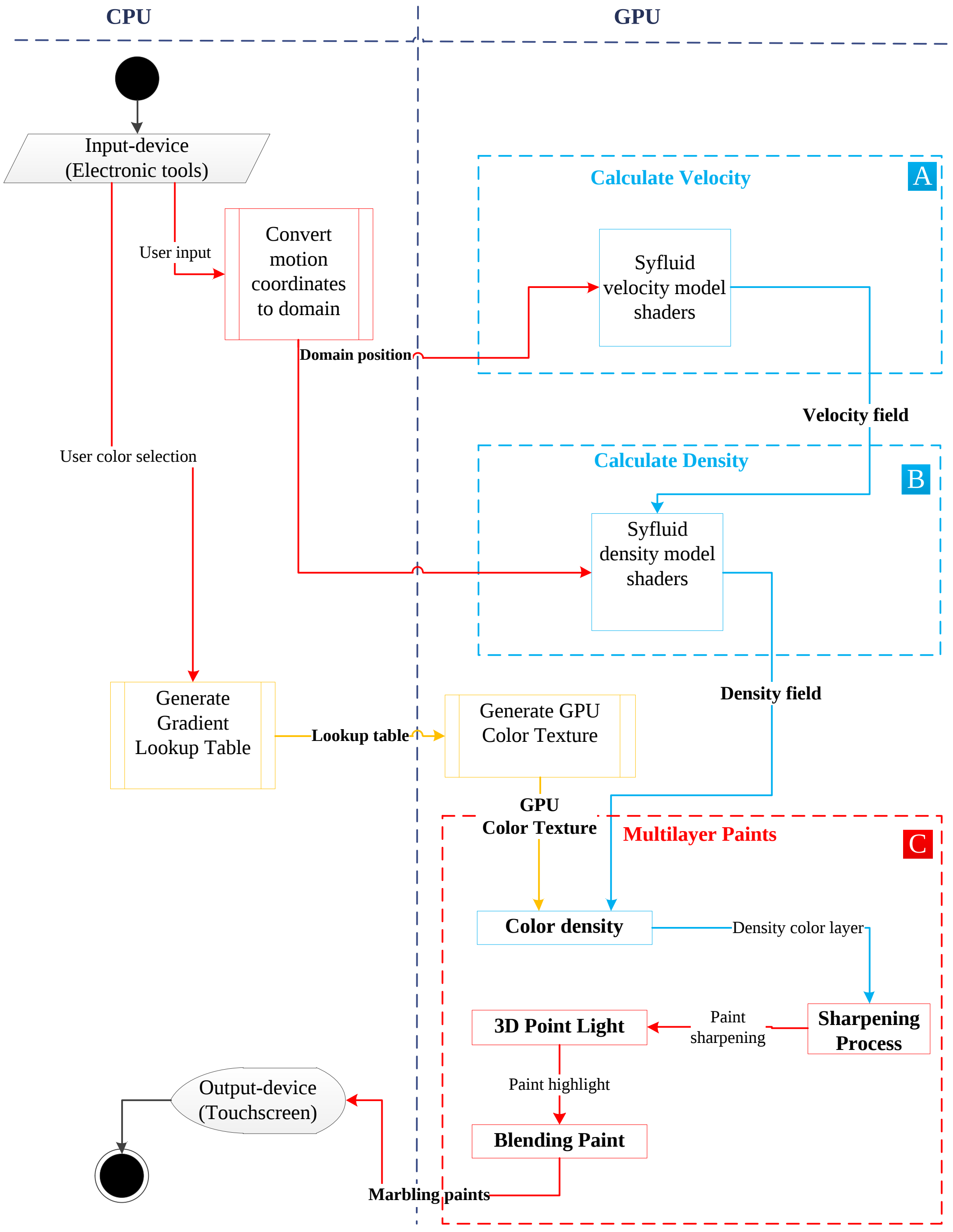

Figure 3. Our digital marbling process: distribution of tasks of each processor in which the interaction between shaders is realized in the GPU processor.

Block A aims at creating the velocity texture and ensuring the incompressibility property. At this block, the displacement force vectors the users provided from the interface interaction is received. This block is the first to be executed, and the velocity field is sent to Block B.

Block B has the purpose of creating of the density texture, that is the digital paints that will be generated according to the colors selected by the users. This process is the second to be executed. Both Blocks A and B have routines that employ shader programs that solve Eq. 1-2 [11]. It is worth to mention that shaders of this library were adapted to our purpose to support multi-layer densities, allowing manipulating effectively several color paints simultaneously.

Block C provides the multi-layer paint rendering (Sec. III-E). This block processes two input textures: one for the color assigned to the paint (**GPU Color Texture**) and other for the fluid data density obtained from Block B (**Density field**). These inputs are adequately blended to define the final rendering painting. In the next section, we detail such a rendering pipeline.

E. Multilayer Rendering

To guarantee that the distinct paint colors inserted by the artist are preserved – the property of the traditional paper marbling, named ox-gall effect [13] – we provide in our system a GPU-based multilayer paints technique [2]. Besides, our multilayer rendering incorporates 3D lighting and paint undulation to become the painting rendering more realistic.

For each layer color, the rendering pipeline is triggered. It begins with a blending of the **GPU Color Texture** with the **Density field**. Then, *Paint Sharpening* (next section) is applied to correct the paint contours.

The next step relies on simulating the 3D lighting and paint undulation. In this stage, we employ the technique proposed by [12] that computes light effects and gradient colors on lookup tables on CPU. These data are transferred to the GPU as textures where we apply a shader-based a point light rendering.

Paint Sharpening: In the process of creating textures by fluid flow simulation, blurring effect is expected to arise (Fig. 4-(a)) due to the dissipation effect produced by the solver [14]. Therefore, to improve the quality of the paint contours, we applied a shock filter [15], [16], which we also implemented in a shader program. The effect of this filter can be seen in Fig. 4-(b). However, the shock filter can produce aliased contours, as expected. Indeed, we also needed to apply the anti-aliasing of the OpenGL, as depicted in Fig. 4-(c).

IV. Results

To illustrate the use of our prototype during the creation of digital marbling patterns, in the next we present functionalities of our system.

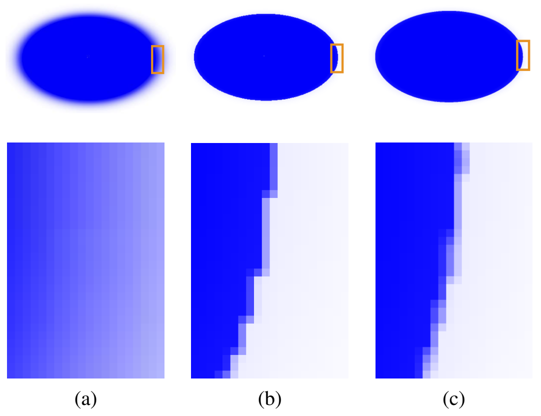

Figure 4. Paint Sharpening: The top row presents a initial blue drop of paint and the bottom row presents a zoom-in at the contour of the drop. (a) Initial paint flow, (b) paint with Shock Filter, (c) paint with shock filter and antialiasing.

A. Creating paints

Fig. 5 provides a sequence of steps to create paint using the Eyedropper Tool. In detail, the user chooses the *Eyedropper Tool* from the painting toolbar. After the user selects paint color and then optionally sets the maximum size of the drop paint. Next, the user puts the paint on the design area with a single click (with the touchscreen, mouse or drawing table) in which a small drop is defined. If the user holds (or increase the pressure with of the pen of the drawing table), the drop increases until the maximum size previously defined. If the user holds and drags, a stripe-shape of paint is inserted. Finally, the user saves and exit of the window.

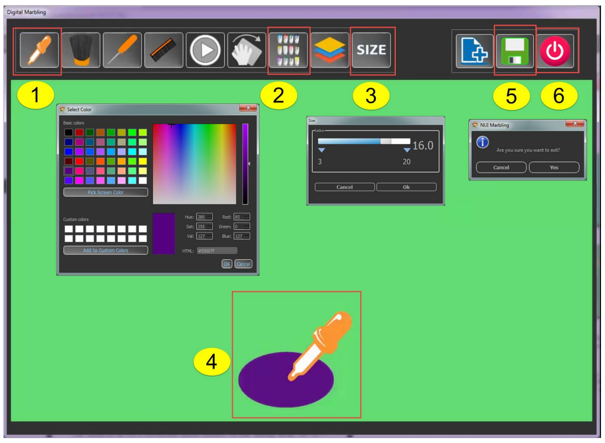

Figure 5. Create paint with Eyedropper Tool.

Figure 6 depicts the creation of multiple (two in the example) paints at the same time using a multi-touch screen. Specifically, the user chooses the tool *Brush* from the painting toolbar. After that, optionally, the user selects paint color since the default color is blue and chooses the size of the paint to draw. Next, the user creates the paint in the design area with two touches onto the screen. Finally, the user saves and exit of the window.

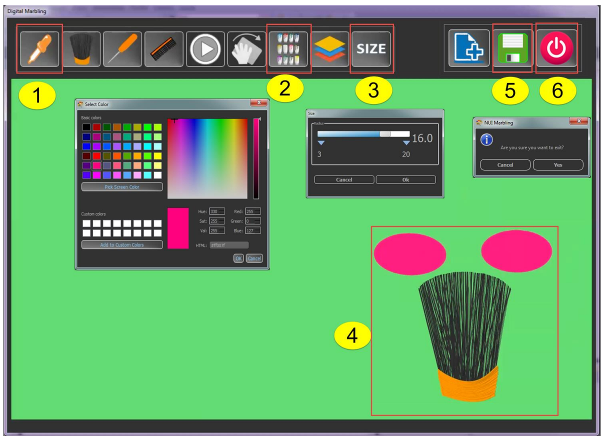

Figure 6. Create paint with brush.

B. Marbling Patterns

Digital marbling patterns are the results of paint movements performed by the user while using the needle and comb tools mentioned in Sec. III-A.

Needle tool allows making one movement per time (Fig. 7), whereas the comb tool allows to multiple movements (Fig. 8).

In Figure 7, it is presented the use of the needle tool and the eyedrop. First, it is created the layers, *i.e.* each color with eyedropper tool (Fig. 7-(b), (c), (d), (i)) and then is made a texture spreading (Fig. 7-(e), (f), (g), (h)), creating the final pattern (Fig. 7-(j)).

The comb tool allows moving the paints in different directions. These movements are essential to get unique marbling patterns. Fig. 8 shows this tool and the final pattern obtained with five colors (layers) with bottom-up movements.

C. Change Color Paint

Our system provides color changes for existing paints, *i.e.*, in the case where a user wants to change the color of the paints created in the design area, the user can accomplish this task. In Fig. 9-(a) can observe up a butterfly in the design area. It is possible to change the paint color by layer option of our system. For this, consider a sequence the steps to accomplish this task, starting when the user creates the paints or design in the design area. After that, the user chooses the *Layers* option from the painting toolbar, then selects the color layer to change in the pop-up window (Layer in Fig. 9-(a)). Finally, the user selects the color wanted to replace (Fig. 9-(b)).

With previous information, artists can create their designs. Some results obtained through our systems are shown in Fig. 10.

D. Performance

We develop our application entirely on C++ and GLSL shader language. We performed our tests for two PC configurations: a PC with an Intel i5 processor, 4GB memory and NVIDIA GeForce GT 540M graphics card; and a PC with an Intel i7 processor, with 32GB memory and NVIDIA GeForce GTX 970 4GB graphics card, indicated as **GPU1** and **GPU2**, respectively, in Table I. The results are presented in frames per second (FPS), in which eight layers were used for **GPU1** and 20 layers for **GPU2**. In each layer was applied the comb tool for the creation of marbling patterns.

Table I. Fluid Flow Solver Domain vs. PC configuration: Results presented in FPS.

	GPU1	**GPU2**
256×256	10	41.6
512×512	1.42	27
1024×1024	0.43	6.36

In Table II we present the processing time footprint when shock filter is applied, considering the **GPU2**. It can be observed that shock filters only increase 4% for the lowest grid resolution and 16% for the highest-resolution grid. On average, when considering the 512×512 grid, which is the default resolution employed in our tests, the shock filters costs in average 10%.

Table II. Fluid Flow Solver Domain vs. Visual Effect: Results presented in FPS.

	With Shock Filter	**Without Shock Filter**
256×256	43.4	41.6
512×512	29.5	27
1024×1024	7.4	6.36

Even though our system can create digital paintings with multiple colors, it is limited to the capabilities of the GPU. In particular, the **GPU2** delivers up to 25 colors with a 512×512 grid resolution.

Finally, this performance is reflecting in Fig. 11, when we present the employment of our application by users in a digital table in real-time and in Fig. 12 we show a blending of results of our technique with paper-like texture, simulating the transfer of the (digital) painting to a paper.

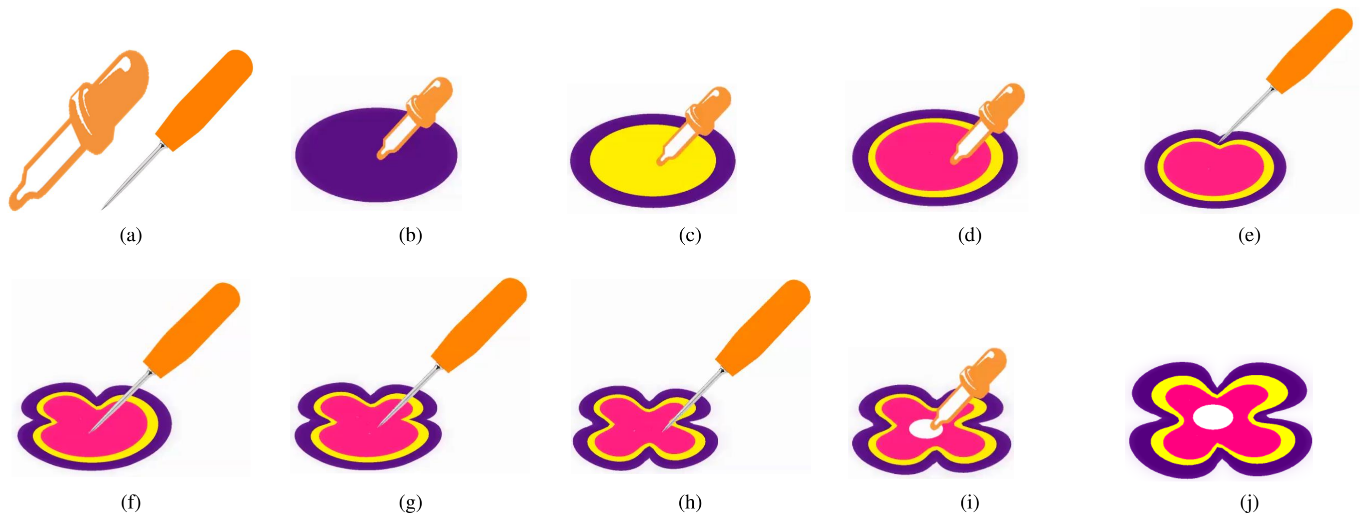

Figure 7. Pattern Marbling using the needle tool: (a) Eyedropper and needle tools, (b) First layer, (c) Second layer, (d) Third layer, (e) Up-bottom movement, (f) Left-right movement, (f) Right-left movement, (e) Bottom-up movement, (g) Fourth layer, and (h) Final design.

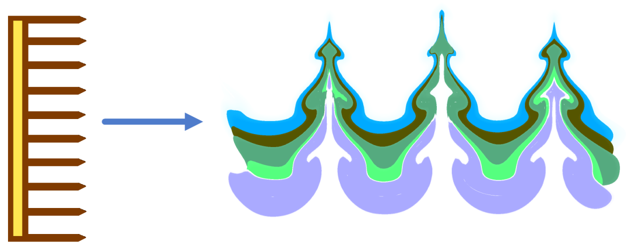

Figure 8. Pattern Marbling using comb tool with 3 teeth.

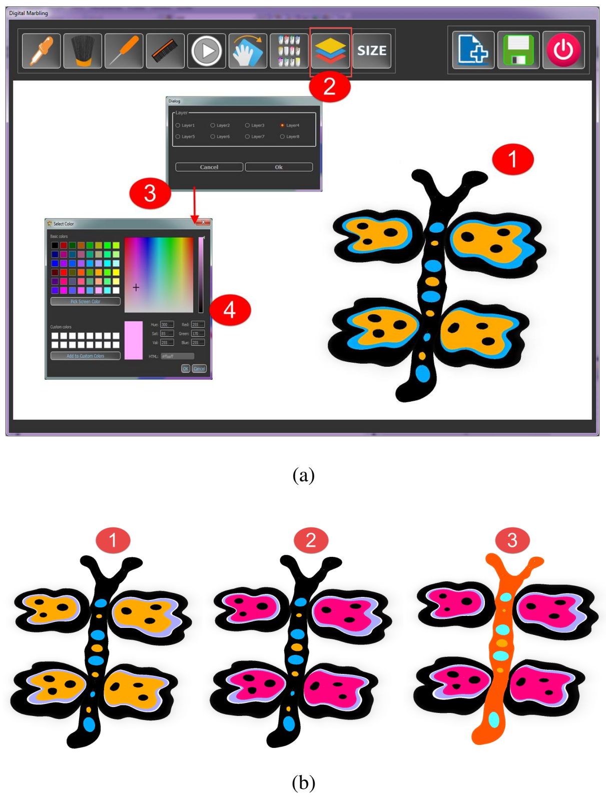

Figure 9. Change the colors of a layer of paint: (a) Initial painting; (b) re-painting after changing the color.

V. Conclusion

We have successfully developed a real-time digital marbling system, which provides the artists with an more natural work interface, with the same working tools on its digital version.

Crucial future work is the development of a *undo* simulation tool. There are enormous challenges in the development of an effectively undo approach. For instance, we should track both previous time-steps of the fluid simulation and the artist manipulation. Trivial solutions lead to large memory footprint, with is prohibitive for real-time simulations. Indeed, an in-depth investigation related to GPU-based data structures [17], [18] and systems of fluid flow simulation with support to restarting fluid simulation [19], [10] must be conducted.

Acar & Boulanger [20] presented a multi-scale fluid model that allows simulating both laminar and turbulent flows. This approach also enables to control the effect of the paint fluctuations at distinct scales. The provided results were exciting. However, to the best of our knowledge, the authors did not discuss how to tailor their approach to an easy user-interface. Specifically, the behavior of their

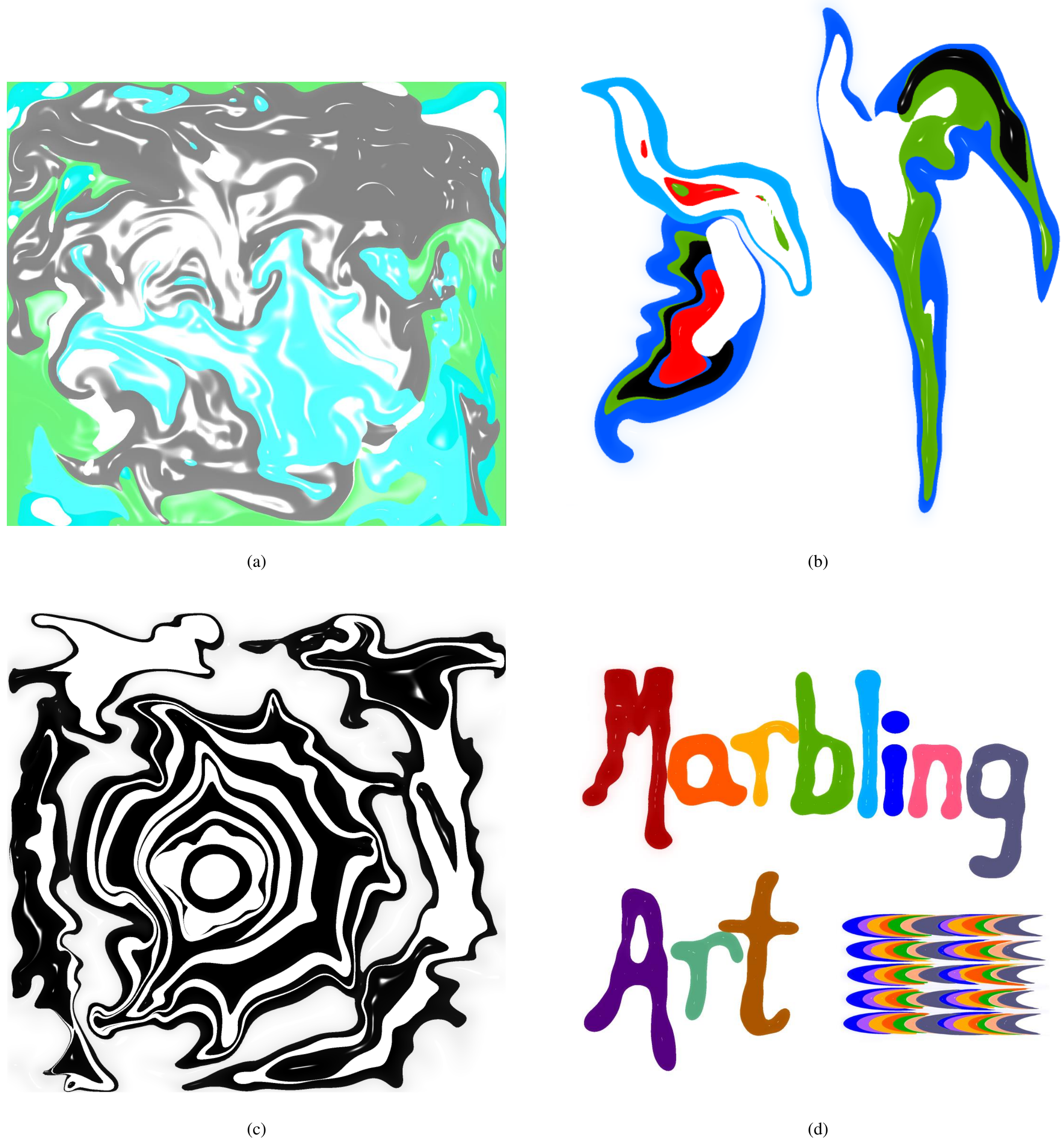

Figure 10. Results from our Digital Marbling System: (a) 5 layers, (b) 6 layers, (c) 17 layers and (d) 13 layers.

technique when simulating the traditional marbling patterns as well as arguing about real-time simulation and GPU-based implementations. We believe this work opens up essential avenues to improve the fluid-simulation of the present technique.

Moreover, to provide users with greater options for the creation of patterns, other spreads on the paints could be implemented, based on the implementation performed in this work. The University of Washington Library [21] provides 32 distinct patterns drawn around the world through paper

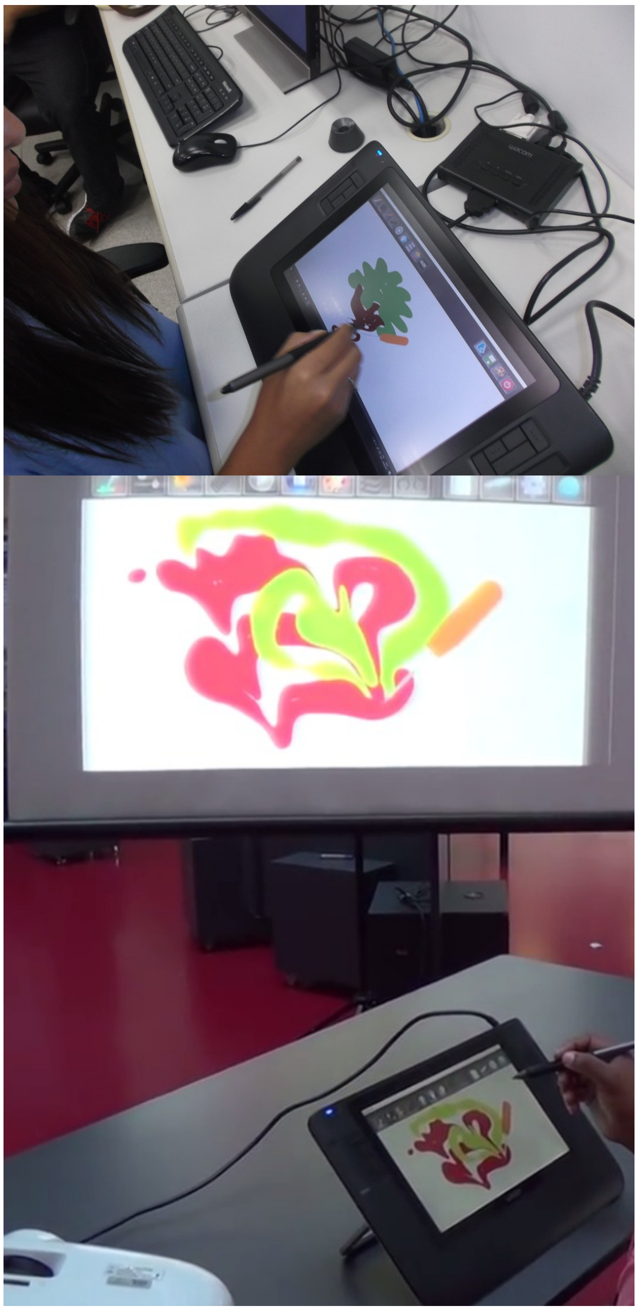
Figure 11. Examples of the use of our application in a Digital Drawing Table

marbling.

Acknowledgements

We thank FAPESP-Brazil (PROC. NO. 2014/11067-1) for grant support and CAPES-Brazil for the scholarship to Candy Tenorio Gonzales.

References

[1] X. Mao, T. Suzuki, and A. Imamiya, “Atelierm: A physically based interactive system for creating traditional marbling textures,” in *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, ser. GRAPHITE ’03. New York, NY, USA: ACM, 2003, pp. 79–ff. [Online]. Available: http://doi.acm.org/10.1145/604471.604489.

[2] X. Jin, S. Chen, and X. Mao, “Computer-generated marbling textures: A gpu-based design system,” *Computer Graphics and Applications, IEEE*, vol. 27, no. 2, pp. 78–84, March 2007.

[3] R. Ando and R. Tsuruno, “Vector fluid: A vector graphics depiction of surface flow,” in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR ’10. New York, NY, USA: ACM, 2010, pp. 129–135. [Online]. Available: http://doi.acm.org/10.1145/1809939.1809954.

[4] S. Wen, “Digital Marbling: a GPU Approach with Pre-computed Velocity Field,” Internet, University of Waterloo, Tech. Rep. [Online]. Available: https://cs.uwaterloo.ca/sites/ca.computer-science/files/uploads/files/CS-2014-08.pdf.

[5] S. Lu, A. Jaffer, X. Jin, H. Zhao, and X. Mao, “Mathematical marbling,” *Computer Graphics and Applications, IEEE*, vol. 32, no. 6, pp. 26–35, Nov 2012.

[6] R. Grossman, *Digital Painting Fundamentals with Corel Painter 11*, 1st ed. USA: Course Technology PTR, 2009.

[7] J. Stam, “Real-time fluid dynamics for games,” in *Proceedings of the Game Developer Conference*, vol. 18. Citeseer, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.6736.

[8] S. Eroglu, B. Weyers, and T. Kuhlen, “Fluid sketching: Bringing ebru art into vr,” in *Mensch und Computer 2018 - Workshopband*, R. Dachselt and G. Weber, Eds. Bonn: Gesellschaft fr Informatik e.V., 2018.

[9] D. Wolff, *OpenGL 4 Shading Language Cookbook*, 2nd ed. Packt Publishing, 2013.

[10] M. Tomé, J. Cuminato, N. Mangiavacchi, S. McKee *et al.*, “GENSMAC3D: a numerical method for solving unsteady three-dimensional free surface flows,” *International Journal for Numerical Methods in Fluids*, vol. 37, no. 7, pp. 747–796, 2001.

[11] sy2002, “Syfuid,” http://www.sy2002.de/, 2014, [Online; accessed 2016-06-30].

[12] Phagor, “Simple Fluid Dynamics Sketch,” https://www.openprocessing.org/sketch/197, 2013, [Online; accessed 2016-06-20].

[13] H. Zhao, X. Jin, S. Lu, X. Mao, and J. Shen, “Atelierm++: A fast and accurate marbling system,” *Multimedia Tools Appl.*, vol. 44, no. 2, pp. 187–203, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11042-009-0290-z.

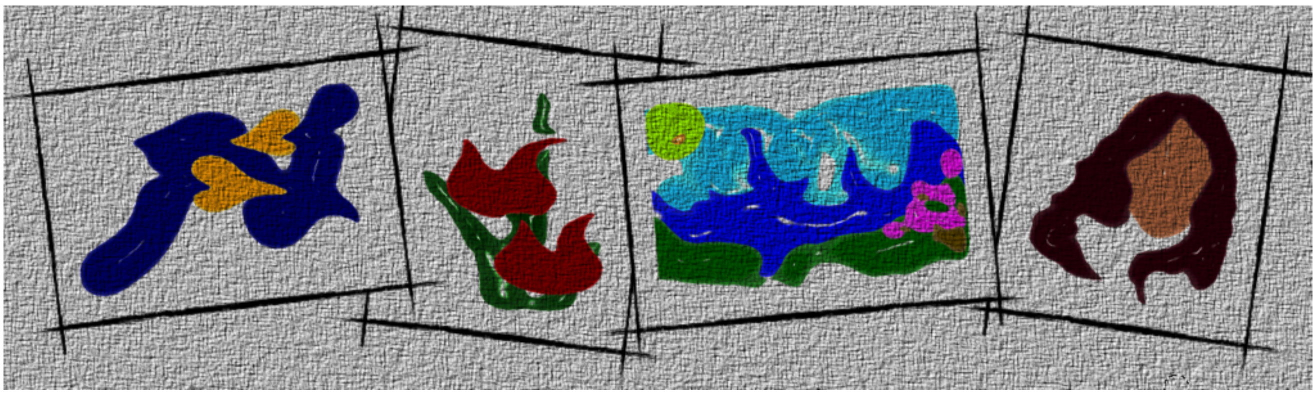
Figure 12. Blending of results of our technique with a paper-like texture, simulating the transfer of the (digital) painting to the paper.

[14] J. Stam, “Stable fluids,” in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128. [Online]. Available: http://dx.doi.org/10.1145/311535.311548

[15] J. Novosad, “Advanced High-Quality Filtering,” 2005.

[16] S. Osher and L. I. Rudin, “Feature-oriented image enhancement using shock filters,” *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 919–940, Aug. 1990. [Online]. Available: http://dx.doi.org/10.1137/0727053.

[17] A. E. Lefohn, S. Sengupta, J. Kniss, R. Strzodka, and J. D. Owens, “Glift: Generic, efficient, random-access gpu data structures,” *ACM Transactions on Graphics (TOG)*, vol. 25, no. 1, pp. 60–99, 2006.

[18] M. Labschütz, S. Bruckner, M. E. Gröller, M. Hadwiger, and P. Rautek, “Jittree: a just-in-time compiled sparse gpu volume data structure,” *IEEE transactions on visualization and computer graphics*, vol. 22, no. 1, pp. 1025–1034, 2016.

[19] H. Jasak, A. Jemcov, Z. Tukovic *et al.*, “OpenFOAM: A C++ library for complex physics simulations,” in *International workshop on coupled methods in numerical dynamics*, vol. 1000. IUC Dubrovnik, Croatia, 2007, pp. 1–20.

[20] R. Acar and P. Boulanger, “Digital marbling: a multiscale fluid model,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 600–614, July 2006.

[21] U. of Washington Libraries, “Marbled Paper Patterns,” http://content.lib.washington.edu/dpweb/patterns.html, 2017, [Online; accessed 2017-01-11].