Moving with the Flow: Wave Particles in Flowing Liquids

Hilko Cords
Institute of Computer Science, University of Rostock, Albert-Einstein-Str. 21, 18051 Rostock, Germany
hilko.cords@uni-rostock.de

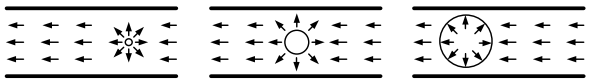
ABSTRACT

This paper presents a real-time water simulation framework with respect to wave propagation in flowing liquids. The presented method is based on the coupling of liquid flow simulation and simulation of surface waves. Physically, our system combines the solution of the Navier-Stokes Equations with the solution of the 2D Wave Equation. Numerically, we combine the concept of wave particles with a FDM-based flow simulation. Thus, wave propagation in fast flowing liquids (e.g., a creek) can be simulated – in real-time. Therefore, it is very suitable for today's video games and VR-environments. Finally, we discuss the coupling of the liquid simulation with a rigid-body simulation and a fountain simulation.

Keywords: Real-time liquid simulation, rigid-body simulation, water simulation, wave particles, stable fluids, fountain.

1 INTRODUCTION

Due to the immense computational costs of simulation, surface extraction and rendering the real-time interactive simulation of liquids is an ongoing challenge. This paper presents a new real-time method to simulate detailed surface waves moving with the flow. Think e.g., of a moving river: Rain drop impacts create radial propagating surface waves. These circular waves are moving with the river flow.



This effect depends on two interfering physical properties of the liquid:

- 3D flow,
- 2D surface wave propagation.

The river flow depends on moving water masses. These masses act as the transport medium for the surface waves. The global surface movement can be described physically by superposition of both movements. In complex flows or vortexes, this effect can result in spectacular chaotic wave movements. In a fast flowing creek, the mentioned circular waves (rain drop impacts) can be deformed strongly by the flow.

The flow is described by the Navier-Stokes-Equations and the propagation of surface waves can be described by the 2D Wave Equation. In the field of computer graphics, the Navier-Stokes-Equations are typically solved with a finite difference approach (FDM) or the smoothed particle hydrodynamics (SPH) method. Usually, the 2D Wave Equation is solved with a finite difference approach. The coupling of both simulations requires the handling of velocities in the 2D Wave Equation solver. The straight forward introduction of velocities in a finite difference solver results in numerical instabilities due to the discretization of the 2D wave field.

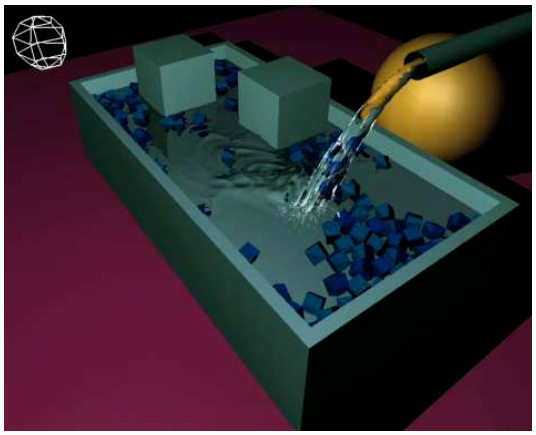


Figure 1: Real-Time Water at 33 FPS.

This paper presents a stable and fast method to combine the flow simulation with a surface simulation, including the movement of surface waves with the flow and the deformation by the flow. Our work builds on the concept of *wave particles*, which has been presented by Yuksel et al. [YHK07]. They use a particle system for approximating the 2D Wave Equation and achieve realistic and plausible results for water surfaces in real-time. In principle, we add to their concept of surface wave propagation the handling of flowing liquids. Hence, surface wave propagation in moving liquids can be simulated (Fig. 1).

Moreover, we present a fast liquid–rigid body coupling and a liquid-fountain coupling. The concepts has been evaluated in practice, achieving good performance and plausible results. Hence, our approach is practical for interactive environments, e.g., VR's or video games.

2 RELATED WORK

Two dimensional radial propagating waves are described by the Wave Equation. Until recently, the Wave Equation is solved with a finite difference approach [Gom00] to simulate liquids in the field of computer graphics. Lately, Yuksel et al. presented an alternative approximating method to solve the Wave Equation [YHK07]. They use particles to simulate the propagating waves with reasonable results at high frame rates. They also present a coupling of rigid body simulation and wave particles – resulting in convincing motor boat simulation in real-time.

The Navier-Stokes equations are usually solved with the Lagrangian approach (particle-based systems, e.g., Smoothed Particle Hydrodynamics - SPH), the Eulerian approach (finite difference approach, FDM) or hybrid methods – at least in the field of computer graphics. Large quantities of water or complex effects are still impossible to simulate physically based at high frame rates on current personal computers. Stam et al. presented the first real-time approach for simulating gaseous phenomena using SPH [SF95]. This approach has been extended to the interactive simulation of liquids in [MCG03] allowing simulations with a few thousand particles at interactive rates. A GPU based implementation has pushed the limit of simulated particles in real-time immensely [HKK07]: Tens of thousands of particles are simulated on the GPU (GeForce 8800GTX). However, the particles are just drawn as points – no surface reconstruction algorithm in real-time has been proposed, demonstrating the complexity of the surface reconstruction itself. Highly detailed, but still interactive liquids at real-time framerates (including surface extraction and rendering) can be reached by superposition of the SPH method and a FDM based wave equation solver [Cor07a].

The FDM approach for simulating liquids has been introduced to the computer graphics community as the marker and cell method [HW65] and has also become popular [CMT04] [FF01], whereas the implicit technique for interactive simulation of fluids was introduced in [Sta99]. This approach has been enhanced for execution on the GPU with reasonable frame rates [Har05].

To increase performance, approaches have been presented that replace the 3D Navier-Stokes equations by 2D ones and extract a height-field from density values [CdVL95]. In this context, column-based height-field approaches have become popular [OH95] [MFC06]. The 3D simulation space is displaced by columns, decreasing the computational costs immensely. Beside liquid flow simulation, these approaches can also simulate the propagation of surface waves at interactive rates. Lately, the Lattice Boltzmann method has been introduced to the field of computer graphics [TSS+07]: Instead of solving the Navier-Stokes equations directly, the Boltzmann equation of kinetic theory of gases is solved. For neatly chosen parameters, the resulting flows are equivalent and can be simulated at interactive rates.

One of the main bottlenecks for interactive animations is the surface or volume reconstruction from simulation data. In the majority of cases a height-field based rendering approach is used. The benefit of such a 2,5D approach is

- fast surface construction and rendering,
- high resolution, and
- the possibility of neat approximations of reflection and refraction effects.

However, specific complex liquid phenomena, such as breaking waves or splashes, cannot be visualized as height-fields. Real-time rendering of height-field based *water* surfaces (including reflection and refraction effects) is usually either based on environment mapping techniques or approximating raytracing techniques. The former approach includes a fast technique, splitting the environment along the water surface [Sou05] [SW01]. Even though non-physical, this approach achieves good visual results and performance [Bel03] when visualizing lakes and ocean. This approach has recently been expanded for better approximation of physical refraction for objects intersecting the water surface [Cor07b]. Using two height-fields (ground and surface field), a simplified raytracer can be realized on the GPU [BD06]: The intersection with refracted and reflected rays is calculated on the GPU with reasonable frame rates. Other GPU-based approaches towards real-time raytracing built upon some major simplifications have been presented [SKALP05] [PMDS06].

Alternatively to the height-field based approaches, an expensive marching cubes algorithm [LC87] can be used, substantially reducing the possible interactive simulated amount of liquid but is extracting a real 3D surface. However, rendering a 3D refractive surface in a complex environment is still an unsolved problem. An image-based double refraction (front and back faces), for instance, can be realized with a two pass rendering approach using an environment map [Wym05]: The refractive object is split into back and front faces and is rendered GPU-based. Restricted to static environment maps (lying at infinite distance), the approach achieves reasonable results at high frame rates. However, this approach cannot refract objects intersecting the water surface. In other cases, a surface splatting technique [ZPvBG01] is used for rendering the surface – anticipating complex refraction effects in real-time as well. Recently, the projected grid method [Joh04] has been extended to the rendering of particle based liquids [MSD07], featuring good performance and splashing effects.

3 OUR APPROACH

According to liquid simulations solving the full 3D Navier-Stokes equations for incompressible flows in real-time, we observe the following problem: Due to the constraints of real-time, a relatively low sampling density of the liquid volume has to be used (Eulerian methods: small grid-size, Lagrangian methods: few particles). Thus, real-time simulated, fast-flowing liquids tend to show rough surface waves. Hence, we propose the coupling of a 2D flow simulation and a surface wave simulation to up-sample the surface (Sect. 3.1).

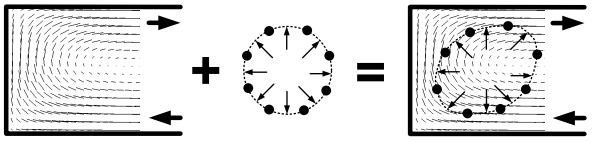


Figure 2: The basic idea: A 2D flow simulation (here: In- and Outflow, left) is coupled with a surface wave simulation (here: circular wave, center), resulting in wave deformation (right).

We aim at fast moving liquid masses with surface waves moving with the flow. Hence, fast and detailed surface effects (e.g., a moving boat, or the wave effect of a static object in the flow) can be simulated. To reach a global flow propagation of a liquid with low viscosity (e.g., water), we use an implicit grid-based Navier-Stokes simulator (Sect. 3.1.1) – similar to [Sta99]. Due to the particle based approach, the Wave Particle concept [YHK07] (Sect. 3.1.2) is well suited to be coupled with such a Navier-Stokes simulation (Fig. 2, Sect. 3.1.3). Our rendering method, including the approximation of refraction effects, is described in Sect. 3.1.4.

Finally, we demonstrate the application of our method in combination with a rigid body simulation (Sect. 3.2) and a fountain simulation (Sect. 3.3).

3.1 Liquid

The key idea of our liquid simulation is shown in the following Table:

Region	Dim.	Simulation	Principle
Liquid Flow	2D	FDM	N.-S. Eq.
Surface	2D	Wave Particles	Wave Eq.

In the next two Sections both principles are described separately. In Sect. 3.1.3 the coupling of both concepts is introduced (Fig. 2).

3.1.1 Flow

The dynamic flow of liquids is described physically by the conservation of momentum (Navier-Stokes equations)

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t}+(\mathbf{v}\cdot\nabla)\mathbf{v}\right)=-\nabla p+\mu\triangle\mathbf{v}+\rho\mathbf{f} \qquad (1)$$

and the conservation of mass (continuity equation):

$$\frac{\partial \rho}{\partial t}+\nabla\cdot(\rho\mathbf{v})=0, \qquad (2)$$

where $\mathbf{v}$ is the velocity field, ρ the density field, p the pressure field, $\nabla=(\frac{\partial}{\partial x_1},\frac{\partial}{\partial x_2},\frac{\partial}{\partial x_3})^T$, $\triangle$ is the Laplacian with $\triangle=\nabla^2$, μ is the viscosity and $\mathbf{f}$ is the acting external force (e.g. gravity). Since we assume incompressible liquids, the density is constant ($\frac{\partial\rho}{\partial t}=0$), resulting in the mass conservation $\nabla\mathbf{v}=0$.

As mentioned above, we use an Eulerian method to solve these differential equations to reach a global flow propagation. A lot has been written about algorithms for solving these equations. We use the implicit Stable Fluids algorithm presented by Stam et al. [Sta99]. This algorithm features a good trade-off between accuracy, stability and performance. It is based on an implicit finite difference solving scheme (for a detailed description of the algorithm we refer to [Sta99]). Additionally, to determine the pressure physically, a Poisson Equation has to be solved numerically – we use the Gauss-Seidel method for good performance. Furthermore, collision objects (or boundary conditions) can be handled as slip or no slip objects: The velocity-elements along the contour of collision objects are set to the opposite value of the direction of the neighboring liquid grid-element (slip) or they are set to zero (no-slip).

3.1.2 Surface Waves

The general Wave Equation describes the propagation of waves in time t and space $\mathbf{x}$. For liquid surface waves the 2D Wave Equation can be used, describing radial wave propagation (e.g., a rain drop impact on flat water surface):

$$\triangle f(\mathbf{x},t)-\frac{1}{c^2}\frac{\partial^2 f(\mathbf{x},t)}{\partial t^2}=0. \qquad (3)$$

Here, $\triangle=\nabla^2=\sum_1^2\frac{\partial^2}{\partial x_i^2}$ is the Laplace operator in 2D and c is the velocity at which waves propagate across the surface. The idea of wave particles is to describe this propagation with particles (Fig. 2). In the following, we give just a brief overview of wave particles (we refer to [YHK07] for more details): An impact results in the creation of circular aligned wave particles with radial velocity. Hence, the particles are moving radially and describe the propagation of a radial wave. Due to the particle discretization, the sampling of the propagating circle decreases. In other words, the distance between particles increases. If the distance reaches a threshold d, each particle is subdivided into three particles with distance $\frac{d}{3}$. This value becomes the new threshold d as well. Hence, a minimum sampling of the wave is guaranteed. The distance d can be determined without expensive neighbor comparisons. The

distance only depends on the radius r and the number of particles n of the actual wave (circumference: $2\pi r$):

$$d = \frac{2\pi r}{n}. \tag{4}$$

Thus, every particle is animated independently, resulting in very fast simulation. By creating several impacts, wave trains, rain drop impacts or the typical wave propagation of a moving boat are simulated. Beside circular waves, the technique is also useful to simulate the wave creation process of swimming objects or linear waves. The dispersion can be influenced by several parameters: In combination with different maps, the damping or the velocity of propagation can be described according to the type of ground.

Boundary conditions (e.g., collision objects) can be included easily as particle reflections from boundaries: Adapting positions and velocities at boundary collisions models the wave reflection process. Due to the independent movement of the wave particles, physical wave interference is guaranteed. Merely, diffraction has to be modeled separately at edges. But in curved water this inaccuracy is hardly noticeable.

To generate a height-field from a wave particle system, all particles are rendered anti-aliased into a 2D texture $t(x,y)$, which is filtered in both directions (normalized filter). The surface normals are calculated straightforwardly. As mentioned in [YHK07] an extended height-field can enhance the realism of waves: Waves under influence of wind become spiky. Therefore, the regularity of x and y positions of the height-field grid is resolved. We manipulate the vertexes of the height-field as follows:

$$\begin{pmatrix} x \\ y \\ t(x,y) \end{pmatrix} \rightarrow \begin{pmatrix} x + c \cdot \frac{\partial}{\partial x} t(x,y) \\ y + c \cdot \frac{\partial}{\partial y} t(x,y) \\ t(x,y) \end{pmatrix}. \tag{5}$$

The intensity of the height-field extension can be controlled by the parameter c.

3.1.3 Coupling

This work targets surface waves on fast flowing liquids. Thus, the wave particles have to be coupled with the flow. Since the surface waves are moving with the flow but do not create a flow by themselves, the coupling is one-way:

$$\text{Flow} \rightarrow \text{waves}.$$

That is, the flow simulation can be executed autonomously, whereas the wave simulation depends on the flow simulation. Since the waves are represented by autonomous wave particles, the wave particles have to be manipulated according to the velocity of the liquid flow at the same position (Fig. 2). Since the discretization of the flow depends on a discrete grid of size $n \times m$, the flow velocity $\mathbf{v}_{\text{flow}}(\mathbf{x})$ at an arbitrary position $\mathbf{x}$ should be determined using a bilinear filtered grid. Thus, aliasing artifacts are avoided. The position $\mathbf{p}^n$ of each particle n is influenced explicitly by the determined floating velocity $\mathbf{v}_{\text{flow}}$:

$$\mathbf{p}^n = \mathbf{p}^n + \triangle t \cdot \mathbf{v}_{\text{flow}}(\mathbf{p}^n). \tag{6}$$

Hence, the coupling of wave propagation and liquid flow is simulated according to nature. The collision handling of wave particles is not affected by the described method and can persist unchanged.

3.1.4 Rendering

This work aims at real-time liquid simulation at high frame rates. Thus, fast rendering is crucial and fast approximating techniques are favored. Non-transparent liquids can be rendered straightforwardly. The visual behavior of those substances can be approximated using specific shader programs. For some liquids (e.g., milk), a subsurface-scattering algorithm can increase realism. Standard reflection effects can be handled by environment mapping techniques – at least for surfaces with no intersecting objects.

For transparent liquids, refraction becomes important: We approximate refraction with a simple two-pass algorithm [Sou05]. The environment without liquid is rendered into a texture. This texture is accessed during the rendering of the height-field according to the surface positions in projection space and a slight variation according to the belonging surface normals. Thus, the refraction is roughly approximated – but without the typical lens and depth effects. However, this approach results in plausible results of refraction of several swimming objects or collision objects – and requires only one additional render pass. Furthermore, the use of effects on a per pixel basis (e.g., bumpmapping, chaotic reflexes, motion blurring) can enhance realism even more. In general, the animator can use the whole scope of shader effects to achieve desired effects.

3.2 Rigid Body Dynamics

One main application of flow simulation in computer graphics is the simulation of objects swimming in the flow. The realistic motion of objects in water depends on rigid body dynamics coupled with the flow and buoyant forces. The buoyant forces $\mathbf{F}_b$ equal the magnitude of the weight of liquid volume displaced by the rigid body V_{displ} (Archimedes' Principle):

$$\mathbf{F}_b = \rho_l \mathbf{F}_g V_{\text{displ}}, \tag{7}$$

where ρ_l is the average density of the liquid and $\mathbf{F}_g = m \cdot \mathbf{g}$ is the gravity force with mass m and $\mathbf{g} \approx (0\ 0\ -9{,}81)^T \frac{N}{kg}$). For a swimming object in

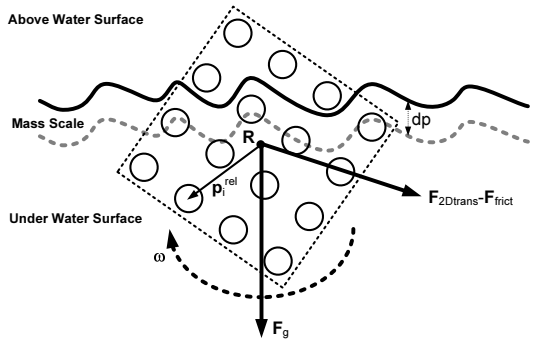


Figure 3: Rigid Body Simulation.

equilibrium, the gravity force equals the buoyant force: $\mathbf{F}_b = \mathbf{F}_g$. If the fluid's density exceeds the average density of the object, the object floats – otherwise, the object sinks. The rigid body motion for dynamic liquid surfaces can be determined by the local differences of $\mathbf{F}_b$ and $\mathbf{F}_g$. The movement of a rigid body can then be separated into a translational displacement $T(\mathbf{x})$ and a rotational movement $R(\omega)$. The rotational part R is described by the angular momentum $\mathbf{M}$ and the acting forces $\mathbf{F}$ in relation to the distance $\mathbf{r}$ to the center of mass:

$$\|\mathbf{M}\| = \|\mathbf{r} \times \mathbf{F}\| = \|m r^2 \frac{d\omega}{dt}\|. \tag{8}$$

The translational displacement $T(\mathbf{x})$ can be described by the derivations of velocities and the acting forces

$$\mathbf{F}_{\text{2D trans}} \approx m \cdot \begin{pmatrix} \frac{\partial v_x}{\partial t} \\ \frac{\partial v_y}{\partial t} \end{pmatrix}, \tag{9}$$

where v_x and v_y are the partial derivatives of the 2D velocity field $\mathbf{v}$ of the simulated fluid (see Sect. 3.1.1). However, this approximation neglects the orientation and immersion depth of the object for translation. But the influence of these physical properties during animation is hardly noticeable and would result in significant higher computational costs. Moreover, the buoyant forces during wave animation produce similar movements. Thus, the rigid body motion can be described according to the center of mass

$$\mathbf{R}_{\text{CoM}} = \frac{1}{m} \int \mathbf{r}\, dm, \tag{10}$$

with an angular acceleration and a translational acceleration. Finally, we use a standard damping or friction force $\mathbf{F}_{\text{frict}}$ depending on the velocity of the rigid body $\mathbf{v}_{\text{RB}}$ with different damping parameters b for object parts lying beneath or above water level:

$$\mathbf{F}_{\text{frict}} = -b \cdot \mathbf{v}_{\text{RB}}. \tag{11}$$

We discretize all these equations of rigid body motions with particle systems. The rigid body is described

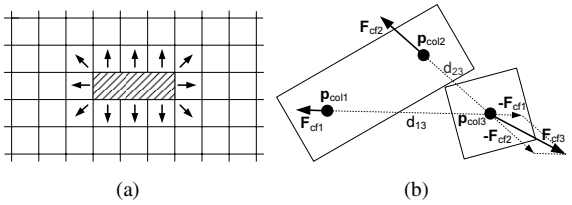


Figure 4: Rigid Body Collisions.

by its center of mass $\mathbf{R}_{\text{CoM}}$, the n relative constant particle positions $\mathbf{p}_i^{\text{rel}}$ $(1 \leq i \leq n)$ and the rotation matrix $\mathbf{M}_\mathbf{R}$ describing the orientation of the rigid body:

$$\mathbf{p}_i = \mathbf{R}_{\text{CoM}} + \mathbf{M}_\mathbf{R}\, \mathbf{p}_i^{\text{rel}}. \tag{12}$$

$\mathbf{M}_\mathbf{R}{}^t$ is updated within a time-step t according to the angular acceleration:

$$\mathbf{M}_\mathbf{R}{}^t = \mathbf{M}_\mathbf{R}{}^{t-1} + \Delta\mathbf{M}_\mathbf{R}, \tag{13}$$

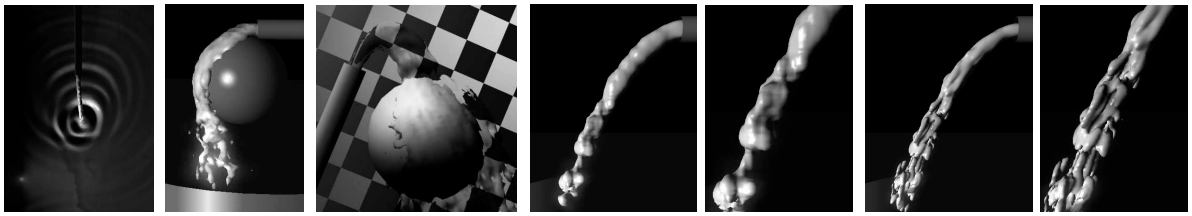
where $\Delta\mathbf{M}_\mathbf{R}$ is the rotation matrix of $\Delta t \cdot \|\mathbf{M}\|$ about the vector $\mathbf{M}/\|\mathbf{M}\|$. Physically, the angular momentum of a collection of particles is the sum of the angular momenta of each particle. Thus, $\Delta\mathbf{M}_\mathbf{R}$ can be determined by the sum of the angular accelerations of each particle. Hence, the mathematics is simplified considerably for arbitrary swimming objects and different object shapes can be simulated easily. The acting forces for a 2D quad are shown in Fig. 3. The acting particle masses within a depth dp under water surface are linearly scaled down to zero at the free surface, to avoid instantaneous buoyant force changes if a particle cuts the liquid surface. In other words, the volume of object parts lying under water is determined smoothly, even for objects represented by just few particles.

For collision handling, we use a force based approach. We distinguish between

- static object - dynamic object collisions and
- dynamic object - dynamic object collisions.

The former are handled by a static force field surrounding static objects (Fig. 4a). Thus, the acting collision force on each particle can be determined with one single memory access.

Dynamic object collisions are handled by an approximation: Depending on the distance between dynamic objects, a force acts between them in opposite direction (Fig. 4b). Therefore, we introduce collision particles $\mathbf{p}_{\text{col}}$ to dynamic objects. The particles are used to determine the distance d and the acting collision forces $\mathbf{F}_{\text{cf}}$. Cubes as rigid bodies can be simulated well with just one collision particle in the center of the cube, but more collision particles should be used for objects with complex shape. A linear kernel describes the acting collision force. The neighbors can be found efficiently within the kernel size using a regular grid, where the grid-size equals the kernel size. The described method



(a) (b) (c) (d) (e) (f) (g)

Figure 5: Fountain: Coupled with liquid simulation (seen from above, a), collisions (b,c). Surface extraction: spherical potentials (d,e), ellipsoidal potentials (f,g).

does not prevent overlapping of rigid bodies and simulates conservation of momentum just roughly. However, for moderate object velocities, the described handling reaches fast and pleasing results without any overlapping problems (Fig. 6).

3.3 Fountain

In the following, we describe the particle creation process for fountains of external inflow, e.g., water splashing out of a pipe. Our particle based fountain simulation is based on the assumption that the interaction between falling liquid particles, splashing particles or drops is negligible. The only acting force is gravity and no liquid specific forces have to be calculated: $\mathbf{F}_g = m\mathbf{g}$. Hence, the integration over time is trivial and very fast. The intensive process of neighbor searches and numerical calculations within a SPH simulation or the free surface determination within a FDM simulation leaves out. Thus, more particles can be used in splashes (compared to full physics based simulation), allowing a plausible real-time simulation.

However, when omitting interactions between particles for rapid simulation, the creation process of particles becomes significant. The natural chaotic appearance of splashes should result in a creation process with caution. Imitating nature, the particle creation in fountains or splashes depends among others on a random function. Thus, the chaotic nature of liquids in such situations can be faked.

Particles are created in a discrete cubic volume $(\triangle x, \triangle y, \triangle z)$ around $\mathbf{x}_{\text{offs.}}$ using random positions:

$$\mathbf{x} = \mathbf{x}_{\text{offs.}} + \begin{pmatrix} r_{\text{neg}} \triangle x \\ r_{\text{neg}} \triangle y \\ r_{\text{neg}} \triangle z \end{pmatrix}. \tag{14}$$

The initial velocities also depend on random functions: The particle velocities are distributed according to one main direction $\mathbf{R} * (1,0,0)$ and two apex angles:

$$\mathbf{v}_{\text{init}} = \mathbf{R} \cdot \begin{pmatrix} r \cdot v_{\text{main}} + v_{\text{main offs.}} \\ r_{\text{neg}} \cdot v_{\text{apex 1}} \\ r_{\text{neg}} \cdot v_{\text{apex 2}} \end{pmatrix}. \tag{15}$$

$\mathbf{R}$ is a rotation matrix changing main direction, r and r_{neg} are random functions ($0 \leq r \leq 1$; $-1 \leq r_{\text{neg}} \leq 1$) and $v_{\text{main offs.}}$ is the minimum speed of liquid outflow. Using these parameters, a flow can be modelled anywhere with any direction and any main velocity. However, other particle creation functions are possible and can be used to model special situations.

Another parameter for controlling the flow is the number of created particles n_c per time step. Modeling flow intensity variations, we use the following formula:

$$n_c = n_{\text{min}} + r \cdot \triangle n. \tag{16}$$

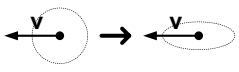
n_{min} is the minimum number of created particles per time step and $\triangle n$ is the intensity of variation. Instantaneous variations of n_c models the change of flow intensity. If n_c equals to zero, no liquid flow occurs.

If a fountain particle hits the liquid surface, it is destroyed and creates wave particles. Hence, a radial wave propagates, starting from the impact position (Fig. 5a). Using collision handling for fountain particles, realistic splashes even at moving objects can be modeled in real-time (Fig. 5b,c). Thereby, we use the same collision handling as described in Sect. 3.2.

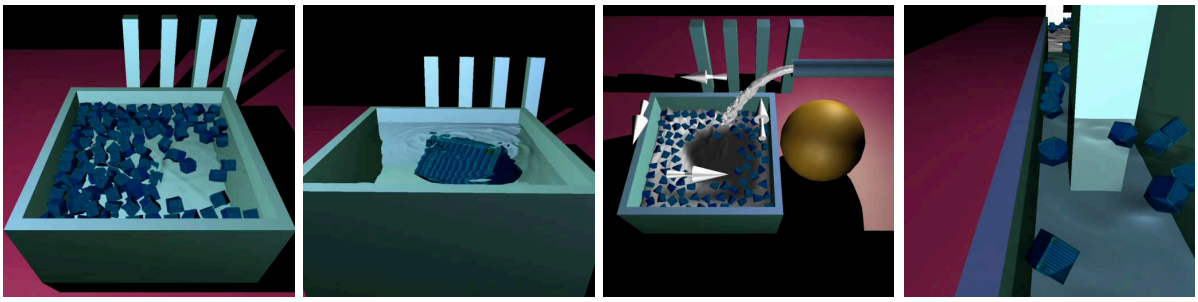
For 3D surface reconstruction of the fountain we use a traditional Marching Cubes (MC) algorithm [LC87]. The isosurface for n particles with positions $\mathbf{x}_i$ ($i = 1 \ldots n$) is determined by the following potential (h: isoradius):

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \sqrt{1 - \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2}} \tag{17}$$

The square root (Eq. 17) can be approximated to increase performance. The performance is good, due to a small, adaptive splashing volume being used for implicit function and iso-surface generation. Additionally, performance increases at low sampling and small particle potentials used in implicit function generation. To increase details and decrease the blobby-ness of the surface, we don't use strict spherical potentials but ellipsoidal potentials along the particle velocities.



Hence, the fountain becomes more detailed and filigree along the velocity direction (Fig. 5d-g).



(a) (b) (c) (d)

Figure 6: Examples of the proposed method – please see accompanying video for animations.

4 IMPLEMENTATION AND RESULTS

The proposed algorithms were implemented using OpenGL 2.0 and the related shading language GLSL in C++. The presented examples were performed on a dual-core desktop PC with a 2,6 GHz AMD Athlon 64 CPU, 2GBs of RAM and a graphics card based on an ATI Radeon x1900 GPU. We use a parallel implementation and split the simulation between two cores as shown in the following Table:

Core 1:	Surface Extraction		Interaction	Rendering
Core 2:	Wave Eq.	N.-S. Eq.	Rigid-B.	Fountain

The GPU is used for fast height-field rendering and refraction simulation. The physically based simulation tasks are performed CPU based. Hence, the performance could be increased immensely, using GPU-based implementations for parts of the physical simulation. The FDM flow simulation is especially suited to be performed on the GPU. However, the measured frame rates achieved with the actual implementation are quite satisfactory, as shown in Table 1 (Resolution: 1024×1024 pixels). Take note that already small grid-sizes NS-GS lead to convincing flow results – due to the explicit handling of surface waves. The performance and complexity of the proposed technique mainly depends on the following parameters:

- Eulerian grid-size ($O(n^2)$),
- number of wave particles ($O(n)$),
- surface resolution ($O(n^2)$),
- number of rigid body particles ($O(n)$),
- number of rigid b. collision particles ($O(n \log n)$),
- number of fountain particles ($O(n)$) and resolution of associated reconstruction volume ($O(n^3)$).

However, a problem of the technique shared with many other height-field based liquid simulation techniques is the impossibility to visualize 3D liquid effects (e.g., splashes or breaking waves). At least, the mentioned extended height-field allows the rendering of surfaces, slightly exceeding the 2,5D surface and the presented fountain model solves this problem in special scenarious.

Figure	NS-GS	2. WP (max)	S-GS	FPS
1	32×64	40000	128×256	33
6a,b,c	32×32	20000	128×128	74,76,23
6a	64×64	20000	128×128	70
6a	128×128	20000	128×128	27
6a	256×256	40000	256×256	4
6d	16×128	40000	64×512	26

Table 1: Performance measurements of the presented algorithms in different scenarios (Gridsize of Navier-Stokes Eq. solver (NS-GS), maximum number of used wave-particles (WP), surface grid-size (S-GS), examples 6a,c,d use 100 rigid body objects).

The presented method allows the simulation of a 2D flow (e.g., a river) at high performance, whereas the surface remains detailed and the surface waves and numerous rigid bodies are propagating with the flow – indicating e.g. the flow direction. Recapitulating, the benefits of the presented combination of flow and surface simulations are:

- Physically based flow simulation,
- physically based wave simulation,
- flow interactions (e.g., moving boat, turbine),
- objects moving with the flow (e.g., leafs, objects),
- automatic, natural and global flow (e.g., creek),
- interactive wave creation (e.g., rain, moving obj.).

The low viscosity of real water cannot be reached in real-time with full 3D Navier-Stokes simulation methods (due to large time-steps and the necessary damping), resulting in a liquid behavior more like oil than real water. The presented method reduces these viscous effects, due to the fast calculation of the 2D flow and the 2D Wave Equation. Hence, the viscosity of the simulated fluid appears to be low. In addition, while decreasing the time-steps or reducing speed the convincing simulation of liquids with a viscosity greater than the viscosity of physical water, such as oil, honey or molten wax, can be achieved easily.

5 CONCLUSION/FUTURE WORK

We have presented a concept for real-time liquid simulation with focus on wave propagation in flowing liquids. The basic idea is the coupling of a 2D Navier-Stokes simulation for flow description and a 2D Wave Equation simulation for surface waves propagation. The solver is based on the Stable Fluids algorithm [Sta99] and the wave particles algorithm [YHK07]. Furthermore, we present methods for real-time rigid body simulation of several hundred objects swimming in the liquid and a real-time fountain simulation interacting with the liquid. The resulting liquid is rendered height-field based at high frame rates.

We demonstrated the potential of combining existing methods and a new coupling scheme according to the simulation of plausible, complex liquid effects in real-time. Of course, those effects have been used in computer animated films with perfect quality (based on slow off-line techniques), but not in real-time environments at high frame rates. Hence, our approach is practical for interactive environments, e.g., VR's or video games. In view of multi-core architectures, our approach can enhance realism of such environments immensely. Although aiming for real-time environments, our fast method is interesting for high quality off-line rendering applications as well.

Our future investigation includes a better rendering approach (e.g., GPU-based approximated raytracing), the use of a column based approach or a SPH approach for flow simulation. As well, the collision handling of many colliding objects at the same time should be enhanced. In those situations, our method tends to unwanted intersection effects. We would also like to use the GPU for numerical calculations to increase performance immensely. Finally, the use of adaptive surface rendering and simulation techniques would decrease complexity and much larger liquid volumes could be simulated.

REFERENCES

[BD06] L. Baboud and X. Decoret. Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena*. Eurographics, 2006.

[Bel03] V. Belyaev. Real-time simulation of water surface. In *GraphiCon-2003*, pages 131–138, 2003.

[CdVL95] J. X. Chen and N. d. V. Lobo. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graph. Models Image Process.*, 57(2):107–116, 1995.

[CMT04] M. Carlson, P. Mucha, and G. Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics*, volume 23, pages 377–384, 2004.

[Cor07a] H. Cords. Mode-splitting for highly detailed, interactive liquid simulation. *Proceedings of GRAPHITE'07*, pages 265–272, 2007.

[Cor07b] H. Cords. Refraction of water surface intersecting objects in interactive environments. *Proceedings of the 4th Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS'07)*, pages 59–68, 2007.

[FF01] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of SIGGRAPH'01*, pages 23–30, 2001.

[Gom00] Miguel Gomez. Interactive simulation of water surfaces. In *Game Programming Gems*, pages 187–195. Charles River Media, 2000.

[Har05] M. Harris. Fast fluid dynamics simulation on the gpu. In *ACM SIGGRAPH 2005 Courses*, 2005.

[HKK07] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Computer Graphics International*, 2007.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182–2189, 1965.

[Joh04] C. Johanson. Real-time water rendering - introducing the projected grid concept. *Master of Science Thesis (Lund University)*, 2004.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH '87*, pages 163–169, 1987.

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of Symposium on Computer animation (SCA '03)*, pages 154–159. Eurographics Association, 2003.

[MFC06] M. M. Maes, T. Fujimoto, and N. Chiba. Efficient animation of water flow on irregular terrains. In *GRAPHITE '06*, pages 107–115, 2006.

[MSD07] M. Müller, S. Schirm, and S. Duthaler. Screen space meshes. In *Symposium on Computer Animation*, 2007.

[OH95] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Symposium on Computer Animation*, 1995.

[PMDS06] V. Popescu, C. Mei, J. Dauble, and E. Sacks. Reflected-scene impostors for realistic reflections at interactive rates. *Computer Graphics Forum*, 25(3):313–322, September 2006.

[SF95] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. *Computer Graphics*, 29:129–136, 1995.

[SKALP05] L. Szirmay-Kalos, B. Aszódi, I. Lazányi, and M. Premecz. Approximate ray-tracing on the gpu with distance impostors. In *Computer Graphics Forum (Proc. of Eurographics 2005)*, volume 24, 2005.

[Sou05] T. Sousa. Generic refraction simulation. In *GPU Gems 2*, pages 295–305. Addison-Wesley, 2005.

[Sta99] J. Stam. Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings*, pages 121–128. Addison Wesley Longman, 1999.

[SW01] J. Schneider and R. Westermann. Towards real-time visual simulation of water surfaces. *Proceedings of the Vision Modeling and Visualization Conference*, pages 211–218, 2001.

[TSS+07] N. Thuersey, F. Sadlo, S. Schirm, M. Mueller, and M. Gross. Real-time simulations of bubbles and foam within a shallow-water framework. 2007.

[Wym05] C. Wyman. An approximate image-space approach for interactive refraction. *ACM Trans. Graph.*, 24(3):1050–1053, 2005.

[YHK07] C. Yuksel, D. H. House, and J. Keyser. Wave particles. *ACM Siggraph 2007 Conference Proceedings*, 2007.

[ZPvBG01] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH'01*, pages 371–378. ACM Press, 2001.