HONEY SIMULATION

TNCG13, SFX - TRICKS OF THE TRADE

Karim Samim Karim, Rebecca Cedermalm

karsa755@student.liu.se, rebce973@student.liu.se

Monday 17th December, 2018
08:10

Abstract

Fluid simulations are important component for effects in film and games. In nature almost all fluids are viscous which makes viscoelastic fluids especially interesting to simulate. The project this report is based uses the Position Based Fluid method [3] to simulate how honey moves and acts when falling into a box. The Position Based Fluid simulation method is an integration of the Smoothed Particle Hydrodynamics method [1] into the Particle Based Dynamics framework [2]. Materials, lights and environments were added to the simulation to make the it look visually pleasing. The result is a stable and realistically simulates the motion of honey, see figure 1.

1 Introduction

Fluid simulations are one of the most popular simulations in the computer graphics field. They are heavily used components in many films and games in the form of water, fire or smoke. Almost all fluids are at least a little viscous in nature even though they might not be simulated as such. Water is a good example of that since it usually is simulated without viscosity but exhibits some characteristics of it when experiments have been made in real life. This report explains the work and results from a project made by two students at Linköping University. The goal of the project was to simulate a viscous fluid such as honey.

The simulation was made using the method Position Based Fluids (PBF) presented by Müller at al. in 2013 [3]. PBF is a density solver that is the final result after a merge between the methods Position Based Dynamics (PBD), presented by Müller et al in 2007 [2], and Smoothed Particle Hydrodynamics (SPH), presented by Monaghan in 1992 [1]. It uses

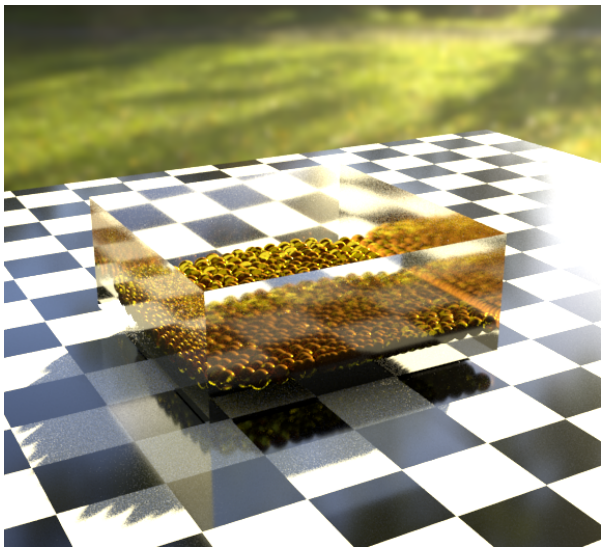

Figure 1: A simulation of honey using the Position Based Fluid method and 512 particles.

positional constraints that enforces constant density and gets the inherited stability from the PBD framework and the convergence and incompressibility from the SPH method. Because of the integration with the PBD framework, which enables larger time steps while still being stable, the PBF method is possible to use for real-time purposes.

Originally, constraints and improvements made for viscoelastic fluids presented by Barreiro et al. [4] was planned to be integrated into the PBF simulation. This would mean that constraints derived from a constitutive model of polymer conformations would be added together with some adjustments to the solver such as adding the possibility to also use velocity-based constraints. Unfortunately, there was no time left to do that. Instead, parameters of the PBF were tuned to make the simulation look like viscous honey.

2 Method

In order to simulate honey, Autodesk Maya was used. It is a tool used for creating interactive 3D applications. Scripting in Autodesk Maya was also done using Python, in order to simulate honey using PBF.

2.1 Position Based Fluids

Following the paper from Macklin and Müller [3] on PBF, constant density needs to be enforced in order to enforce incompressibility. This is done by enforcing constraints on each particle, see the equation below.

$$C_i(P_0, P_1, ..., P_n) = \frac{\rho_i}{\rho_0} - 1 \tag{1}$$

Where $P_0, P_1, ..., P_n$ is the position of a particle and its neighbours, ρ_0 is the rest density and ρ_i is a density estimator from SPH [1], see the equation below.

$$\rho_i = \sum_j m_j W(P_i - P_j, h) \tag{2}$$

Where m_j is the mass of each particle (set to 1 for all particles in this case), h is the radius and W is a smoothing kernel. For gradient calculation the Spiky kernel is used and the Poly6 kernel is used for density estimation [3]. The sum loops through the neighbour particles P_j of particle P_i.

In order to gain stability, PBD needs to be added to the density estimator. PBD tries to satisfy the constraint $C(P + \Delta P) = 0$ where ΔP is a particle position correction. This constraint can be found by the equation below [2].

$$C(P + \Delta P) \approx C(P) + \nabla C^T \nabla C \lambda = 0 \tag{3}$$

Where ∇C is the direction which ΔP is restricted to and λ is a scaling factor [2]. Afterwards, the gradient of equation 1 is calculated for a particle k.

$$\nabla_{P_k} C_i = \frac{1}{\rho_0} \sum_j \nabla_{P_k} W(P_i - P_j, h) \tag{4}$$

Where ∇_{P_k} is the particle gradient. There are also two different cases for this equation, whether k is a neighbouring particle or not.

$$\nabla_{P_k} C_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{P_k} W(P_i - P_j, h), if k = i \\ -\nabla_{P_k} W(P_i - P_j, h), if k = j \end{cases} \tag{5}$$

Afterwards, equation 5 is inserted in 3 and solving for the scalar λ for each particle gives the equation:

$$\lambda_i = -\frac{C_i(P_0, P_1, ..., P_n)}{\sum_k |\nabla_{P_k} C_i|^2} \tag{6}$$

In equation 6, the denominator causes instability because the constraint function is non linear where the gradient vanishes at the smoothing kernel boundary when particles separate. By constraint force mixing which softens the constraint, the problem can be solved [3]. Equation 3 will therefore change and the new equation will be:

$$C(P + \Delta P) \approx C(P) + \nabla C^T \nabla C \lambda + \epsilon \lambda = 0 \tag{7}$$

Where ϵ is the relaxation parameter. The scaling factor can now be updated, see the equation below.

$$\lambda_i = -\frac{C_i(P_0, P_1, ..., P_n)}{\sum_k |\nabla_{P_k} C_i|^2 + \epsilon} \quad (8)$$

After calculating the scaling factor, the position update can be calculated, see the equation below.

$$\Delta P_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(P_i - P_j, h) \quad (9)$$

In SPH algorithms there is a common problem with particle clumping. This is caused by a negative pressure when a particle has few neighbours and is unable to satisfy the rest density. In order to solve this problem, an artificial pressure term is added, see the equation below [3].

$$S = -k \left(\frac{W(P_i - P_j, h}{W(\Delta Q, h)} \right)^n \quad (10)$$

Where k and n are positive constants and ΔQ is a point. This term is than inserted into the position update equation:

$$\Delta P_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j + S) \nabla W(P_i - P_j, h) \quad (11)$$

Afterwards, vorticity confinement is used in order to replace lost energy [3]. This is done by first calculating the vorticity at a particles position.

$$\omega_i = \nabla \times V = \sum_j (V_j - V_i) \nabla_{P_j} W(P_i - P_j, h) \quad (12)$$

Where V_i is the velocity of a particle and V_j is the velocity of the particles neighbours. After calculating the vorticity, the corrective force can be calculated, see the equation below.

$$F_i^\omega = \delta(N \times \omega_i) \quad (13)$$

Where $N = \frac{n}{|n|}$, n is $n = \nabla |\omega|_i$ and δ is a constant which is used to decide the amount of small details which is added to the flow field [3].

Finally, XSPH viscosity is added which is needed for coherent motion. [3].

$$V_{i+1} = V_i + c \sum_j (V_j - V_i) \cdot W(P_i - P_j, h) \quad (14)$$

Where c is a parameter which was typically chosen to be 0.01 in the paper by Macklin and Müller [3].

2.2 PBF algorithm

The algorithm below shows how PBF was implemented in the honey simulation.

forall *particles i* **do**
  $v_i = v_i + \Delta t F_{ext}$, apply external force
  $x_i^n = x_i + v_i \Delta t$, get the new position
end
forall *particles i* **do**
  $imposeBoundaryConstraint(i)$, impose collision detection and response
end
forall *particles i* **do**
  $findNeighbour(i)$, find particle i neighbour
end
$iterations = 0$
while $iterations < maxIter$ **do**
  forall *particles i* **do**
    $calculate\lambda(i)$
  end
  forall *particles i* **do**
    $calculate\Delta P(i)$
  end
  forall *particles i* **do**
    $x_i^n = x_i^n + \Delta P_i$
  end
  iterations = iterations + 1
end
forall *particles i* **do**
  $imposeBoundaryConstraint(i)$
end
forall *particles i* **do**
  $v_i = \frac{1}{\Delta t}(x_i^n - x_i)$
  apply vorticity and XSPH viscosity
  $x_i = x_i^n$
end

Algorithm 1: PBF simulation loop

Algorithm 1 is done once for each step in the

simulation loop, where the velocity and position are saved for the next step.

2.3 Simulation in Autodesk Maya

First of all, a scene was created with a box without a roof and 512 particles which are placed above it. The box has a glass shader and the particles has a honey shader. Gravity is added to the particles as a external force and the PBF algorithm. There is also collision detection and response between the particles and the box. The scene is rendered using 150 keyframes in order to see how the particles behave when it falls on the box and after it has landed on the box.

3 Result

In order to make the fluid behave like honey, some variables had to be chosen. Their values are presented in a table below.

Variable	Value
ρ_0	1000
Δt	0.016
h	0.4
ϵ	200
k	0.001
n	4
ΔQ	h*0.3
δ	0.25
c	0.001
$maxIter$	10

Table 1: Table showing the values of different simulation variables.

The result of the PBF simulation in Autodesk Maya can be seen in the figures. They are taken during different keyframes in order to show how the particles propagate.

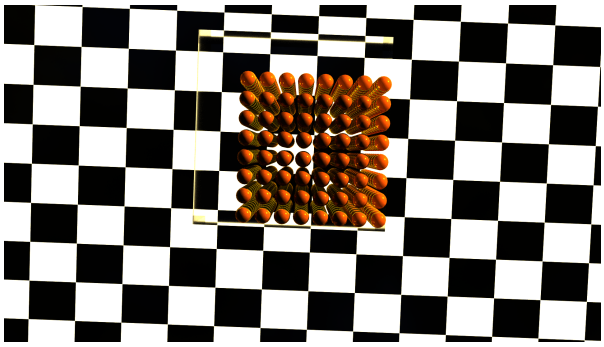

Figure 2: PBF simulation of honey at keyframe 1.

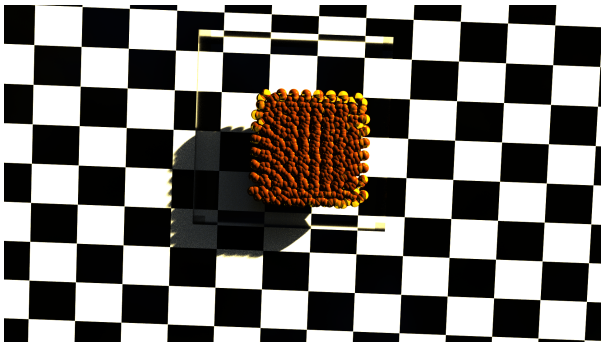

Figure 3: PBF simulation of honey at keyframe 13.

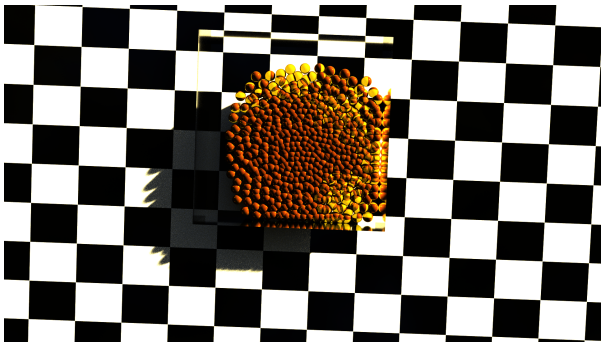

Figure 4: PBF simulation of honey at keyframe 29.

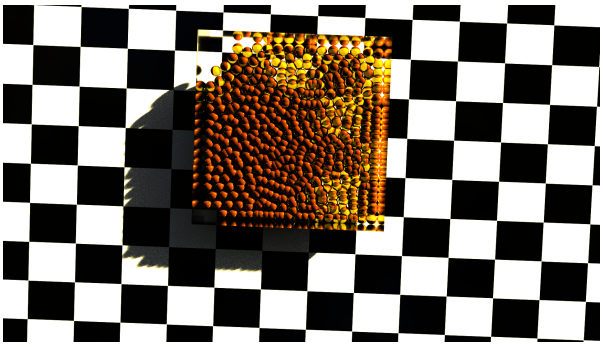

Figure 5: PBF simulation of honey at keyframe 110.

4 Conclusion and Future Work

The result from the honey simulation looks realistic if honey would be divided into smaller spherical particles in nature. The particles move slowly over the surfaces just as honey would do in the real world.

The algorithm proposed in the PBF method was easy to follow since it was well documented and cited by others. Parameters were however hard to set right, since they did not say any particular ranges for what the parameter value could be. This lead to a very time consuming work of getting the parameters right.

The paper by Barreiro et al. [4], where improvements and constraints for viscoelastic fluids were proposed, was harder to understand. Some variables were not explained and since the paper brought up several different kinds of materials it was somewhat hard to find what was needed to simulate honey. The paper did not have any citations at the time as well, which made it impossible to see if anyone else has understood it and explained it better. When the paper was understood there was too little time left to be able implement the parts needed.

Future work includes tuning the parameters better, implementing the constraints and improvements proposed by Barreriro et al. [4] and turning the particles into a mesh to make the result look smoother and more realistic.

References

[1] Monaghan, J. J. 1992. Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics 30, 1, 543574.

[2] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. 2007. Position based dynamics. J. Vis. Comun. Image Represent. 18, 2 (Apr.), 109118.

[3] Macklin, M., Müller, M. 2013. Position Based Fluids. ACM Trans. Graph. 32, 4, Article 104 (July 2013), 5 pages.

[4] Héctor Barreiro, Ignacio Garca-Fernández, Iván Alduán, and Miguel A. Otaduy. 2017. Conformation Constraints for Efficient Viscoelastic Fluid Simulation. ACM Trans. Graph. 36, 6, Article 221 (November 2017), 11 pages.