Real-Time Water Wave Simulation with Surface Advection based on Mass Conservancy

Dongyoung Kim
Korean Minjok Leadership Academy
KMLA, 1334 Sosa, Anheung, Hoengseong, Gangwon, South Korea

Kwan-Hee Yoo
Department of Computer Education, and Department of Information Industrial Engineering
Chungbuk National University, Cheongju, 361-763, Chungbuk, Korea

ABSTRACT

In this paper, we present a real-time physical simulation model of water surfaces with a novel method to represent the water mass flow in full three dimensions. In a physical simulation model, the state of the water surfaces is represented by a set of physical values, including height, velocity, and the gradient. The evolution of the velocity field in previous works is handled by a velocity solver based on the Navier-Stokes equations, which occurs as a result of the unevenness of the velocity propagation. In this paper, we integrate the principle of the mass conservation in a fluid of equilateral density to upgrade the height field from the unevenness, which in mathematical terms can be represented by the divergence operator. Thus the model generates waves induced by horizontal velocity, offering a simulation that puts forces added in all direction into account when calculating the values for height and velocity for the next frame. Other effects such as reflection off the boundaries, and interactions with floating objects are involved in our method. The implementation of our method demonstrates to run with fast speed scalable to real-time rates even for large simulation domains. Therefore, our model is appropriate for a real-time and large scale water surface simulation into which the animator wishes to visualize the global fluid flow as a main emphasis.

Keywords: *Water Surfaces, Navier-Stokes, Real-Time Simulation*

1. INTRODUCTION

In general, water is around us everywhere. From drinking cups to swimming pools, water is very plain to us yet it has strikingly unique physical properties, creating visually impressive effects in the real life. The research is driven by a motivation to catch such visual scenes caused by water, and to render the effects on the computer screen by mimicking how natural water would act [6].

Computer graphics algorithms are adapted for their uses. In computational cost blind areas where they ask for the most accurate detail as possible, they use a complex and accurate algorithm to spend hours of offline computation to create the most accurate simulation of graphics effect [3]. On the other hand, areas such as video or online games require an instant simulation at an interactive level with the user, known as the real-time simulation [2], [4], [7], [9], [10], [11]. Our work is focused on the latter, where we seek to provide a convincing physical simulation of water with a high performance, there to be applied for practical uses with large domains. We do not cover every aspect of water phenomena. Instead, advection of floating objects, generation of water waves, and interaction with surface boundaries are part of water effects we aim to simulate on our virtual water surface model.

In general, algorithms for creating a large scale simulation that are handled by real-time often entail abbreviation of physical principles [1-4], [7-11]. Tendencies in previous research show that water surface simulation techniques for large domains neglect significant fluid flow. This tendency can be explained that the fluid flow might be considered unimportant on a water surface of extensive size. Also, physically depicting the velocity field often brings heavy computational cost. However, neglecting the velocity property of the water surface puts limitation in animating some wider range of surface behavior, such as objects traversing along a flowing water from one point to another, or wave being generated from horizontal wind blow. Therefore, we depart from the common trend that concentrates on the depiction of waves, which is physically defined as the transport of energy through space not associated with motion of the medium occupying this space as a whole. Instead, we design a water surface when the fluid flow is of the main importance, and still embodying the wave effects and other surface phenomena as

* *Corresponding author. E-mail : khyoo@chungbuk.ac.kr*
Manuscript received May. 06, 2008 ; accepted Jun. 13, 2008

well. Of course, all of this is kept in real-time performance.

2. RELATED WORK

Computational fluid dynamics has a long history after that Navier and Stokes formulated the famous Navier-Stokes equation that describe the dynamics of fluids [2-4], [9], [10]. A basic mathematical model used in real-time simulation of surface wave is the wave equation [7], [11]. Specifically, the wave equation is a description of how vertical tension force acts on a medium to create wave effects. As wide as it is used, its implementation on a computer gives a realistic and fast simulation of any wave propagation effect on the surface of a fluid. However, it has a shortcoming that the represented water is static. In a water surface modeled by the wave equation [7], [11], the animation cannot render tides caused from wind blown on the surface nor can it make floating objects drift along the surface.

The recently developed smoothed particle hydrodynamics method gives a real time simulation for the water body based on particles [8]. However, the running speed of the algorithm directly depends on the water volume, so the method is very limited in the extent of amount of water it can simulate while maintaining real time, i.e. a glass of water. Calculations based on the water volume are not adaptable for surface simulation which is characterized by the dynamics that happen on the surface, not the entire water body itself. The research developed by Yuksel, *et. al* [11] presents a novel concept of wave particles as it stresses in its title. Focused on the surface behavior of large bodies of fluids, the implementation demonstrates credible realism with time steps scalable for real-time of detailed wave simulation. However, the simulation represents a globally flowless fluid body, and again this proves to be a limitation in simulating dynamics integrated with horizontal force addition.

3. STABLE FLUIDS

A prestigious *stable fluid* model was developed by Stam[9]. Based on the Navier-Stokes equation, Stam's work provides an accurate and comprehensive simulation method for fluids, and now stands as a bible figure in today's computer graphics society. It is a general purposed model for any type of simulation that exhibits fluid flow. Applications of his work are endless, and the model can be applied for both 2D and 3D fluid simulation. However, the running speed of the algorithm is exponentially proportionate to the number of the dimensions it deals with, so while the model can be run on real time rates at two dimensions, it takes much slower time scale for a three dimensional simulation.

Given an initial state for a fluid in a space, the evolution of its velocity $\boldsymbol{u}$ can be described by the Navier-Stokes equation as Eq. (1).

$$\frac{\partial u}{\partial t} = -(u \bullet \nabla)u - \frac{1}{\rho}\nabla p + \mu \nabla^2 u + f \qquad (1)$$

In (1), $\boldsymbol{p}$ and $\boldsymbol{f}$ are the pressure and the force, respectively, ρ (μ) is a constant for density value (resp., viscosity value).

The four terms on the right side of Eq. (1) each refer to advection, pressure, diffusion, and external force of the fluid in order. Stam's method solves the equation along with another indispensable equation as Eq. (2).

$$\nabla \bullet u = 0 \qquad (2)$$

This equation refers to the incompressibility of fluid, which means the total inflow of fluid should be equal to the total outflow, making the net flow of the fluid zero, and thus conserving mass in an incompressible status. Solving these two equations Eq. (1) and Eq. (2) gives a velocity solver that adds force, advections, diffuses, and finally projects the velocity field, where 'project' means stabilizing the velocity field to make it into a mass conserving field. The projection function is derived from the Hodge decomposition [5], a mathematical result which states that any vector field $\boldsymbol{w}$ can uniquely be decomposed into the equation as Eq. (3).

$$w = u + \nabla q \qquad (3)$$

In Eq. (3), the velocity field $\boldsymbol{u}$ is a mass conserving field, and ∇q is the gradient field. Any vector field is a sum of a mass conserving field and a gradient field. From the Eq. (3), the projection function with respect to the velocity would be derived as follows:

$$u = \mathrm{P}(w) = w - \nabla q \qquad (4)$$

This solution of the Navier-Stokes can be directly applied to simulation of the Vector field either 2D or 3D. The main process of a time step in a 3D application would look like the following pseudocode,

```
Simulator(float* u, float* v, float* w)
{
  addForce(u,v,w);
  Diffuse(u,v,w);
  Project(u,v,w);
  Advect(u,v,w);
  Project(u,v,w);
}
```

The 3D Navier-Stokes equation can be used to calculate velocities in a three dimensional space. In this equation, it can be noticed that the velocity calculation requires iteration through three for-loops to process the velocity in all *x-y-z* direction. Even though the Navier-Stokes equation is the best physically accurate way to handle the velocity, it involves extensive numerical calculations and puts obstacle in real-time simulation.

The current velocity method is not optimized method for both 3D advection effects and real-time performance, so we take a new novel scheme to drastically improve our running time performance while achieving the similar 3D effect based on Navier-Stokes, which will be described in the next section.

4. CONSRUCTING WATER SURFACES

Some observations characterize the water surface in its physical uniqueness. On a water surface, the net mass transportation occurs only in the ***x-y*** direction, while the ***z*** displacement eventually returns to original zero level in the

long run. Thus, we are allowed to simplify the velocity vector field ***w*** which uses first order differentiation function, to a scalar height field that represents the magnitude of the water surface level.

Based on this idea, the water surface where other physical attributes such as density and temperature are kept uniform can be described by the velocity vector <***u,v***>, and a height value ***z***. The spatial coordinates are denoted by ***X=(x, y)***, and these are fixed dimensions. Height value and the velocity vector components are assigned to every point on the water surface, as are the definition of height field and velocity field. Considering that we are interested only in the surface simulation, the important feature of fluid motion we want to see in the vertical direction is the height displacement, which is the representation of water waves and oscillations of interacting objects. On the other hand, the main issue in the horizontal planar motions is the velocity. The velocity becomes a driving source for wave generation and object traversal.

The velocity vector field and the simplified height field correspond with one another while constructed and processed independently. The complexity of the either fields is two dimensional, so the maximum complexity of the simulation can be considered two dimensional as well. This optimization allows us to achieve our main goal to produce 3D effects while the overall complexity is reduced from n^3 to n^2 compared to applying the full vector field <***u,v,w***>. However, the system also leaves some incurred issues to solve.

4.1 Mass Conservation

As we have seen in the previous section, a general velocity field assumes a perfect mass-conserving field. When a certain amount of fluid is flown into a given area, it pushes out the same amount of the fluid to flow out from that area, making a mass-equivalent medium by stabilizing the velocity field. However, the assumption of perfectly equilateral mass distribution does not hold on the proposed system. Our method of water modeling uses only two dimensional velocity field to represent a three dimensional space. Unequal net flow of fluid is bound to occur in such velocities because the velocity field contains only the horizontal planar vector component while it is also correlated with the vertical height dimension, which we simplified as the height field.

When initial force on the water surface creates a random velocity field, the water mass, carried unevenly by the velocity, lump together to build a higher elevated height depending on the net flow of the velocity. If the velocity field surrounding a given unit area has net flow running into the space, the mass would build up in that particular area which would make the water height would rise. Correspondingly, water height will sink if the net fluid flow runs out from the given space.

To embody a mass-conserving simulation, we put both the velocity and height field into account when creating new version of the projection step in Navier-Stokes equation. Taking a view from the law of conservation of mass, a higher water level can result from and only from the equal amount of net inflow into the region. This interpretation can be supported that when a surface is divided into smaller pieces containing the unit area, the water level on a certain point indicates how much water mass, ***M***, is accumulated at the time frame. So for a given velocity field ***u***, the calculation can be made as Eq. (5).

$$\nabla \bullet u = \Delta(\sum M) = \rho \Delta V \qquad (5)$$

As described in Eq. (5), the velocity divergence determines the mass flux and then, based on a constant water density ρ, determines the change, ΔV in volume. Once the volume is determined from the divergence, it is easy to find out the change in height value and the unit area $\bar{s}$ as Eq. (6)

$$\Delta V = \Delta z \times \bar{s} \qquad (6)$$

We can draw the relationship between the elevated height and the velocity using Eq. (5) and Eq. (6). Setting a constant $\kappa = \bar{s} \times \rho$, the relation is made as Eq. (7).

$$\kappa \Delta z = \nabla \bullet u \qquad (7)$$

The Eq. (7) shows the manner that height is changed directly according to the magnitude of divergence ***u***.

4.2 Height Fields

The previous step on velocity calculation, while providing the basic physics system between, also allows us to calculate the height deviation, which we will now fully extend into a visual wave system using a height field. A height field is defined as a continuous function of fluid surface level ***z*** over a planar position $\mathbf{x} = (x, y)$ and a time ***t***. With a ripple speed c that represents the rate at which the wave propagates through the medium, the height field function satisfies the second order wave equation like as Eq. (8).

$$\frac{\partial^2 z}{\partial t^2} = c^2 \left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}\right) \qquad (8)$$

We include an additional term representing viscosity, formulating the wave experiencing through the surface experiencing viscous force. Being that ν stands for the viscosity, Eq.(8) is changed into Eq.(9)

$$\frac{\partial^2 z}{\partial t^2} = c^2 \left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}\right) - \nu \frac{\partial z}{\partial t} \qquad (9)$$

Eq. (9) is the wave equation with viscous damping force. To obtain the height ***z(x,y,k+1)*** of a point (***x,y***) at specific time ***k+1*** using Eq. (9), we can discretize Eq.(9) as the following Eq. (10).

$$\begin{aligned} z(x,y,k+1) = {} & \frac{4 - 8c^2t^2/d^2}{\nu t + 2} z(x,y,k) \\ & + \frac{\nu t - 2}{\nu t + 2} z(x,y,k-1) \\ & + \frac{2c^2t^2/d^2}{\nu t + 2} [z(x+1,y,k) \\ & + z(x-1,y,k) + z(x,y+1,k) \\ & + z(x,y-1,k)] \end{aligned} \qquad (10)$$

The constants in Eq. (10) are defined as the follows.

- d : the distance between adjacent vertices in the grid in both the ***x*** and ***y*** directions
- t : the time interval between calculations of the grid - higher for a faster simulation, lower for slower
- ν : the viscosity of the fluid - a smaller value allows ripples to exist on the surface for a long time, a larger value makes

them diminish faster

c : the speed at which ripples travel across the surface

This numerical algorithm of the wave equation gets the new height value for the $(k+1)^{th}$ time step using the values from k^{th} and $(k-1)^{th}$ time steps. For convenience, we represent the wave equation function in simpler terms as Eq. (11).

$$z_{k+1} = W(z_k, z_{k-1}) \qquad (11)$$

The deviation, Δz, of the height was calculated from the divergence of the velocity field, *i.e.*, Eq. (7) and then the value is added to the fluid level for calculating the height at the k^{th} frame over a planar position $\mathbf{x} = (x, y)$ as Eq. (12).

$$z_k = z(X,k) + \Delta z \qquad (12)$$

By substituting Eq. (12) for Δz_k in Eq. (11), the following equation can be obtained as Eq. (13).

$$z_{k+1} = W(z(X,k) + \Delta z, z_{k-1}) \qquad (13)$$

This is the final form of our function for our general behavior of height field. This way, the height deviation Δz is not directly added up into the rendering scene, but rather included implicitly through the wave equation algorithm to be accounted for the next time step.

4.3 Velocities

According to a wide consent among computer graphics researchers, to provide a physically accurate foundation for general fluid velocity, one must employ the Navier-Stokes equations [2,4,7,9,10], which are the full embodiment of Newton's second law in fluids. For the evolution of our velocity field, we use the stable velocity solver [9,10] which is given detail in Section 3. We give a brief redefinition of the steps in the implementation of the method, along with presenting some optimization for the specific conditions of our model. As proposed in [9], the velocity evolution is largely a three step procedure for a velocity field on a grid: advection, diffusion, and projection.

The advection step is used as it is in [9]. The diffusion step is considered as follows. Diffusion on velocity has a blurring effect on the fluid. The degree of diffusion can be controlled by a user-defined diffusion constant. While our model is already a simplification where losses of detailed effects are likely to occur, we do not want an unnecessary process to impede in the preservation of detailed water vortices, which was explained in Section 4.1. For advantages in both computational cost and simulation quality, the diffusion will be removed from the velocity solver.

Including projection step is questionable, for in the previous section we have pointed out that our model conserves mass not by a single physics property, but by the entire framework. We have seen that an initial force creates a turbulent velocity that disregards the mass conservation, and the height field is constructed by its net flux. However, after the height field is constructed from the initial forces, it receives a second force from the gravity which tries to put it back to an even level. The gravitational forces are included in the wave equation that we apply for our height field after adding the height deviation. As the height field is forced back to an equilateral state, the velocity is also gradually restored with its mass conserving property. Simply put, neglecting the projection step would cause discordance between the height field, which spreads out, and the velocity field, which lump together. The link between the velocity and the height field is a two-way correlation, so our mass conservation is complete when the velocity projection step takes place after the wave equation is applied to the height.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The algorithm is straightforward in its implementation. The water surface is modeled by a planar 2D grid, where values for physical and visual properties are stored in each cell. A two dimensional array or any similar data structure is constructed for a field of each significant physical property (e.g. the velocity). We construct the height field-*height*, velocity field-*(u,v)*, divergence field-*div*, and the pressure field-*p*. We denote the manager including the five arrays as `ArraySet()`.

```
ArraySet() : height(N, N), u(N, N), v(N, N),
             div(N, N), p(N, N)
```

After the data structures are established, the main procedure of each time frame is composed of three steps: update new velocities after adding force, generate heights using wave equation, and finally, project the water's velocities to a mass-conserving state. Each function of those is implemented with the corresponding parameters on the two-dimensional grid structure.

```
NewSimulator(float* u, float* v, float* div,
float* height)
{
  AddForce(u,v);
  Advect(u,v);
  UpdateHeight(u,v,height,div);
  Project(u,v);
}
```

In our system, the force can be given by users directly or by computers automatically. After adding the force to the velocity, `Advect()` and `Project()` are applied which might seem familiar to the steps directly derived from the previously introduced velocity solver. The `Advect()` and `Project()` are individual parts of the large velocity evolving process.

Next, the height field computation is integrated into a whole function `UpdateHeight()`. This step first calculates the height deviation from the divergence of the velocity (Eq. (7)), and adds the height deviation to the k^{th} step of the wave equation (Eq. (13)). The calculation of divergence follows the following finite difference form:

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial y}{\partial x} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{v_{i+1,j} - u_{i-1,j}}{2\delta y}$$

The divergence calculation goes into the `UpdateHeight()` function in our system.

Although `UpdateHeight()` and the two functions `Advect()` and `Project()` operate on completely different fields, the outcome can be very different depending on in what

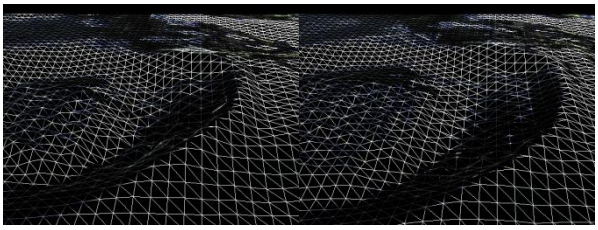
order the functions are executed. It is very important that as shown in the `NewSimulator()`, the `UpdateHeight()` is executed after when the `Advection()` is complete, but `Project()` is not. The `Advection()` basically receives forces and puts the water in motion. After the velocity vector field $\boldsymbol{u}$ and $\boldsymbol{v}$ goes through advection, it may be in a highly physically aroused and unstable status. In reaction to water's sudden turbulence, the following height solver instantly does operations on the vertical dimension to keep the mass-conserving property. Moreover, the `Project()` forces the velocity to have a zero divergence(Eq. (4)), which in that case it would be impossible to derive the height deviation at all. Thus, the height calculation is performed at the end of every force addition, yet before the velocity projection.

The memory requirements for storing the results of the height fields at a specific time frame are rather vast. To do it, it needs two copies for $h(t-\Delta t)$ and $h(t-2\Delta t)$ to be stored for every vertex of the height field $h(\mathbf{x})$. The vector field does not need a duplicate copy, yet it requires two sets of grids to supplement the two directional vector notation. On the other hand, the entire information is only 2D, so the actual requirement of memory capacity is much lower than of any 3D representation.

The proposed method was implemented in C++, using a graphics rendering engine OGRE [12]. All experiments were run on a standard laptop with Dual Core CPU running at 1.83GHz, with 1.00 GB of memory, and ATI Mobility Radeon X1400 with 512MB of video RAM.

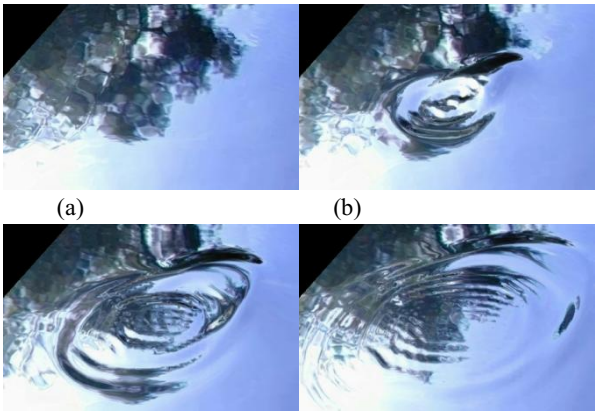
Experiments were held on grid sizes of 64×64, 128×128, 196×196, and 256×256, each populated with 200 boxes of floating objects to show advection. For a grid size of 64×64 populated with 200 boxes, the simulation ran at 300 fps. The simulation ran at 200 fps when the complexity was increased to 128×128, ran at 60 fps for a 196×196, and ran at 30 fps for a 256×256. The timing studies show that our approach achieves plausible real-time performance, and the running speed of the application of the algorithm depended linearly on the size of the simulation, which enables implementation on larger simulations as well without significant increase of running cost.

Figures 1 and 2 show sample experiments tested to prove our major point that the algorithm successfully shapes the water waves under the presence of water advection. Movable objects were placed on a water surface under the effect of a force field induced by wind. The major observation issues were shapes of velocity-induced waves, manner of advection, and boundary reflections.



(a) (b)

Fig. 1: Snapshots of a traversing large sized water wave in a wireframe view mode.



(a) (b)

(c) (d)

Fig. 2: Snapshots of a traversing large sized water wave in a shading view mode

6. CONCLUDING REMARKS

We propose a novel method for simulating water surfaces. The main focus presented here concentrates on creating a real-time simulation of water that deals with full 3D hydrodynamics motions. Our work to achieve it consists of two parts: the independent evolutions of different physics fields, and the integrator combining the attributes of water into a whole framework. In special, the integrator is based on the mass conservation of water, and provides the basic fundamentals of how horizontal forces can influence the vertical dimensions of the surface. The water waves are directly converted from the values of the commonly used height field. The velocities are governed by the Navier-Stokes equations. The sets of algorithms were all based on the 2D Cartesian grid. The global advection across the surface can be handled for large water domains such as the ocean surface, and also in environments that is enclosed within boundaries that may collide with the fluid flow. For a grid size of n, the operation efficiency exhibits performance of $O(n^2)$, which is an efficient performance for a 3D simulation, which makes it possible to render a large-scale simulation while maintaining real-time frame rate.

The future extensions of our approach include involving water particles to generate splashes, breaking of waves, shallow surface simulation [1], [11]. And we are planning to do the stabilization of the entire system. Our current method only offers application to deep water simulations. Also while it provides an unconditionally stable approach to velocities, the

height model is capable of blowing up, therefore our system as a whole involves possible instability. Our approach is highly compatible with any of the fluid simulation techniques that are based on a grid. And with further experiments, we can incorporate other fluid techniques that realize these subsets of fluid behavior that we lack to simulate.

REFERENCES

[1] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, *"Adaptively sampled particle fluids,"* ACM Transaction on Graphhics(Proceedings of ACM SIGGRAPH '07). Vol. 26, No.3, 2007, pp. 48-54.

[2] J. X. Chen and N. V. Lobo, *"Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations,"* Graphical Models and Image Processing, Vol.57, No.2, March, 1995, pp. 107-116.

[3] D. Enright, S. Marschner and R. Fedkiw, *"Animation and Rendering of Complex Water Surfaces,"* Proceedings of ACM SIGGRAPH '02, 2002, pp. 736–744.

[4] N. Foster and R. Fedkiw, *"Practical Animation of Liquids,"* Proceedings of ACM SIGGRAPH '01, 2001, pp. 23–30.

[5] W. R. Fox and A. T. McDonald, *Introduction to Fuild Mechanics*, 4th Edition, Wiley, New York, 1992.

[6] A. Iglesias, *"Computer Graphics for Water Modeling and Rendering: a Survey,"* Future Generation Computer Systems, Vol.20, 2004, pp. 1355-1374.

[7] A. T. Layton and Michiel van de Panne, *"A numerically efficient and stable algorithm for animating water waves, "* The Visual Computer, Vol. 18, No. 1, pp. 41-53, 2002.

[8] J. J. Monaghan, *"Simulating free surface flows with SPH,"* Journal of Computational Physics, Vol.110, No.2, 1994, pp. 399-406.

[9] J. Stam, *"Stable Fluids."* Proceedings of ACM SIGGRAPH '99, 1999, pp. 121–128.

[10] J. Stam, *"Real-Time Fluid Dynamics for Games."* Proceedings of Game Development Conference '03, 2003.

[11] C. Yuksel, D. House and J. Keyser, *"Wave Particles,"* Proceedings of ACM SIGGRAPH '07, Vol.26, No.3, 2007, pp. 1-8.

[12] OGRE, Object-Oriented Graphics Rendering Engine, http://www.ogre3d.org

Dongyoung Kim

He is a student attending in Korean Minjok Leadership Academy, and a national top ranker of Korean Olympiad Informatics(Top Gold Medal, 2007). His research experiences include long-term stays in KAIST(Korea Advanced Institute of Science and Technology) (2007-2008), Korea University(2007), and Chungbuk National University (2008). While interested in computer science altogether, his interest lies primarily on computer graphics and physical modeling of fluids.

Kwan-Hee Yoo

He is a professor of computer education and IIE(information industrial engineering) at Chungbuk National University, Korea. He received the B.S. in computer science from Chonbuk National University, Korea in 1985, and also received M.S., Ph.D. in computer science from KAIST (Korea Advanced Institute of Science and Technology), Korea in 1988 and 1995, respectively. His research interests include computational geometry, computer graphics, 3D character animation, dental/medical applications.