EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Real-Time Animation of Particles and Seaweeds in Underwater Scenes

Y. Coulais
D. Ghazanfarpour
O. Terraz
Laboratoire MSI
Université de Limoges - France
coulais@msi.unilim.fr

S. Thon
Laboratoire LSIS
Université de Provence - France

Abstract

Water is one of the most important natural phenomena to be rendered in computer graphics. Although ocean waves animation has been well studied in Computer graphics, only few studies have been done for underwater animation. In this paper, we present a new real-time method for animation of suspended particles and seaweeds in underwater scenes. One of the main advantages of this method, compared to other approaches, is the real-time animation of submerged objects by taking into account some natural underwater forces that govern their movements, such as forces generated by water surface in deep and shallow waters as well as underwater currents. Taking into account forces generated by ocean surface is an original approach in Computer Graphics. In addition, real-time animation of complex underwater scenes composed of a great number of particles and seaweeds can be performed by the use of appropriate levels of details.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

One of the most ambitious research fields in computer graphics is the realistic representation of the real world. An interactive real-time animation of natural phenomena is a crucial problem to tackle in many domains such as virtual reality, simulation and video games. This is the case of water due to its importance and its omnipresence in the real world.

In computer graphics, some works have been presented for real-time animation of water surfaces in some cases, such as ocean waves far from the coast [HNC02] as well as running waters [TG02]. However, there are few methods in which submerged objects can be animated in real-time by taking into account some natural phenomena. With modification of the damping coefficient, Stam [Sta97] simulated movements of underwater plants by a method for flexible structures animation like trees. Nevertheless, physically based forces generated by surface movements have never been taken into account to animate underwater objects. But this phenomenon is essential for realistic underwater ocean animation, especially in the case of shallow water. Yu [Yu98] (based on Tu's work [Tu96]) presented interactive underwater scenes inhabited by realistic artificial fishes. For improving the realism of the scene, simple seaweeds and plankton are animated by forces generated by underwater currents [WH91] and fishes movements. Brutzman [Bru02] presented an interactive aquarium scene where movements of seaweeds are based on empirical equations deduced by the observation of movements of seaweeds in the real scenes.

We present in this paper a new and simple real-time method for animation of underwater scenes with submerged objects such as seaweeds and suspended particles. In ocean, submerged objects are subject to forces that interact with them and cause their movements. Compared to other approaches, we take into account the two most important underwater forces at the origin of submerged objects movements: water surface movement and underwater currents. As our main goal is a real-time animation, we neglect the interaction and influence that these forces may have on each other. For a low computation time of realistic water surface (where water movements are based on physical data), we use the surface model for deep [TDG00] and shallow water [Gon99]. Flow primitives [WH91] are used to generate underwater currents. In addition, the model supports movement of complex seaweeds (who are animated like [GCF01]) and a characteristic phenomenon of underwater scenes: the presence of suspended particles whose movements are governed by underwater forces. Moreover, real-time animation of complex scenes (presence of hundreds particles and complex seaweeds) can be performed thanks to levels of details for seaweeds and particles.

In this paper we survey the previous works in section 2. In section 3, we study equations used for the animation of suspended particles in deep and shallow ocean waters. Then we present our method for a real-time modelling and rendering of underwater scenes with hundreds of particles with different levels of details. In section 4 we present our method for the animation of seaweeds in underwater scenes using levels of details that allow real-time animation. Section 5 concerns conclusions and futures works.

2. Previous works

In the first part of this section we survey the previous works related to underwater objects animation and in the second part we quickly survey the real-time water surface models.

2.1. Underwater objects animation

There are few previous works related to real-time underwater objects animation.

Tu [Tu96] presented a virtual underwater world inhabited by a variety of realistic artificial fishes. To increase realism of the underwater scene, seaweeds and planktons are animated according to forces generated by underwater currents. Fluid flow primitives presented by [WH91] were used for simulating underwater currents. Interaction between seaweeds and fishes is also supported by [Tu96]. Yu [Yu98] obtained real-time rendering with Tu's method by the use of levels of details on fishes. Nevertheless, this model supports only simple seaweeds (like blade of grass) modelled by segments like a mass-spring chain. Instead of computing forces acting on each seaweed's geometric surface, forces acting on each segment of a seaweed are approximated by fluid viscosity and fluid velocity. Stiffness and damping of each part of a seaweed are not taken into account in forces computation, like force generated by a given joint (intersection between two parts). In addition, forces generated by water surface are not supported.

Stam [Sta97] simulated the effects of turbulence generated by wind on flexible structure, such as tree. For solving the dynamical equation of movement without integrating it for each frame, the equation was transformed in the frequency domain. With this method, movement during a time interval can be precomputed and used for real-time animation. By increasing the damping coefficient, the movement of flexible structures looks like the movement of underwater plants.

Brutzman [Bru02] presented an interactive VRML simulation of the Monterey Bay Aquarium kelps forest (seaweeds with heights up to 50 meters). Seaweeds in aquarium are modelled and animated interactively, but only empirical equations of seaweeds displacement were used for animation and only the force generated by piston pump in aquarium was taken into account in these equations. Moreover, seaweed body is principally modelled by a simple segment, which does not contain articulated junction, so seaweed movements are very restricted.

2.2. Water surface models

In order to compute forces generated by water surface (which depend on water depth), we need models for modelling and animating water surface. In fact, water surface movement influences water volume, causing interaction with objects present in the water volume (see section 3.2.1 for more details). We also need models which combine realism and low cost computation to permit real-time animation. Various models have been proposed in Computer Graphics to tackle these cases.

Fournier-Reeves [FR86] presented a model for animated surface water in deep ocean water. Based on this model, Gonzato [Gon99] presented a real-time model of wave propagation in open sea (deep ocean water) and coastline (intermediate and shallow waters). Cieutat *et al.* [CGG01] presented a maritime training simulator, using model of Gonzato.

Thon and Ghazanfarpour [TDG00] presented a water surface who was modelled with trochoids using Gerstner model [Ger09]. A spectral approach was used in order to parameterise trochoids variables correctly. Hinsinger *et al.* [HNC02] presented a model which allows animation of realistic deep ocean water in real-time. The wave model is similar to [TDG00]: an adaptive scheme was used to animate a large stretch of water in real-time.

Belyaev [Bel03] presented a real-time simulation of water surface, organized in three parts: simulation of water surface was done based on [Tes01]; optical effects like refraction of objects above the water surface and sky reflection on surface were performed; rendering technique of water surface was performed by triangle meshes creation.

In underwater, visibility is rarely more than sixty meters so we don't need to model a large water surface like [HNC02]. In order to perform animation in different ocean waters types, we use the surface model of Thon [TDG00] for deep ocean water and Cieutat [CGG01] for shallow water.

3. Modelling and Animating particles

A characteristic phenomenon of underwater scenes is the presence of suspended particles in water volume. There are many varieties of particles in ocean, like organic (plankton, dissolved organic particles...), inorganic (chemical molecules, dissolved minerals...) or colloidal particles (sand, earth...). As each variety of particles has its own localization in the ocean, each variety should be set at a correct localization for a more realistic underwater scene. For example, colloidal particles are mainly at the bottom of water, contrary to phytoplankton (plant) that is near the water surface in order to absorb sunlight for photosynthesis.

In the other parts of this section, we present modelling and animating suspended particles. Then we present levels of details and results for underwater particles animation.

3.1. Modelling particles

Two representations could be used for modelling suspended particles: 2D texture (low memory and computation cost) or 3D representation (realistic representation). In our case, we use particles with small size (up to a centimetre) and low mass so the volume information is negligible for this kind of particles. It is the same case for particle deformation so we neglect this phenomenon. Moreover, hundreds of particles could easily be present in an underwater scene. A 3D representation will unnecessarily increase computation and memory costs. Therefore, we decide to represent particles with 2D textures (witch orientation could be random or follow the user's vision).

A bounding box system was used to delimit easily the localization of groups of particles form the same type, with random generation of particles positions inside the bounding box. For easier parameterisation, there is only one type of particle per bounding box (bounding boxes can intersect).

3.2. Particles animation

Underwater particles movements are submitted to forces that act on them like gravity, Archimedes's principle, underwater currents or influence of water surface movement. In our case, we animate suspended particles with low mass and low volume, so only the influence of water surface movement and underwater currents are taken into account. Moreover, in order to reduce computation cost, we neglect the interaction and influence that water may have to itself. So interaction and influence of surface movement and underwater currents are neglected. These forces are taken into account separately, this allow us real-time animation with good visual effects.

3.2.1. Water surface movement

For computing and animating ocean waves in deep water, we used the same method that we proposed in [TDG00]. For wave modelling, the model of Gerstner [Ger09] is used: it describes the movement of every water particle by a circle of radius r in a vertical plane around a fixed point (x_0, y_0):

$$\begin{aligned} x &= x_0 + r.\sin(k.x_0 - \omega t) \\ y &= y_0 + r.\cos(k.y_0 - \omega t) \end{aligned} \tag{1}$$

where t is time, k is the spatial frequency (also called wave number) and ω is the temporal frequency. Superposition of these wave trains (also called trochoids) is performed to create a water surface, as stated by the linear theory of ocean waves. A spectral approach is used to correctly parameterise trochoids variables, based on the oceanographical spectrum model of Pierson and Moskowitz [PM64] for ocean waves.

Interaction between wind and water surface causes the movement of fluid particles, the resulting movement depends on the wavelength λ of each wave train (figure 1).

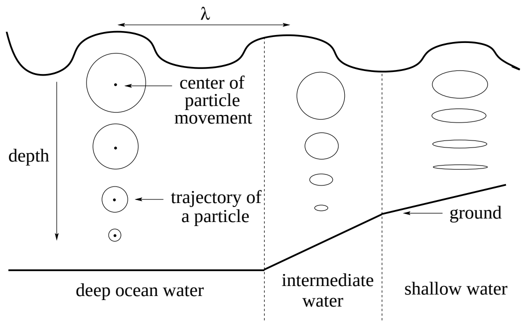

Figure 1: *Particles trajectory in different ocean water types for a single wave train.*

In deep ocean water ($depth > \lambda/2$), water particle movement is circular and its radius decreases exponentially when the depth increases. For intermediate water ($\lambda/20 < depth < \lambda/2$), the movement is elliptic and semi major and semi minor radius decreases exponentially when the depth increases. The movement is also elliptic in shallow water ($depth < \lambda/20$) but only semi minor radius decreases exponentially.

3.2.2. Force of underwater currents

In order to model underwater currents, we use, like [Tu96], fluid flow primitives [WH91] based on an approximation of the Navier Stokes equations. Each fluid flow primitive has its own directions and velocity (figure 2): *uniform flow* has a constant velocity following a single direction, *source flow* is a point in which flow moves out in all direction, *sink flow* is the opposite of source flow and *vortex flow* is a fluid movement in concentric circles.

The sum of these four different flow primitives allows us to model complex underwater currents. Force exerted by flows on particles and objects (composed of vertex) can be computed, these forces depend on the position, area and orientation of underwater objects.

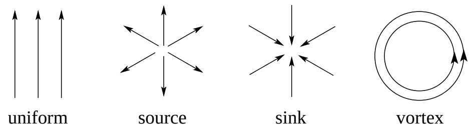

Figure 2: *The four flow primitives.*

3.2.3. Equation of movement

According to equation (1) and the exponentially decrease of the radius with depth y_0, equations of movements of both suspended and water particles, due to forces generated by water surface in deep ocean water, are:

$$\begin{array}{rcl} x & = & x_0 + \sum_{i=1}^{nb} \cos(\theta_i).r_i.\exp(k_i.y_0) \\ & & \times \sin(k_i.(x_0.\cos(\theta_i) + z_0.\sin(\theta_i)) - \omega.t) \\ y & = & y_0 - \sum_{i=1}^{nb} r_i.\exp(k_i.y_0) \\ & & \times \cos(k_i.(x_0.\cos(\theta_i) + z_0.\sin(\theta_i)) - \omega.t) \\ z & = & z_0 + \sum_{i=1}^{nb} \sin(\theta_i).r_i.\exp(k_i.y_0) \\ & & \times \sin(k_i.(x_0.\cos(\theta_i) + z_0.\sin(\theta_i)) - \omega.t) \end{array} \qquad (2)$$

where (x_0, y_0, z_0) is the initial position of the particle (and the center of the particle's circular movement), t is time, k is the spatial frequency, ω is the temporal frequency, θ is the direction of propagation and nb the number of wave trains that compose water surface. In the same way, we can have equations of particles movements in shallow water by using those presented by Gonzato [Gon99].

Forces generated by underwater currents disturb movements of submerged particles generated by water surface. Particles movements only depend on fluid flow velocity, causing movement of initial position of particles. Position of the particles movements at time $t + dt$ depends on position of particle at time t:

$$P_o(t + td) \quad = \quad P_0(t) + V_{uc}dt \qquad (3)$$

where $P_0 = (x_0, y_0, z_0)$ is the center of the particle's circular movements and V_{uc} is the sum of all fluid flow velocity that act on particle P_0. Perturbations generated by underwater currents are taken into account before forces generated by water surface.

3.3. Levels of details

In order to reduce computation cost of particles movements, we use two different kinds of levels of details. The first one concerns movement computation of particles (with an adaptive selection of wave trains), the second one concerns displaying particles (using trompe l'oeil).

3.3.1. Adaptive selection of wave trains

In the same way as [HNC02], we select the number of wave trains nb used in movement computation (equation 2) in order to significantly reduce computation time. Wave trains have not the same amplitude, so their influence on the final trajectory are not the same: the lowest wave train amplitude could have no visible influence on trajectory so this wave train can be ignored.

A particle trajectory of amplitude A at distance d of the point of view is projected on the projection plane (with a perspective projection as shown in figure 3).

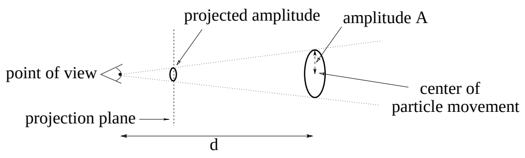

Figure 3: *Perception of particle trajectory.*

If the projected amplitude is lower than a reference amplitude A_{min} (lower than the size of a pixel), the wave train is ignored. In a precomputation step, each wave train is sorted in decreasing amplitude order. For a set of distance intervals, the number of significant wave trains is precomputed and stored in a table. During the animation step, the number of significant wave trains is simply known with the distance between particle and point of view.

3.3.2. Adaptive selection of particles

The idea of adaptive selection of particles is to take only into account visible particles during animation. The distance of visibility in clear waters is around 60 meters, but suspended particles cannot be distinguished separately at a distance of more than 10 meters. Above this limit, humans can only see the interaction between light and particles, the agglomeration of particles acts as a participating medium in this case.

The display of particles is performed in two steps. First, we select only the bounding box of particles which intersects the visual field box. Then, for each selected box, we check particles whose trajectory could be in the visual field: distance between the center of trajectory P_0 of a particle and the visual field is less than, or equal to, the maximum amplitude of a particle movement (figure 4). Only the trajectory of checked particles is performed by equation 2. For simplifying the computations, faces of bounding box are parallel to base axes.

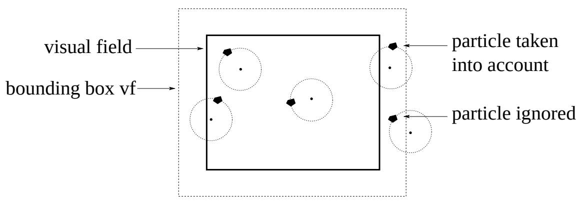

Figure 4: *Adaptive selection of particles.*

3.3.3. Trompe l'oeil

We present here two trompe l'oeil methods for reducing computation cost.

The first one, that we will call the master-slave method, consists of using volumetric agglomeration of particles. Instead of computing the movement of each particle, for a set of particles only the movement (equation 2) of one of them (the master) is computed. Other particles (called slaves) positions are obtained by a single translation from master position. Slave particles are randomly distributed around the master particle in a range between $d_{p_{min}}$ and $d_{p_{max}}$ (figure 5).

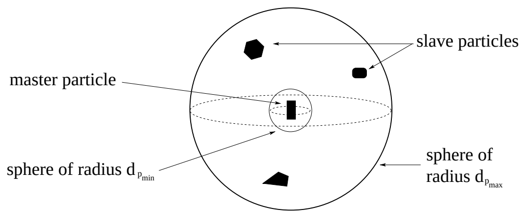
slave particles
master particle
sphere of radius $d_{p_{min}}$
sphere of radius $d_{p_{max}}$

Figure 5: *The master / slave method.*

This method allows us to move several particles with a single movement computation and to provide realistic visual results, even for particles near the user point of view. This method also prevents significantly any agglomeration problem that could appear by precomputing $d_{p_{min}}$ and $d_{p_{max}}$ values by taking into account the number of particles in the bounding box and the particles concentration in this box. Nevertheless, the efficiency of this method depends on the particles concentration in water: if the distance between the master and his slaves is very important, the movement does not look realistic, manual parameterisation of $d_{p_{min}}$ and $d_{p_{max}}$ is the best way to prevent this case.

We use a second trompe l'oeil for a great agglomeration of particles which are very far from the point of view. At a certain distance (D_M), particles are too small to be perceived individually. They act as a local fog. So instead of representing each particle, we use RGBA textures to simulate the presence of these particles.

Smooth transition among these levels of details is performed with an alpha blending on a short distance. Near the point of view, each particle trajectory is computed. At middle distance D_m the "master-slave" modelling is used. For long distance D_M, a local fog (by using RGBA texture) is displayed (figure 6).

3.4. Results

The model runs under Linux systems (AMD AthlonXP 2400+, 512MB RAM, nVIDIA GeForce FX 5700 LE) and OpenGL libraries are used. The test scene is composed of 4.000 particles (in a $10m \times 1m \times 10m$ bounding box), 30 wave trains are used for modelling water surface, the distance of visibility is 30 meters (particles are ignored beyond this distance), there is a simple underwater current in this scene (uniform current). Results, measured in frame per second (FPS), of our method are shown in table 1 with or without wave train selection and levels of details (LoD): *distance* is the minimum distance between point of view and the nearest particle and s/m is the number of slave particles per master particle. The levels of details presented in section 3.3 and transition among these levels of details (3 slaves per master, $D_m = 2$ and $D_M = 12$) are used for the LoD test scene. This method allows us to move inside the scene in real-time with good visual effects despite a great number of particles.

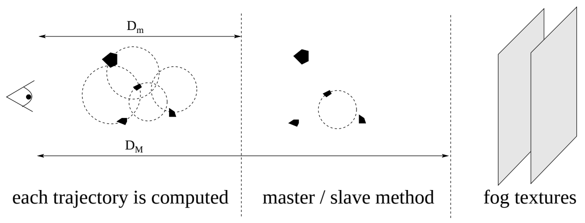

Figure 6: *Transition between levels of details.*

distance (meters)	0	3	6	9	12
no wave train selection					
- no levels of details	11	11	11	11	11
- LoD (with 3 s/m)	20	23	29	45	97
wave train selection					
- no LoD	13	16	21	28	30
- LoD (with 3 s/m)	22	28	38	62	122

Table 1: *animation of particles (in FPS).*

4. Modelling and animating seaweeds

Like particles, seaweeds movement is submitted to external forces that act on it in deep and shallow water. Nevertheless, seaweeds are rather rare on depth higher than sixty meters and they are not always submitted to forces generated by water surface (as seen in section 3.2.1). In deep ocean water, only seaweeds with important lengths (like kelp forest) or seaweeds which are fixed on objects (like a rock) are influenced by forces generated by water surface.

First, we survey animation methods of plants in order to introduce modelling and animating seaweeds. Then we present levels of details and results for real-time animation of seaweeds.

4.1. Animation methods

In computer graphics, animation of other plants than seaweeds has been studied. Perbert and Cani [PC01] used wind primitives [WH91] and precomputed step for animating grass of a prairie. The action of a wind primitive on a blade of grass is precomputed during a physically-based simulation, and computed movement of a blade of grass is stored in memory (each position of a blade of grass is referenced by an index for each wind primitive type). With the use of this data and levels of details, large prairies can be animated in real-time. Endo *et al.* [EMF03] presented a real-time animation model of underbrush such as grass or flowers. In this model, a simple physically based model [WB97] and wind primitives [WH91] are used for animation. During a precomputation step, an elastic force opposed to gravity is used to reach an equilibrium position. The concentration of nutrients on the ground is used in order to define a realistic distribution of underbrush into the scene. Precomputation and levels of details are used in the same way as [PC01]. Di Giacomo *et al.* [GCF01] combined two animation methods for animating a forest in real-time with no precomputed movement data. A procedural method is used for fast animation of tree and a physically-based method for realistic animation and interaction with the user. The transition between these two methods is performed (during a period of about one second) by interpolating angles computed with procedural and physically-based methods. Levels of details are used for both rendering and procedural animation.

In the following, we propose an animated model of seaweeds based on [GCF01].

4.2. Modelling seaweeds

We propose a seaweed model, composed of a skeleton of segments and meshes. The skeleton segments define the topology of seaweeds and the meshes define seaweeds geometry (each segment has an associated mesh). Three geometric parameters (h, θ_i, ϕ_i) are used for defining topology of each segment i with respect to its parent (figure 7), each segment i has its own local base $(0, X_i, Y_i, Z_i)$. The initial position of segment i is obtained by translation h along the Y_{parent} axis of his parents. Then local base of segment i is obtained by two transformations: a rotation by θ_i along Y_p and then a rotation by ϕ_i along X_i.

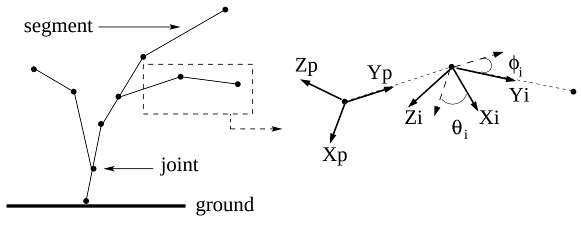

Figure 7: *Local base of segment 'i' with respect to its parent.*

4.3. Seaweeds animation

Animation of a seaweed is performed at each frame by computing θ and ϕ for all segments i of the skeleton. In order to perform animation, external forces must be computed. In this section, we present two animation methods for seaweeds: procedural (for fast animation) and physically-based (for realistic animation).

4.3.1. External forces action on seaweeds

Seaweeds are submitted to the same external forces as particles (sections 3.2.1 and 3.2.2). The force F_{ws} of water surface that acts on each point of an seaweed can be deduced with equation (2), so we have for each segment i:

$$\begin{aligned} F_{ws} &= \int_{solid} F_j \\ &\approx L_i F_c \end{aligned} \tag{4}$$

where F_j is the external force that acts on a point j of the solid (a segment in this case). We approximate F_{ws} by the simplifying assumption $F_j = F_c \ \forall j$ where c is the center of the considered segment and L its length. Area of segment mesh (or an approximation) could be used instead of its length for better approximation.

Forces generated by underwater currents can be obtained as in equation 4 with underwater current [WH91].

4.3.2. Procedural animation

This method consists of applying directly torque generated by external forces without time integration of acceleration, movement is performed around an equilibrium position of seaweeds. External force $F(t)$ applied to a segment i creates a torque at his base, causing motion. The torque is approximated by:

$$\tau(t) = L_i Y_i \times F(t) \tag{5}$$

where t is time, L_i the length of segment i, Y_i axe of local base of segment i and $F(t)$ is external force. Torque is then projected to Y_p and X_i (the two rotation axes for a single segment) in order to compute rotation amplitude:

$$\begin{aligned} \theta_i &= \theta_{i_0} + \tfrac{1}{m_i}\tau(t) Y_p \\ \phi_i &= \phi_{i_0} + \tfrac{1}{m_i}\tau(t) X_i \end{aligned} \tag{6}$$

where θ_{i_0} and ϕ_{i_0} are equilibrium angles of segment i, m_i the mass of segment i, Y_p and X_i the rotations axes and $\tau(t)$ the torque created by F_t.

4.3.3. Physically-based animation

In order to compute physically-based animation, external force F_{ext}, mass m, stiffness k, damping ν and forces generated by joint (link between two segments) are taken into account.

External forces are considered as real forces and second time derivative of these forces are taken into account for computing physical torque in the same way as equation (5).

A torque generated by a joint for a segment i is modelled using linear damped angular springs:

$$(k\theta + \nu\dot{\theta})Y_p + (k\phi + \nu\dot{\phi})X_i \tag{7}$$

The net torque applied to a segment i is the sum of the actions of external forces and joint forces (generated by parents and children):

$$\begin{aligned} \tau \quad = \quad & -\tau_{F_{ext}} - (k\theta + \nu\dot{\theta})Y_p - (k\phi + \nu\dot{\phi})X_i \\ & +(\textstyle\sum_{j \in children}(k_j\theta_j + \nu_j\dot{\theta}_j)Y_i \\ & +(k_j\phi_j + \nu_j\dot{\phi}_j)X_j) \end{aligned} \tag{8}$$

Angular acceleration $\dot{\Omega}_i$ of segment i (with respect to angular acceleration $\dot{\Omega}_p$ of its parent) is then projected to rotation axes of segment i to obtain $\ddot{\theta}_i$ and $\ddot{\phi}_i$:

$$\begin{aligned} \dot{\Omega}_i &= \tfrac{1}{m_i}\tau - \dot{\Omega}_p \\ \ddot{\theta}_i &= Y_p\dot{\Omega}_i \\ \ddot{\phi}_i &= X_i\dot{\Omega}_i \end{aligned} \tag{9}$$

θ_i is then obtained by time integration using standard Euler method (ϕ_i is obtained similarly):

$$\begin{aligned} \dot{\theta}_i(t+dt) &= \dot{\theta}_i(t) + \ddot{\theta}_i(t)dt \\ \theta_i(t+dt) &= \theta_i(t) + \dot{\theta}_i(t)dt \end{aligned} \tag{10}$$

4.3.4. Perturbation generated by seaweeds

When moving, seaweeds cause some perturbations in surrounding water volume. In order to simulate this phenomenon, we use appropriate fluid flow primitives: a single source field is placed at seaweed center of gravity for perturbing objects near seaweed, the force of the source field is proportional to seaweed acceleration and volume influence is limited to the bounding box of seaweed.

More complex perturbations can be done by using various fluid flow primitives such as uniform fields and vortex fields. Thus, it allows to simulate the flow perturbation due to the presence of submerged objects.

4.4. Levels of details

4.4.1. Transition between two animation methods

Transition between physically-based animation and procedural animation for a seaweed movement can be performed like [GCF01]. The idea is to animate only seaweeds near the point of view with physically-based method, seaweeds far from the user point of view are animated procedurally. During a time interval $[t_0..t_1]$, smooth transition is performed between computed θ_i and ϕ_i of the two methods by linear interpolation. The segments of seaweed skeleton could be animated with the combination of these methods: motion of major segments is computed physically and other segments could be fixed or animated with procedural method.

Our approach is slightly different. Instead of computing two movements for a seaweed during transition, we selected segments which movements are computed. Only significant segments of seaweed skeleton are taken into account during the animation for seaweeds far from the point of view. For a seaweed branch, the most important segment is the first one (the segment which is join to branch root) because its movement has repercussion on other segments of the branch (section 4.3.2 and 4.3.3). So for seaweeds far from the point of view (at $distance > FD$), only movement of the first segment of each branch is computed with procedural method, relative positions of other segments i depend on the last (θ_i, ϕ_i) computed. At middle distance ($FD > distance > ND$), a half of the segments (closer to branch root) are supported and near the point of view ($ND > distance$), the movement of each segment is computed.

4.4.2. External forces and modelling

We also use some other approximations in order to reduce computation costs. First, in the same way as section 3.3.1 we only take into account significant wave trains for computing forces of water surface adaptively with respect to the distance, and only seaweeds in visual field are animated (such as particles in figure 4).

Secondly, our modelling technique is more complex than [GCF01] and mesh polygons near the joint of the same axis or between two axes must be computed. During movement computation, we stored in memory data like the local base of each segment (figure 7) in order to compute relative position of polygons involved in joint, this number of polygons depends on the distance between seaweed and the point of view. At short distance we used smooth skinning (with GPU cards possibilities) in order to obtain good visual effects, at middle or long distance single rigid skinning is enough for good visual effects.

4.5. Results

The specifications of the test scene are the same than the ones in section 3.4. The scene takes place in shallow water: there is no particles animation and there is a uniform flow fluid. Seaweeds (simple seaweeds) are placed arbitrarily on the ground in an area of 40×10 meters (length is oriented in the visual field direction), results are shown in table 2.

number of seaweeds	25	50	100	200	300
no Levels of details	63	41	24	13	9
Levels of details	82	59	38	22	16

Table 2: *Animation of seaweeds (in FPS)* .

The levels of details presented in section 4.4 are used with $ND = 5$ meters and $FD = 20$ meters. With the use of these levels of details, the animation of more than a hundred simple seaweeds (Figure 8) could be performed in real-time and the animation of three hundred seaweeds is performed at interactive frame rate. With our methods presented in section 3 and 4, we can animate 100 seaweeds and 2400 particles (3 s/m) at about 20 FPS.

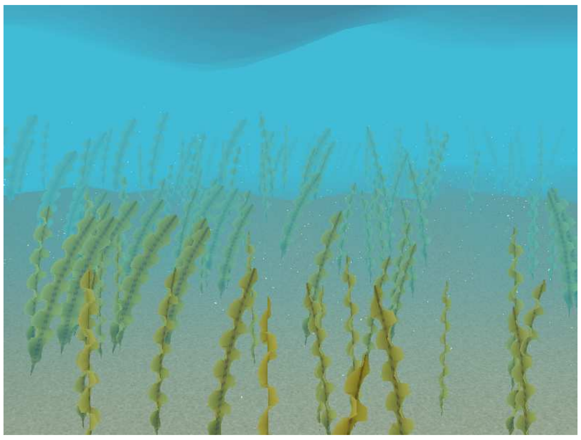

Figure 8: *Scene with particles and simple seaweeds (Laminaria saccharina).*

5. Conclusions and future works

We presented in this paper a real-time method for animation of underwater scenes composed of suspended particles and seaweeds. The forces generated by water surface and underwater currents are taken into account and are computed on the fly for each frame. The animation of complex underwater scenes with a great number of particles and seaweeds is performed in real-time by the use of appropriate levels of details. Compared to other approaches, we took into account the force generated by water surface (in deep and shallow waters), which is one of the most important underwater forces causing submerged objects movements.

For the moment, flow fluid primitives are placed arbitrarily on water volume; later, primitives could be placed automatically as a function of underwater ground. In order to achieve a complete physically-based method for animation of underwater scenes, we will investigate a real-time physically-based method for computing underwater currents, based on a real-time fluid simulation. We will also investigate other underwater forces (like forces generated by fishes movements).

6. Acknowledgements

We would like to thank the Regional Council of Limousin for their financial support of this work.

References

[Bel03] BELYAEV V.: Real-time simulation of water surface. *GraphiCon* (2003).

[Bru02] BRUTZMAN D.: Teaching 3d modeling and simulation : Virtual kelp forest case study. *7th International Conference on 3D Web Technology* (2002), 93–101.

[CGG01] CIEUTAT J., GONZATO J., GUITTON P.: A new efficient wave model for maritime training simulator. *Proc. of Spring Conference on Computer Graphics'2001* (2001), 202–210.

[EMF03] ENDO L., MORIMOTO C., FABRIS A.: Real-time animation of underbrush. *WSCG'2003 11(1)* (2003).

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. *SIGGRAPH'86 20* (1986), 75–84.

[GCF01] GIACOMO T., CAPO S., FAURE F.: An interactive forest. *Eurographics Workshop on Computer Animation and Simulation* (2001), 65–74.

[Ger09] GERSTNER F.: Theorie der wellen. *Annalen der Physik 32* (1809), 412–440.

[Gon99] GONZATO J.: Modélisation des scènes océaniques. *Ph.D Thesis, LaBRI, Université Bordeaux I* (1999).

[HNC02] HINSINGER D., NEYRET F., CANI M.-P.: Interactive animation of ocean waves. *Symposium on Computer Animation* (2002), 161–166.

[PC01] PERBERT F., CANI M.: Animating prairies in real-time. *ACM Interactive 3D Graphics 24(4)* (2001), 103–110.

[PM64] PIERSON W., MOSKOWITHZ L.: A proposed spectral form for fully developped wind seas based on the similarity theory of s.a. kilaigorodskii. *Journal of Geophysical Research* (1964), 5181–5190.

[Sta97] STAM J.: Stochastic dynamics : Simulating the effects of turbulence on flexible structures. *Computer graphics Forum 16* (1997), 159–164.

[TDG00] THON S., DISCHLER J.-M., GHAZANFARPOUR D.: Ocean waves synthesis using a sea spectrum controlled turbulence. *Computer Graphics International 2000 proceedings* (2000), 65–72.

[Tes01] TESSENDORF J.: Simulating ocean water. *Siggraph courses notes* (2001).

[TG02] THON S., GHAZANFARPOUR D.: Real-time animation of running waters based on spectral analysis of navier-stokes equations. *Proceedings of Computer Graphics Internatinal'2002* (2002), 333–346.

[Tu96] TU X.: Artificial animals for computer animation: Biomechanics, locomotion, perception, and behavior. *Ph.D Thesis, Department of Computer Science, University of Toronto* (January 1996).

[WB97] WITKIN A., BARRAF D.: Physically based modeling : principes and practice. *Siggraph'97 Course Notes* (1997).

[WH91] WEJCHERT J., HAUMANN D.: Animation aerodynamics. *Computer Graphics 25* (1991), 19–22.

[Yu98] YU Q.: Synthetic motion capture for interactive virtual worlds. *Ph.D Thesis, Department of Computer Science, University of Toronto* (1998).