

# Accelerated Animation of Liquid Splash

Roman Ďurikovič<sup>\*†</sup>      Shinya Abe  
Software Department, The University of Aizu

## Abstract

We present a method for simulation and visualization of a liquid splash motion. The proposed system is composed of a computational fluid dynamics and a polygon based renderer using multi texturing graphical acceleration. The system well supports the *Navier-Stokes* solver with adaptive time step and viscosity. The final liquid surface is obtained from the velocity field by rough estimation of the surface and then subdividing the surface. Rendering of the surface is performed by multi texturing method using the caustics, reflection, liquid and object textures.

**Keywords:** animation, computational fluid dynamics, Navier-Stokes equations, natural phenomena, multi-texturing

## 1 Introduction

Physical based simulation, animation and visualization are playing an important role in the field of computer graphics (CG) in recent years. The improvements of physical based methods support CG-designers in producing realistic CG animation of natural phenomena; for example, rigid body collision [8], destruction of walls [1], explosions [10], fluid dynamics, steam motion and flames.

The time needed for simulating and rendering the physical based effects consumed several hours per frame using ordinary ray-tracing method. Many researchers apply computational fluid dynamics (CFD) for physical based animation to generate realistic fluid motions. *Foster* [2, 3, 4] developed a modeling and controlling method based on CFD, which reduces simulation time costs and implements a high controllability of fluid.

*Kunimatsu* [7] reduced the rendering time of fluid media to 60 seconds per frame. Their team utilized modern high performance Personal Computer (PC) and graphics acceleration hard-ware. However this

<sup>\*</sup>Contact person: Roman Ďurikovič, Tsuruga, Ikki-machi, Aizu-Wakamatsu City, 965-8580 Japan. Email: roman@u-aizu.ac.jp, Tel: +81 (242) 37-2641, Fax: +81 (242) 37-2706.

<sup>†</sup>On leave from Department of Computer Graphics and Image Processing, Faculty of Mathematics, Physics and Computer Science, Comenius University, Bratislava, Slovakia.

method can not treat small fluid objects such as splash and spray.

Our final goal is to develop a real-time system for realistic fluid animation. The system will employ the following specifications:

1. Dynamic fluid motion including mixture, splashing, self-collision, etc.
2. Realistic appearance including th refraction, reflection and caustics.
3. Real time simulation and animation.

For satisfying these terms, we planned to combine and improve the methods described in Foster [4] and Kunimatsu [7]. We add two new ideas to this hybrid method. The first, is the adaptive time step iteration and automatic control of system stability, which reduces simulation time costs. The second idea, is the introduction of reflection texture mapping to realistic water appearance.

The main application of the proposed method are in the motion picture industry. This system gives high quality previews for CG-designers therefore, results are easily evaluated and instantly modified. To render final high quality animation sequences, we focused on the appearance and dynamics of water using a non-real-time rendering algorithms.

The papers has the following structure. Section 2 describes the simulation method. Section 3 describes the polygonization of water surface capable of following the splash changes and the hard-ware accelerated rendering method. Section 4 shows the rendering results and the time costs of simulation and rendering.

## 2 Simulation Method

The key point of this simulation method is to combine the application of Navier-Stokes equations on low resolution finite difference method and use marker particles for tracking water surface. This method requires only low calculation cost and it provides high controllability for water motions.

### 2.1 Physical Model

At human scale, the behavior of water is influenced by the following forces:

**Momentum :** Water moves under the laws of energy and momentum conservation.

**Gravity :** Gravity and pressure cause most water waves.

**Viscous drag :** This is the key producing the real behavior of water. This part leads directly to self-propagating rotation.

**Pressure :** Pressure is the important fact in the behavior of the surface of water. The motion of the water surface is influenced by changes in pressure within the volume.

Other motion factors, turbulence and surface tension, cause few effects relative to the above four forces at human scale.

The *Navier-Stokes* equation includes these four factors and completely describes fluid motion. The equation consists of two parts. The first part is a non-linear equation, which shows relations among velocity  $\mathbf{u}$ , pressure  $p$  and forces:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla \cdot (\nabla \mathbf{u}) + \mathbf{f} - \frac{1}{\rho} \nabla p, \quad (1)$$

where velocity field  $\mathbf{u} = (u, v, w)$ ,  $\nu$  is kinematic viscosity and  $\mathbf{f}$  is the sum of external force vectors. Constant  $\rho$  is density of fluid, which is equal to 1 for water. Time is denoted as  $t$  and operator  $\nabla$  is

$$\nabla = \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right).$$

The second part, of *Navier-Stokes* equation, models mass conservation of liquids,

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

This equation states that the net inflow and outflow is zero. Together with suitable boundary conditions, the *Navier-Stokes* equation can be used for simulation.

Our simulation system runs on 3-dimensions and treats only the gravity force  $g = 9.8 \text{ m/s}^2$  as an external force. Thus, Eqs. 1 and 2 can be written as [2]:

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} = & \\ & - \frac{\partial p}{\partial x} + \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \\ \frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} = & \\ & - \frac{\partial p}{\partial y} + \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) \\ \frac{\partial w}{\partial t} + \frac{\partial wu}{\partial x} + \frac{\partial wv}{\partial y} + \frac{\partial w^2}{\partial z} = & \\ & - \frac{\partial p}{\partial z} + g + \nu \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) \end{aligned} \quad (3)$$

and

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \quad (4)$$

## 2.2 Simulation Grid

In order to solve the *Navier-Stokes* equations using central finite-difference method, all solid obstacles and the atmosphere in the scene are approximated using a series of fixed cubic grids. Velocity and pressure are defined on the grids.

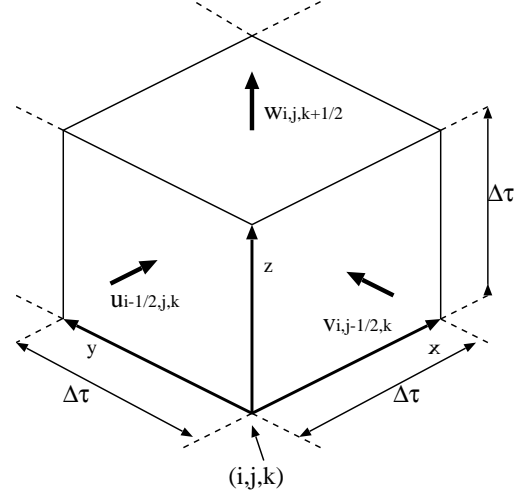


Figure 1: Location of staggered velocity components and pressure on a cell.  $\Delta\tau$  is grid size.

Pressure is defined at the center of each cell. Each velocity component  $u$ ,  $v$  and  $w$  in  $x$ ,  $y$  and  $z$  directions, is defined at the centers of each face of a cell, shown in Figure 1. While solving equations in discrete form, velocities which do not lie on faces are required. In this case, unknown velocities are averaged among the nearest values, e.g,  $v_{i, j, k} = \frac{1}{2}(v_{i, j-\frac{1}{2}, k} + v_{i, j+\frac{1}{2}, k})$ .

Each cell has four content attributes which are *Full*, *Surface*, *Obstacle* and *Empty*. An *Empty* cell represents atmosphere, an *Obstacle* cell contains a solid obstacle, a *Surface* cell contains fluid, but faces at least one *Empty* cell, finally a *Full* cell is filled with fluid and does not face any *Empty* cells.

## 2.3 Boundary Conditions

The *Navier-Stokes* equation can be applied automatically without testing surface or obstacle positions when the boundary conditions are set. Boundary conditions are determined from the attributes of a cell. A boundary is the interface between water and the atmosphere, or water and an obstacle. To simplify the explanation, we use a 2-dimensional model in  $(x, z)$  coordinate system.

### 2.3.1 Obstacle and water

Left image of Figure 2 shows an obstacle and a water surface. To avoid water passing into the obstacles, normal velocity on the obstacle face must be zero namely,

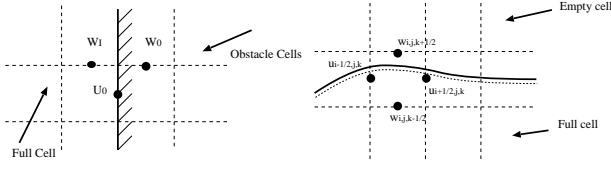


Figure 2: Boundary Condition. Right: The obstacle and the full cell. Left: The free surface cell.

$u_0 = 0$ . If the surface of an obstacle is non-slip, tangential velocities are also zero. This condition is applied indirectly by setting tangential cell face velocity inside the obstacle equal to the opposite value of tangential velocity of the neighbor cell,  $w_0 = -w_1$ . If the surface of an obstacle is smooth and water can slip, inner tangential velocities are set equal to outer tangential velocity,  $w_0 = w_1$ .

For eliminating any acceleration across the obstacle boundary, the pressure of the obstacle cell is set the same as the pressure of the adjacent fluid cell.

### 2.3.2 Water and atmosphere

Equation 4 is used to set boundary conditions at *Surface* cell. If the cell is surrounded by three *Surface* or *Full* cells, the velocity of the open side is calculated from the divergence of the target *Surface* cell. For example, consider the situation on right of Figure 2, the velocity  $w_{i,j,k+1/2}$  can be derived from explicit finite difference method of Eq. 4 as

$$\frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x} + \frac{w_{i,j,k+1/2} - w_{i,j,k-1/2}}{\Delta z} = 0,$$

where  $\Delta x$  and  $\Delta z$  are grid sizes. Considering  $\Delta x = \Delta z$ , the above equation can be written as

$$w_{i,j,k+1/2} = w_{i,j,k-1/2} - (u_{i+1/2,j,k} - u_{i-1/2,j,k}).$$

In the case of cell surrounded by two fluid cells with attribute either *Surface* or *Full*, each open side velocity is set equal to the opposite side velocity. This modification satisfies Eq. 2. If the case when three sides of the cell are open, velocity of the side facing the fluid is carried to the opposite side and, the other two sides are not changed.

In the 3-dimensional system, there are 64 different combinations of *Empty* and *Full* cells. Please, note that the pressure of the *Surface* cell is set equal to the atmospheric pressure.

## 2.4 Numerical Method

The calculation step is divided into two parts. The first part calculates new velocities using Eq. 1 with the velocities and the pressures of the previous time step. The resulting velocities will not satisfy mass conservation, Eq. 2. Therefore, the second part, modifies the resulting velocities and pressures of former calculation using Eq. 2.

### 2.4.1 Calculation of velocity field

To calculate the velocity field, explicit finite difference approximations are applied to Eq. 3. The velocity  $\tilde{u}_{i+1/2,j,k}$  is a new velocity in next time step,  $\Delta t$  is time step length and  $\Delta x$ ,  $\Delta y$  and  $\Delta z$  is grid size,  $\Delta x = \Delta y = \Delta z = \Delta \tau$ . Velocity component of  $u_{i+1/2,j,k}$  is then updated by

$$\begin{aligned} \tilde{u}_{i+1/2,j,k} = & u_{i+1/2,j,k} + \Delta t \left\{ \frac{1}{\Delta \tau} [(u_{i,j,k})^2 - (u_{i+1,j,k})^2] \right. \\ & + (uv)_{i+1/2,j-1/2,k} - (uv)_{i+1/2,j+1/2,k} + (uw)_{i+1/2,j,k-1/2} \\ & - (uw)_{i+1/2,j,k+1/2} \left. \right\} + \frac{1}{\Delta \tau} (p_{i,j,k} - p_{i+1,j,k}) \\ & + \frac{\nu}{\Delta \tau^2} [u_{i+3/2,j,k} + u_{i-1/2,j,k} + u_{i+1/2,j+1,k} \\ & + u_{i+1/2,j-1,k} + u_{i+1/2,j,k+1} + u_{i+1/2,j,k-1} - 6u_{i+1/2,j,k}]. \end{aligned}$$

Similarly, we can write  $v_{i,j+1/2,k}$  and  $w_{i,j,k+1/2}$ , except the gravity component is added to  $w_{i,j,k+1/2}$ . In each iteration, new velocities are updated by calculating the above equations with previous velocities and pressures.

### 2.4.2 Mass conservation

The result of the calculation described in previous section does not satisfy Eq. 2. The efficient way of enforcing incompressibility is to modify pressures using the Laplacian operator [5],

$$\nabla^2 p = \frac{\rho \nabla \cdot \mathbf{u}}{\Delta t}.$$

Applying explicit finite difference approximations at the center of the simulation cell, the above equation can be discretized to

$$\begin{aligned} p_{i-1,j,k} + p_{i+1,j,k} + p_{i,j-1,k} + p_{i,j+1,k} + p_{i,j,k-1} \\ + p_{i,j,k+1} - 6p_{i,j,k} = \rho \frac{\Delta \tau}{\Delta t} (u_{i+1/2,j,k} - u_{i-1/2,j,k} \\ + v_{i,j+1/2,k} - v_{i,j-1/2,k} + w_{i,j,k+1/2} - w_{i,j,k-1/2}). \end{aligned}$$

The above equation forms a linear system  $\mathbf{A}\mathbf{P} = \mathbf{b}$ .  $\mathbf{A}$  is a regular and symmetric matrix. The diagonal coefficient  $a_{ii}$  is equal to negative number of cells adjusted to cell  $i$ , and non-diagonal elements are set  $a_{ij} = a_{ji} = 1$  if the cell at  $j$  is the neighbor of the cell at  $i$ ,  $\mathbf{P} = (p_{0,0,0}, p_{0,0,1}, \dots, p_{i,j,k}, \dots)$  is a vector of unknown pressures. The SOR [6] iterative method can solve this sparse linear system.

After calculating new pressures, the velocities of each cell are updated by

$$\tilde{\mathbf{u}} = \mathbf{u} - \Delta t \frac{1}{\rho} \nabla \cdot p.$$

## 2.5 Tracking Water Surface

To utilize massless marker particles is a simple and effective way for tracing free water surface. Marker particles are introduced into the simulation system at inflow points and follow the velocities of the cell which the particle is in. The particle does not have mass and does not influence calculations. Marker particles are just used for free water surface detection. The position and velocity of particles are calculated from velocities of adjacent cells using an area weighting interpolation scheme demonstrated in Figure 3. After new particle positions are found, attributes of each cell are changed as follows:

- A cell which does not contain any particles is an *Empty* cell.
- A cell which has at least one particle and faces at least one *Empty* cell is a *Surface* cell.
- A cell which has at least one particle and does not face an *Empty* cell is a *Full* cell.

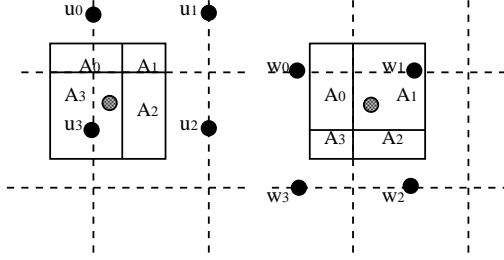


Figure 3: Gray point is position of particle  $p$ . New velocities of  $p$  are delivered from  $u_p = A_0u_0 + A_1u_1 + A_2u_2 + A_3u_3$  and  $w_p = A_0w_0 + A_1w_1 + A_2w_2 + A_3w_3$ .

## 2.6 Stability

Long time step makes total simulation time shorter but can introduce the instability. We utilize an adaptive time step which guarantee the largest time step with stability. To keep this simulation system stable, viscosity  $\nu$  and time step  $\Delta t$  must satisfy the following conditions [3],

$$\Delta t |\mathbf{u}| < \Delta \tau, \quad (5)$$

$$\nu \Delta t < \frac{\Delta \tau^2}{6} \quad (6)$$

and

$$\nu > \max \left[ \left( \frac{\Delta t}{2} u^2 + \frac{\Delta \tau^2}{2} \frac{\partial u}{\partial x} \right), \left( \frac{\Delta t}{2} v^2 + \frac{\Delta \tau^2}{2} \frac{\partial v}{\partial y} \right), \left( \frac{\Delta t}{2} w^2 + \frac{\Delta \tau^2}{2} \frac{\partial w}{\partial z} \right) \right]. \quad (7)$$

If the situation that viscosity does not satisfy Eq. 7 occurs, viscosity will be increased for satisfying this equation at an unstable cell. Each cell has This modification causes few visual effects to the simulation result. After modification of viscosity, time step  $\Delta t$  is checked by the Eqs. 5 and 6. If a time step does not satisfy them, it will be decreased to satisfy both conditions.

## 2.7 Implementation of Algorithm

The simulation system is composed of two parts. The first part initializes simulation grids, the initial water position and its velocities, obstacles, inflow and outflow positions. The second part iterates the calculation of the *Navier-Stokes* equation. The outline of the second part is as follows:

1. Determine the contents of each cell using the surface tracking method.
2. Set boundary conditions for the free surface and obstacle cells.
3. Check stability conditions and modify viscosity and time step.
4. Compute velocity for all cells filled with water using the *Navier-Stokes* Equations.
5. Compute new pressures and velocities which satisfy mass conservation.
6. Compute marker particle positions
7. Update the position of the surface.
8. Repeat

## 3 Rendering Method

The combination of texture mapping techniques and surface subdivision methods provides a good visualization of liquid surfaces. The rendering consists of following steps:

1. Polygonization of water free surface.
2. Generation of caustics textures.
3. Mapping refraction textures
4. Blending caustics, reflection and refraction textures.
5. Drawing water surface and scene.

Using this method, the refracted image is directly mapped to the water surface.

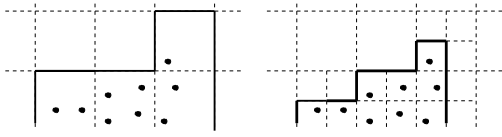


Figure 4: Subdivided grid and surface detection.

### 3.1 Polygonization of Water Surface

In this simulation system, the environment is divided into a low resolution grid. We divide the grid into smaller grids and determine more accurate surface positions, shown in Figure 4.

### 3.2 Subdivision Surface

The resulting polygon mesh is very rough for rendering realistic images. The mesh surface needs to be modified to be smoother by applying a surface subdivision algorithm. The *Catmull-Clark* or *Loop* subdivision surface [9] were used. Both subdivision algorithms are very simple to implement.

We use the Cutmull-Clark method for subdividing surface. The method can not handle non-manifold edges. To overcome the problem, non-manifold edges detected on the coarse-mesh are split into separate cells.

### 3.3 Caustics Texture

The effects of caustics make a more realistic appearance. At first, for each object two textures are created, one is the original texture which contains the objects surface image, and the other is the caustics texture initialized as zero. For each polygon of water surface mesh, an illumination volume is defined by sweeping in the light direction as shown in Figure 5. If an intersection occurs between the illumination volume and the object surface, intensity will be added to texels of caustics texture on the projected area of the illumination volume. The intensity is obtained from Equation [7]

$$I_{caustics} = (\mathbf{N} \cdot \mathbf{L}) \frac{S_1}{S_2} I_{light}, \quad (8)$$

where  $I_{caustics}$  is an intensity of caustics and  $I_{light}$  is an intensity of light.

### 3.4 Reflection Texture

The reflection effect can be calculated and stored in to a independent texture. Consider Figure 6, the intensity of each texel is calculated as follows:

$$T_{reflection}(\theta) = \cos^n \theta, \quad (9)$$

where  $n$  is the user defined value. It determines the sharpness of reflection.

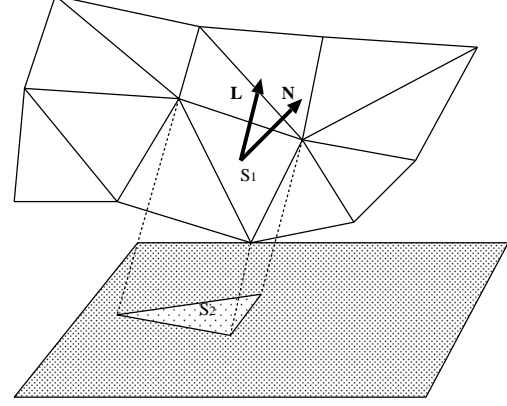


Figure 5: Generation of caustics texture.  $\mathbf{N}$  is a normal vector of surface polygon,  $\mathbf{L}$  is the light direction,  $S_1$  is an area of surface polygon and  $S_2$  is a projected area of illumination volume and the gray plane is the texture of an object.

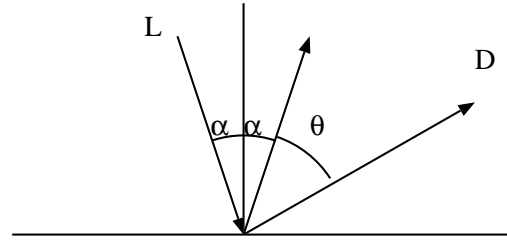


Figure 6:  $\mathbf{L}$  is a light direction vector and  $\mathbf{D}$  is a camera direction vector.

### 3.5 Refraction Mapping

To simulate refraction effects, the texture of an object is directly mapped to the water surface polygons. The texture coordinates,  $u$  and  $v$ , are calculated by tracing the ray's route from camera to target object by

$$n_{air} \sin \theta_1 = n_{water} \sin \theta_2, \quad (10)$$

where  $n_{air}$  is a refractive index of air and  $n_{water}$  is a refractive index of water. In Figure 7, the texture of an object is directly mapped to the water surface and the texture coordinates,  $u'$  and  $v'$ , are mapped to the vertex  $P$  of a triangle patch.

### 3.6 Blending of Textures

Reflection, caustics and object textures are blended by

$$\begin{aligned} T_{result}(u, v) &= T_{reflection}(u, v)T_{water}(u, v) \\ &+ (1 - T_{reflection}(u, v)) \\ &\times \{T_{object}(u', v')(T_{caustics}(u', v') + I_{ambient})\}, \end{aligned}$$

where  $T_{result}(u, v)$  is a final texture color at  $(u, v)$ ,  $T_{reflection}$  is a reflection texture,  $T_{water}(u, v)$  is a water original texture color  $T_{object}(u', v')$  is an original

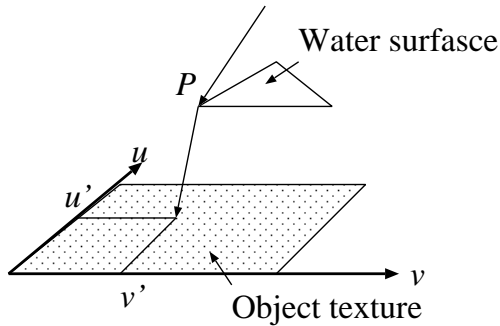


Figure 7: Projected mapping.

texture color of the object at projected  $uv$ -coordinate of  $(u, v)$  and  $T_{caustics}(u', v')$  is the intensity of the caustics texture.  $I_{ambient}$  is an ambient intensity for all surfaces in a scene. The resulting texture is directly mapped onto water surface polygons and rendered directly, using graphic accelerated hard-ware.

## 4 Results

The system are executed on a PC which has Athlon 1.2G-Hz CPU, 512MB main-memory and GeForce2 GTS with 32MB video-memory. Figure 8 shows an example scene which water is falling into a small pool. The simulation grid resolution is  $20 \times 20 \times 20$  and the grid size  $\Delta\tau$  is  $10\text{cm}$ . Initial water velocity is  $1.4\text{ m/s}$  and the inflow shape is a circle with a radius of  $40\text{cm}$ . The time step  $\Delta t$  range is from  $0.0003\text{s}$  to  $0.01\text{s}$ . The average rendering time is  $27\text{s}/f$  and the simulation time is  $11\text{s}/f$ .

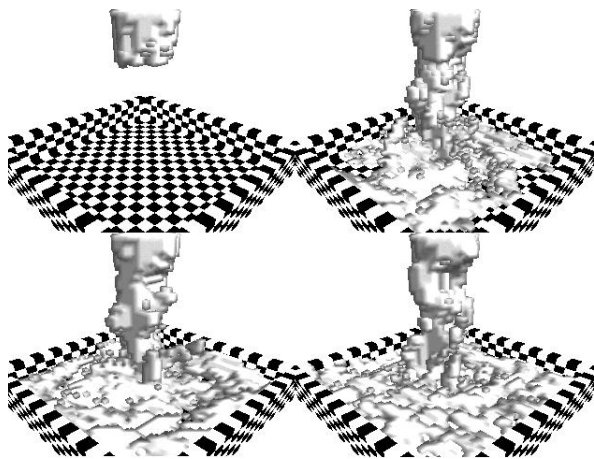


Figure 8: An example of resulting image

## 5 Conclusion and Future Work

We presented full solver of the Navier-Stokes equation with an adaptive time step method and fast rendering technique using graphical-hardware. The simulation and rendering method is examined and we can get

even faster frame rates after tuning the techniques. The works under progress include the improving of caustics and refraction mapping.

## References

- [1] J.F. O'Brien, J.K. Hodgins, *Graphical modeling and animation of brittle fracture*. *Computer Graphics*, Proceedings of SIGGRAPH 1999 (Los Angeles, California, August 8–13, 1999) In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pp. 137–146, 1999.
- [2] N. Foster, D. Metaxas, *Realistic animation of liquids*, *GRAPHICAL MODELS AND IMAGE processing*, 58(5), (1996), pp. 471-483.
- [3] N. Foster, *Modeling and animating fluid phenomena for computer graphics and special effects*, Ph.D. dissertation, University of Pennsylvania, Philadelphia, 1997.
- [4] N. Foster, D. Metaxas, *Controlling fluid animation*, *Proceedings CGI 1997* (1997), pp. 178-188.
- [5] N. Foster, R. Fedkiw, *Practical animation of liquids*. *Computer Graphics*, Proceedings of SIGGRAPH 2001 (Los Angeles, California, August 12–17, 2001) In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pp. 23–30, 2001.
- [6] David R. Kincaid, Jhon R. Respass and David M. Young *ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods*, Netlib Repository, <http://www.netlib.org/>.
- [7] A. Kunitatsu, Y. Watanabe, H. Fujii, T. Saito, K. Hiwada, T. Takahashi and H. Ueki *Fast simulation and rendering techniques for fluid objects*, *EUROGRAPHICS 2001*, page 57-66 (2001).
- [8] B. Mirtich, *Timewarp rigid body simulation*. *Computer Graphics*, Proceedings of SIGGRAPH 2000 (New Orleans, Louisiana, July 23–28, 2000) In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pp. 193–200, 2000.
- [9] P. Shroder, D. Zorin *Subdivison for Modeling and Animation*, SIGGRAPH 98 Course Notes, ACM SIGGRAPH, 1998.
- [10] G.D. Yngve, J.F. O'Brien, J.K. Hodgins, *Animating explosions*. *Computer Graphics*, Proceedings of SIGGRAPH 2000 (New Orleans, Louisiana, July 23–28, 2000) In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pp. 29–36, 2000.