Graphical Models xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Detail-preserving fluid control

N. Thürey [a,b,*], R. Keiser [a], M. Pauly [a], U. Rüde [b]

[a] Applied Geometry Group, IFW C28, ETH Zurich, Switzerland
[b] System Simulation Institute (LSS), University Erlangen-Nuremberg, Germany

ARTICLE INFO

Article history:
Received 24 February 2007
Accepted 30 December 2008
Available online xxxx

Keywords:
Fluid simulation
Detail preservation
Animation control

ABSTRACT

We propose a new fluid control technique that uses scale-dependent force control to preserve small-scale fluid detail. Control particles define local force fields and can be generated automatically from either a physical simulation or a sequence of target shapes. We use a multi-scale decomposition of the velocity field and apply control forces only to the coarse-scale components of the flow. Small-scale detail is thus preserved in a natural way avoiding the artificial viscosity often introduced by force-based control methods. We demonstrate the effectiveness of our method for both Lagrangian and Eulerian fluid simulation environments.

1. Introduction

While realism is an important aspect for physically based animations, the practical use of fluid simulations for animation is also determined by the ability to efficiently control the behavior of the fluid. In general, animators prefer to have uncontrolled physical simulations only where absolutely necessary. Since fluid motion is typically very hard to predict, it is difficult to achieve a specific fluid behavior only by changing the corresponding global parameters. For some cases, such as animations of characters consisting of fluid, there is no real-world reference, and thus no way to really validate such a simulation. Still, it is required that such an animation looks physically plausible. Animators are mostly interested in modifying the large-scale motion of the fluid, without having to specify fine-scale detail such as small vortices or drops. High-level control is thus crucial in production environments (see Fig. 1).

The method presented in this chapter makes use of control particles, similar to [1]. Particles are a natural choice, as they are established tools in all major 3D-applications, can be intuitively handled, and animators are familiar with particle systems for creating various other effects. Since control particles are independent of the underlying fluid model they can be integrated easily in different flow simulation environments. In addition, many different control scenarios can be implemented, such as scripting, keyframing, or coarse-to-fine simulations. It will furthermore be shown how control particles can be automatically generated from pre-computed functions, an animated target shape, or an existing flow simulation. Directly enforcing control from the particles onto the fluid can lead to noticeable distortions of the velocity field, which is noticeable as an increased viscosity. To avoid this artificial viscosity, the velocity field is decomposed into coarse- and fine-scale components and the control forces are only applied to the low-frequency part. High-frequency components are largely unaffected, thus small-scale detail and turbulence are significantly better preserved. An example of this effect can be seen in Fig. 2. Techniques that make use of similar decompositions are used for example in geometry processing or motion capture editing. The decomposition is achieved by smoothing the velocity field using a low-pass filter given by the influence kernels of the control particles. Velocity control forces are then computed with respect to the smoothed velocity field. Although the following description will focus on the grid-based LBM, the control mechanism is also applicable to other grid-based solver (e.g., level set methods) or Lagrangian fluid solvers, such as smoothed particle hydrodynamics (SPH).

* Corresponding author. Address: Applied Geometry Group, IFW C28, ETH Zurich, Switzerland.
E-mail address: thuereyn@inf.ethz.ch (N. Thürey).

1524-0703/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2008.12.007

Fig. 1. A controlled fluid character.

In [3] Foster and Metaxas were the first to propose the embedding of controllers to control pressure and velocity of the flow. well as the fluid surface. This concept is further extended in [1] by sampling 3D parametric space curves with oriented points to locally alter the velocity of the fluid. Space curves are also used in [10] to model flames. These curves evolve according to physics-based, procedural, and manually defined wind fields. Feldman et al. [4] already demonstrates the capabilities of particle-based fluid control for animating explosions. Rasmussen et al. [18] introduce viscosity, velocity divergence and level set particles for melting, expansion and contraction of the liquid. Treuille et al. [23] presents an optimization technique to solve for control parameters such that simulated smoke matches the given density and velocity keyframes. The efficiency is improved by adopting the adjoint method for solving the nonlinear optimization problem in [14]. As control with the adjoint method is a more general framework, it can also be applied to, e.g., particle systems and cloth [24]. The authors of [17], on the other hand, demonstrate an approach that makes use of radial basis functions to control flow simulations. Fattal and Lischinski [2] proposed the idea of driving smoke towards target smoke density states by introducing a force term and counteract diffusion of smoke by adding a gathering term to the Euler equations. This simple technique is significantly faster than the previously mentioned approaches. Hong and Kim [5] derive potential fields from the initial distribution of smoke and a target shape. The force field is then defined as the gradient of this potential field. Shi and Yu control both smoke [21] and liquids [22] by matching the level set surface of the fluid with static or moving target shapes. Velocity constraints at the boundary force the fluid into the desired shape. While for smoke a compressible fluid model can be used [21], the velocity field needs to be divergence-free for liquids to guarantee mass preservation, as described in [22]. This is achieved by changing the boundary forces such that the flux at the boundary is zero, yielding a constraint minimization problem.

2. Fluid simulation models

We briefly discuss the two fluid simulation models, smoothed particle hydrodynamics and the Lattice–Boltzmann method that we use to demonstrate our control method.

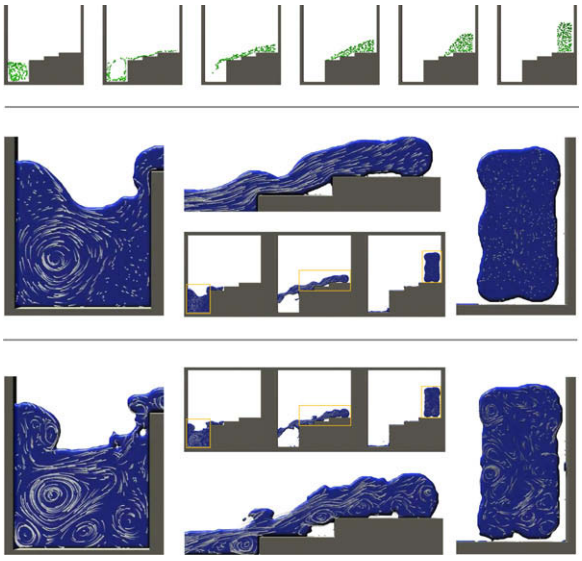

Fig. 2. Comparison of direct velocity control (middle) and scale-separated velocity control (bottom). Both simulations are controlled by the same 250 control particles (shown in the top row).

Smoothed Particle Hydrodynamics (SPH) is an approximation method that can be used to numerically solve the Navier–Stokes (NS) equations. The fluid is sampled with particles that serve as interpolation points. A fluid property $A(\mathbf{x})$ at an arbitrary position $\mathbf{x}$ in space is computed from the fluid properties A_j of the neighboring particles p_j as

$$A(\mathbf{x}) = \sum_j A_j V_j W(\mathbf{x} - \mathbf{x_j}, h), \tag{1}$$

where V_j and $\mathbf{x_j}$ are the volume and position of p_j, respectively. The normalized kernel function W depends on the distance of a particle to $\mathbf{x}$ and a length scale h. Usually, kernels similar to a Gaussian but with compact support are chosen for computational efficiency reasons. For detailed descriptions of the SPH method we refer to [13,11]. For our SPH simulations we used the framework presented by Müller et al. [12].

The Lattice–Boltzmann Method (LBM) is a grid-based technique that was derived from discrete gas molecule simulations. Each grid cell stores a set of *distribution functions* (DFs), which represent an amount of fluid moving with a fixed velocity. We use the common three-dimensional LBM model D2Q19 with nineteen grid velocities $\mathbf{e}_i$, $i = 1,\ldots,19$. The macroscopic fluid properties, such as density ρ and velocity $\mathbf{v}$ can be calculated by summation of the DFs, and are needed to calculate the equilibrium DFs for a cell:

$$f_i^{eq}(\rho, \mathbf{v}) = w_i \left[\rho + 3\mathbf{e_i} \cdot \mathbf{v} - \frac{3}{2}\mathbf{v}^2 + \frac{9}{2}(\mathbf{e_i} \cdot \mathbf{v})^2 \right]. \tag{2}$$

For the D3Q19 model, the weights w are given as $w_1 = 1/3$, $w_{2\ldots7} = 1/18$, and $w_{8\ldots19} = 1/36$. The LBM algorithm proceeds by first handling the movement of the DFs, which is equivalent to copying them to their adjacent neighbors,

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

and then computes the molecular collisions that would have occurred during this movement. The collisions are handled by a relaxation towards the equilibrium with a relaxation factor ω that is given by the physical fluid viscosity:

$$f_i(\mathbf{x}, t + \Delta t) = (1 - \omega) f_i(\mathbf{x} - \mathbf{e}_i, t) + \omega f_i^{eq}. \quad (3)$$

More details of the algorithm and a derivation of the model can be found in [6,25]. To simulate a fluid with a free surface we use a method similar to the *Volume-of-Fluid* approach for NS solvers [9,15].

The control framework uses a set of particles that locally exert forces on the fluid. Control particles are generated using an implicit function, a sequence of target shapes, or another fluid simulation. They directly induce forces to attract the fluid or influence the velocity field. It will be demonstrated that these two forces can be used for a wide variety of effects. It will furthermore be shown how the small-scale detail can be retained by applying the control forces only on the coarse flow of the fluid. An overview of the framework can be seen in Fig. 3.

3. Generating control particles

In the following, three different methods to generate control particles will be described. The easiest way to create control particles is with a given pre-computed function as described in [3,1]. To perform fluid simulations with a given target shape, the initial triangle shape mesh is first regularly sampled. The control particles are then displaced for each mesh of the animation sequence using mean value mesh coordinates, as described in [7]. Control particles can also be generated from other, possibly coarser, fluid simulations. Within an LBM simulation, massless tracer particles can be tracked in the fluid velocity field. Their positions are then used as control particles in a second simulation pass. Such a control simulation can usually be very coarse, and may even run in realtime to give instant feedback to an animator. It can likewise be controlled to yield the desired result. Control particle sets generated by a fluid simulation can be used to easily control large volumes of fluid. Furthermore, by changing or reversing the timing of the control particles interesting effects can be achieved. Both Figs. 2 and 9 make use of this approach.

4. Control forces

A control particle p_i is given by its position $\mathbf{p}_i$, velocity $\mathbf{v}_i$, and influence radius h_i. A constant radius h is chosen as 2.5 times the average sample distance of the control particles.

Fluid attraction is controlled using a force that pulls fluid towards the control particles. In order to preserve as much of the natural fluid behavior as possible, this force is scaled down when the influence region of the control particle is already sufficiently covered with fluid. Let V_e denote the volume of a fluid element e (a filled grid cell for the LBM, a mollified particle for SPH). A scale factor is defined for the attraction force as

$$\alpha_i = 1 - \min\left(1, \sum_e V_e W(d_{i,e}, h)\right), \quad (4)$$

where $d_{i,e} = \|\mathbf{p}_i - \mathbf{x}_e\|$ is the distance between p_i and the center $\mathbf{x}_e$ of fluid element e, and W is the control particle kernel function. A linear falloff function of width $h/2$ can be sufficient for W:

$$W(d,h) = \begin{cases} 1 & : d \leqslant h/2 \\ 2 - \frac{2d}{h} & : d > h/2,\ d < h \\ 0 & : d \geqslant h \end{cases} \quad (5)$$

However, in the following a normalized spline kernel with support h [12] will be used

$$W(d,h) = \begin{cases} \frac{315}{64\pi h^9}(h^2 - d^2)^3 & : d < h, \\ 0 & : d \geqslant h. \end{cases} \quad (6)$$

Summing up the attraction forces exerted by control particles p_i on a fluid element e then yields

$$\mathbf{f}_a(e) = w_a \sum_i \alpha_i \frac{\mathbf{p}_i - \mathbf{x}_e}{\|\mathbf{p}_i - \mathbf{x}_e\|} W(d_{i,e}, h), \quad (7)$$

where w_a is a global constant that defines the strength of the attraction force. If w_a is negative, Eq. (7) will result in a repulsive force.

While the attraction force pushes fluid towards control particles that are not covered with fluid yet, a second force is used to modify the velocity of the fluid according to the flow determined by the control particles. A velocity force

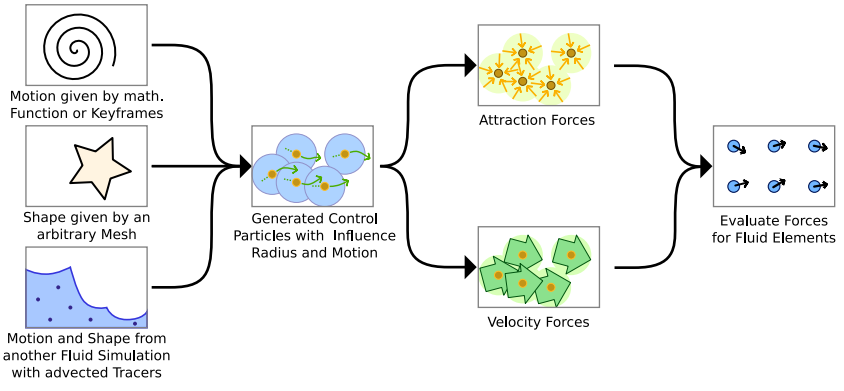

Fig. 3. Here an overview of the particle-based control framework can be seen.

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

per volume $\mathbf{f}_v$ for the fluid element e is defined similar to the attraction force

$$\mathbf{f}_v(e) = w_v \sum_i [\mathbf{v}_i - \mathbf{v}(e)] W(d_{i,e}, h), \tag{8}$$

where $\mathbf{v}(e)$ is the velocity of the fluid element e, and w_v a constant that defines the influence of the velocity force. Finally, the new total force per volume $\mathbf{f}(e)$ acting on teh fluid element is given by the sum of attraction, velocity and fluid forces

$$\mathbf{f}(e) = \mathbf{f}_a(e) + \mathbf{f}_v(e) + \mathbf{f}_f(e). \tag{9}$$

Here $\mathbf{f}_f(e)$ is the force given by the physical fluid simulation (e.g., gravity). Integrating $\mathbf{f}(e)$ gives the new velocity $\mathbf{v}'(e)$ of a fluid element that is then used for the next SPH step, or the calculation of the equilibrium distribution functions for the LBM. In order to apply the method to a level set based solver it will be necessary to ensure that the velocity field is divergence-free, e.g., as in [22]. For SPH based solvers the existing acceleration structures could be used to evaluate the control forces at the fluid particle positions.

5. Detail-preserving control

The velocity force of Eq. (8) effectively leads to an averaging of the fluid velocities with the control velocities, which introduces undesirable artificial viscosity. This effect is illustrated in Fig. 2, where 250 control particles force the fluid to flow up the stairs. The middle row of pictures is calculated with control forces as described above. Within the influence of the control particles, the fluid is unable to develop small-scale vortices and turbulent behavior.

It needs to be ensured that the fluid velocity matches that of the control particles without unnecessarily disturbing the natural small-scale fluid motion. To achieve this goal the overall fluid motion is separated from fine-scale detail using a low-pass filter on the current velocity field. Velocity forces are then computed with respect to the smoothed fluid velocities. The smooth velocity field $\tilde{\mathbf{v}}_f$ is obtained using an approximation of discrete convolution with the kernel W of the control particles:

$$\tilde{\mathbf{v}}_f(e) = \frac{\sum_i \tilde{\mathbf{v}}_i W(d_{i,e}, h)}{\sum_i W(d_{i,e}, h)} \quad \text{with} \quad \tilde{\mathbf{v}}_i = \frac{\sum_e \mathbf{v}'_f(e) W(d_{i,e}, h)}{\sum_e W(d_{i,e}, h)},$$

where the filtered velocity for each control particle $\tilde{\mathbf{v}}_i$ is computed with the current fluid velocities $\mathbf{v}'_f(e)$ given by the simulation. The process of evaluating these two equation is illustrated in Fig. 4. Note that $\tilde{\mathbf{v}}_f$ is computed with velocities of the control particles, while $\tilde{\mathbf{v}}_i$ is computed with a sum of fluid element velocities. The smoothed fluid velocity $\tilde{\mathbf{v}}_f(e)$ then replaces $\mathbf{v}(e)$ in Eq. (8).

To show that this new control force only modifies the low-frequency part, while retaining the high-frequencies, the control force $\mathbf{f}_c(e)$ is integrated separately, yielding $\mathbf{v}'_c = k(\mathbf{v}_p - \tilde{\mathbf{v}}_f)$. Here $\mathbf{v}_p$ is the interpolated velocity of the control particles at a fluid element e and k is a constant depending on the user parameter w_v from Eq. (8). By decomposing $\mathbf{v}'_f$ into the low-pass filtered velocity $\tilde{\mathbf{v}}_f$ and the high-frequency part $\Delta\mathbf{v}_f$, i.e. $\mathbf{v}'_f = \tilde{\mathbf{v}}_f + \Delta\mathbf{v}_f$, the new fluid velocity is given by

$$\begin{aligned}\mathbf{v}' &= \mathbf{v}'_f + \mathbf{v}'_c = \tilde{\mathbf{v}}_f + \Delta\mathbf{v}_f + k(\mathbf{v}_p - \tilde{\mathbf{v}}_f) \\ &= (1-k)\tilde{\mathbf{v}}_f + k\mathbf{v}_p + \Delta\mathbf{v}_f. \end{aligned} \tag{10}$$

Hence, the low-frequency part of the fluid velocity is blended with the velocity of the control particles, while the high-frequency part is retained, as sketched in Fig. 5. The bottom row of Fig. 2 shows the effect of the scale-separated force control that significantly better preserves fine-scale fluid motion.

To fine tune the effect of the velocity control, another parameter can be introduced to linearly blend between direct and detail-preserving control. Using a weight slightly larger than one, this method can even be used to artificially increase and reinforce the fluid details. This technique can be used similar to the vorticity confinement and vortex particle methods of [20,16,19].

6. Results

For the implementation, the control particles are rasterized to the LB grid using early reject tests that prevent unnecessary evaluations of the influence forces. Due to the small changes of a single LB step it is sufficient to update the control force array in intervals. For the simulations presented here the forces are updated every 32 LB steps. To include the control forces into the LBM, the equilibrium DF is computed with the modified fluid velocity. Eq. (3) for the collision is thus changed to

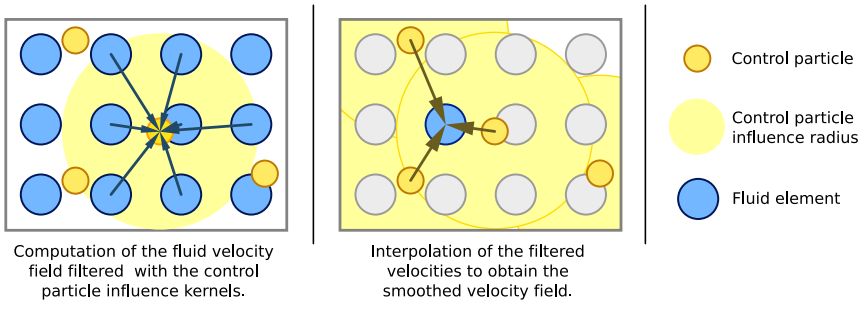

Fig. 4. This figure illustrates the process of computing the filtered fluid velocity field for each control particle, and its interpolation back to the fluid elements.

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

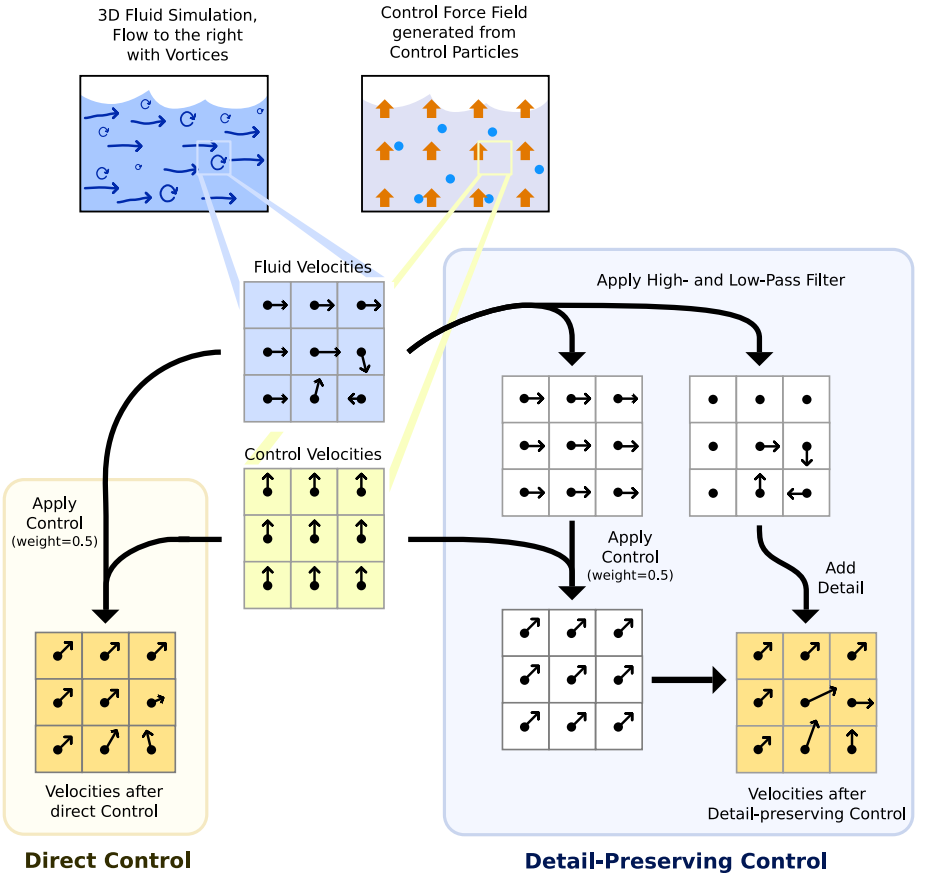

Fig. 5. Overview of the two possibilities for applying velocity control – the lower left velocity field shows the effect of direct velocity control. The velocities of the vortex are distorted significantly. The lower right velocity field shows the preserved vortex with the improved velocity control method.

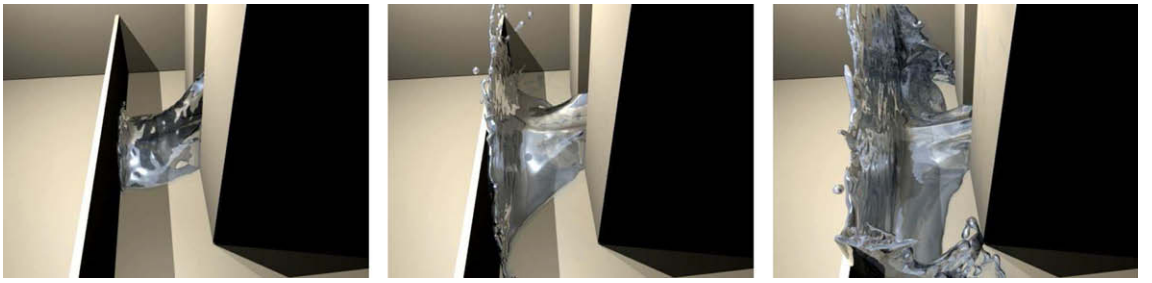

Fig. 6. The uncontrolled breaking dam simulation. The water splashes to both sides at the T-junction.

$$f_i(\mathbf{x}, t+\Delta t) = (1-\omega) f_i^*(\mathbf{x}, t+\Delta t) + \omega f_i^{eq}(\mathbf{v}+\mathbf{f}(x), \rho). \quad (11)$$

The effect of the detail-preserving control approach is shown in Fig. 7, where a column of fluid splashes against a wall at a T-junction. Without control, the water symmetrically distributes on both sides, as can be seen in Fig. 6. Fig. 7 shows two simulations where a coarse set of 216 control particles forces the fluid to flow only towards the left. As can be seen in the upper two pictures, the direct velocity control introduces artificial viscosity, smoothing out the turbulence and vortices of the flow. The pictures in the lower row show the resulting fluid motion using the detail-preserving approach described in Section 5. The controlled flow with detail-preservation retains small-scale fluid features and therefore yields a more natural and interesting behavior. The simulation was performed with a $240 \times 120 \times 120$ grid resolution, which took 38 s per frame on average (without rendering) on a standard Pentium IV 3 GHz PC. The computation of the control forces took 2–4% of the total computation time.

In Fig. 8 we demonstrate target matching on a running horse example, similar to [22]. Note that unlike in [22], the mesh is only used to generate a sequence of control particles as described in Section 3. During the simulation, the

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

Fig. 7. Comparison of direct velocity control (left column) and detail-preserving control (right column) for the T-junction breaking dam.

Fig. 8. An SPH fluid following the shape of an animated model. Control particles are created by sampling the interior of the horse.

attraction forces cause the fluid to flow into the given shape sampled with control particles, while the fluid follows the motion of the mesh due to the velocity force. We sampled the initial horse mesh regularly with 69 k control particles, and used 266 k particles for the SPH simulation which took 102 s per frame, including the computation of the control forces which took 14 s per frame.

In Fig. 9 the fluid is forced to flow up several stairs and form a human figure. For the first part of the animation, 500 control particles are used. These are generated from a time-reversed coarse simulation of fluid flowing down the stairs. As the fluid reaches the upper platform, these control particles are blended with 5 k control particles sampled from a 3D model of the human figure. The simulation was performed on a 300^3 grid resolution and took 142 s per frame, including on average 4 s for computing the control forces (see Fig. 10).

These examples demonstrate the reduced artificial viscosity of the detail-preserving control. As the width of the influence radius of a control particle is coupled to the filtering of the velocity field, the scale of detail-preservation is determined by the number of control particles. This allows large-scale control with a low number of control particles, while additional sets of finer control particles can be used to modify smaller scales. In the future, the method could be extended to include anisotropic influence kernels, which could allow finer control with fewer control particles. The framework could also be used to control the deformation of elastic bodies, similar to [8]. Furthermore, it would be useful for practical applications to determine the influence parameters directly, e.g., from the motion of a

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

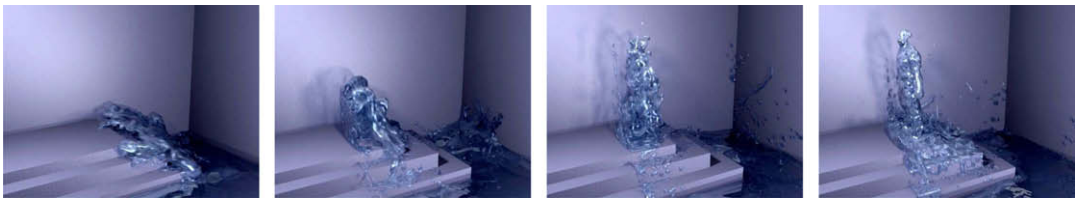

Fig. 9. A fluid simulation is controlled to flow up the stairs and form a human figure.

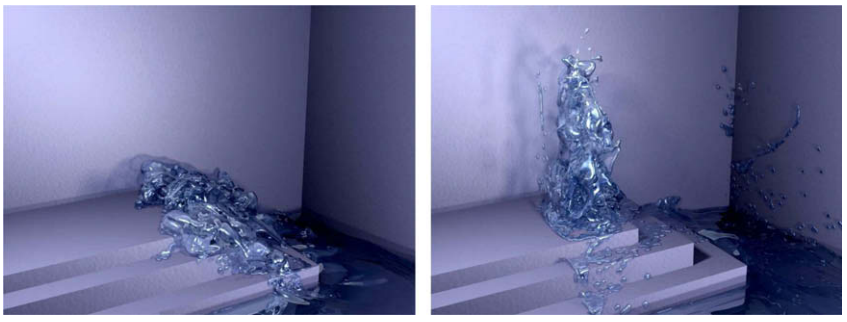

Fig. 10. Two enlarged images from the sequence shown in Fig. 9.

target shape. This could help users to more easily achieve a desired fluid motion. There are number of limitations and possible improvements for the future. Since control is force-based, it is difficult to define hard constraints on the flow, e.g., to guarantee that fluid does not enter a certain region of the simulation environment. So far the control particles have a radial symmetric kernel. Anisotropic kernels would enable better directed control of the simulation. Finally, for grid-based methods, Fourier techniques could be an interesting alternative to explicit smoothing of the velocity field.

7. Conclusions

We presented a detail-preserving approach for controlling fluids based on control particles. The fluid flow is modified by attraction and velocity forces exerted from the control particles. Several possibilities for generating control particles have been used. This simple control scheme showed to be very flexible and efficient, and allows to simulate various effects such as filling and following moving target objects or time-reversed flows. We solve the problem of artificial viscosity introduced by the control forces by applying these forces on the low-pass filtered velocity field. Therefore, only the coarse-scale flow of the fluid is modified while the natural small-scale detail is preserved, resulting in more natural looking controlled simulations.

References

[1] Nick Foster, Ronald Fedkiw, Practical animation of liquids, in: Proc. of ACM SIGGRPAH, 2001, pp. 23–30.
[2] Raanan Fattal, Dani Lischinski, Target-driven smoke animation, ACM Trans. Graph. 23 (3) (2004) 441–448.
[3] Nick Foster, Dimitris Metaxas, Controlling fluid animation, in: Proc. of CGI, 1997.
[4] Bryan E. Feldman, James F. O'Brien, Okan Arikan, Animating suspended particle explosions, in: Proc. of ACM SIGGRAPH, 2003, pp. 708–715 (August).
[5] Jeong-Mo Hong, Chang-Hun Kim, Controlling fluid animation with geometric potential: research articles, Comput. Animat. Virtual Worlds 15 (3-4) (2004) 147–157.
[6] X. He, L.-S. Luo, Lattice Boltzmann model for the incompressible Navier–Stokes equations, J. Stat. Phys. 88 (1997) 927–944.
[7] Tao Ju, Scott Schaefer, Joe Warren, Mean value coordinates for closed triangular meshes, ACM Trans. Graph. 24 (3) (2005) 561–566.
[8] Ryo Kondo, Takashi Kanai, Ken-ichi Anjyo, Directable animation of elastic objects, in: Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation, 2005.
[9] C. Körner, M. Thies, T. Hofmann, N. Thürey, U. Rüde, Lattice Boltzmann model for free surface flow for modeling foaming, J. Stat. Phys. 121 (1–2) (2005) 179–196. October.
[10] Arnauld Lamorlette, Nick Foster, Structural modeling of flames for a production environment, in: Proc. of ACM SIGGRAPH, 2002, pp. 729–735.
[11] G.R. Liu, M.B. Liu, Smoothed particle hydrodynamics, A Meshfree Particle Method, World Scientific Publishing, 2003.
[12] Matthias Müller, David Charypar, Markus Gross, Particle-based fluid simulation for interactive applications, in: Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation, 2003.
[13] J.J. Monaghan, Smoothed particle hydrodynamics, Rep. Prog. Phys. 68 (2005) 1703–1758.
[14] Antoine McNamara, Adrien Treuille, Zoran Popovic, Jos Stam, Fluid control using the adjoint method, ACM Trans. Graph. 23 (3) (2004) 449–456.
[15] N. Thürey, C. Körner, U. Rüde, Interactive Free Surface Fluids with the Lattice Boltzmann Method, Technical Report 05-4. Technical Report, Department of Computer Science 10 System Simulation, 2005.
[16] Duc Quang Nguyen, Ronald Fedkiw, Henrik Wann Jensen, Physically based modeling and animation of fire, in: SIGGRAPH'02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, NY, USA, 2002, pp. 721–728.
[17] Frederic Pighin, Jonathan M. Cohen, Maurya Shah, Modeling and editing flows using advected radial basis functions, in: Proc. of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM Press, 2004, pp. 223–232.
[18] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, R. Fedkiw, Directable photorealistic liquids, in: Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation, 2004.

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007

[19] Andrew Selle, Nick Rasmussen, Ronald Fedkiw, A vortex particle method for smoke, water and explosions, ACM Trans. Graph. 24 (3) (2005) 910–914.
[20] J. Steinhoff, D. Underhill, Modification of the Euler equations for vorticity confinement: application to the computation of interacting vortex rings, Phys. Fluids 6 (8) (1994) 2738–2744.
[21] Lin Shi, Yizhou Yu, Controllable smoke animation with guiding objects, ACM Trans. Graph. 24 (1) (2005).
[22] Lin Shi, Yizhou Yu, Taming liquids for rapidly changing targets, in: Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation, 2005.
[23] Adrien Treuille, Antoine McNamara, Zoran Popovic, Jos Stam, Keyframe control of smoke simulations, ACM Trans. Graph. 22 (3) (2003) 716–723.
[24] Chris Wojtan, Peter J. Mucha, Greg Turk, Control of complex particle systems using the adjoint method, in: Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2006.
[25] Dazhi Yu, Renwei Mei, Li-Shi Luo, Wei Shyy, Viscous flow computations with the method of Lattice Boltzmann equation, Prog Aerospace Sci. 39 (2003) 5.

Please cite this article in press as: N. Thürey et al., Detail-preserving fluid control, Graph. Models (2009), doi:10.1016/j.gmod.2008.12.007