Vision, Modeling, and Visualization (2015)
D. Bommes, T. Ritschel, and T. Schultz (Eds.)

Simulation of Water Condensation based on a Thermodynamic Approach

Sebastian-T. Tillmann and Christian-A. Bohn

Wedel University of Applied Sciences, Wedel, FR Germany

Abstract

We introduce a novel approach for physically based simulation of water drops on surfaces considering the thermodynamical laws like mixing temperature, specific heat capacity, water vapor, saturation, and additional material properties in an adiabatic environment. The algorithm is able to robustly handle huge scenes of complex closed and also non-closed objects defined as implicit surfaces and thus it is ideally suited for extending classical, well-known fluid simulation models. A subset of thermodynamic rules based on a static grid is used. Other approaches use buoyant force and other equations based on motion, e. g. [HBSL03].

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction and Related Work

Fluid simulation for computer graphics [Li14, EMF02, FM96] has widely been used in films and even real-time scenarios, and has been an important research topic for several decades. While many of the related approaches focus on the simulation of fluid surfaces, this work is concerned with the *generation* of fluid, i.e. water drops, as result of a condensation process. To enable a physically based simulation our approach is based on the areas of fluid mechanics, thermodynamic laws. The simulation model mainly bases on implicit surfaces [BW97] using a grid based method [Bra10], and is capable to be transposed to a representation by explicit surface meshes [BB06].

Another important aspect of our approach is usability and integration into an existing 3D modeling software like Blender. The simulation should be able to work also on explicit mesh object data.

Although research of simulating fluids has a long history, to the authors' knowledge the proposed physical model has not been implemented so far. Similar fields are investigated by the simulation of cloud dynamics [HBSL03] and water surfaces [WMT05].

2. Physical Background

Thermodynamics is an important factor in water condensation simulations based on the consideration of heat and temperature in relation to energy. It defines macroscopic parameters such as internal energy, entropy, and pressure. The phenomena of water condensation basically occurs when the amount of water in the air is higher than the air is able to receive. The amount of water in the air to reach the saturation point depends on temperature. If the saturation point is reached the water vapor starts to condensate and the generation of water drops begins. Unlike methods which use the Navier-Stokes equations [Sta99] our approach combines an Euler grid with the thermodynamic laws.

2.1. Navier-Stokes Equations

Navier-Stokes is a system of nonlinear partial differential equations and may be used in addition to our approach to simulate fluid mechanics. We assume a constant pressure leading to the first incompressible Navier-Stokes equation (Eq. 1), which calculates the pressure variation.

$$\frac{\partial \cdot \rho}{\partial \cdot t} = -(\vec{u} \cdot \nabla) \cdot \rho + \kappa \cdot \Delta \cdot \rho + S \qquad (1)$$

The partial derivation $\frac{\partial \cdot \rho}{\partial \cdot t}$ defines the pressure variation over time. $(\vec{u} \cdot \nabla) \cdot \rho$ is the pressure shift with the flow. ∇ is termed

DOI: 10.2312/vmv.20151267

the gradient, the spatial first partial derivation $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}$ and calculates the gradient from the scalar field of pressures. The diffusion of the pressure is calculated through $\kappa \cdot \Delta \cdot \rho$ and S are external pressures. Δ represents the Laplace operator and is a shortcut for ∇^2. The second equation (Eq. 2) describes the change of velocity over the time.

$$\frac{\partial \cdot \vec{u}}{\partial \cdot t} = -(\vec{u} \cdot \nabla) \cdot \vec{u} + \nu \cdot \Delta \cdot \vec{u} + \vec{f} \tag{2}$$

The left part is equivalent to the first equation, with the difference that we now deviate the velocity instead of the pressure. $(\vec{u} \cdot \nabla) \cdot \vec{u}$ is the velocity shift by flow. $\nu \cdot \Delta \cdot \vec{u}$ describes viscosity. The kinematic viscosity of the fluid is ν and $\vec{f}$ defines external forces influencing the fluid. The proposed simulation uses a finite difference method as a convenient numerical way to solve partial differential equations.

2.2. Heat Transfer

In the branch of thermodynamics heat transfer describes the exchange of thermal energy between physical systems depending on the temperature and pressure by dissipation. The fundamental modes of heat transfer are conduction or diffusion, convection, and radiation. The transported energy is called heat or thermal energy. Heat transfer always follows the negative temperature gradient.

We differentiate between three types of heat transport processes [LR11]. *Advection* is the transport mechanism of a fluid substance depending on motion and momentum. *Conduction* is the transfer of energy between objects that are in physical contact. Thermal conductivity is the property of a material to conduct heat and is evaluated primarily in terms of Fourier's Law for heat conduction. *Convection* describes the transfer of energy between an object and its environment caused by fluid motion. The average temperature serves as reference for evaluating properties related to convective heat transfer.

In real systems more than one type of transfer act together. Within solid bodies there exist thermal conductivity and also heat radiation. In fluids an additional convective flow of heat is added. Heat radiation takes place between surfaces while gases are nearly not concerned with it. It should be noted that the thermal transfer also happens in a thermal equilibrium, but of course without changing the temperature.

2.2.1. Heat Transfer Coefficient

The heat transfer coefficient α is described as the heat flow *(thermal output)* [Hel79]. It passes on a surface $A = 1m^2$ with the temperature difference of $\Delta t = 1K$ on a liquid or gas *(fluid)* and vice versa. The heat transfer coefficient is able to adopt many different values summarizing all influences of the properties and movement states belonging to temperature, pressure, velocity, thermal conductivity, density, specific heat capacity, and the viscosity of the fluid, as well as the shape and surface of the body.

2.3. Richmann's Rule of Mixture

An important physical law for this approach is the rule of mixture for calculating the mixing temperature [Hel79] when pooling multiple bodies with different temperatures (Eq. 3). Under this condition the aggregate state is not changing and the system is secluded out of the bodies

$$m_1 \cdot c_1 \cdot (T_1 - T_m) = m_2 \cdot c_2 \cdot (T_m - T_2) \tag{3}$$

leading to the mixing temperature

$$T_m = \frac{m_1 \cdot c_1 \cdot T_1 + m_2 \cdot c_2 \cdot T_2}{m_1 \cdot c_1 + m_2 \cdot c_2}. \tag{4}$$

m_1 and m_2 are the masses, c_1 and c_2 the specific heat capacities of the involved bodies. Assuming temperature T_1 being greater than T_2, the first body dispenses heat to the second. T_m stands for joint temperature of both bodies after mixture.

2.4. Specific Heat Capacity

We need a certain amount of energy *(specific heat)* (Eq. 5) in order to warm up $1kg$ of a material at $1K$

$$c = \frac{Q}{m \Delta T}. \tag{5}$$

Q is the thermal energy attached to or detracted from the material — the heat quantity. m is the mass of the material. The deviation from the starting to the final temperature is $\Delta T = T_2 - T_1$. The SI unit of the specific heat capacity (Eq. 6) is

$$[c] = \frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{K}}. \tag{6}$$

See Tab. 1 for some example values of certain media. Note that getting a value for air needs to take the relative humidity in relation to the temperature to be taken into account.

material	temperature	rel. humidity	spec. heat cap.
air	20	45	1.0054
air	20	100	1.0300
glass	–	–	0.7000
water	0	–	4.2280
water	10	–	4.1880
water	20	–	4.1860
water	30	–	4.1830
water	40	–	4.1820

Table 1: *The Table shows an extract from the chemistry library [Wag04] with specific heat capacities in $kJ/(kg \cdot K)$ for different materials. The temperatures are given in $^\circ C$ and the relative air humidity in percentage.*

The values of Tab. 1 are retrieved under certain conditions, like, i.e., a constant atmospheric pressure of $1013.25hPA$.

2.5. Humidity depending on Temperature

If we examine air as an ideal gas, humidity ρ in kg/m^3 is given by

$$\rho = \frac{p \cdot M}{R \cdot T}, \tag{7}$$

with air pressure p, molar mass M, universal gas constant R, and the temperature T in Kelvin. Inserting $R_s = 287.058 \frac{J}{kg \cdot K}$ for the dry air [Hel79] delivers

$$\rho = \frac{p}{R_S \cdot T}. \tag{8}$$

For example, assuming a temperature of 25 °C and humidity of $1.184 kg/m^3$ delivers $\rho_{t25} = 1.184 kg/m^3$. See Tab. 2 [Hel79] for specific values for humidity.

temperature	air humidity
0	1.2920
10	1.2466
20	1.2041
30	1.1644

Table 2: *The Table shows examples for air humidity [Hel79] in relation to temperature in °C on sea level conditions.*

2.6. Structure of Humid Air

Air humidity comes from water vapor of the gas mixture in the earth atmosphere or in rooms [LR11]. As a function of temperature, a given volume of air is able to contain a certain amount of water vapor. At the maximum amount of water vapor, air is saturated. The common dimension of air humidity is the relative humidity given in percentage of saturation. Relative humidity depends on the current temperature and the current pressure. The amount x of water in the air is

$$x = \frac{water}{air} = \frac{water\ vapor + liquid\ water + ice}{air}, \tag{9}$$

whereas "water" means vapor, liquid, and ice in general. To find the amount of vapor, liquid, or desublimated vapor (ice), virtually the degree of saturation of the air must be determined. The pressure of saturation linearly relates to the water vapor pressure, termed the relative humidity

$$\varphi = \frac{partial\ pressure\ of\ water\ vapor}{partial\ pressure\ of\ saturation}. \tag{10}$$

The partial pressure of saturation is the maximum partial pressure that the water can adopt by a given temperature. A higher partial pressure results in the condensation as liquid or desublimation as ice, depending on the temperature and the atmospheric pressure. With ambient pressure as an environment property we get liquid water at temperatures above zero °C and below that ice.

Relative humidity range is $0 \leq \varphi \leq 1$, like

$$\varphi = \begin{cases} 0 & \text{dry air} \\ 0 < \varphi < 1 & \text{unsaturated humid air} \\ 1 & \text{saturated humid air.} \end{cases} \tag{11}$$

If the absolute air humidity exceeds the maximum possible value it is called supersaturation and results in a "spontaneous condensation" of water vapor to water drops without a required condensation nucleus. This approach provides the option of an adiabatic system, that occurs without transfer of heat or matter between a system and its surroundings, which is capable to render supersaturation which mainly appears in the earth's atmosphere due to neighboring variations of air parameters.

This approach is capable of simulating an adiabatic system where heat is kept inside the simulation domain, as well as an system where heat may radiate outside the simulation representation which we term an 'open system'.

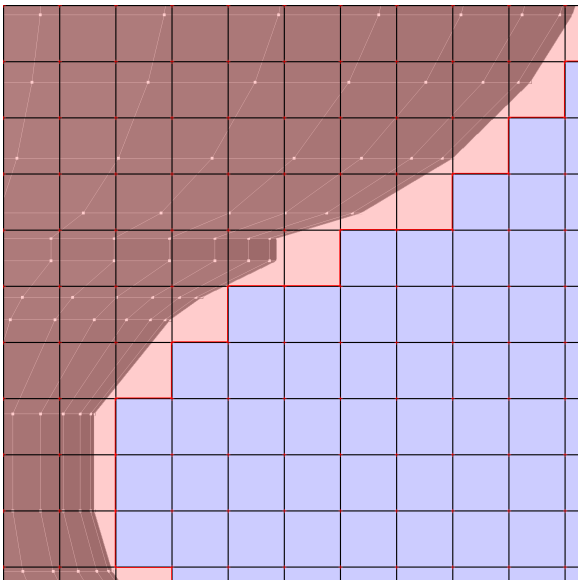

Figure 1: *Red filled voxels mark detected geometry of the mesh within the grid.*

3. Workflow

The general simulation process exchanges water vapor and temperature between cells in a regular grid until saturation is reached. The level of saturation is important for the generation of water drops. Upon generation the water droplets are able to be rendered.

Computations are based on a regular voxel grid of voxels which have aggregate states (liquid or solid), material states (air, water or glass), and concerning parameters like

temperature or absolute humidity. Initial values at the beginning of the simulation generate an adiabatic system at 20 °C room temperature at the center and a relative humidity between 20% and 60%. In each cell exactly one nucleation at a random position is set. Figure 2 exposes work steps before, during, and after a simulation process. The *Blender*-related steps are optional to our actual simulation.

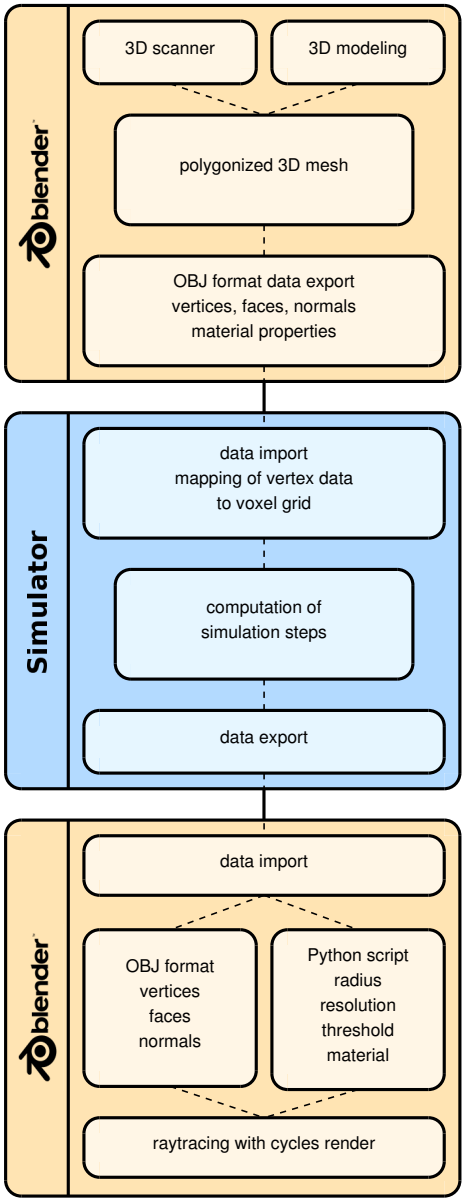

Figure 2: *Workflow with individual sub-steps.*

The simulator works also without importing data before the simulation starts, however itself can create only rudimentary objects.

4. Algorithm Overview

The complete iterative refinement algorithm of our thermodynamic approach is executed for each calculation step within the voxel grid (Alg. 1). Be aware that all physical processes described before are based on the *International System of Units (SI)* and involve time in reality for each calculation step. Every call to our mixturing temperature function calculates one second in reality between two voxels. We exchange material properties of bordering cells in a non-

Algorithm 1 function UPDATESIMULATION

```
1: if opensystem then
2:     SETCELLS(positions, properties)
3: for all cells do
4:     UPDATETEMPERATURE(cell)
5:     UPDATESTEAM(cell)
```

adiabatic environment (see Section 4.2). After that the temperature exchanges with the surrounding cells and the water vapor transfer happens and executes.

4.1. Temperature Balance

Richmann's Rule of Mixture (see Section 2.3) is applied between neighboring grid cells, like exposed in Alg. 2. Temperatures are propagated from the cell under consider-

Algorithm 2 function UPDATETEMPERATURE

```
1: for all neighbors do
2:     if GETTEMP(cell) ≥ GETTEMP(neighbor) then
3:         mixTemp ← GETMIXTEMP(cell, neighbor)
4:     else
5:         mixTemp ← GETMIXTEMP(neighbor, cell)
6:     SETTEMP(cell, mixTemp)
7:     SETTEMP(neighbor, mixTemp)
```

ation to its neighbors and vice versa depending on the temperature deviation. The procedure GETMIXTEMP calculates the required properties of a cell, like the specific weight (see Section 2.5) and the specific heat capacity (see Section 2.4). These values are stored for reusing them during one cycle through the whole grid. The calculation of the specific heat capacities are accomplished in the same manner. Since we use a static grid to traverse all cells, ordering the cells affect the solution in no way.

4.2. Environmental Heat Exchange

This step is a boundary condition and will only be executed when a non-adiabatic environment is assumed where bordering cells exchange heat with its surrounding environment (see Figure 3). At any time the system can be biased from outside (red cells) by, i.e, by introducing heat and humidity.

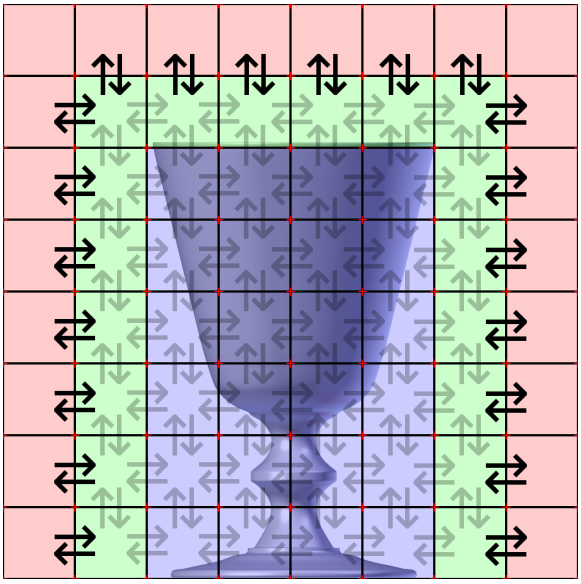

Figure 3: *An open system exchanges heat with the environment which is instantiated by a frame (red cells) at 28 °C.*

4.3. Steam Delivery

Also vapor is propagated to neighboring cells when saturation is reached. Algorithm 3 shows the implementation of the topics described in Section 2.6. Propagation of vapor to

Algorithm 3 function UPDATESTEAM *(part 1)*

```
1:  for all neighbors do
2:      if GETKIND(neighbor) = water then
3:          CONTINUE                          ▷ Ignore water cells
4:      if GETKIND(cell) = air then
5:          if GETKIND(neighbor) = air then
6:              if GETRELHUM(cell) ≤ 100 then
7:                  if GETRELHUM(neighbor) ≥ 100 then
8:                      nh ← GETABSHUM(neighbor)
9:                      ch ← GETABSHUM(cell)
10:                     ns ← GETSAT(neighbor)
11:                     cs ← GETSAT(cell)
12:                     ml ← ABS((nh − ns) − (cs − ch))
13:                     SETABSHUM(cell, ch + ml)
14:                     SETABSHUM(neighbor, nh − ml)
```

neighboring cells increases humidity there until saturation is reached. If vapor exceeds saturation or the temperature in a cell decreases (relative air humanity rises) condensation at a nucleus particle is initiated (see Section 4.4).

4.4. Condensation

Algorithm 4 shows relevant physical calculations in detail, i.e., the association of air humidity and water vapor in exchange with the amount of saturation between air and glass cells which were described in Section 2.6. Water drops are

Algorithm 4 function UPDATESTEAM *(part 2)*

```
1:  for all neighbors do
2:      if GETKIND(neighbor) = water then
3:          CONTINUE                          ▷ Ignore water cells
4:      if GETKIND(cell) = air then
5:          if GETKIND(neighbor) = glass then
6:              if GETRELHUM(cell) ≥ 100 then
7:                  nh ← GETABSHUM(neighbor)
8:                  ch ← GETABSHUM(cell)
9:                  cs ← GETSAT(cell)
10:                 nr ← GETDROPRAD(neighbor)
11:                 ml ← GETDROPVOL(cell, ch − cs)
12:                 SETABSHUM(neighbor, nh + ml)
13:                 SETABSHUM(cell, ch − ml)
14:                 SETRAD(neighbor, nr)
```

created at a nucleation site by delivering vapor within humid air onto a solid body material (e.g. glass).

The function GETDROPRAD(CELL) calculates the radius of a given water drop according to Archimedes' sphere volume formula based on the available water volume

$$V = \frac{4}{3} \cdot \pi \cdot r^3 \Rightarrow r = \sqrt[3]{\frac{3 \cdot V}{4 \cdot \pi}}. \tag{12}$$

A created water drop will exist forever. Reversing the process (*evaporation*) is not included in our approach. It is, for example, not possible to monitor volume (*radius*) changes while setting a higher temperature.

5. Results

We proposed a time-dependent water condensation method into a suitable modeling workflow process and simulated several different small-scale aggregate state scenarios. The implemented simulator is able to handle different materials like air, glass, and water. On the left hand side of Figure 4 the simulator shows a tumbler with a temperature of 20 °C. The glass is filled with cold water of 5 °C. On the right hand side, one can see generated condensation drops on the glass material, which sizes are independent from the grid resolution. As expected more water drops can be found at cold locations. Figure 5 shows a further result of condensed water drops on glass surface locations which are cooled by the contained water. The Blender Cycles renderer was used to render the droplets on the glass. The data of the droplets was exported from our implemented simulator into a file format specifically invented for metaballs. As shown in the last step of Figure 2 our simulator is able to handle metaballs and

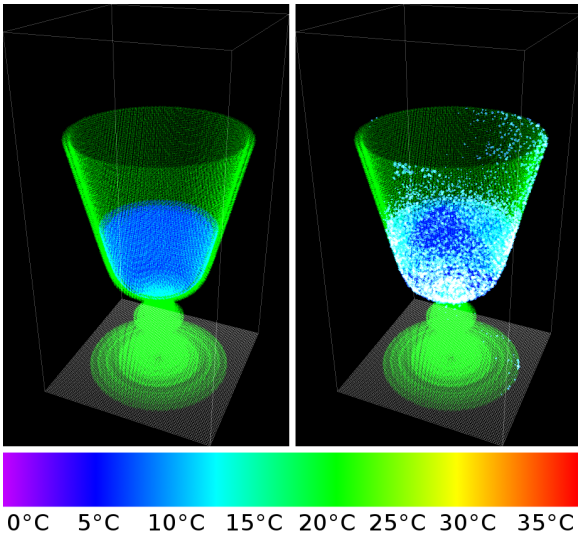

Figure 4: *Thumbler filled with water and generated condensation drops. The temperature color gradient is used in 3D simulation space.*

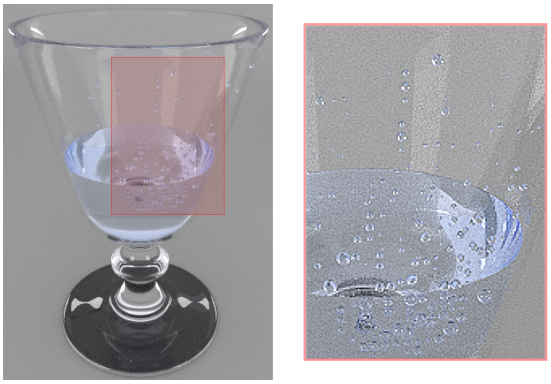

Figure 5: *Rendering with drop-metaballs using implicit surfaces.*

also explicit mesh objects. In Figure 6 a test series of 8 million cells has been carried out. After 1,500 execution steps and a computation time of 2 hours we stopped the simulation at a maximum residuum of 10^5% in the adiabatic simulation. 77,537 water drops were created during this time. The computation process took 2 hours on an Intel Haswell processor with 2.6 GHz with 12GB of RAM and an Intel HD 4400 GPU. Using a grid resolution of $200 \times 200 \times 200$ voxels, the simulation was 4.8 times slower than the thermodynamic process will need in reality. Table 3 shows the data in more detail. One can see that the droplet count doesn't get any much higher when using more steps and a longer com-

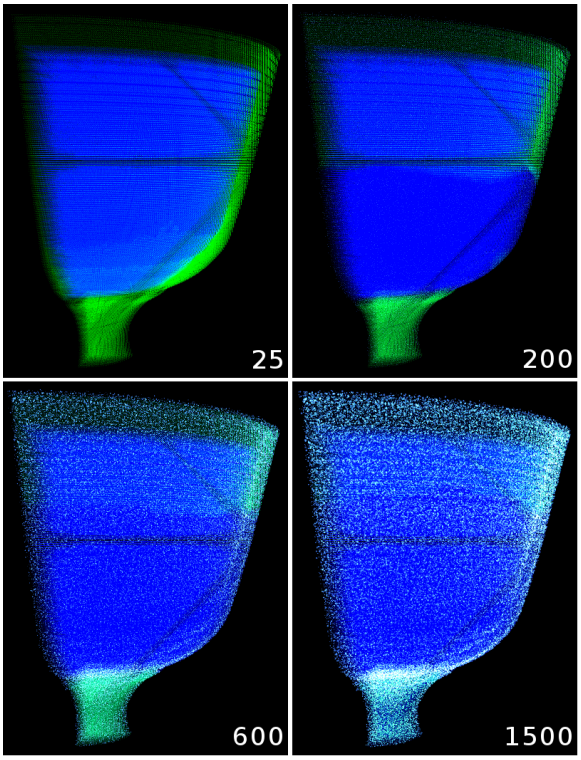

Figure 6: *Results of generated condensation water drops on the glass surface filled with water.*

putation time. This is because either there is no more water vapor in the air to receive from or every voxel around the object already has the maximum size and amount of droplets reached. Instead of generating even more droplets, the size of the existing droplets increases.

steps	comp. time	drop count	reality
5	22.0 sec	74649	5.0 sec
25	1.9 min	75312	25.0 sec
200	15.0 min	76170	3.3 min
600	45.2 min	76820	10.0 min
1500	2.0 hrs	77537	25.0 min

Table 3: *Results of an example computation with 8 million cells using a grid resolution of* $200 \times 200 \times 200$ *voxels. The inital starting temperature of the air and glass is 20 °C. The cold water has a temperature of 5 °C.*

6. Conclusion and Future Work

In this paper, we have presented a new algorithm to simulate water drops based on thermodynamics. Since the method is based on a physical model it delivers realistic results and may perfectly be suited to extend classical approaches of, i.e., simulating clouds [HBSL03] or general fluid mechanics. Due to the moderate execution times compared to a real-world scenario, the method would ideally fit into interactive modeling tools.

Future work should include a comparison to a reference where the analytic solution is already acquired. Additionally the animation of drop formation process over time could be computed, e.g. using [WMT05].

Also we will investigate other, more complex scenes to get an impression of numerical robustness in real world scenarios.

Other materials like marble, oil and cement could be implemented in a future version of the simulator.

Like many finite difference methods, even this approach seems to be able to efficiently be parallelized which will be subject to future investigations.

References

[BB06] BROCHU T., BRIDSON R.: Fluid animation with explicit surface meshes. 1

[Bra10] BRALEY S.: Fluid simulation for computer graphics: A tutorial in grid based and particle based methods. URL: http://www.colinbraley.com/Pubs/FluidSimColinBraley.pdf. 1

[BW97] BLOOMENTHAL J., WYVILL B. (Eds.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. 1

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. 736–744. doi:10.1145/566570.566645. 1

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (Sept. 1996), 471–483. doi:10.1006/gmip.1996.0039. 1

[HBSL03] HARRIS M. J., BAXTER W. V., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. 92–101. URL: http://www.markmark.net/cloudsim/harrisGH2003.pdf. 1, 6

[Hel79] HELL F.: *Grundlagen der Wärmeübertragung*. VDI-Verlag, 1979. 2, 3

[Li14] LI X.: Fluid simulation for computer graphics. *Institute of Software (Chinese Academy of Sciences)* (2014). URL: http://lcs.ios.ac.cn/intranet/images/2/2a/S-lxs.pdf. 1

[LR11] LABUHN D., ROMBERG O.: *Keine Panik vor Thermodynamik!* Vieweg Verlag, Friedr, & Sohn Verlagsgesellschaft mbH, 2011. 2, 3

[Sta99] STAM J.: *Stable Fluids*. SIGGRAPH '99. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999. doi:10.1145/311535.311548. 1

[Wag04] WAGNER W.: *Wärmeübertragung: Grundlagen*. Kamprath-Reihe. Vogel, 2004. 2

[WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. *ACM Trans. Graph. 24*, 3 (July 2005), 921–929. doi:10.1145/1073204.1073284. 1, 7