SPH for Visual Applications

Caio José dos Santos Brito

September 2014

Summer Project Document submitted to the University of Wales, Swansea
in Partial Fulfilment for the Degree of Bachelor of Science

Department of Computer Science
University of Wales Swansea

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted for any degree.

Name: ……..

Signed: …….

Date: ……..

Statement 1

This dissertation is being submitted in partial fulfillment of the requirements of the Science without Borders Programme.

Signed: …….

Date: ……

Statement 2

This dissertation is the result of my own independent work / investigation, except where otherwise stated. Other sources are specifically acknowledged by clear cross referencing to author, work and page(s) using the bibliography / references section. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Signed: …….

Date: …….

Statement 3

I hereby give consent for my dissertation to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed: …….

Date: ……..

Abstract

This project have the goal of implementing a Smoothed Particle Hydrodynamics (SPH) for liquid simulations based in the paper Ghost SPH for Animating Water by Schechter and Bridson. The algorithm creates new layer of particles around the surface using a Poisson Disk scheme to solve the free surface problem. To test the algorithm, three test cases were done using a 2D Dam Break problem and used Metaballs rendering method to show the results. The simulation proved realistic but with a long time to create the new layers of particles and the rendering give a proper result.

Acknowledgment

I want to thank the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the sponsorship and the Universidade Federal de Pernambuco (UFPE) for the suport. I am thankfull to Swansea University for the study and reasearch opportunity and also would like to thank Dr. Mark Jones and Dr. Chengfeng Li for tutoring me in the project.

Contents

1 Introduction

1.1 Motivation

Scenes involving fluid phenomena as oceans, smoke, and fire has been increasingly common in the animation movie scenery and in the game industry and usually are the most spectacular visual effects using realistic rendering or not.
Pixar's movies, for instance, always brings some new improvement related to physics simulation either in fluids simulation as in many other kinds of computer simulation like hair [1], clothes [2] or deformable models [3].
In the movie Cars 2, the studio implemented a simulation for ocean surface using the J. Tessendorf's wave system [4]. In Ratatouille the studio used their in-house simulator, *splasht* [5], which is used to create particles-based simulation.
Although Computation Fluid Dynamics (CFD) is quite stablished and well used, there are still some open problems in the field as turbulence, free surface, surface tension, high performance simulation.
This project have the focus in Smoother Particle Hydrodynamics (SPH), which is a CFD approach that represent fluids as a fixed number of particles that interact via kernel functions

1.2 Related Work

SPH method was created by Lucy, Gingold and Monaghan in 1977 to simulate astrophysical phenomena [6], and later was used for fluids. Initially used for simulation just a small quantity of particles, the method was extended to a greater number of particles and used for interactive application [7].
In [8] SPH is used to simulate fluids with turbulence using a modified way to compute the velocity of the particles and so creating a better stability in the simulation and a removing noisy motion. Solenthaler [9] proposed a modified SPH for a simulation for different fluids mixed and Teschner [10] implemented a SPH algorithm to simulate rigid-fluid simulation.
SPH is also used to simulation material that have a similar behaviour, as in [11] wich the technique is used to simulate sand.
Most of those simulations are done offline but there are many applications research for real time simulations, which is the case of [12] which uses GPU programming to accelerate the simulation process.

1.3 Objectives

The goal of this project is to implement a SPH simulation in 2D based on the approach proposed by Schechter and Bridson, Ghost SPH for Animating Water [13].
These proposal describe how to sample ghost particles in the air and how to extrapolate the fluid characteristics to the ghost particles. Therewith, the free surface problem [6] can be solved and the simulation can have a greater liquid behaviour.
Other aim is to render the simulation results, to achieve this purpose Metaballs (Blob) technique was used to give a proper rendering [14].

2 Method

2.1 Basic SPH

Smoothed Particle Hydrodynamics is a Lagrangian approach that treats the continuum as a finite number of particles. The fluid in the continuum form has position x, velocity u, density ρ, pressure P, mass m and their movement are described by equation 1[6].

$$\rho \frac{d\vec{u}}{dt} = \nabla P + \nabla \cdot \vec{\tau} + Fext \quad (1)$$

where τ is the viscosity, $Fext$ is the body force and t is the time.
The SPH method discretize the fluid into a set of particles i with the same attributes described above and with a radius of action h, as illustrated in Figure 1

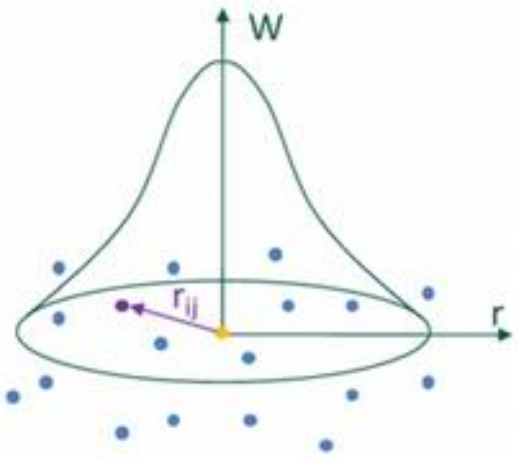


Figure 1-Scheme showing a particle with its radios of action

In the method any attributes A(x) can be described as (2)

$$A(i) = \sum_j m_j \frac{A_j}{\rho_j} W(x_i - x_j, h) \quad (2)$$

where W is the kernel function and j are the neighbour particles.
The kernel function W is a smooth, symmetric, normalized function with finite support, the same function can be used to many variables or a different kernel to an each variable
From this equation the gradient and the laplacian can be calculated as in (3) and (4)

$$\nabla A(i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(x_i - x_j, h) \quad (3)$$

$$\nabla^2 A(i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(x_i - x_j, h) \quad (4)$$

To get the neighbourhood from a single particle, a geometric comparison is used between the distances of two particles. If a couple of particles are within a distance less than the radius of action, those particles are neighbours of each other.
The density of a particle can be calculate with respect to its neighbourhood, using equation (5) or using the convergence equation (6).

$$\rho_i = \sum_j m_j \, w_{ij} \tag{5}$$

$$\frac{d\rho_i}{dt} = \sum_j m_j (v_i - v_j) \, \nabla \mathrm{w}_{ij} \tag{6}$$

There are many form of calculation the pressure exerted on a particles by the other particles. For high compressibility an ideal gas equation (7) can be used; if the simulation requires an incompressibility of the fluid the Poisson equation (8) can be used; or the Tait's equation (9) can be used to enforce a very low density variation and leads to an efficient computation [15].

$$P_i = k_p(\rho_i - \rho_0) \tag{7}$$

$$\nabla \cdot \left(\frac{1}{\rho} \nabla P^{n+1}\right)_i = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}_i \tag{8}$$

$$P_i = B((\frac{\rho_i}{\rho_0})^{\gamma} - 1) \tag{9}$$

Where k_p and B are the pressure constants and ρ_0 is the rest density of the fluid and γ is a constant that usually has a value 7.

The pressure force can be calculate using the discretization of the pressure gradient, which will result in (10), this formula is largely used because ensures a modular equality between two particles and conserves linear and angular momentum, creating a greater stability in the simulation [6].

$$F = -m_i \sum_j m_j \left(\frac{P_i}{\rho^2{}_i} + \frac{P_j}{\rho^2{}_j}\right) \nabla \mathrm{w}_{ij} \tag{10}$$

To calculate the viscosity usually an artificial viscosity is used, as in Becker and Teschner research [15], that uses equation (11) which conserves linear and angular momentum.

$$\frac{dv_i}{dt} = \begin{cases} -\sum_j m_j \Pi_{ij} \nabla \mathrm{w}_{ij} & v_{ij}^T x_{ij} < 0 \\ 0 & v_{ij}^T x_{ij} < 0 \end{cases}$$

To prevent the fluid to escape from its recipient, the Lennad-Jones method is used to create a repulsive force near boundary particles [6].

The particle velocities is calculated doing a simple integration using (11) and the new position is found using (12).

$$v_i = v_i + a_i \nabla t \tag{11}$$

$$x_i = x_i + v_i \nabla t \tag{12}$$

2.2 Ghost SPH

These technique is a variation of the Basic SPH created by Schecter and Bridson [13], that dynamically create a number of ghost particles in the surface of the fluid with a blue noise distribution and extrapolate the technical features of the particles into the new ghost particles to use those particles in some of the calculations. The algorithm can be summarized as in the Figure 2

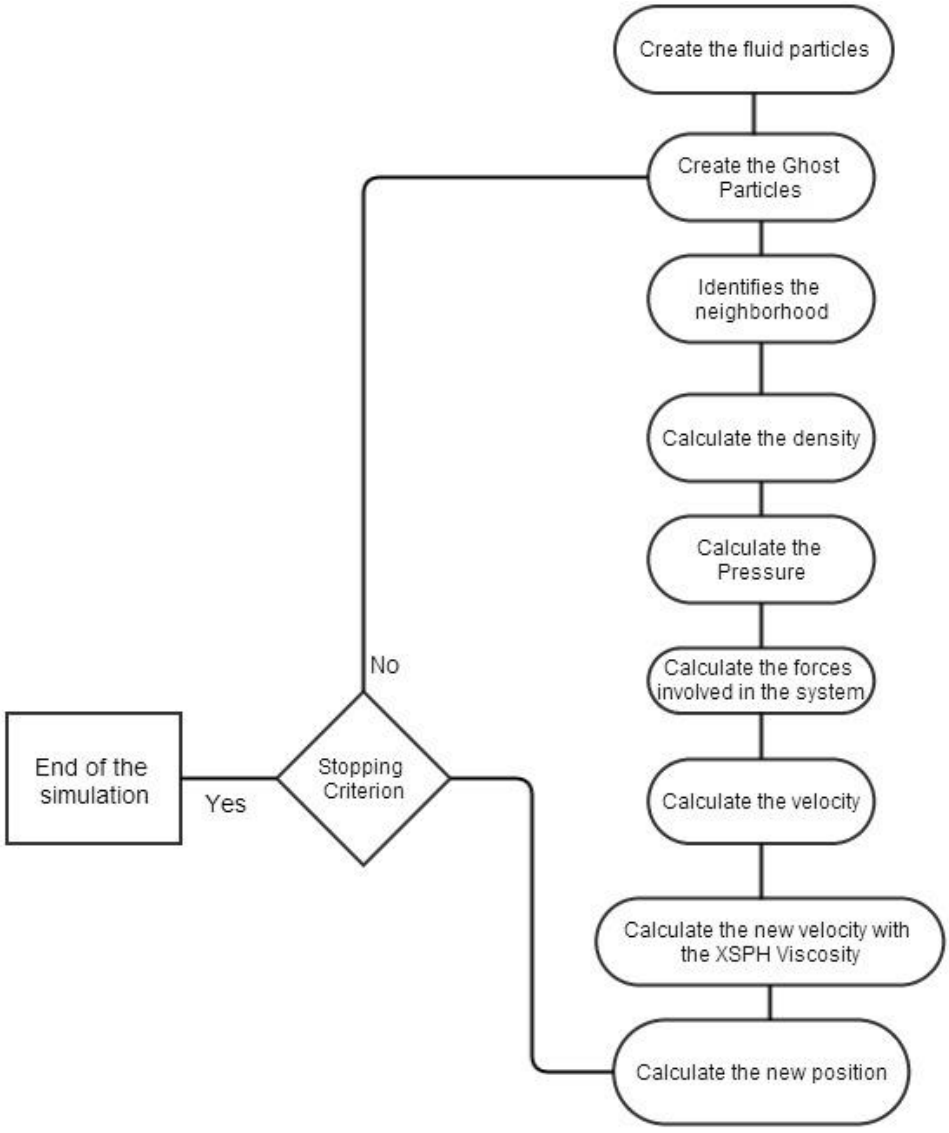


Figure 2 – Flow diagram from the Ghost SPH

In the Ghost SPH, uses a Cubic Spline kernel (13) , the density is calculated by using the convergence equation and it uses Tait's equation (9), with a B = 2000, to solve the pressure equation. As in the Basic SPH, the pressure force is calculated using (6) and the velocities and position are calculated by a simple integration.

$$w(\vec{r}-\vec{r},h)=w(R,h)=\alpha_d\times\begin{cases}\frac{2}{3}-R^2+\frac{1}{2}R^3 & 0\le R<1\\ \frac{1}{6}(2-R)^3 & 1\le R<2\\ 0 & R\ge 2\end{cases} \quad (13)$$

2.2.1 XSPH Artificial Viscosity

Different of the basic algorithm uses a simpler XSPH style of damping noise which is cheaper and easier to tune [13].
With that change, an intermediate velocity v^* is created (14) and is used to calculate the velocity with the viscosity force (15)

$$v_i^* = v_i + a_i \nabla t \quad (14)$$

$$v_i = v_i^* + \in \sum_j m_b \frac{(v_i^* - v_j^*)}{\bar{\rho_j}} w_{ij} \quad (15)$$

2.2.2 Ghost particles in the air

To solve the free surface density problem, air particles are created in the beginning of each time step inside the kernel radius of the liquid, as can be seen in Figure 3

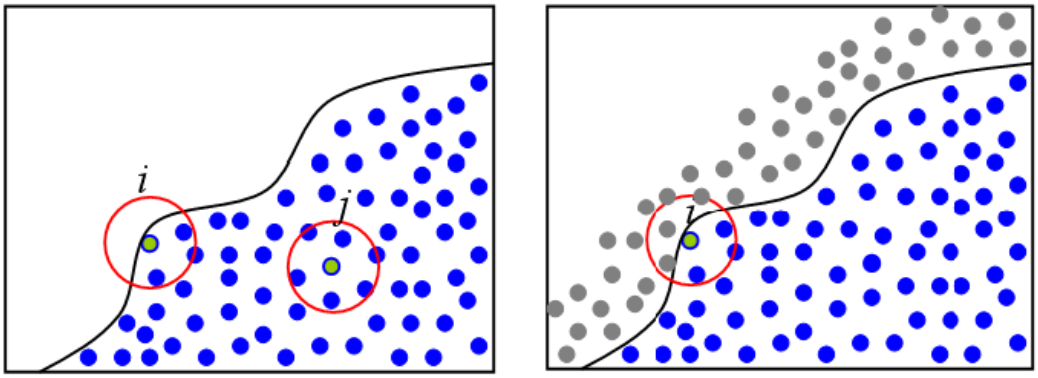
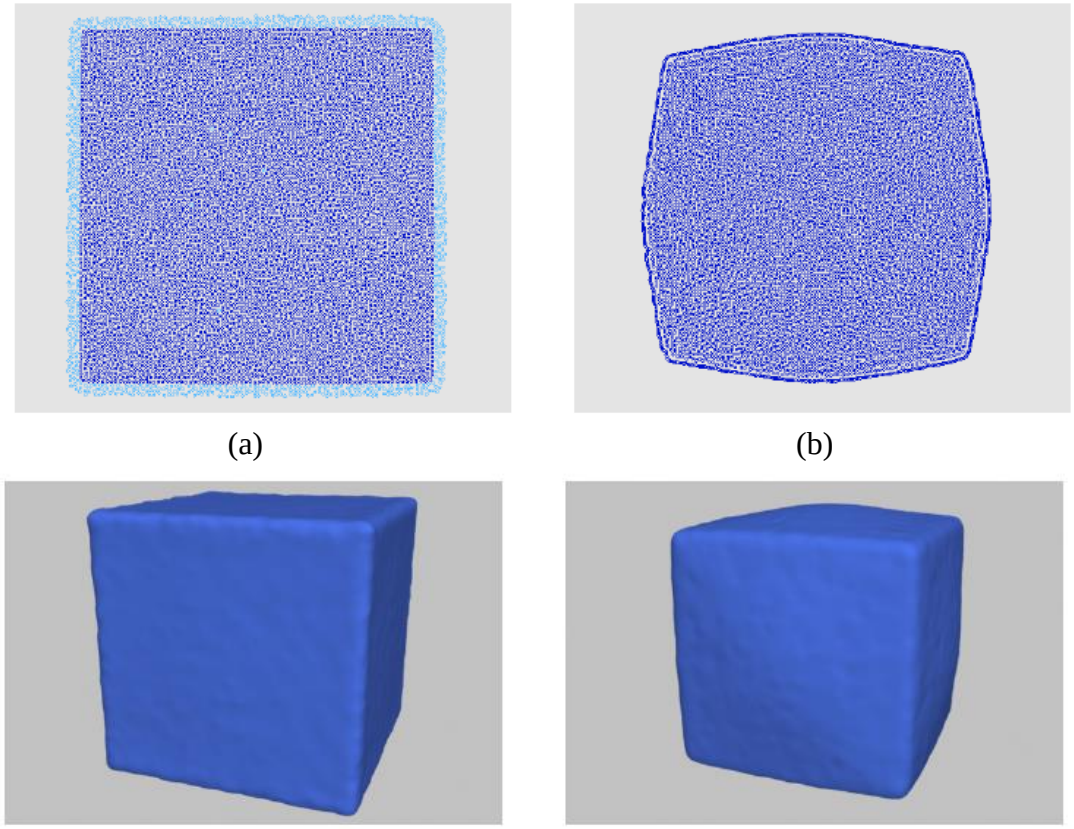


Figure 3 – Ghost particles in the air. Fluids particles are shown in blue and ghost particles in grey. Figure taken from [13]

Each new ghost particle has some characteristic:

- Mass of a liquid particle
- Velocity equals of the nearest liquid particle
- Density equal to the rest density, with that will be under P = 0 creating a free surface boundary condition.
- Contributes to the density calculation
- No contribution to the pressure force
- Explicitly excluded from the XSPH artificial viscosity

Those new particles in the simulation fill a enough layer around the surface, with that, the free surface density problem is solved, as it can be seen in the Figure 4 that shows the result from a zero-gravity hydrostatic test with Basic SPH vs Ghost SPH. The Ghost method keeps the fluid stable, keeping the shape and the volume as the original, meanwhile the Basic SPH does not conserves the shape and volume, creating a (free surface problem) [13].



(a) (b)

(c) (d)

Figure 4 – Hydrostatic Test.(a) Ghost SPH in 2D; (b) Basic SPH in 2D; (c) Ghost SPH in 3D (d) Basic SPH in 3D. Figure taken from [13]

2.2.3 Sampling Algorithm

To sample the new particles in the simulation a Poisson disk pattern is used, with that, is guaranteed that each new particle is at a minimum distance from other and to create this pattern the Brinson's fast rejection-based approach is used [16].
These algorithm can be explained in a series of steps:
Select an initial sample from the domain and insert on an "active list" (array of samples)
While the list is not empty
    Choose a random sample, x, from the active list
    Generate up to k points inside a sphere with radius r and 2r around x
    For each new point, check if it is within a distance r of all sample
    If the new point is far enough from all samples
        The new point is created and is add to the active list
    If after k attempts no new point is created
        Remove x from the active list
In the SPH simulation, instead of using random fluid particles, every surface particles is inside the "active list"

The ghost particles algorithm above compared with a grid sampling gets better results because solve the anisotropic shell development problem [15], with some time the free surface gets with a shorter spacing between particles which can lead to instability in the simulation, as can be seen in Figure 5 .

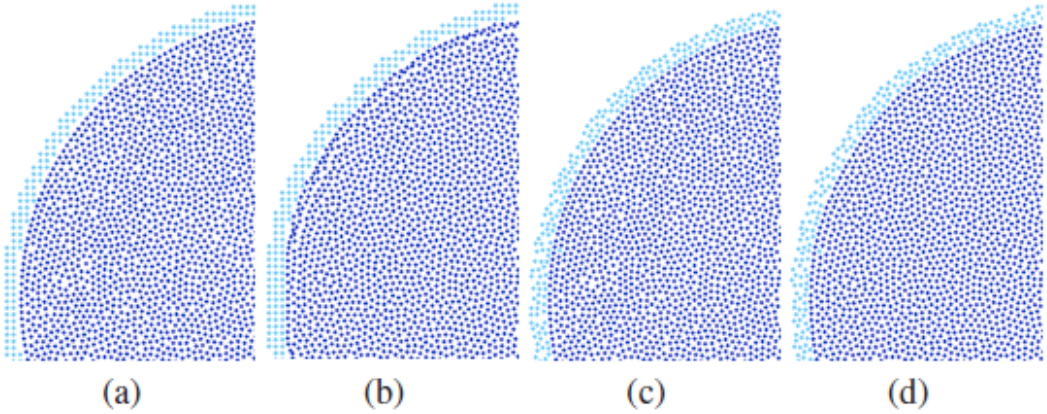


Figure 5 - Grid sampling the air vs Poisson dis. (a) Initial grid (b) 500 frames later, with an anisotropic shell developed. (c) Initial Poisson disk. (d) 500 frames later, essentially unchanged. Figure taken from [13]

2.2.4 Free Surface Tracking

To find the surface particles, the divergence of the positing (16) is applied.

$$\nabla x_i = \sum_j \frac{m_j}{\rho_i} \boldsymbol{r}_{ij} \cdot \nabla \mathrm{w}_{ij} \tag{16}$$

If a particle is far from the free surface, the divergence has a value bigger than a threshold and much less than the threshold due to the lower number of neighbouring particles on your domain. Usually the value used is 2, but in the simulations made, a value 1.5 is used [17]

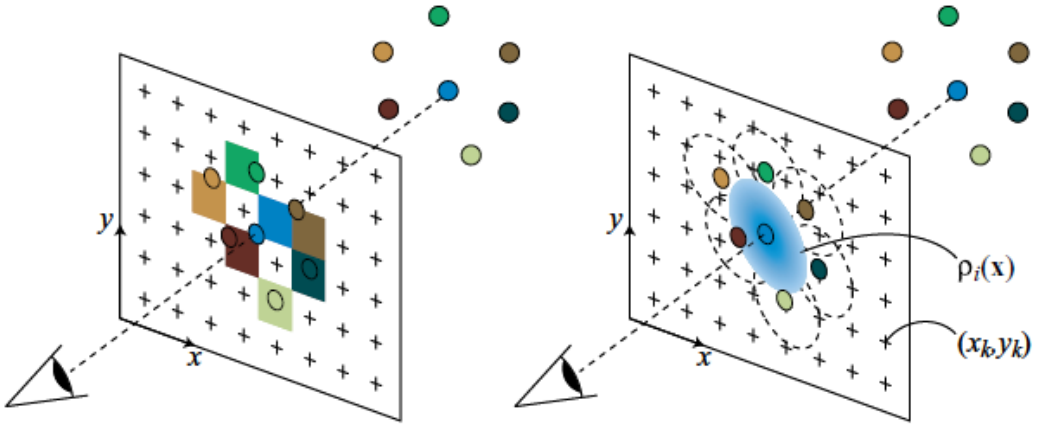
2.3 Rendering

Most of the techniques used to render a SPH result is based in surface reconstruction like Marching Cubes or Point Splatting [7]. Those techniques render the result reconstructing an iso-surface.
In the project the technique used to render the results from the simulation was Metaballs, also named as Blobs [14]. The technique was chosen since can simply render 2D points with a low cost, because different of other techniques, this does not generate a list of polygons, instead, it uses only the points of the simulation.

2.3.1 Surface Splatting

Splatting [21] is a simple and efficient algorithm for rendering point surfaces. The method uses z-buffer to resolve visibility, can be process the data set without any additional structures of acceleration but can be combined with some structures of acceleration to obtain a better result in the rendering.

In a naive approach of a projection point based method, each 3D point of the scene and give the color of the point to the pixel found in the projection, as can be seen in *Figure 6*.a).That approach leads to holes in the rendered image if the surface do not have a high level of density points.
To solve this problem, the spatting technique uses a footprint function that gives an intensity to the contribution of the point in the pixels, illustrated in *Figure 6*.b).



(a) (b)

Figure 6 – (a) Scheme of a naïve point based projection method; (b) Scheme of the splatting technique. Figure taken [21]

If each pixel of the image can be denoted by a function $\theta(x, y)$, to calculate the color of each pixel, for each channel (R.G.B) we use (17) to find the color.

$$\theta(x, y) = \sum_i c_i \rho_i(x, y) \tag{17}$$

where i represents each point of the scene, ρ is th footprint function and c are the color value of each point.

2.3.2 Marching Cubes

The marching cubes technique [22] uses a divide to conquer approach which the scene is processed from cells (voxels), equivalents to cubes. In each cell, the intersections between each of its 12 edges its isosurface
The vertices values are compared with a value associated to the isosurface, each vertex is classified as “in” or “out” of the isosurface and with the vertices a set of triangles is made to approximate the isosurface, as can be seen in *Figure 7*

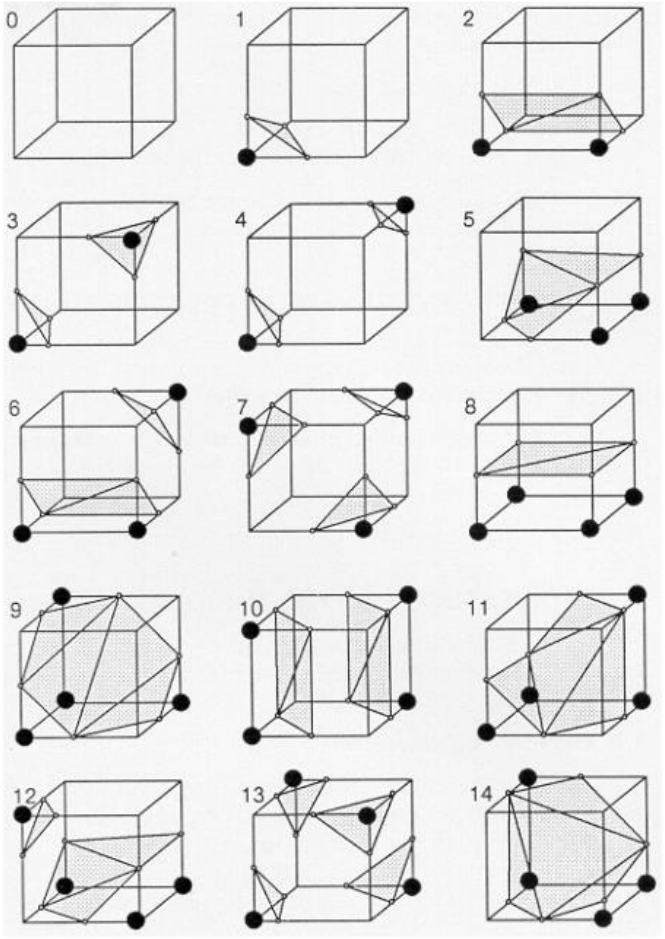


Figure 7 – Triangulated Cubes. Figure taken from [22]

The main advantage of this technique lies in the fact that the processing of a cell is independent of each other, allowing their parallelization, however, as a drawback, it can a cause holes in the isosurface due to topological ambiguities between cases
In general, many case processed by the marching cubes algorithm are empty, which will lead to unnecessary cost, to minimize this cost, spatial data structures are used to process only the active cells.

2.3.3 Metaball

Metaball is an iso-surface of a scalar fields, in that case, their surface has an influence in each point. If the influence is equal to a threshold values, a surface is found, otherwise it isn`t. With a system with more than one metaball their influence can be added to get the spheres influence together [14].
The strength of a blob can be positive or negative and it is relative to the core of the blob. If a blob have a positive strength, the blob creates a force of attraction in the other metaballs; if it is a negative value, the blob repels the other; if it has a 0 value, does not creates any force to the system.
The threshold determines the strength of each blob and it has some characteristics [14]:

1. Must have a positive value.
2. If is a large number, the surface gets closer to the center of the blob.

3. If is a small number, the surface gets close to the surface of the blob.

The influence of a metaball can be calculate by the density function [14], the influence can be calculated as (18)

$$\text{density} = \ strenght \ \cdot (1 - \left(\frac{distance}{radius}\right)^2)^2 \tag{18}$$

3 Test Case

To see the performance of the algorithm, three 2D dam break test case were made, all of them were based in the paper of Staroszczyk [18] which are illustrated in the Figure 8 with the parameters bellow:

- L = H = 0.6
- 3600 fluid particles
- Initial Spacing between the particles = 0.01m
- $\nabla t = 0.0005 \ s$
- Mass = 0.225g
- No initial velocity or acceleration
- Boundary were created with the same parameters but with half of the initial spacing

D

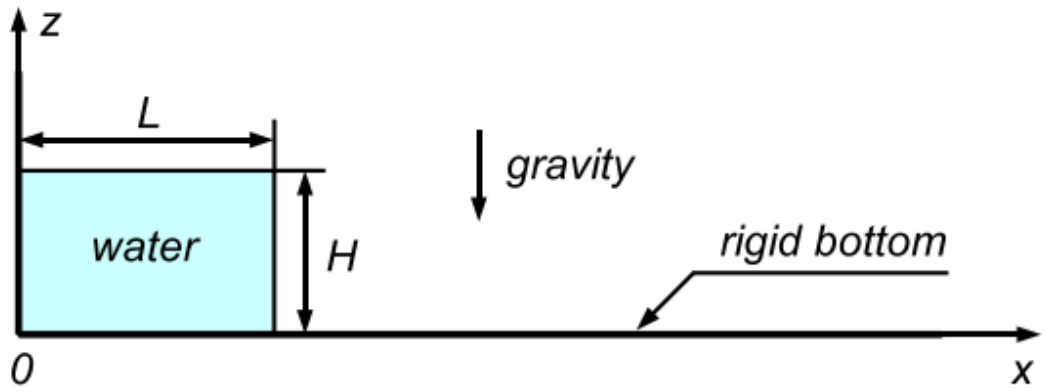


Figure 8 – Scheme of the Dam Break Test. Figure taken from [18]

This test case is a classic test, this comes from the fact that the test can be compared to the experimental and analytical results, as can be seen in [23], [24] and [18].

4 Results

Each test case was simulated were run on an Intel Core i7 with 8 GB of memory. To render the results, a PovRay [19] code were produced and send it to the software to render the simulation result as can be seen in Figure 9.c) and also a xml file were created to see the results in the software ParaView [20] that can be make a simple animation of the results just rendering the points like in Figure 9.a)
To implement the method was used C++, the code was modularized so new formulas can be easily added or in case of new test case, this part of the code can be seen below

```
densityCalculator22D(particles, h); //calculate the density

pressureCalculator2D( particles, rho0); //calculate the pressure

pressureForceCalculator2D( particles ); //add pressure and gravity forces

velocityCalculator2D( particles, dt ); //calculate the velocity

velocityXSPHCalculator2D( particles, dt, 0.05); //Calculate the viscosity

positionCalculator2D( particles, dt ); //update the position
```

If it's necessary to include some new component as turbulence or same boundary conditions this can be add just including a new method and passing the particles as a parameter. For instance, if a turbulence calculater were added, the code should look like these:

```
densityCalculator22D(particles, h); //calculate the density

pressureCalculator2D( particles, rho0); //calculate the pressure

pressureForceCalculator2D( particles ); //add pressure and gravity forces

turbulenceCalculator(particles, attributes);

velocityCalculator2D( particles, dt ); //calculate the velocity

velocityXSPHCalculator2D( particles, dt, 0.05); //Calculate the viscosity

positionCalculator2D( particles, dt ); //update the position
```

The particles attributes can be easily accessed from the created struct, as can be seen below:

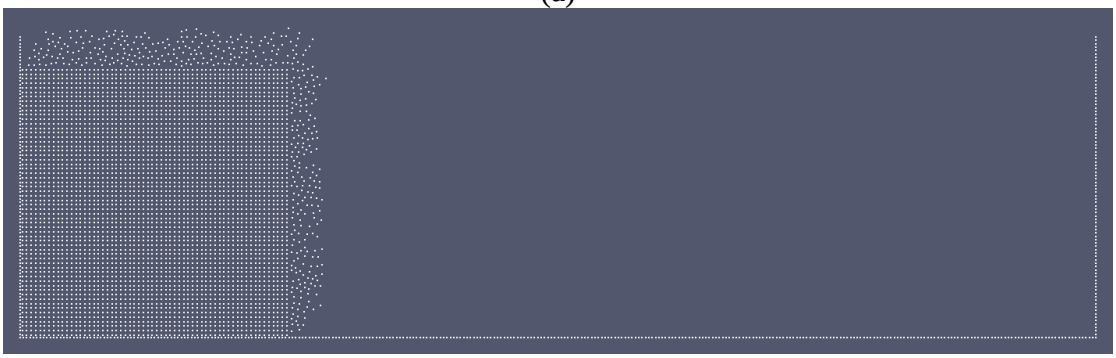
```
struct Particle2D{

	Vector2D <double> position;
	Vector2D <double> velocity;
	Vector2D <double> intVelocity;
	Vector2D <double> acceleration;

	double mass;
	double pressure;
	double initialPressure;
	double density;
```

```
        int type; // 0 for fluid, 1 for board, -1 for ghost
        int freeSurface; //1 for true, 0 for false
        vector<int> neighbors;

        vector< double > kernel;
        vector< Vector2D<double> > kernelGrad;


};
```

4.1 Test 1

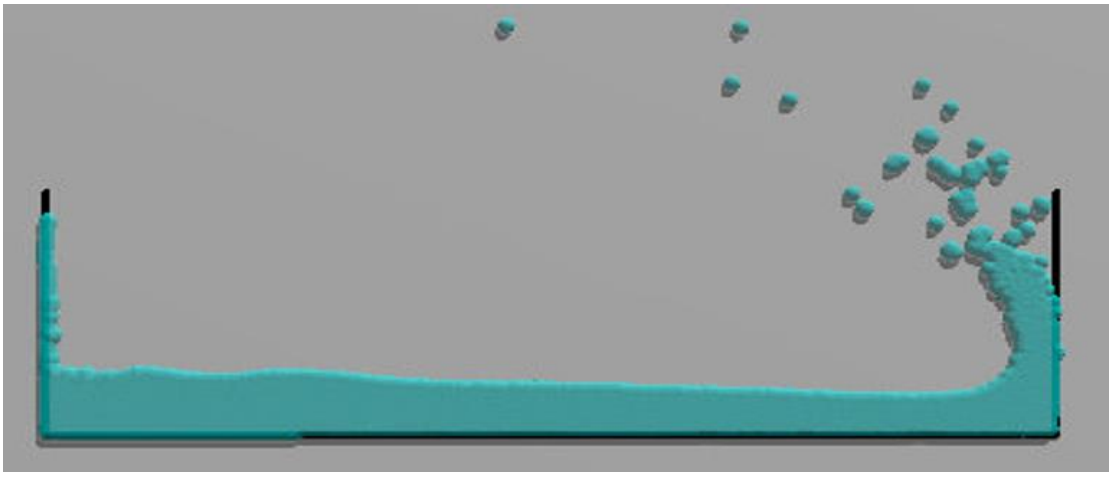
The first test is a common dam break, which, after a while, the particles from the bottom of the column of water have greater velocity than the particles from the top, creating a runoff of the fluid, which can be seen in the Figure 9

(a)



(b)

(c)



(d)

Figure 9 – Teste Case 1: (a) Initial configuration; (b) Initial configuration after ghost particles creation (c) Result after 500 time steps using blobs to render; (d) Result after 2300 time steps using blobs to render

Comparing the result from the Ghost SPH with the Basic SPH, can be seen that the result of the Ghost SPH have bigger stability. This fact can be seen in the Figure 10, the top left corner gets unstable using the Basic SPH, and some particles start to untie from the fluid, that problem was solved using the Ghost SPH.

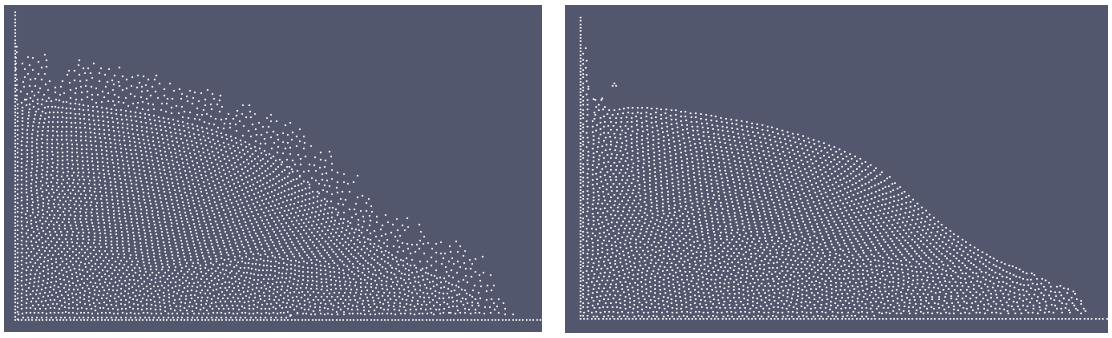
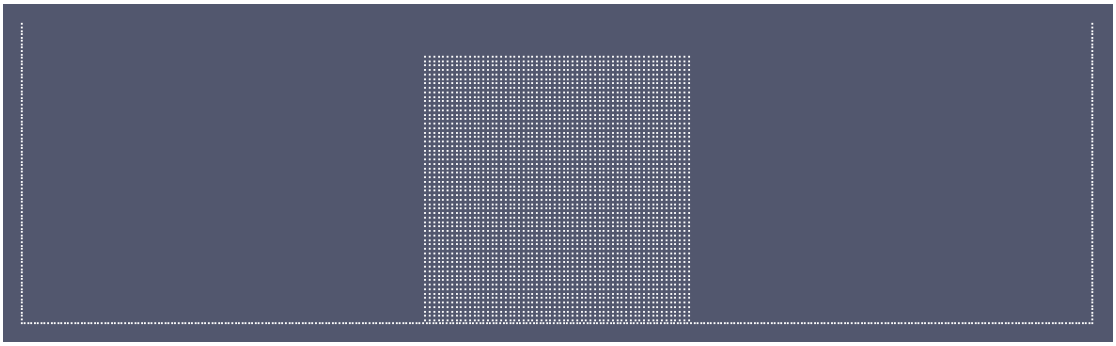


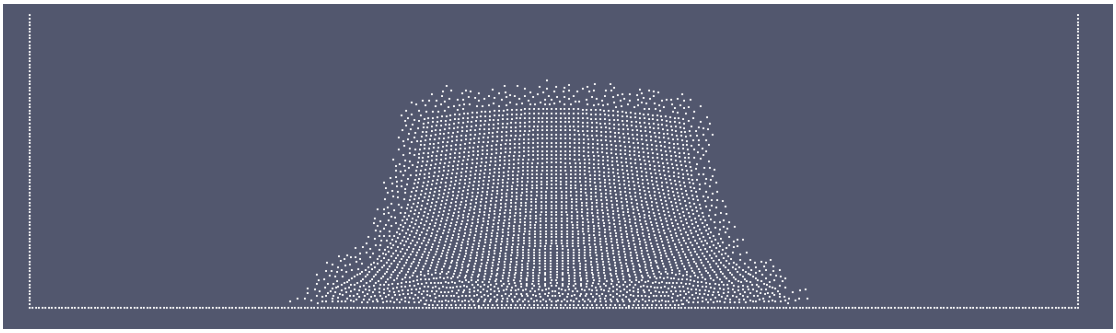
Figure 10 – Test case 1 result comparison: (a) Ghost SPH; (b) Basic SPH

4.2 Test 2

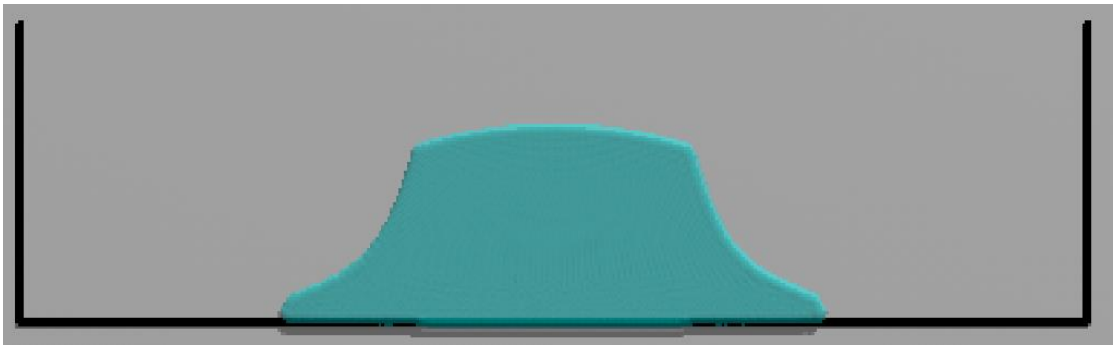
The second test is a dam break with the fluid begging in the middle of the recipient, is expected that each half of the fluid have the same flow, as illustrated in the Figure 11



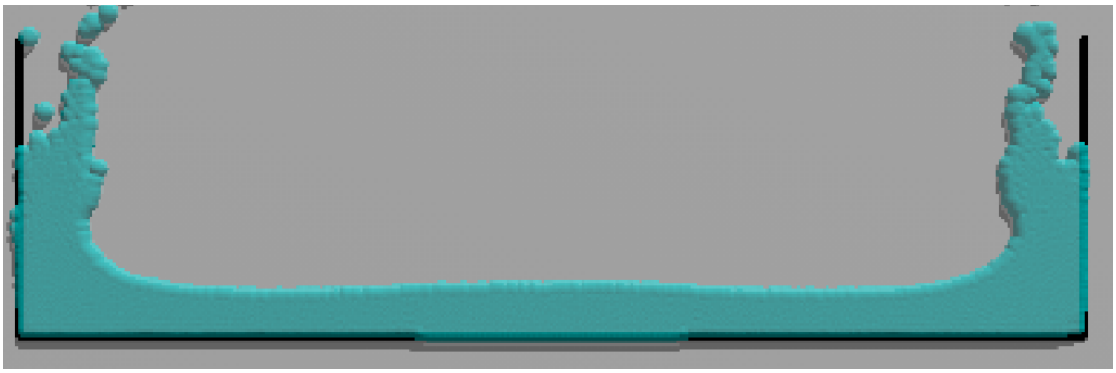
(a)



(b)



(c)

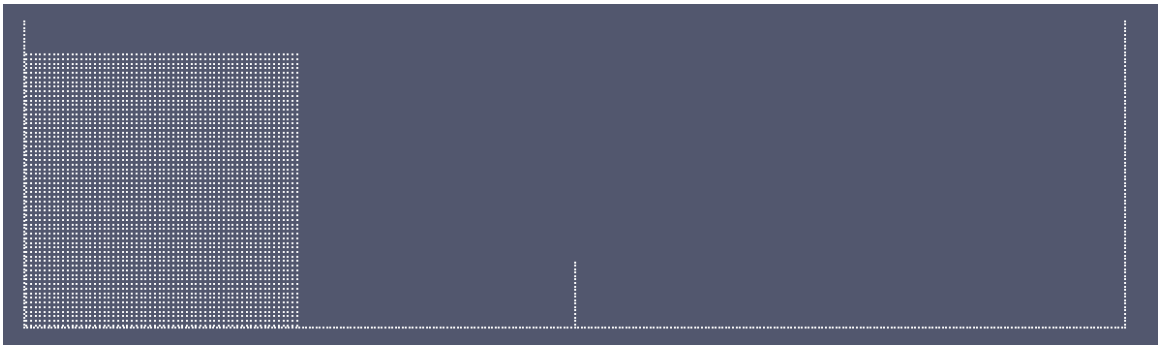


(d)

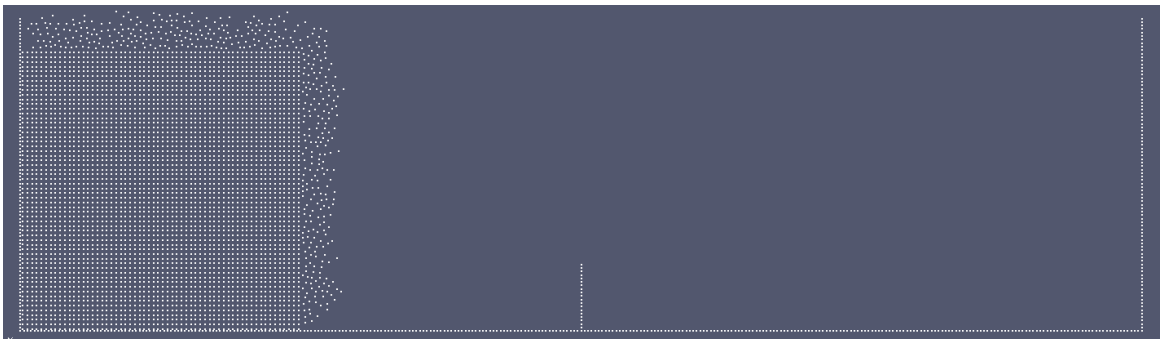
Figure 11 – Test Case 2: (a) initial configuration; (b) initial configuration after ghost creating; (c) simulation after 400 tie steps using metaballs to render (d) simulation after 2000 time steps using blobs to render

4.3 Test 3

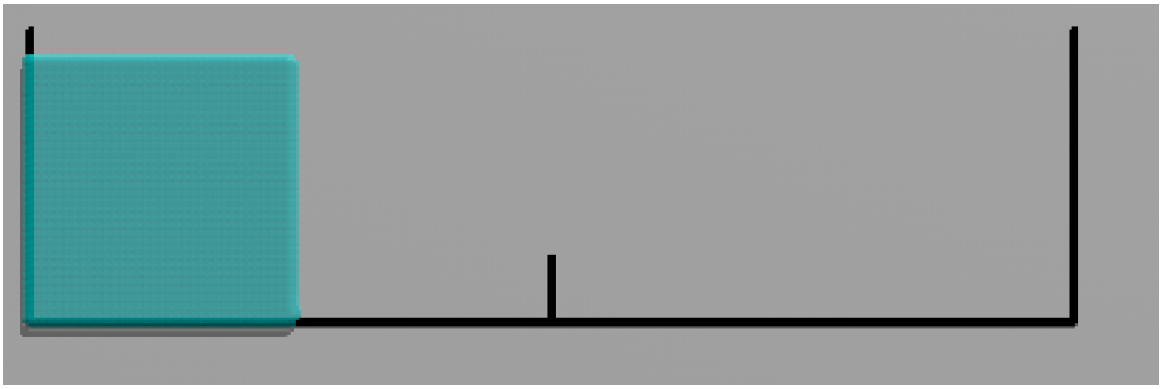
The last test case is similar to the Test 1, but in the recipient there is obstacle that will change the flow of the fluid, as demonstrated bellow (Figure 12)



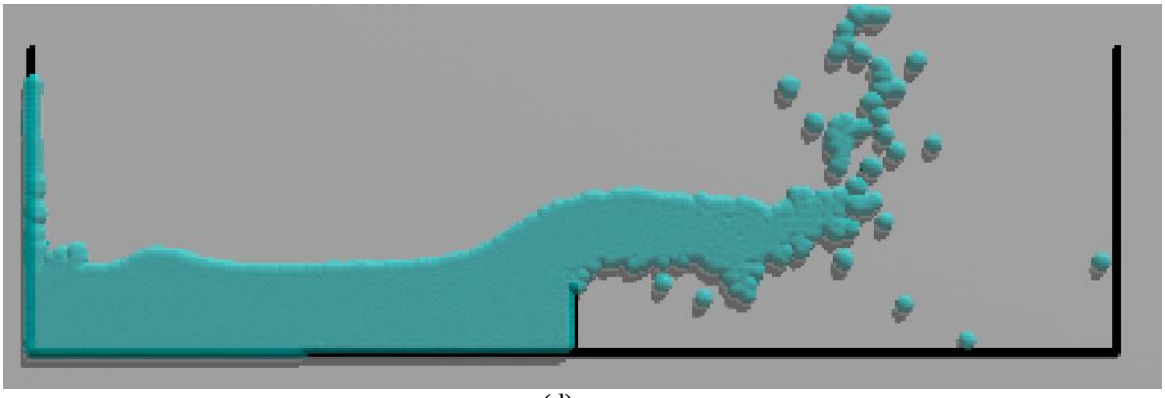
(a)



(b)



(c)



(d)

Figure 12 – Test Case 3: (a) Initial configuration; (b) initial configuration after ghost creation; (c) Initial configuration using blob to render; (d) simulation result after the shock with the obstacle

The results show a realistic look in the simulation but show some problems in the boundary condition since some particles are adhered to the wall though is just a small part of the sample which only create a small instability to the system.

5 Conclusion and future work

This project have the aim to implement a SPH based on the Ghost SPH [13], which generates ghost particles on the surface to solve the free surface problem and create a more realistic simulation.
The test case show that the simulation creates smooth simulation with a real look, solves the free surface problem because completes the action radius with particles and have to be done offline since of the big time consumption with the creation of the new particles at each time step.
In the future, is planned to improve the boundary condition, create simulations in 3D and to solve the problem with the time computation, GPU programing could be used, since the particles are geometric independent of each other, becoming easy to parallelize, e.g., each particle calculation is performed by one thread.

6 References

[1] Baraff D, Witkin A, Kass M. Untangling cloth. ACM Trans Graph. 2003;22(3):862-70.

[2] Iben H, Meyer M, Petrovic L, Soares O, Anderson J, Witkin A. Artistic simulation of curly hair. Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation; Anaheim, California. 2485913: ACM; 2013. p. 63-71.

[3] Irving G, Schroeder C, Fedkiw R, editors. Volume conserving finite element simulations of deformable models. Proceedings of the ACM SIGGRAPH Conference on Computer Graphics; 2007.

[4] Tessendorf J. Simulating ocean water. Simulating Nature: Realistic and Interactive Techniques SIGGRAPH. 2001;1.

[5] Froemling E, Goktekin T, Peachey D. Simulating whitewater rapids in *Ratatouille*. ACM SIGGRAPH 2007 sketches; San Diego, California. 1278862: ACM; 2007. p. 68.

[6] Monaghan JJ. Smoothed particle hydrodynamics. Reports on Progress in Physics. 2005;68(8):1703.

[7] M Muller, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation; San Diego, California. 846298: Eurographics Association; 2003. p. 154-9.

[8] Monaghan JJ. SPH compressible turbulence. Oxford, ROYAUME-UNI: Blackwell; 2002.

[9] Solenthaler B, Pajarola R. Density contrast SPH interfaces. Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation; Dublin, Ireland. 1632623: Eurographics Association; 2008. p. 211-8.

[10] Akinci N, Ihmsen M, Akinci G, Solenthaler B, Teschner M. Versatile rigid-fluid coupling for incompressible SPH. ACM Trans Graph. 2012;31(4):1-8.

[11] Alduan I, Otaduy MA. SPH granular flow with friction and cohesion. Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation; Vancouver, British Columbia, Canada. 2019410: ACM; 2011. p. 25-32.

[12] Krog OE, Elster AC. Fast GPU-Based fluid simulations using SPH. Proceedings of the 10th international conference on Applied Parallel and Scientific Computing - Volume 2; Reykjavík, Iceland. 2187012: Springer-Verlag; 2012. p. 98-109.

[13] Schechter H, Bridson R. Ghost SPH for animating water. ACM Trans Graph. 2012;31(4):1-8.

[14] PovRay (Org). Blob. < http://www.povray.org/documentation/view/3.6.0/275/>. Accessed in 10/08/2014

[15] Becker M, Teschner M. Weakly compressible SPH for free surface flows. Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation; San Diego, California. 1272719: Eurographics Association; 2007. p. 209-17.

[16] Bridson R. Fast Poisson disk sampling in arbitrary dimensions. ACM SIGGRAPH 2007 sketches; San Diego, California. 1278807: ACM; 2007. p. 22.

[17] Chen Z, Zong Z, Liu MB, Li HT. A comparative study of truly incompressible and weakly compressible SPH methods for free surface incompressible flows. International Journal for Numerical Methods in Fluids. 2013;73(9):813-29.

[18] Staroszczyk R. Simulation of dam-break flow by a corrected smoothed particle hydrodynamics method. Arch Hydro-Eng Environ Mech. 2010;57(1):61-79.

[19] PovRay (Org). PovRay. <http://www.povray.org>. Accessed in 10/08/2014
[20] KITWARE (Org). ParaView. < http://www.paraview.org>. Acessed in 20/08/2014.
[21] Gross M, Pfister H. Point-Based Graphics: Morgan Kaufmann Publishers Inc.; 2007.
[22] Lorensen WE, Cline HE. Marching cubes: A high resolution 3D surface construction algorithm. SIGGRAPH Comput Graph. 1987;21(4):163-9.
[23] Zheng, Xing, Duan, Wen-yang. Numerical simulation of dam breaking using smoothed particle hydrodynamics and viscosity behavior. Heidelberg, ALLEMAGNE: Springer; 2010.
[24] Liu MB, Xie WP, Liu GR. Modeling incompressible flows using a finite particle method. Applied Mathematical Modelling. 2005;29(12):1252-70.