Advanced fluid visualisation with DualSPHysics and Blender

Conference Paper · June 2016

CITATION
1

READS
6,897

4 authors, including:

Orlando García Feal
University of Vigo
45 PUBLICATIONS **690** CITATIONS
SEE PROFILE

Alejandro J. C. Crespo
University of Vigo
203 PUBLICATIONS **4,687** CITATIONS
SEE PROFILE

José M. Domínguez
University of Vigo
215 PUBLICATIONS **3,461** CITATIONS
SEE PROFILE

Advanced fluid visualisation with DualSPHysics and Blender

O. García-Feal
Environmental Physics Technologies S.L.
Ourense, Spain
orlando@ephytech.com

A.J.C. Crespo, J.M. Domínguez,
M. Gómez-Gesteira
EPHYSLAB
Universidade de Vigo
Ourense, Spain

***Abstract*— A new tool is proposed to create visualisations with high realism using open-source software. The SPH solver DualSPHysics and Blender are employed in the proposed solution. A Python library interface that can be employed at the Blender console is implemented. This library includes algorithms to import the output data of SPH simulations into a high-quality 3D rendering software where solutions for many visual effects are also provided.**

I. Introduction

SPH has been demonstrated to be an excellent method for free-surface flow simulations. This work explores now its capabilities as a solution for computer fluid animations. The industry of VFX has been using SPH and other methods to simulate water effects over the last years. Some examples can be found in films such as the lava particle simulations for the Gollum drop in "The Lord of the Rings".

On the other hand, visualization is an important tool for the dissemination of scientific results. However, the SPH models oriented to scientific and engineering applications usually present animations with poor visual realism. Most of scientists or engineers are lacking of the required knowledge and skills to handle software oriented to 3D-artists, therefore, more suitable tools should be developed. The solution that we proposed in this work helps to easily create realistic fluid animations based on open-source software. The SPH solver DualSPHysics (www.dual.sphysics.org) and the software for visualisation (rendering, lighting, animations and 3D graphics) Blender (www.blender.org) are used. The aim of the implementation is to improve visualization quality of particle-based fluid simulations.

A graphical user interface (GUI) inside Blender has been developed to be also released as open-source. Hence, the user can perform animations intuitively with only some basic Blender knowledge. With the proposed tool, it is possible to create animations featuring visual effects as realistic materials and lighting, texturing, foam simulation, motion blur and compositing

II. Open-Source Software

A. *DUALSPHYSICS*

DualSPHysics [1] is a free software package for SPH fluid simulations. It has been validated in many scenarios and is highly optimised to be executed in parallel architectures like GPUs and multi-core CPUs. This software also includes several post-processing tools to visualise the output files in the form of isolated points or isosurface. The format of these outputs is VTK (Visualization ToolKit).

B. *VTK format file*

VTK not only supports the particle positions, but also physical quantities that are obtained numerically for the particles involved in the simulations. VTK supports many data types, such as scalar, vector, tensor, texture, and also supports different algorithms such as polygon reduction, mesh smoothing, cutting, contouring and Delaunay triangulation. The VTK file format consists of a header that describes the data and includes any other useful information, the dataset structure with the geometry and topology of the dataset and its attributes. There are many software packages to visualise VTK files like Paraview (www.paraview.org) or Mayavi (http://mayavi.sourceforge.net), however none of them are meant to provide realistic results.

C. *BLENDER*

Blender is a popular open-source 3D graphics software used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. In addition, Blender (since version 2.61) includes a path-tracing render engine (Cycles) that physically simulate light to offer realistic results. The path-tracing method was firstly introduced in 1986 [2] as a Monte Carlo based solution for the Render Equation.

III. OUR CONTRIBUTION

The proposed method is able to import data with the format of scientific visualization tools, like Paraview, into a high-quality 3D rendering software where solutions for many visual effects are also provided.

The workflow shown in Fig. 1 is proposed for this implementation. First, the user will run the simulation with DualSPHysics, then data will be processed to generate visual effects and imported into Blender. Next, the developed Blender plugin will allow the user to work with objects of the simulation and process them as usual Blender objects. Once the user finishes the setup of the scene he can start the rendering of the animation. Finally, the Blender plugin will update the data and object properties at each time step. This implementation can be easily adapted to other SPH based fluid simulators.

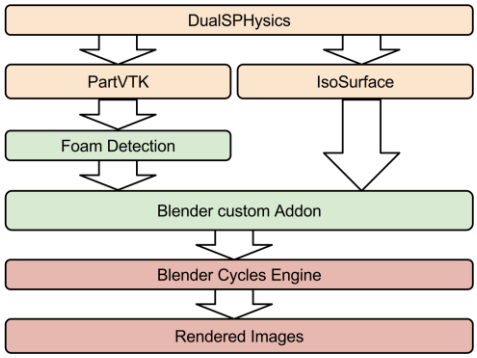

Figure 1. Workflow of the visualisation method.

IV. VISUALIZATION FEATURES

The output files for visualisation are usually particles as seen in Fig. 2a. These particles can be used to create the isosurface where free-surface is detected as shown in Fig. 2b. The isosurface is created using techniques like point splatting [3] or marching cubes [4]. The first one is faster but the second one offers better quality results so it was chosen for non-real time simulations. Note that both output files are VTK and can be imported into scientific visualization software like Paraview. This kind of software displays data in a clear way but without the desired realism. Therefore the following features were implemented in the proposed method to reach higher realism for SPH simulations.

A. *Data import*

Data produced by the simulation software (DualSPHysics or other SPH solver) needs to be converted in some way into a format supported by the rendering software (Blender). However, this approach would use extra memory space and computing time. Thus, the best option is adding support to the rendering software to import those data files directly. For this purpose, data should be loaded into the native data structures, and conveniently updated every time step of the simulation maintaining the object properties given by the user in the rendering software. The Fig. 2c shows rendering performed in Blender with basic materials and lighting.

B. *Texturing*

Textures give realistic appearance to objects of the scene. These textures can even include bump-mapping and other effects to give extra detail to low-poly objects. Bump-mapping consists of giving a rough appearance to the surfaces of objects, modifying the normal of the surface without changing the geometry. On the other hand, textures are easily applied to fixed or external objects, however in the case of moving objects, the texture mapping should be preserved and applied to the geometry each time-step. The texture of wood is applied to the floating box in Fig. 2d.

C. *Materials and lighting*

Proper materials and lighting should be used to achieve realistic results. Complex lighting schemes are supported by Blender including HDRI (High Dynamic Range Images) environment maps. Several materials for water have been designed including glossy, refraction, volume absorption and caustic effects (see Fig. 2e). These effects can be adjusted or even disabled depending on the visualization the user is looking for. For example, the refraction effect can be disabled for a proper visualization of objects inside the water.

D. *Foam effects*

Water flows produce foam, spray or bubbles (*diffuse material* from now on). The method published in [5] was implemented to generate diffuse material particles during post-processing and it will be explained in detail in section V. Fig. 2f shows the result of applying this method.

E. *Motion blur*

Animations generated with computers often seems unnatural and lack of the smoothness of classic film footage. Part of this problem is because blurriness is not considered in fast moving objects. In classic video cameras, the shutter is often not fast enough to freeze the image of moving objects which causes the effect known as motion blur. Blender supports straight forward motion blur for its own moving objects, but not for external animated meshes. This functionality has been implemented to produce smooth animations of the simulated objects. The fast moving elements of the simulation (the box, the fluid and foam) are slightly blurred in the direction of the movement (Fig. 2g).

F. *Post-processing effects and compositing*

Blender supports a wide range of visual effects after rendering. This effects can be, for example, colour correction, contrast, vignetting… This can be useful to adjust the final image (see Fig. 2h). Compositing is also a technique that allows combining images from two sources. For example, backgrounds generated by the computer with live characters in a sci-fi film. This can be used to merge computer fluid simulations with real world structures.

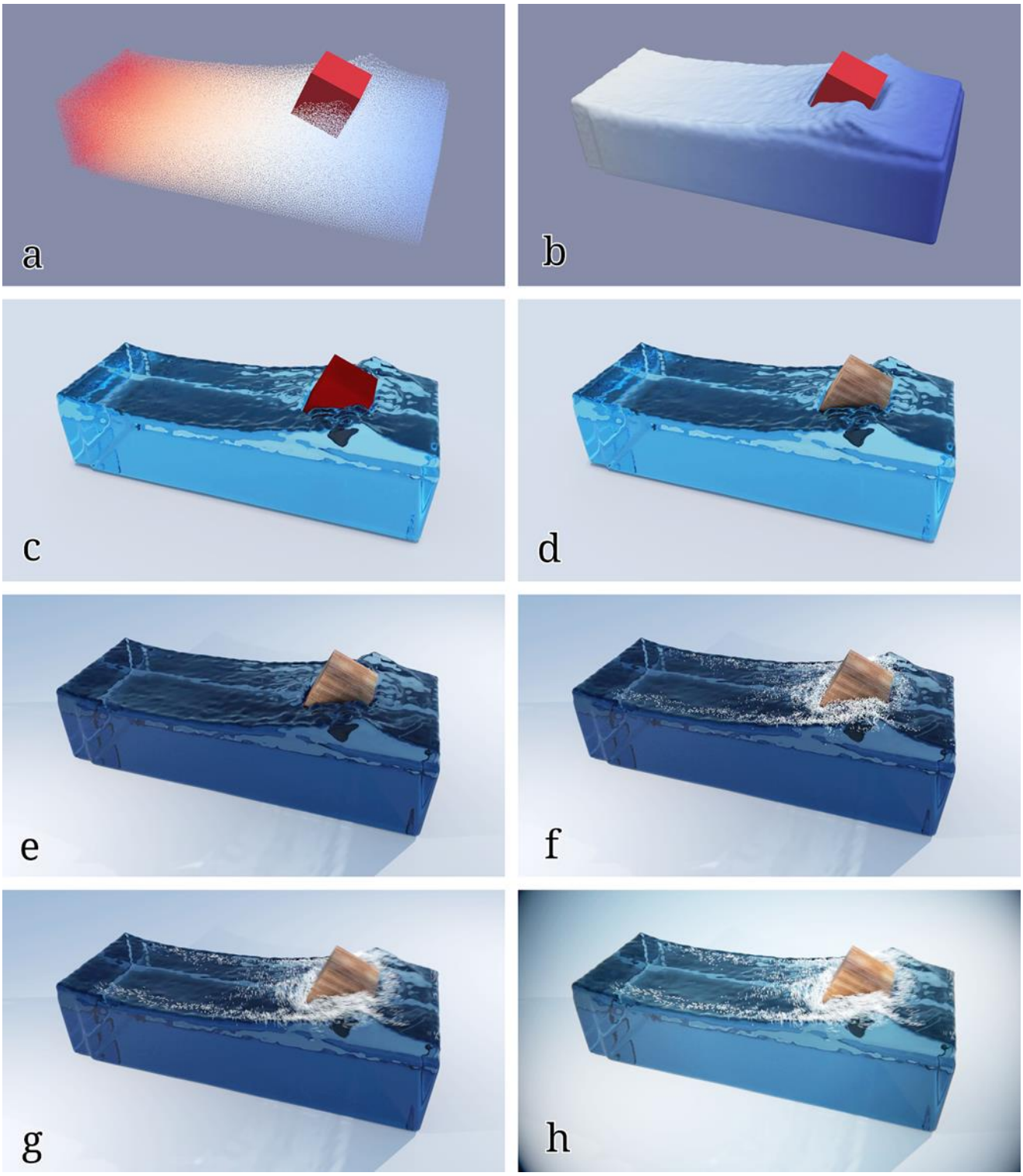

Figure 2. Different levels of improvements (different effects) applied to achieve a realistic visualisation of the SPH simulation.

V. FOAM SIMULATION

Diffuse material is generated in water flows when an air-water mixture is produced (examples can be seen in Fig. 3). This effect can be simulated by means of a two-phase (air-water) fluid simulation [6] or through the surface tension and the Weber number [7]. However, these methods are computationally expensive if we are only focused on visualisation purposes that usually require to be computed during the fluid simulation.

In this solution, the method described in [5] was implemented. The basic idea is to generate new particles (diffuse particles) starting from one existing fluid particle according to some criteria that will be later explained in detail. This approach is computationally efficient and can be executed in a post-processing stage once the fluid simulation is finished. The criteria will define if a fluid particle can be related with air-fluid mixture.

In nature, air-water mixture can be produced when a wave crest becomes unstable or when there are high speed differences in a water flow (see Fig. 3). These new diffuse particles are classified depending on their position in relation with the fluid as; foam, spray or bubbles. Since these particles do not interact with each other, this method is fast and allows the simulation of a high amount of diffuse particles. The number of diffuse particles can be adjusted by changing the values of thresholds given by the user.

Figure 3. Diffuse material generated in nature.

A. Trapped air calculation

Diffuse material is first detected where high differences in the velocity field are observed (due to water-structure impacts or at turbulent zones). The velocity difference for each fluid particle with all its neighbours is computed:

$$v_i^{diff} = \sum_j \|v_{ij}\| \left(1 - \hat{v}_{ij} \cdot \hat{x}_{ij}\right) W\left(x_{ij}, h\right) \quad (1)$$

$$\hat{v}_{ij} = \frac{v_i - v_j}{\|v_i - v_j\|} \quad (2)$$

$$\hat{x}_{ij} = \frac{x_i - x_j}{\|x_i - x_j\|} \quad (3)$$

where $\hat{v}_{ij}$ is the normalised relative velocity between two particles and $\hat{x}_{ij}$ the normalised distance. W is the smoothing kernel The Wendland kernel [8] has shown to provide good results for this task:

$$W(r,h) = \begin{cases} \frac{21}{16\pi h^3}\left(1 - \frac{q}{2}\right)^4 (2q+1), & 0 \le q \le 2 \\ 0, & otherwise \end{cases} \quad (4)$$

where $q = r/h$ and r is the distance between two particles.

B. Wave crests detection

Wave crests are characterised by a high curvature and a locally convex surface. Diffuse material is produced by the action of the wind or the inherent instability of the wave. In order to find these zones, surface particles should be first identified by using the method by [9].

This method uses the concept of the colour field; a value of 1 where there is fluid and 0 where there is no fluid. Since discrete particles are used, a smoothed colour field must be used:

$$c_s(x) = \sum_j m_j \frac{1}{\rho_j} W\left(x - x_j, h\right) \quad (5)$$

The value of the colour field is calculated for a fluid particle considering its neighbours at x_j; m is the mass of each particle, ρ the density and h the smoothing distance.

The colour field gradient can be used to obtain the surface normal for each particle and it is defined as follows:

$$n = \nabla c_s \quad (6)$$

Thereby, the calculation of the curvature k of a set of surface particles can be approximated by:

$$k_i = \sum_j k_{ij} = \sum_j \left(1 - \hat{n}_i \cdot \hat{n}_j\right) W\left(x_{ij}, h\right) \quad (7)$$

where n is a normalised surface normal. The relative angle between $\hat{n}_i$ and $\hat{x}_{ij}$ is considered to identify convex zones:

$$\tilde{k}_i = \sum_j \tilde{k}_{ij} \quad (8)$$

$$\tilde{k}_{ij} = \begin{cases} 0, & \hat{x}_{ij} \cdot \hat{n}_i \ge 0 \\ k_{ij}, & \hat{x}_{ij} \cdot \hat{n}_i < 0 \end{cases} \quad (9)$$

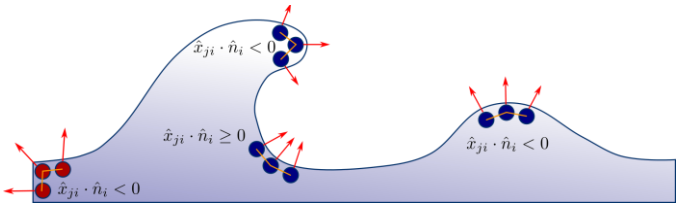

Figure 4. Convex zones are detected using the surface normal of a particle and the position relative to its neighbours.

The expression shown in Eq. (8) is suitable to identify wave crests (Fig. 4). However all water edges are also detected (red points) at the limits of the container. Thus, only fluid particles that move in the same direction as the surface normal will be considered to belong to the wave crest. This can be achieved by multiplying Eq. (8) by the factor:

$$\delta_i^{vn} = \begin{cases} 0, & \hat{v}_i \cdot \hat{n}_i < 0.6 \\ 1, & \hat{v}_i \cdot \hat{n}_i \geq 0.6 \end{cases} \quad (10)$$

C. Kinetic energy

This method does not take into account the surface tension due the excessive computational time and complexity. Constant surface tension is assumed and only the kinetic energy is considered to adjust the amount of diffuse material to be generated. Therefore the amount of kinetic energy of a fluid particle will also be used as a condition to create new diffuse particles.

D. Particle generation

The number of diffuse particles produced by one fluid particle is determined by; i) velocity difference of the fluid particle with all its neighbours, ii) if the fluid particle belongs to wave crests and iii) the value of the kinetic energy of the particle. In addition, the thresholds defined by the user can help to reduce or increase the amount of generated diffuse material.

Considering the position of a fluid particle as $x_f(t)$ at the instant t, the diffuse particles will be initially randomly located in a cylinder whose radius will be the radius of fluid particle volume (r_v) and the height will be defined by the velocity of the fluid particle ($\Delta t \cdot v_f$). Fig. 5 shows the cylinder where 5 diffuse particles are initially located.

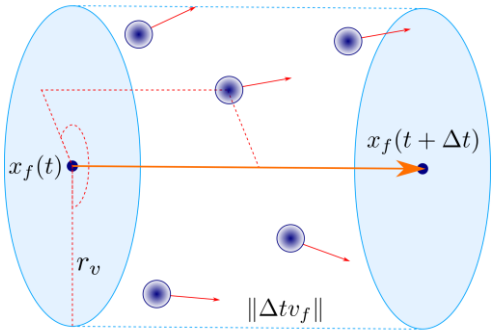

Figure 5. Diffuse particles are located into a cilynder.

Positions and velocity of the new diffuse particles are determined based on random numbers and as modification of the position and velocity of the fluid particle used to generate them.

E. Motion and dissolution

Diffuse particles are classified according to their position relative to the free-surface. If a particle is outside the water it will be classified as spray, if it is inside the water it will be a bubble, and if it is at the free-surface it will be foam. This classification is carried by computing the kernel summation. Fig. 6 shows an example of the three different types.

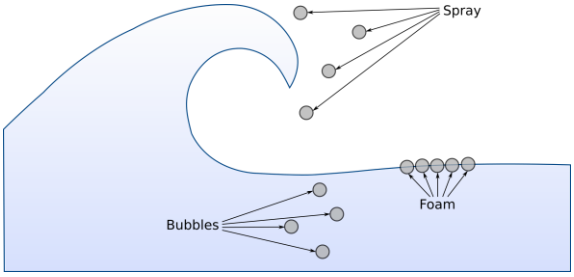

Figure 6. Type of diffuse particles: spray, bubble and foam.

Each type of particle follows a different behaviour. Spray particles will move in a ballistic way, only initial velocity and gravity will be considered. Foam particles are dragged by surrounding fluid particles. Bubbles will be dragged by surrounding fluid particles but a buoyancy force in the opposite direction of the gravity will also be considered.

All diffuse particles are created with a life time. This life time can be adjusted by the user. However, since in nature large masses of foam last longer than shorter ones, this time of life scales with the number of created particles in a certain area. This is, the more diffuse particles are generated by a single fluid particle, the longer they last. At some point, bubbles and spray will become foam, and the life time of foam particles decreases at each time step of the simulation. Once this time is over, the particle is removed.

F. Rendering

A tiny sphere is created for each diffuse particle. Spheres, that are too close, are merged into a larger one, reducing the geometric complexity of the final scene and increasing the randomness appearance of the diffuse material.

G. Implementation details

The implementation of these algorithms was carried out in C++11 and parallelised with OpenMP. A Python library interface was implemented so it can be easily employed with a Python script or at the Blender console.

VI. GRAPHICAL USER INTERFACE

The proposed solution aims to facilitate its use so a graphical user interface (GUI) is crucial. Work with scripts and console programs are a powerful tool but the learning curve is steeper and less intuitive than graphical interfaces. In this way, all the intensive computational parts of the software (like the foam simulator and importing VTK files) are implemented as a C++ library with a Python interface that can be easily integrated into Blender but advanced users can also develop their own custom Python scripts.

All the functionalities have been integrated into the Blender GUI, which allows the user with basic Blender knowledge to intuitively import objects from the SPH simulation and to add effects. Fig. 7 shows the special contextual options in the file browser when files of the SPH simulation are imported. Fluid objects have a special foam simulation menu as shown in Fig. 8.

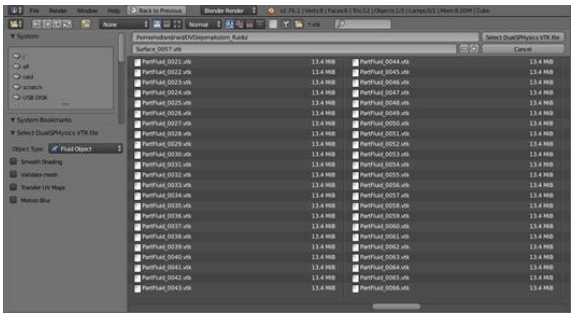

Figure 7. New object types are now included to Blender.

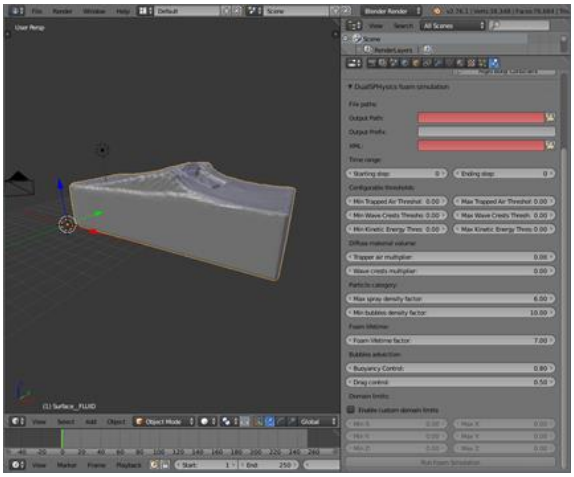

Figure 8. Fluid objects have a special foam simulation menu.

VII. EXAMPLES

Some examples generated with the proposed method are shown in this section to demonstrate the different functionalities.

In the first example (Fig. 9), a boat is under the action of waves. The materials and texture mapping are maintained over time. The resulting animation is visually attractive, however foam and spray that would be expectable in the real world are missing.

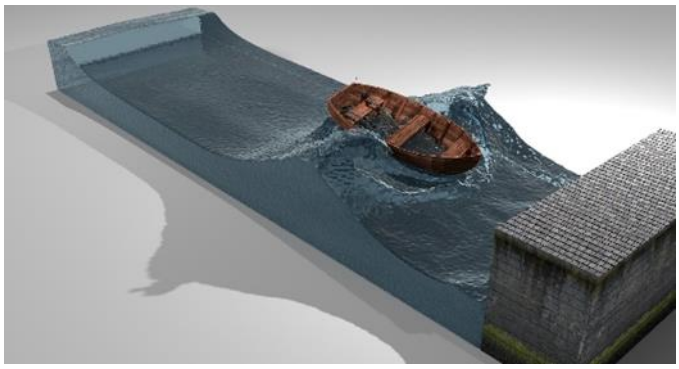

Figure 9. Animation of a floating boat.

In the second example (Fig. 10), the foam effects are included in the simulation of waves interacting with a breakwater. Most of the foam is generated at the wave crest.

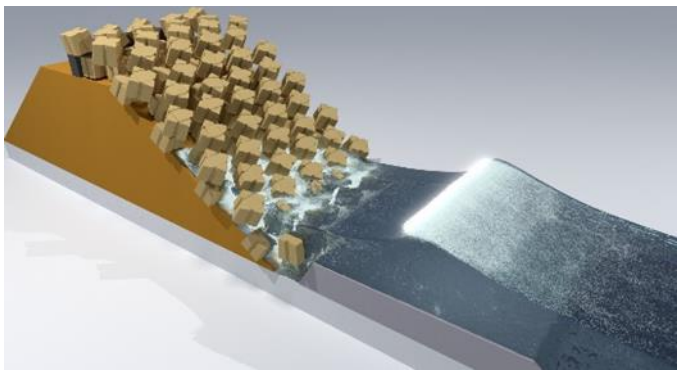

Figure 10. Animation of waves interacting with a breakwater.

In the third example (Fig. 11), a ship is moving. In this case, most of the foam is generated due the high velocity differences and the typical wake can also be observed.

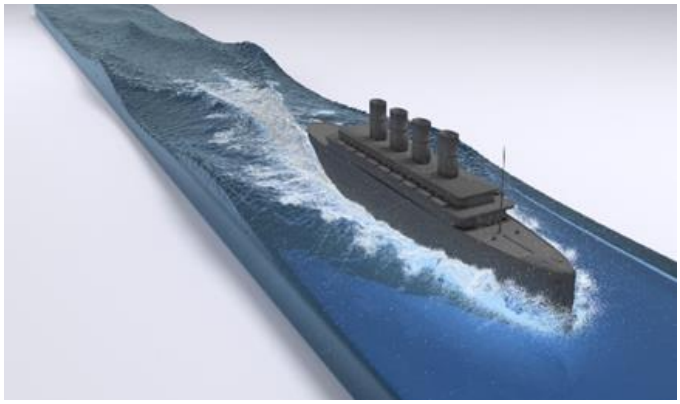

Figure 11. Animation of a boat.

The solution can also be used for the animation of realistic water flowing over complex surfaces. Thus, a natural creek can be seen in Fig. 12. Foam is produced in the waterfalls and the smoothness of the fast moving zones of the water flow due the motion blur can be also observed.

Figure 12. Simulation of a natural creek.

VIII. CONCLUSIONS

This works presents a solution to create high quality animations of water based on open-source tools like DualSPHysics and Blender. The aim of the implementation is to improve visualization quality of particle-based fluid simulations.

The method imports VTK files into a high-quality 3D rendering where the following features were implemented to reach high realism: texturing, materials and lighting, foam effects, motion blur, colour correction, constrast, vignetting and compositing.

A Python library interface is implemented and can be easily employed with a Python script or at the Blender console. In addition, a graphical user interface (GUI) inside Blender has been also developed and will be released as open-source.

The proposed solution is developed to be used in conjunction with DualSPHysics, however it can be easily adapted to other codes based on particle methods.

ACKNOWLEDGEMENT

This work was partially financed by Xunta de Galicia under project “Programa de Consolidación e Estructuración de Unidades de Investigación Competitivas (Grupos de Referencia Competitiva)”. The authors thank to EPHYTECH S.L. (spin-off of the University of Vigo) for the support developing this new tool of visualisation.

REFERENCES

[1] Crespo AJ, Domínguez JM, Rogers BD, Gómez-Gesteira M, Longshaw S, Canelas R, Vacondio R, Barreiro A, García-Feal O. DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH). Computer Physics Communications. 2015, 28;187:204-16.

[2] Kajiya JT. The rendering equation. InACM Siggraph Computer Graphics 1986 Aug 31 (Vol. 20, No. 4, pp. 143-150). ACM.

[3] Zwicker M, Pfister H, Van Baar J, Gross M. Surface splatting. InProceedings of the 28th annual conference on Computer graphics and interactive techniques 2001 Aug 1 (pp. 371-378). ACM.

[4] Lorensen WE, Cline HE. Marching cubes: A high resolution 3D surface construction algorithm. InACM siggraph computer graphics 1987 Aug 1 (Vol. 21, No. 4, pp. 163-169). ACM.

[5] Ihmsen M, Akinci N, Akinci G, Teschner M. Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 2012 Jun 1;28(6-8):669-77.

[6] Takahashi T, Fujii H, Kunimatsu A, Hiwada K, Saito T, Tanaka K, Ueki H. Realistic animation of fluid with splash and foam. InComputer Graphics Forum 2003 Sep 1 (Vol. 22, No. 3, pp. 391-400). Blackwell Publishing, Inc.

[7] Bagar F, Scherzer D, Wimmer M. A Layered Particle-Based Fluid Model for Real-Time Rendering of Water. InComputer Graphics Forum 2010 Jun 1 (Vol. 29, No. 4, pp. 1383-1389). Blackwell Publishing Ltd.

[8] Wendland H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Advances in computational Mathematics. 1995, 1;4(1):389-96.

[9] Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. InProceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation 2003 Jul 26 (pp. 154-159). Eurographics Association.