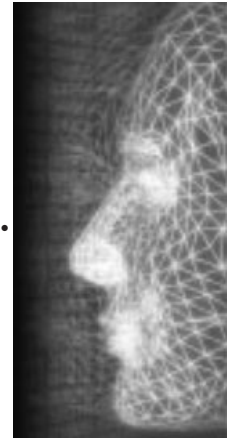


# Controlling fluid animation with geometric potential

By Jeong-mo Hong and Chang-hun Kim\*



*We propose a new fluid control technique that uses a geometrically induced potential field. Instead of optimizing the control forces exerted at each frame, as was done in previous work, a potential is added as an extra dimension to the simulation space which coerces the fluid inside this space to form the target shape. This type of shape control requires practically no additional computing by the Navier-Stokes solver at run-time, and adds little overhead to implementation. The confinement potentials are induced from geometric information given by animators, and so the control forces that take fluids to a lower potential can be decided in a preprocessing step. We show that a slightly generalized Navier-Stokes equation for fluids in potential fields can be simulated without changing the solver itself. A harmonic potential function can be quickly found with the Poisson solver which is already implemented as a part of the Navier-Stokes solver. 2 and 3 dimensional flows designed by common methods such as hand drawing, traditional shape modeling and key-framing, can be animated efficiently with our control technique. Copyright © 2004 John Wiley & Sons, Ltd.*

KEY WORDS: fluid control; potential fields; physically based modeling; simulation technique; natural simulation; computer animation

## Introduction

The characteristics of flow inspire an admiration for nature, but they also represent an intellectual challenge. In computer graphics, many researchers have tried to introduce moving fluids into synthetic scenes such as digital animations and virtual environments. Recently, numerical simulation of the fully three-dimensional Navier-Stokes equation has proved to be a promising technique for the realistic animation of fluids. The most important thing for computer animators is to be able to control such simulations without having to subordinate their creative inspiration to the technical barriers. The control of fluids requires physically based models and numerical simulation techniques, which are more closely related to computational fluid dynamics than to other areas of computer animation.

A keyframing technique for fluid animation is desirable, but poses particular technical problems. Treuille *et al.*,<sup>1</sup> first proposed such a system, and focused on the

optimization of control forces which could lead fluid flows towards target shapes given by keyframing. Although this approach was able to control a realistic fluid animation, the associated optimization requires too much computing time. A Navier-Stokes simulation should not be a time-consuming task on modern computers, and it would be desirable to simulate a controlled flow pattern without incurring such a large overhead.

We have therefore chosen a different strategy. Instead of exerting external control forces on the fluid at each frame, we adjust the physical properties of the simulation space, making it natural for the fluids in this space to follow the required movements (see Figure 1 for an example). This yields several potential advantages. Firstly, the control of fluid shape becomes an initial-value problem. After the initial setting of the space with a given target shape, there is no need for further control of the simulation. But if the target shape changes in time, as is required for keyframing, the simulation space can be reset with the new target shape which is dealt with as a new initial-value problem. Secondly, the clarity of a physically based model is maintained, because no additional control algorithm is attached. Finally, the algorithm is efficient and easy to implement.

\*Correspondence to: Chang-hun Kim, Computer Graphics Laboratory, Korea University, An-am Dong, Seong-book Gu, Seoul, Korea.  
E-mail: chkim@korea.ac.kr

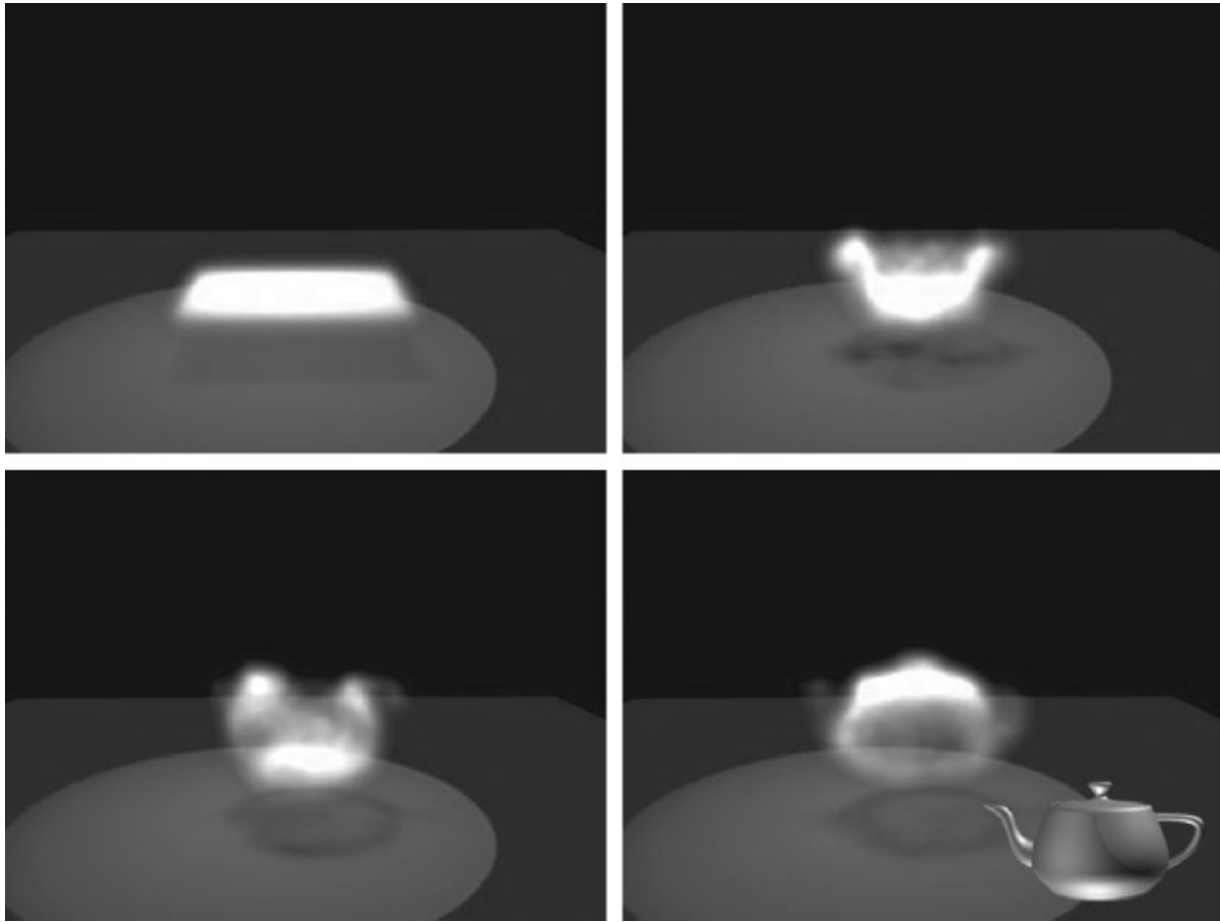


Figure 1. Shape control of fluids—teapot.

We propose a new potential field that will achieve this control. Unlike gravitational or magnetic potentials, this confinement potential field is induced by the geometric information given by animators: the initial position of the fluid, the target shape, and the boundaries of the simulation space. Therefore, we call this field the *geometric potential*.

The Navier-Stokes equation must be slightly generalized to describe the dynamics of flows in a potential field. Because the potential field, and hence the gradient field of the potential field, are decided during preprocessing, this modified equation can be simulated without changing either the implementation or the performance of previous solvers. Furthermore, our geometric potentials can be formulated as a harmonic function. A harmonic potential function without local minima can be quickly found by the Poisson solver which is already implemented as a part of a Navier-Stokes solver for incompressible fluids. Consequently,

our shape control technique requires practically no additional computing by the solver at run-time, and only a little extra implementation effort.

Several examples are presented in 2 and 3 dimensions to show that our algorithm gives a lot of control without losing a realistic fluid appearance. Target shapes and keyframes can be designed without difficulty, using common methods such as hand drawing and traditional shape modeling.

## Related Work

Techniques for the control of fluids rely on developments in physically based models and numerical simulation techniques. Numerical simulation of the Navier-Stokes equation (the governing equation of general fluids) has become a standard technique for the realistic animation of fluids. Foster and Metaxas<sup>2</sup> were

the first to use a fully three-dimensional Navier-Stokes solver in computer graphics. Stationary obstacles, in-flow and outflow, buoyant obstacles and surface pressure history were dealt with by their fluid model. Control was maintained in all those situations by adjusting the properties of the fluid, for instance by manipulating velocities and pressures. Subsequent work<sup>3</sup> by Foster and Metaxas allowed animators to control the fluid without knowledge of the underlying physics of the simulation, because the low-level details were concealed. Despite this improvement in usability, control was still limited by the fluid model.

Subsequent research<sup>4-9</sup> has allowed additional phenomena to be animated within a Navier-Stokes simulation framework. The work of Feldman *et al.*<sup>10</sup> shows an interesting application of a Navier-Stokes solver for incompressible fluids, which is more efficient and stable than the compressible one, to simulate explosions. These solvers look very promising, but the associated control techniques were similar to those of computational physics, and not friendly to computer animation.

Treuille *et al.*<sup>1</sup> introduced a change of paradigm, using new and efficient optimization of control forces so as to enable keyframing to be used in fluid animation. Unfortunately, the number of degrees of freedom in fluids is much larger than in rigid bodies,<sup>11</sup> so this method needs hours of computing time even for simple two-dimensional target shapes such as letters of the alphabet. However, it is likely to become much faster in the near future. Nevertheless, our method offers efficiency and ease of implementation, and in any case availability of a large number of methods allows animators to select an appropriate technique for a particular purpose.

Interestingly, the key idea of our method has been used in some quite different researches. The level-set method<sup>12</sup> and the level-set shrink wrapping algorithm<sup>13</sup> showed that the implicit representation of target shapes can provide a way to persuade fluids to take up a desired shape. This idea was implemented as an alternative to the harmonic potential function. Since it is common for fluid geometry to be stored in implicit forms, such as level-set surfaces for liquid interfaces and density fields for gas, shape transformation by means of implicit functions looks like a good approach to controlling fluids. Turk and O'Brien<sup>14</sup> solved the transformation problem for  $N$ -dimensional shapes in  $N + 1$  dimensions. This encouraged our idea of adding a potential dimension to the simulation space, to represent the deviation of fluid distribution from the target shape. More generally, the idea to explain a phenomenon as a property of space draws on Einstein's theory

of general relativity (see Alder *et al.*<sup>15</sup> for an introduction to this topic).

Blakely<sup>16</sup> provides a good introduction to the characteristics of potential fields, including conservative scalar fields and harmonic functions.

## Overview

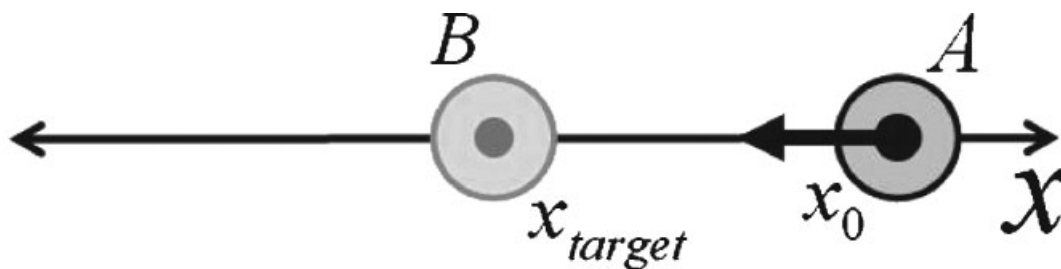
In this section, the basic idea of our control algorithm is explained and we compare it with previous ones.

A simple control problem in one dimension can be modeled as shown in Figure 2(a). There is a physical entity  $A$ , currently at  $x_0$ , that we want to place at  $x_{\text{target}}$ . Control means checking the error and doing something to reduce it. In Figure 2(a), the error  $e$  is  $x_{\text{target}} - x_0$ . Control forces will be exerted on  $A$  continuously to make  $e$  tend towards zero. Determining those forces is the major concern of classical control theory (interested readers can see<sup>17</sup>). This framework was used by Treuille *et al.*,<sup>1</sup> who developed novel techniques to determine correct and optimized control forces for fluids. However, the very existence of an external controller can be burdensome to a Navier-Stokes solver. Furthermore, since fluids have many degrees of freedom, it is questionable whether this approach is applicable to complicated three-dimensional target shapes. The alternative is to insert the uncertainty into the physical model, as was done by Cheney and Forsyth,<sup>18</sup> but this is still computationally burdensome, for similar reasons.

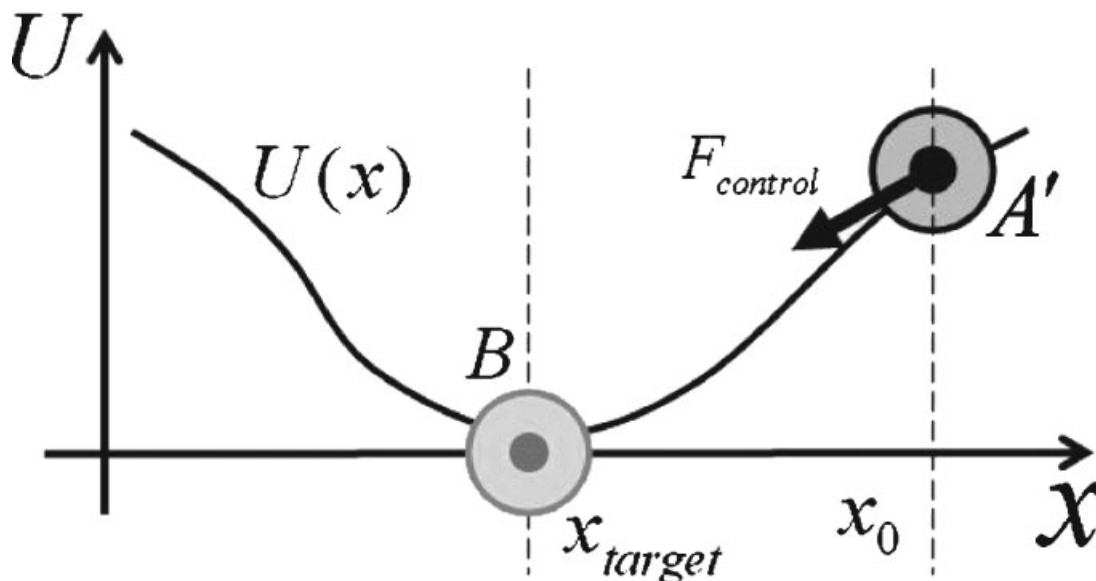
Figure 2(b) shows the key idea of our method. Intuitively, we can confirm that  $A'$  will start moving, and will stop at  $x_{\text{target}}$  at some time without external control, as a rock rolling into a ravine would do.<sup>†</sup> This means that our method—adding a potential dimension to the simulation space and adjusting it to design the animation—reformulates the control problem as an initial-value problem. There is no need for control when the simulation is actually going on: we control the animation by designing  $U(x)$  in a preprocessing step. Since the vertical axis that we have introduced is not a geometric dimension, the dimensionality and size of the simulation space are not increased, and so the performance of the numerical simulation is maintained. The only change to the fluid dynamics that results from the potential field is the introduction of  $F_{\text{potential}}$ , which is

$$F(x)_{\text{potential}} = -\nabla U(x), \quad (1)$$

<sup>†</sup>They call this a potential well in physics.



(a) Sample control problem



(b) Our control framework

Figure 2. The key idea of our control technique.

from the energy-force relationship of Newtonian mechanics. Since the potential scalar field,  $U(x)$ , is decided during preprocessing, the gradient vector field,  $\nabla U(x)$ , can be decided at the same time. This makes it possible to simulate fluids in a potential field without lowering the performance of the Navier-Stokes solver. Details are given in the next section.

### Dynamics of Fluids in a Potential Field

To simulate fluids in a potential field, the governing equation of fluids, the Navier-Stokes equation, should

be generalized to include the potential term. In this section, we will show that this generalization is physically correct and that generalized Navier-Stokes equation can be simulated with existing solvers, by revisiting the derivation of the Navier-Stokes equation.<sup>19</sup>

To derive the differential form of the momentum equation that describes fluid motion, we apply Newton's second law to an infinitesimal fluid particle of mass  $dm$ .

Newton's second law for a finite system is given by

$$\vec{F} = \frac{d\vec{p}}{dt} \Big|_{\text{system}}, \tag{2}$$

where the linear momentum,  $\vec{P}$ , of the system is given by

$$\vec{P}_{\text{system}} = \int_{\text{mass(system)}} \vec{u} dm. \quad (3)$$

Then, for an infinitesimal system of mass  $dm$ , Newton's second law can be written

$$d\vec{F} = dm \left. \frac{d\vec{u}}{dt} \right|_{\text{system}}. \quad (4)$$

Having obtained an expression for the acceleration of a fluid element of mass  $dm$  moving in a velocity field, we can write Newton's second law as the vector equation

$$dm \frac{D\vec{u}}{Dt} = dm \left( \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) = d\vec{F}. \quad (5)$$

We shall now consider the elements of the force,  $d\vec{F}$ , acting on a differential element of mass  $dm$  and volume  $dV = dx dy dz$ . It is generally accepted in fluid mechanics that  $d\vec{F}$  is composed of surface forces and body forces and that

$$d\vec{F} = d\vec{F}_{\text{surface}} + d\vec{F}_{\text{body}}. \quad (6)$$

Surface forces acting on the surfaces of a finite element can be analyzed as viscous forces and pressure differences<sup>‡</sup>, as follows:

$$d\vec{F}_{\text{surface}} = \{ \mu \nabla \cdot (\nabla \vec{u}) - \nabla p \} dV \quad (7)$$

Body forces are the forces exerted on the whole body of a finite element. In the classic Navier-Stokes equation, gravity is the only body force acting on a fluid. Instead of the potential field due to gravity, we introduce a general potential field,  $U$ . Then we can write the more generalized body forces as

$$d\vec{F}_{\text{body}} = -(\nabla U_{\text{potential}}) dV \quad (8)$$

from Equation (1).

Since the potential fields can be superposed, we can say that

$$U_{\text{potential}} = U_{\text{gravity}} + U_{\text{control}} \quad (9)$$

where  $U_{\text{control}}$  is our geometric potential generated for control purposes.

<sup>‡</sup>The details of this step are omitted here so as not to lose the direction of this paper. Interested readers are referred to basic fluid mechanics textbooks.

The generalized Navier-Stokes equation can then be written as

$$\frac{\partial(\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = \mu \nabla \cdot (\nabla \vec{u}) - \nabla P + \rho \vec{g} - \nabla U_{\text{control}} \quad (10)$$

with the incompressibility condition

$$\nabla \cdot \vec{u} = 0 \quad (11)$$

The only difference between Equation (10) and the classic Navier-Stokes equation is the existence of the potential term,  $\nabla U$ . Since  $U(\mathbf{x})$  is decided in a preprocessing step, we can also determine and store  $\nabla U(\mathbf{x})$  at that time. Then all that we have to do at run-time is to add this gradient field to the velocity field, an operation which is easily implemented because the solver already has to deal with a gravity term,  $\rho \vec{g}$ . So this small generalization adds practically no additional cost to a Navier-Stokes solver.

## Geometric Potential

Using our technique, animators can design a fluid animation by adjusting the potential field  $U(\mathbf{x})$ . An example is shown in Figure 3. This target shape was drawn using a mouse, and is drawn in black in Figure 3(a). The red-colored region shows the initial distribution of the fluid. From these user-specified geometric data, which originate in a standard bitmap format, we can obtain the geometric potentials shown in Figure 3(b). The animation in Figure 6 shows a seed that grows into a flower. It was designed and animated using this potential field. We will now describe how we create the potential field.

The initial start distribution of the fluid,  $\mathbf{p}_{\text{start}}$ , and the target shape,  $\mathbf{p}_{\text{target}}$ , are required as input. They can be specified as pixels in bitmap files, as the vertices of polygons or as implicit surfaces. In any case, they will be converted to indices on a computational grid. Equation (10) means that fluids will flow into locations of lower potential. Therefore, we wish the potential function  $U$  to be

$$U(\mathbf{p}_{\text{target}}) < U(\mathbf{p}_{\text{free}}) < U(\mathbf{p}_{\text{start}}); \quad (12)$$

where  $\mathbf{p}_{\text{free}}$  is an unspecified region. To avoid the fluid leaving the simulation space, we need one more condition,

$$U(\mathbf{p}_{\text{start}}) < U(\mathbf{p}_{\text{boundary}}) \quad (13)$$

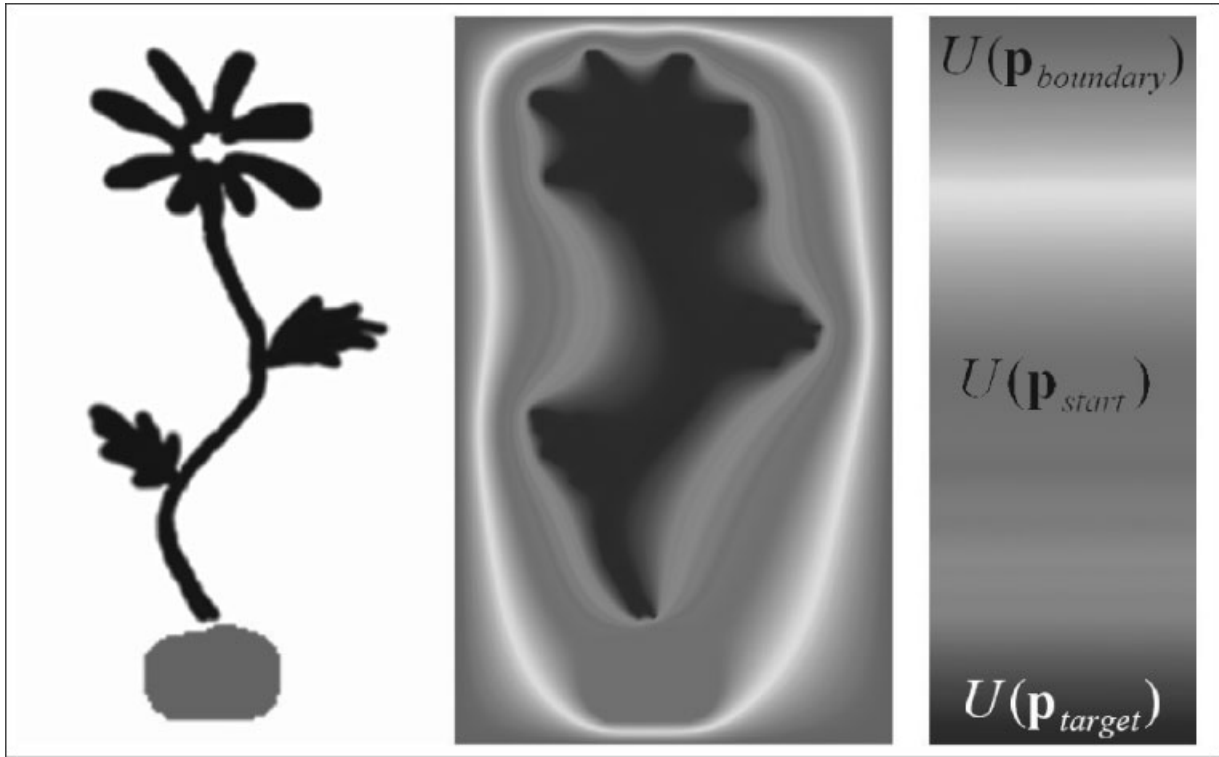


Figure 3. Potential field for the flower model: (a) target shape; (b) geometric potentials; (c) key.

And finally, to avoid local minima in the flow path, we have

$$\nabla U(\mathbf{p}_{\text{free}}) \neq 0. \quad (14)$$

These conditions are very similar to those found in modeling physical fields. In those models, we can obtain a harmonic potential field by solving Laplace's equation,

$$\nabla^2 U = 0, \quad (15)$$

with the Dirichlet boundary condition. In the example of Figure 3, the boundary conditions are

$$U(\mathbf{p}_{\text{target}}) = 0, \quad (16)$$

$$U(\mathbf{p}_{\text{start}}) = k_{\text{start}} \quad (17)$$

and

$$U(\mathbf{p}_{\text{boundary}}) = k_{\text{boundary}}, \quad (18)$$

where  $0 < k_{\text{start}} < k_{\text{boundary}}$ . Laplace's equation then becomes

$$\nabla^2 U(\mathbf{p}_{\text{free}}) = 0. \quad (19)$$

Fortunately, Equation (19), with the boundary conditions given in Equations (16)–(18) can be solved numerically by the Poisson solver which is already implemented as a part of a Navier-Stokes solver. Since this is a standard technique, details are not given here (but see Refs 5,6, and 20).

In some cases, alternative methods are required. As shown in Figure 4(a), our potential function can lose some features of a complicated model. This may be desirable or undesirable, depending on the situation. Such features can be tagged with additional data,  $\mathbf{p}_{\text{check}}$ , supplied by the user such that  $U(\mathbf{p}_{\text{check}}) = k_{\text{check}}$ , where  $0 < k_{\text{check}}$ . In the upper image of Figure 4(b), we used blue pixels to show  $\mathbf{p}_{\text{check}}$ . The lower image in Figure 4(b) shows a very clear potential field. Another method is to use a distance field. Figure 4(c) is similar to Figure 4(b), but this field is created using only  $\mathbf{p}_{\text{target}}$ . These techniques all have their own characteristics and can be used for different sorts of animation.

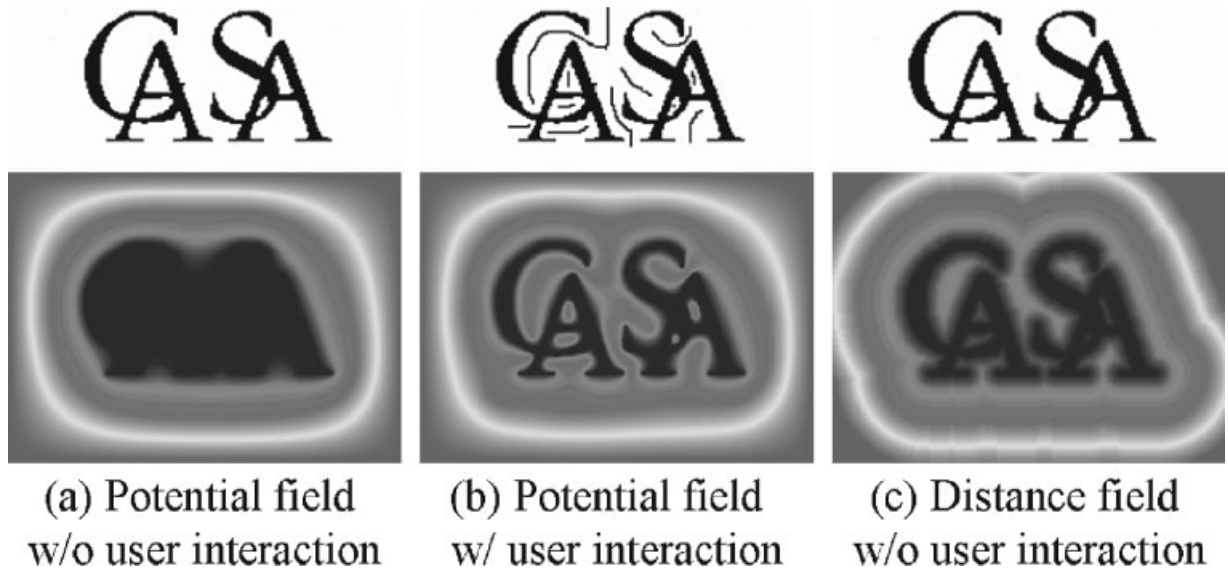


Figure 4. Several techniques used to create a potential field.

## Implementation

Our solver follows previous implementations.<sup>5,6,9</sup> Neither the implementation of the Navier-Stokes solver, or of the Poisson solver, are explained in this paper, because they are already widely used in computer graphics. Instead, we show how to combine our algorithm with an existing Navier-Stokes solver. We believe that our control techniques are compatible with almost all kinds of Navier-Stokes solver. First subsection describes the pipeline of our algorithm. Next subsection deals with some visualization topics related to our implementation.

### Simulation Pipeline

In an actual implementation, the  $\nabla U$  term in Equation (10) should be replaced by  $\rho \nabla U$ , where  $\rho$  is the density or indicator function<sup>9</sup> of fluids, since  $U$  is based on the assumption that  $\rho = 1$ . This assumption was made to obtain values of  $U$  and  $\nabla U$  which are independent of time, while  $\rho$  changes with time. Therefore, these quantities could be calculated in a preprocessing step. In our implementation, the density field  $\rho$  is determined by counting the particles near each computational grid. If  $\rho$  is updated by the Navier-Stokes solver,<sup>5,6</sup> it can also be used in calculating  $\rho \nabla U$ .

We present the pseudo code for this algorithm.

```

Preprocessing for NSSolver;
Preprocessing for GeometricPotential {
  —getControlInput  $\mathbf{p}_{start}$ ,  $\mathbf{p}_{target}$  and  $\mathbf{p}_{check}$ ;
  —makePotentialField  $U$  with  $\mathbf{p}_{start}$ ,  $\mathbf{p}_{target}$  and  $\mathbf{p}_{check}$ ;
  —determineGradientField  $\nabla U$  from  $U$ ;
While ( Simulating ) {
  —add  $\rho \nabla U$  to  $\vec{u}$ ;
  —update  $\vec{u}$  and  $\rho$  with NSSolver for dt;}

```

The cost of adding together the two vector fields,  $\rho \nabla U$  and  $\vec{u}$ , is negligible. In any time during the simulation,  $U$  can be reset using the same method used to set it initially. This allows our method to be used for both keyframe animations and interactive applications.

### Visualization

Various methods have been developed to capture the geometric shape of fluids convected by numerically simulated velocity fields. As shown in the following section, our results are rendered to appear as gases. Although it is more general to simulate the density function  $\rho(\mathbf{x})$  directly, as was done by Stam<sup>5</sup> and Fedkiw *et al.*,<sup>7</sup> suspended inertialess particles were used in our applications (see Figure 5) to capture the movements of fluids, and then converted to  $\rho(\mathbf{x})$  for improved rendering. The initial density field  $\bar{\rho}$  was determined by counting the particles near each sampling grid. Since  $\bar{\rho}$  is not smooth in general, it should be smoothed by the diffusion equation,

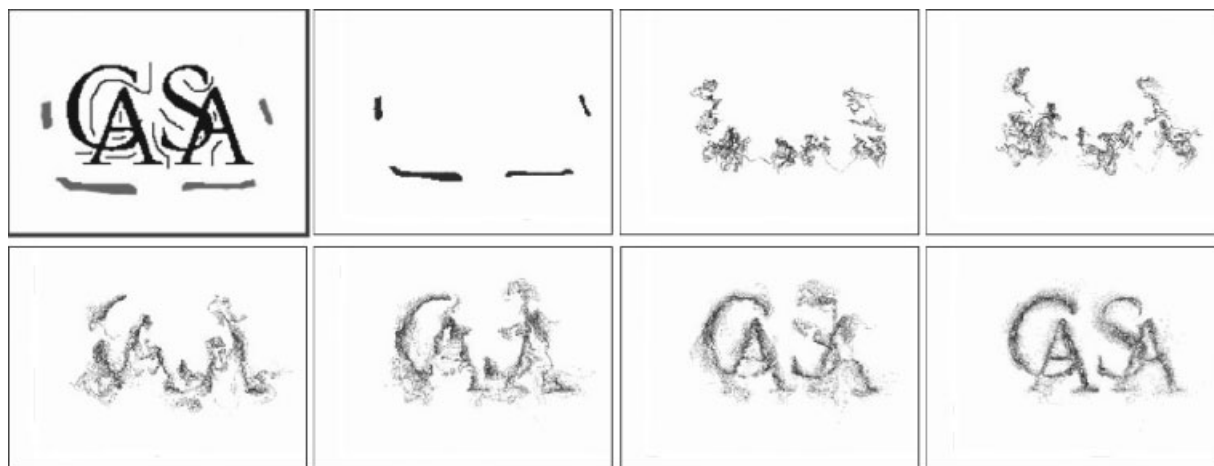


Figure 5. CASA logo example captured by marker particles.

$$\frac{\partial \rho}{\partial t} = k \nabla^2 \rho. \quad (20)$$

There are several advantages in capturing density field using particles instead of solving a partial differential equation to update  $\rho(t)$ .

First of all, the number of particles is constantly managed. There are no worries about undesirable numerical effects such as dissipation or mass loss. This is very helpful because our application requires drastically large deformations of the fluid. However, we could add or remove particles if it were necessary. Furthermore, while it is common to define the density field on the same grid for a Navier-Stokes solver, our geometric shapes are stored as a set of points. Thus we can easily control the size of the sampling grid for  $\rho(\mathbf{x})$  and change the level of detail of the scene.

## Results and Discussion

We have implemented our simulation system using MS Visual C++ 7.0, with OpenGL APIs for basic rendering. We tested this system on a Windows PC with 1 GB of RAM and an Intel Pentium IV processor running at 2.66 GHz. It uses an NVIDIA GeForce4 Ti graphics card with 128 MB of video RAM.

The CASA logo example in Figure 5 shows the flows resulting from the potential field and initial distribution shown in Figure 4(b). The fluid was simulated using a  $200 \times 150$  grid and rendered as particles with OpenGL. The final image of Figure 5 clearly shows the CASA logo. The whole process, including initial setting,

simulating, rendering and storing of  $800 \times 600$  bitmap files took 5 minutes. We simulate these alphabetic characters in a single simulation space within a much shorter time than Treuille *et al.*<sup>1</sup> Although there are differences in results between theirs and ours, this confirms the value of our technique.

Figure 3 shows a further two-dimensional example that uses a  $120 \times 200$  grid. The leftmost image shows the user input. Target shapes are marked by black pixels and start points by red pixels. These input originated as standard bitmap images and were then converted to a potential field (central image in Figure 3) during pre-processing. Figure 6 shows scenes from the animation resulting from this potential field. Gas initially located at high potential areas flows towards regions of low potential, transforming a seed to a stem, then to branches and leaves, and finally to a bud. This demonstrates how a flow path, as well as shapes can be created by a designer. The whole simulation process was done in 7 minutes.

This animation was also recorded using particles, which were subsequently converted to a density field, which was rendered by a conventional ray-tracer. We can preview the animation results quickly by looking at the particles, and then ray-trace the final images off-line after a conversion and smoothing process. This conversion process was described in the Implementation section. Postprocessing, including conversion, smoothing and rendering, took about 1 minute for 2 frames.

Figure 7 is a three-dimensional example, using three-dimensional polygon models as input data. The shape was successfully changed to various keyframe shapes, a knot, and rabbit, randomly repeated. Our method maintains the characteristics of the fluid flow during and after the deformation process. The fluid can be seen



Figure 6. Two-dimensional flow control—a blossoming flower.

moving even when the shape is constant. Since we generate the potential fields from the surface of each model (not its volume), the fluid flows along the surface of models.

Changing the keyframe (target shape) takes a few seconds, a quick process compared to the simulation itself. But if we store a pre-calculated potential field, even these delays can be eliminated.

This example took 21 minutes to simulate. Postprocessing and rendering took one minute per frame.

Our techniques strongly restrict the fluid motion to the given shapes. We believe that this is desirable because it is easy to make the flows more relaxed but not so easy to increase control.

Table 1 shows the preprocessing times for our examples, including file reading and the calculation of the

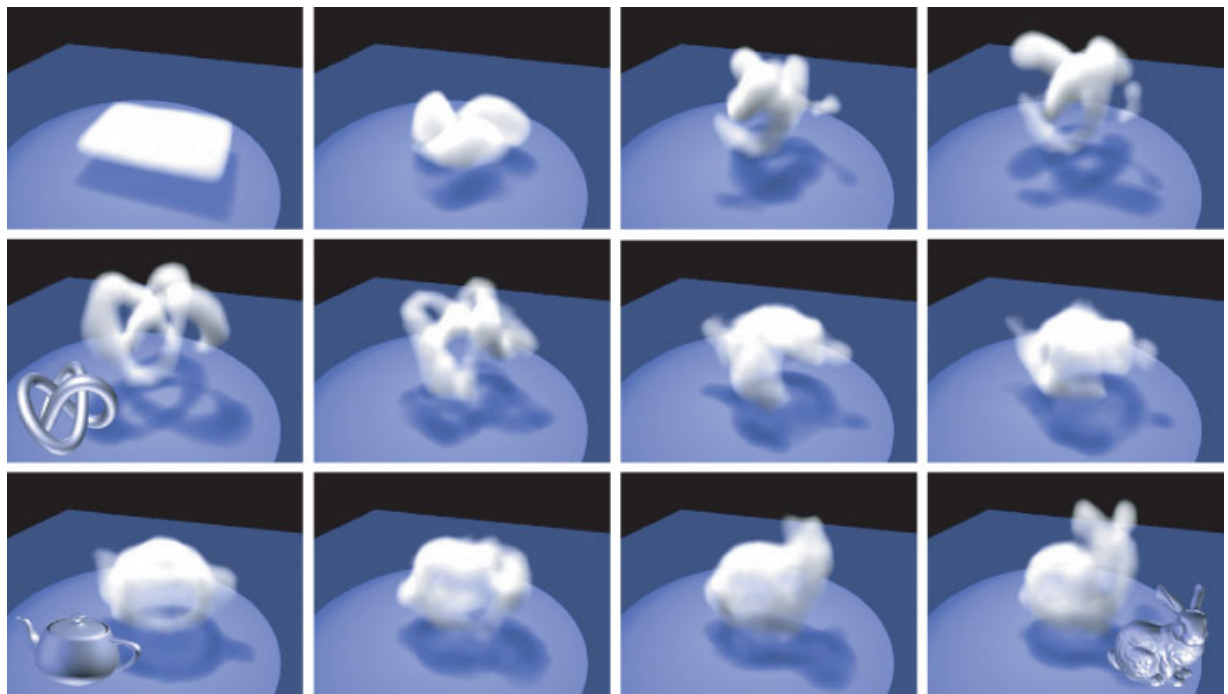


Figure 7. Three-dimensional keyframing—knot, teapot and rabbit model.

Model	Figure	Grid size	# of $p_{\text{target}}$	# of $p_{\text{start}}$	# of $p_{\text{check}}$	Time (ms)
Flower	3	120 × 200	1801	822	0	2361
Logo (a)	4(a)	200 × 150	2152	454	0	2189
Logo (b)	4(b)	200 × 150	2152	454	297	2233
Logo (c)	4(c)	200 × 150	2152	0	0	9106
Knot	7	38 × 28 × 28	401	0	0	1747
Teapot	7	38 × 28 × 28	1170	0	0	4952
Bunny	7	38 × 28 × 28	1383	0	0	5812

**Table I. Preprocessing times**

Example	Figure	Grid size	# of frames	# of markers	Mean fps
Teapot	1	40 × 35 × 30	141	6000	1.57
Logo	5	200 × 150	135	2270	0.590
Flower	6	120 × 200	350	8220	1.00
Keyframing	7	38 × 28 × 28	555	18000	0.449

**Table 2. Running times**

potential and gradient fields; all are negligible compared with the simulation time. If we have to change the target shape several times, it is more efficient to store the potential or gradient field in memory. Table 2 shows the run-time performance but, since our algorithm requires negligible run-time, this merely shows the performance of the Navier-Stokes solver. If we used a faster solver, then the whole process could be accelerated.

The example flows presented here are driven only by potential fields. Since our system can be implemented as a plug-in to an Navier-Stokes solver, it is obviously possible to combine our results with existing fluid effects, such as fluid shaping by interactions of dynamic objects or an externally specified wind.

## Conclusion

We have proposed a new fluid control technique which generates a virtual potential field to satisfy the geometric constraints given by a designer, but places little additional load on the Navier-Stokes solver. The novelty of our algorithm derives from our use of a potential dimension to configure the simulation space, thus reformulating the control problem as a common simulation problem with given initial values. We were able to control the fluid to form specific shapes without losing its realistic appearance, and we can do this at interactive

speeds. Our examples indicate how our method can help animators design creative fluid animation scenes.

We think that our techniques could be improved in various directions. More controllability can be obtained by adjusting  $k_{\text{start}}$ ,  $k_{\text{boundary}}$  and  $k_{\text{check}}$  using mathematical and engineering analysis developed for control theory. Furthermore, time-varying potential fields or velocity potentials could be applied to achieve particular flows, still without harming the performance of the solver. We will pursue these goals in future work.

## ACKNOWLEDGEMENTS

This research was supported by University IT Research Center Project. We would like to thank Young-rang Yoon for designing the elegant flower model. We would also like to thank Sun-jeong Kim, Jun-yong Ji and Jin-kyung Hong for discussing many aspects of this paper. We are deeply grateful to reviewers for their precise comments, which have improved the quality of this paper and will benefit our future work.

## References

1. Treuille A, McNamara A, Popović Z, Stam J. Keyframe control of smoke simulations. *ACM Transactions on Graphics (In Proceedings of ACM SIGGRAPH 2003)* 2003; **22**(3): 716–723.
2. Foster N, Metaxas D. Realistic animation of liquids. *Graphical Models and Image Processing* 1996; **58**(5): 471–483.
3. Foster N, Metaxas D. Controlling fluid animation. In *Proceedings of CGI '97, 1997*; 178–188.
4. Foster N, Metaxas D. Modeling the motion of a hot, turbulent gas. In *Proceedings of ACM Siggraph 1997, 1997*; 181–188.
5. Stam J. Stable fluids. In *Proceedings of ACM Siggraph 1999, 1999*; 121–128.
6. Foster N, Fedkiw R. Practical animation of liquids. In *Proceedings of ACM Siggraph 2001, 2001*; 23–30.
7. Fedkiw R, Stam J, Jensen HW. Visual simulation of smoke. In *Proceedings of ACM Siggraph 2001, 2001*; 15–22.
8. Enright D, Marschner S, Fedkiw R. Animation and rendering of complex water surfaces. *ACM Transactions on*

- Graphics (In Proceedings of ACM SIGGRAPH 2002)* 21(3): 736–744.
9. Hong J, Kim C. Animation of bubbles in liquid. *Computer Graphics Forum (In Eurographics 2003 Proceedings)* 2003; 22(3): 253–262.
  10. Feldman BE, O'Brien JF, Arikan O. Animating suspended particle explosions. *ACM Transactions on Graphics (In Proceedings of ACM SIGGRAPH 2003)* 2003; 22(3): 708–715.
  11. Popović J, Seitz SM, Erdmann M, Popović Z, Witkin A. Interactive manipulation of rigid body simulations. In *Computer-Graphics (ACM SIGGRAPH 2000)*, 2000.
  12. Sethian JA. *Level Set Methods and Fast Marching Methods*. Cambridge University Press: Cambridge, 1999.
  13. Zhao H-K, Osher S, Fedkiw R. Fast surface reconstruction using the level set method. In *Proceedings of 1st IEEE Workshop on Variational and Level Set Methods, 2001*; Vancouver: Canada, 194–202.
  14. Turk G, O'Brien JF. Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH 2000, 1999*; 335–342.
  15. Adler R, Bazin M, Schiffer M. *Introduction to General Relativity*, 1975. McGraw-Hill.
  16. Blakely RJ. *Potential Theory in Gravity and Magnetic Applications*. 1995. Cambridge University Press: Cambridge.
  17. Franklin GF, Powell JD, Emami-Naeini A. *Feedback Control of Dynamic Systems*, 2002. Prentice Hall: Englewood Cliffs, New Jersey.
  18. Chenney S, Forsyth DA. Sampling plausible solutions to multi-body constraint problems. In *Proceedings of ACM SIGGRAPH 2000, 2000*; 219–228.
  19. Fox RW, McDonald AT, Pritchard PJ. *Introduction to Fluid Mechanics*, 1993. John Wiley and Sons: Chichester.
  20. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C*, 1988. Cambridge University Press: Cambridge.