AIR CURRENT SIMULATION IN DESKTOP FLIGHT SIMULATORS.

Author : **Manuel Dobusch**

Abstract

Flight simulation is an important mean of training for flight crews. One factor of flight simulation is the simulation of air flow. The interaction of wind and terrain is often only approximated. Resulting upwind, downwind and turbulence is only of limited believability. Based on fluid simulation a new method is proposed in this work that aims to give a more detailed and believable simulation of wind-terrain interaction. The smoothed particle hydrodynamics (SPH) algorithm comes to use. A way to contain the SPH simulation in a relatively small volume is presented. Finally an evaluation of the terrain interaction of the algorithm is given.

Key words:

Flight Simulation, Wind Simulation, Fluid Simulation

1. Introduction

The goal of flight simulation is to recreate the systems and environment of aircrafts for visualization, research or training purposes (Robinson, Mania and Perey 2004). One aspect of flight simulation is the simulation of air currents and its effects on aircrafts. This work aims to find methods on how to improve upon common methods in desktop flight simulators for simulating these currents.

This work focuses on desktop flight simulations run on common PCs. The strength of those programs lies in their portability and affordability. They allow flight crews to train almost anytime and anywhere and are suitable for basic aircraft familiarization and procedure training (Robinson, Mania and Perey 2004). For hobby pilots they are often the only available method for training, so convincingly recreating as many aspects of flying as possible on these programs may be very valuable to flight crews.

Specifically the simulation of air masses at low altitudes is the topic of this work. Special attention lies on airflow above uneven terrain such as hills and mountains. In these environments airflow causes effects like ridge lift, upwind caused by wind hitting obstacles like hill or mountain slopes, and turbulent air which are influencing the behavior of aircrafts. To the author’s best knowledge these effects are usually only approximated and of limited believability. It is however very important for pilots to know when and where they occur. Glider pilots need to know where to find upwind and lift. Strong downwinds are a danger for any airplane. Especially for sports pilots they are a known cause of accidents.

A method will be proposed that can simulate wind-terrain interaction in more detail than methods used in desktop flight simulators today. Take as an example FlightGear, an open source flight simulator originally released in 1997 and still further developed today. The program simulates effects like ridge lift and wave lift based on the terrains shape, wind properties like speed and direction and the aircraft's position. One method available in the simulator is the one proposed by Forster-Lewis (2007). Four probes are placed on the ground at different horizontal distances from the aircraft along the wind direction. The slopes between pairs of probes are calculated. The steepness of those slopes and the distance of the probes to the aircraft determines how strong the up- or downwind is. Wind speed and aircraft altitude are also factored in.

To simulate gusts and turbulence FlightGear offers various models like the Dryden Turbulence Model. This model calculates the linear and angular velocity components of gusts based on a white noise signal. It can be parameterized to simulate different magnitudes (Department of Defense 2004). While this model deals with recreating the effects of turbulence as described, it does not deal with determining where turbulences occur. In practice they are often simply globally present. Therefore determining where airflow becomes turbulent is part of the challenge.

By dividing space into discrete areas FlightGear enables locally varying weather conditions. Each of these areas, referred to as weather tiles, can have different wind and turbulence conditions. This allows to approximate effects of the surface shape on air flow. (FlightGear Wiki n.d.) However those conditions are still the same within the whole tile. Again, proposing a more detailed depiction is the goal of this work.

Flight simulators usually cover vast terrains, Microsoft’s Flight Simulator X for example allows to fly around the whole earth. So the spatial domain the wind simulation needs to cover is potentially very large. This is a challenge because to cover a large area with fluid simulation a trade off between detail and performance is necessary.

The method proposed to simulate wind is based on computational fluid dynamics (CFD). Air currents at altitudes below 1000 ft above ground are simulated with attention to terrain interaction. A particle based fluid simulation algorithm called smoothed particle hydrodynamics (SPH) is used. Said particles carry physical properties of the simulated fluid. The current state of the fluid is estimated by interpolating said properties between particles via weighting kernel functions to account for spatial distribution of particles.

While SPH solves some issues involved in fluid simulation implicitly, some others that arise in this works specific setup need special treatment. One such problem is to reduce the spatial domain to reduce computational cost without limiting the particle’s ability to follow the air current. The SPH algorithm in and of itself does not limit the particle’s movement, they can move freely in space. This work deals with potentially large terrains though, so the area in which wind will be simulated will be much smaller. Otherwise the computational cost would be potentially too high. A method to confine the SPH simulation based on the work of Federico et.al. (2012) is proposed. The region of interest, in which wind is supposed to be simulated, is surrounded by more particles that do not follow the SPH simulation. They rather follow a predefined flow direction, referred to as global wind direction. They carry the same physical properties as the SPH particles. Once they enter the region of interest, they will be simulated as SPH particles. When an SPH particle leaves this area it becomes one of the non-simulated particles. This allows the SPH particles to move through the region of interest, and even leave it, without the need to have this region cover the whole terrain. Additionally a simple method based on the work of Marrone et. al. (2011) is proposed for solid boundaries. Solid boundaries refers to boundaries which particles may not cross. Immobile particles are created underneath the terrain to repell SPH particles on the ground level. They are only immobile relative to the fluid simulation domain, they move along with it though as it travels over the terrain.

2. Related Work

This work is based upon CFD and explores its usefulness for flight simulation. More specifically the simulation of air flow using CFD is explored. There are several examples for previous works dealing with similar cases. Lee, Sezer-Uzol, Horn, Long and Lyle (2005) used CFD to simulate the air wakes around ships during helicopter take offs and landings. What they did was somewhat similar to what is planned to be done in this work, although they did focus on a very specific setting and a small spatial domain, the area directly around a ship. Also they worked on high fidelity flight simulations, which commonly include specialized hardware and cockpit replications, rather than desktop flight simulations.Galway (2009) uses CFD to predict wind and turbulence patterns in urban environments. He uses these predictions to enhance the performance of unmanned aerial vehicles (UAVs) in urban environments.

Porté-Agel et al. (2011) make use of large-eddy simulation (LES) to predict airflow at planned wind turbine sites. Their focus is on predicting how the air flow will be influenced by placement of wind turbines and the wakes caused by these turbines. The goal is to predict an optimal configuration of wind turbine sites so that each turbine can achieve a maximum efficiency. LES is a field of CFD that deals with simulating the large-scale part of turbulent flows (Mathew 2010).

In this work the SPH algorithm is used as the basic method for simulating fluids, more specifically wind, over a wide terrain. Fluid simulation in general is based on the Navier-Stokes equations. Those equations describe the movement of bodies of fluid and will be discussed in chapter 2.1. After covering those basics the SPH algorithm will be discussed in some detail in chapter 2.2, followed by a discussion on techniques for solid and open boundaries in chapters 2.3 and 2.4.

2.1. Fluid Simulation

There are two basic viewpoints commonly used in fluid simulation, the Eulerian and the Lagrangian viewpoint.

The Eulerian viewpoint is named after the Swiss mathematician Leonhard Euler. Fixed points are placed in space on which properties of the fluid, like density, velocity or temperature are measured. When the fluid is moving those measurements will change over time, as fluid moves past the fixed points. For example, there could be a volume of warm fluid followed by cold fluid. As they move by a measurement point the temperature at that point will change from warm to cold. Besides movement of the fluid the measurements may also change because of dissipation or a global change in the fluid. If warm water is poured into a pool of cold water the temperature of the warm water will dissipate until the pool has an even temperature. The temperature may also decrease or increase globally over time, e.g. because of sun light.

The Lagrangian viewpoint is named after the mathematician Joseph-Louis Lagrange. Fluids are treated as sets of particles. One might think of each particle as being one molecule of the fluid, although those particles may be much bigger than actual molecules. So one should keep in mind that one particle may represent a whole packet of fluid. Besides having a position and velocity those particles may also have other properties like density or temperature. Besides collision particles may interact in other ways like attraction or dissipation.

Bridson (2008) provides an intuitive example on how to think of those two methods. One may imagine two ways of doing a weather report. The Lagrangian way would be to release a weather balloon that will move with a volume of air and measure change of temperature, humidity and whatever is of interest in that volume. The Eulerian way would be to place a weather station on the ground and measure the change of interesting properties over time at this fixed point.

From a practical point of view both ways of thinking have advantages and disadvantages. The Lagrangian view in practice would be to simulate fluids as particle systems. This resembles closer how one might think of fluids in physics. The Eulerian view on the other hand would be implemented as a fixed grid. It is easier to work out spatial derivatives on such a grid. Desbrun, Kanso and Tong (2006) describe how to do so on arbitrarily shaped grid meshes, while Elcot et. al. directly apply those methods to fluid simulation. Conservation of mass and boundary conditions on the other hand are solved easier in particle based methods. Lagrangian simulations are also free to move in space, while Eulerian simulations are confined to the space covered by their grid. Choosing one or the other approach comes down to the requirement of the individual use case.

The motion of fluids is described by the partial differential Navier-Stokes equations. The equation describing the acceleration of the fluid, called momentum equation, is

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla\vec{u} + \frac{1}{\rho}\nabla \mathrm{p} = \vec{g} + v\nabla \cdot \nabla\vec{u} \tag{1}$$

where $\vec{u}$ stands for velocity, ρ stands for the fluid's density and p for pressure. $\vec{g}$ is called the body force and v is the kinematic viscosity of the fluid.

The different terms of the equation describe different forces that influence the fluid's motion. Bridson (2008) provides an intuitive way of thinking about each term. Imagine any fluid as being made up from small particles. Each particle would have mass m, volume V and velocity $\vec{u}$. Now the terms of the momentum equation can be seen as forces acting on those particles.

To support this way of looking at it, the momentum equation is rewritten as

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla \mathrm{p} + V\mu\nabla \cdot \nabla\vec{u} \tag{2}$$

The term $m\vec{g}$ describes an external force that acts evenly on the whole fluid. A typical example would be gravity. It can also be the sum of multiple forces, like gravity and wind. $\mathrm{V}\nabla p$ describes the pressure force acting upon a particle of fluid. ∇p is the gradient of pressure at a given point. Since the gradient of a function has the direction of steepest ascent, and the pressure force would act in dirction of lower pressure, the negative of the gradient is used. The particle is supposed to move towards low pressure, so in direction of steepest descent. To approximate integration of the pressure force over the volume of the particle the gradient is multiplied by the particle volume V.

$V\mu\nabla \cdot \nabla\vec{u}$ describes the viscosity of the fluid. Fluids with high viscosity try to resist deformation while fluids with low viscosity deform quickly. Honey for example has a higher viscosity than water. $\nabla \cdot \nabla$ is the so called Laplacian operator, which measures the difference of a quantity at a point from the average of this quantity around that point. The Laplacian is often alternatively written as ∇^2. In this case the quantity in question is the velocity $\vec{u}$. The factor μ is the dynamic viscosity coefficient. As described above for the pressure force, V is used to approximate integration over the particles volume.

To reduced errors introduced of the approximations done in the above terms the number of particles in a volume of fluid can be increased. The more particles, the smaller the volume and mass per particle gets. This poses a problem when the number of particles goes to infinity because the volume and mass per particle goes to zero. Dividing by V takes care of this problem. This results in the equation

$$\frac{m}{V}\frac{D\vec{u}}{Dt} = \frac{m}{V}\vec{g} - \nabla \mathrm{p} + \mu\nabla \cdot \nabla\vec{u} \tag{3}$$

m/V equals the fluid density ρ. By dividing by ρ, defining the kinematic viscosity v as μ/ρ and rearranging the terms the equation becomes

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla \mathrm{p} = \vec{g} + v\nabla \cdot \nabla\vec{u} \tag{4}$$

Now the equation is almost back to the first form in equation . The difference is the first term $\frac{D\vec{u}}{Dt}$. This is called material derivative. To get the rate of change in time t for a value described by a function at a point $\vec{x}$ in space of the form $q(t,\vec{x})$ one has to take the derivative of this function

$$\frac{d}{dt}q(t,\vec{x}) = \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\vec{x}}{dt} \tag{5}$$

$$= \frac{\partial q}{\partial t} + \nabla q \cdot \vec{u} \tag{6}$$

$$\equiv \frac{Dq}{Dt} \tag{7}$$

Pedley (1997) shows how the momentum equation can be derived from Newton's laws of motion. The first law of motion states that whenever an object changes its state of motion a force is applied. The second law states that the acceleration from said force equals the force divided by the object's mass.

$$F = m * a \tag{8}$$

The momentum equation gives the momentum m_p for a body fluid caused by internal and external forces. Momentum is defined as

$$m_p = m * v \tag{9}$$

Changes in the velocity v are caused by internal and external forces. So equation -1 describes how the momentum within a body of fluid changes over time. Note that p is the common symbol for momentum. In the rest of this work p is used for pressure though so m_p is used instead for momentum.

Many fluids have an almost constant volume. Small changes in density and pressure, and therefore volume, occur in fluids in the form of sound waves. That is true for liquids and gases alike, although it is harder to change the volume of a liquid. Extreme situations, like putting the fluid into a pump or sonic booms may change the volume more visibly (Bridson, 2008). The change of volume caused by sound waves is negligible when dealing with fluid motion on a macroscopic level (like waves on a sea shore). Other than that fluids do not change volume under normal circumstance, so for many simulation cases it is acceptable to assume that a fluid is incompressible. This incompressibility can be expressed as

$$\nabla \cdot \vec{u} = 0 \tag{10}$$

In literature this is often referred to as the incompressibility constraint.

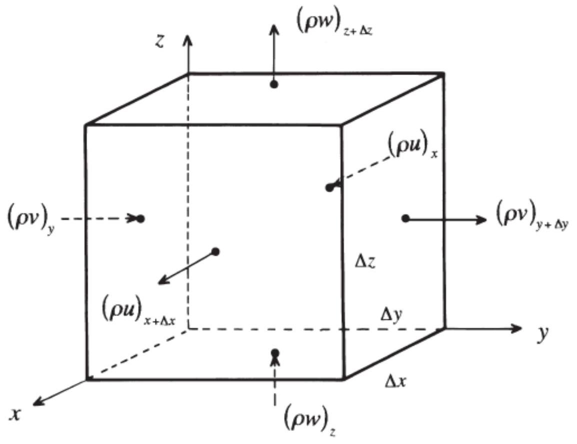


Figure 1: Flow in and out of a rectangular region in space, Pedley 1997

To derive this formula imagine a constant arbitrary rectangular volume of a fluid. The amount of mass within this volume will change with the rate of fluid entering the volume across its surface. Figure 1 illustrates the flow in and out of such a volume in accordance to equation -11.

$$\Delta x \Delta y \Delta z \frac{\partial \rho}{\partial t} = \Delta y \Delta z([\rho u]_x - [\rho u]_{x+\Delta x}) \\ +\Delta z \Delta x\big([\rho v]_y - [\rho v]_{y+\Delta y}\big) \\ +\Delta x \Delta y([\rho w]_z - [\rho w]_{z+\Delta z}) \tag{11}$$

Δx, Δy Δz are the side lenght in each dimesion of the volume. u, v and w are the components of velocity vector $\vec{u}$. So equation -11 describes how much mass is entering the volume at each face. Dividing this by $\Delta x \Delta y \Delta z$ and taking the limit results in

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x}(\rho u) - \frac{\partial}{\partial y}(\rho v) - \frac{\partial}{\partial z}(\rho w) \tag{12}$$

In short equation -12 can be written as

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{u}) \tag{13}$$

since the divergence operator $\nabla \cdot$ is defined as

$$\nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \tag{14}$$

Equation -13 expresses that the density in a given volume increases if divergence in that field is negative. In other words, if the fluid converges on that volume density increases. It was stated earlier that it is sufficient to assume that the simulated fluid is incompressible, so the density may not change over time. Equation -15 expresses that the divergence in any volume of fluid must be zero. It is called the incompressibility constraint.

$$\nabla \cdot \vec{u} = 0 \tag{15}$$

2.2. Smoothed Particle Hydrodynamics

Smoothed particle Hydrodynamics (SPH) is a lagrangian method that was introduced in 1977 by R. A. Gingold and J. J. Monaghan (1977). It was initially applied to complex problems in astrophysics that lack spherical symmetry. Solving said problems numerically on a grid would have greatly increased complexity. A symmetric problem would need N grid points on a 1 dimensional grid while a asymmetric problem would need N^3 grid points.

SPH uses parcels of fluid that move according to forces such as pressure gradient, gravity or magnetic forces. Those parcels will be called particles. The fluid equation only needs to be solved at the positions of said fluid particles. They are distributed randomly according to the fluid's density. To actually get the density at each particle position the smoothed kernel method introduced by Bartlett in 1963 and Paren in 1962 is used. This method can be thought of as an approximation to an integral according to the Monte Carlos procedure. (Gingold and Monaghan 1977)

To interpolate an arbitrary quantity A, the SPH algorithm uses the equation

$$A_I(r) = \int A(r')W(r-r',h)dr' \tag{16}$$

where r is the position at which to interpolate and r' is the position from which to interpolate. W is the kernel function and h is the kernels support radius. W weighs the quantity A at distance r' to account for decreasing influence at increasing distance. The interpolation is integrated over the kernel's support radius. To apply this interpolation to a fluid, and ultimately to SPH, the fluid needs to treated as a set of elements. An element a has mass m_a, density ρ_a and position r_a.

$$A_I(r) = \int \frac{A(r')}{\rho(r')}\rho(r')dr' \tag{17}$$

When discretizing the fluid into particles the integral of equation -17 is approximated by

$$A_I(r) = \sum_N m_N \frac{A_N}{\rho_N} W(r- r_N,h) \tag{18}$$

N are theoretically all other particles. Since the kernel function W is 0 beyond h, in practice only the particles closer than h have to be considered. Therefore N will only refer to the particles closer than h, the neighbours.
As an example consider the interpolation of density ρ at point r.

$$\rho(r) = \sum_N m_N W(r-r_N,h) \tag{19}$$

In practice r is the position of a particle, theoretically it can be a non-particle position as well. Considering this it becomes clearer that when estimating the density at the particle with position r, N must include this particle as well. It's own mass influences the density at r just like the mass of the neighbours.
Density estimation is the first step when calculating pressure based forces in fluids. The next step is to calculate the pressure at each particle and based upon that the pressure gradient. Assuming no other forces, like gravity or magnetism, the particles will move along the pressure gradient. Pressure p is calculated by

$$p = k\rho \tag{20}$$

where k is a gas constant. The influence of temperature on pressure forces can be accounted for by modifying k accordingly. Desbrun and Gascuel (1996) proposes an alternative equation that allows to define the density the fluid has when settled

$$p = k(p-p_0) \tag{21}$$

where p_0 is said defined density. It is called rest density. The pressure force follows the negative gradient of the pressure. The gradient of a function is directed towards the steepest ascent. Pressure however pushes objects towards lower pressure, therefore the negative of the pressure gradient comes to use.

$$f_i^{pressure} = -\nabla p(r_i) \tag{22}$$

In terms of integration approximation this becomes

$$f^{pressure} = -\sum_N m_N \frac{p_N}{\rho_N} \nabla W(r - r_N, h) \qquad (23)$$

Where ∇W is the gradient of the kernel function. In practice this calculation will most likely result in asymmetric forces. Consider two particles a and b. a uses the pressure at b to calculate the gradient and b uses the pressure at a. If the pressure is not equal at those particles it will result in different forces, violating Newton's third law. An alternative formulation that respects Newton's third law is proposed by Müller, Charypar and Gross (2003)

$$f^{pressure} = -\sum_N m_N \frac{p + p_N}{2\rho_N} \nabla W(r - r_N, h) \qquad (24)$$

$\frac{p+p_N}{2\rho_N}$ means that the pressure at the particle of interest and the current neighbour are considered. Therefore the calculation results in the same pressure force at both particles.

To evaluate the viscosity term $\mu\nabla^2 v$ of the Navier-Stokes equation at a particle the velocities at its neighbours are considered.

$$f^{viscosity} = \mu \sum_N m_N \frac{v_N}{\rho_N} \nabla^2 W(r - r_N, h) \qquad (25)$$

Equation -26 may again result in asymmetric forces. This becomes clear when again considering two particles a and b, which may have different velocities. Müller, Charypar and Gross (2003) propose an alternative to this as well. They point out that viscosity forces depend on velocity differences rather than absolute velocities. Considering this the following alternative to equation -25 can be expressed as

$$f^{viscosity} = \mu \sum_N m_N \frac{v_N - v}{\rho_N} \nabla^2 W(r - r_N, h) \qquad (26)$$

Body forces g are trivial to apply. Their calculation, if there is any necessary, is not part of the fluid simulation. They are applied equally to the velocity of each particle.

As Gingold and Monaghan (1977) pointed out kernels are required to have compact support. Beyond the support radius they evaluate to zero. This is a performance requirement, this way particles beyond the support radius can be ignored.

Another requirement for kernel functions is to be continously derivable twice. Considering equations -24 and -26 one can see that the gradient $\nabla W(r, h)$ and the Laplacian $\nabla^2 W(r, h)$ comes to use. The gradient and the Laplacian of a bilinear function $f(x, y)$ are defined as

$$\nabla f(x, y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) \qquad (27)$$

$$\nabla^2 f(x, y) = (\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}) \qquad (28)$$

Gingold and Monaghan (1977) used a Gaussian kernel in there original paper. In one dimension this kernel has the form

$$W(r, h) = \frac{1}{h\sqrt{\pi}} e^{-(x^2/h^2)} \qquad (29)$$

Lucy (1977), who in 1977 independently developed a method equivalent to SPH used the kernel

$$W(r,h) = \frac{105}{16\pi h^3}(\frac{1-r}{h})^3(\frac{1+3r}{h}) \tag{30}$$

The Poly6 kernel was designed by Müller, Charypar and Gross (2003) with computational performance in mind. r only appears squared, therefore taking the square root when calculating the eulerian distance between two particles can be avoided.

$$W(r,h) = \frac{315}{64\pi h^9}(h^2 - r^2)^3 \tag{31}$$

2.3. Solid Boundary

Fluids often need to interact with solid objects, such as containers. In a lagrangian method such as SPH it would be easy to simply implement rigid body collision between the fluid particles and the solid objects. However this causes simulation errors. The calculation of density at particles close to the solid boundary would be erroneous. Since there are no particles present beyond the solid boundary to influence density estimation at those particles close to the boundary, it will result in an underestimation of density.

A common solution is to use particles on the other side of the solid boundary to fix the mentioned issue in density calculation. One such scheme is presented by Colagrossi and Landrini (2003). They propose a method called ghost particles. At each timestep the particles within a layer with thickness of the kernel support radius are mirrored on the opposite side of the boundary. The position of the mirroring particles is based on the position of the mirrored particle and the boundary. The distance of the mirrored particle to the boundary is the same as the distance of the mirroring particle to the boundary.

$$r_{iM} = r_w - r_i \tag{32}$$

r_i being the position of the mirrored particle, r_{iM} being the position of the mirroring particle and r_w the position of the point of the boundary by which to mirror.

Velocity components tangential and normal to the boundary are mirrored in the following way

$$v_{iMn} = V_{wn} - v_{in} \tag{33}$$

$$v_{iMt} = v_{it} \tag{34}$$

Where $_n$ denotes the normal velocity components and $_t$ tangential component to the boundary. So the tangential velocity component of the mirroring particle v_{iMt} is equal to the tangential velocity component of the mirrored particle v_{it}. Therefore the boundary will behave as a free-slip boundary. There is no friction tangential to the surface. A no-slip version has not been proposed by Colagrossi and Landrini (2003). The normal velocity component v_{iMn} however is influenced by the velocity of the solid object V_{wn}.

The pressure of the mirroring particle is simply identical to the pressure of the mirrored particle

$$p_{iM} = p_i \tag{35}$$

Marrone et. al. (2011) propose an alternative method based on ghost particles called fixed ghost particles. Other than in the normal ghost particle method the fixed ghost particles are created only once at the beginning of the simulation at fixed positions. The particles are arranged in a layer beginning just below the surface with a thickness of support kernel radius. To find the positions of particles the surface is assumed to be a spline discretized into regularly spaced points. Normals and tangents along the surface can therefore be calculated based on said spline, with the normals facing into the solid object. The points defining the discretized spline are duplicated in direction of the normals to define a new spline. This new spline needs to be discretized again into evenly spaced points to repeat the process until enough points are created to fill a layer of kernel support radius thickness. The fixed ghost particles take the position of the discrete points along each spline. Figure 2 illustrates fixed ghost particles along a bend surface.

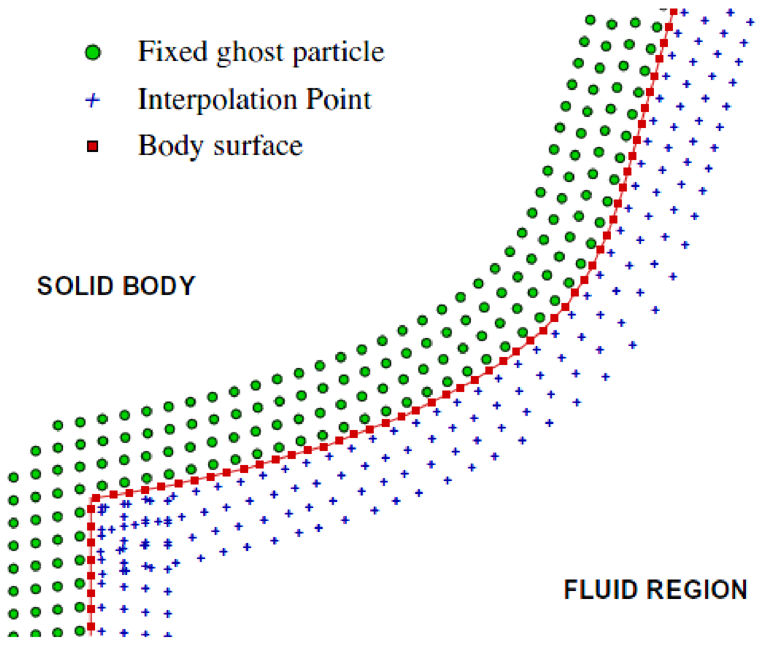


Figure 2: Fixed ghost particles and interpolation points along a surface, Marrone et.al. 2011

Other than in the normal ghost particle method the physical properties of the fixed ghost particles are not based on a paired SPH particle. Interpolation points within the SPH domain are used to calculate said properties instead. The positions of those interpolation points are the positions of the fixed ghost particles mirrored by the solid boundary surface.

Free-slip and no-slip surfaces are both easily realized. For free-slip surfaces the particle velocity calculated at the ghosts corresponding interpolation point is used as is. No-slip surfaces however are realized by reversing the tangential velocity component. In any case the normal component of the velocity is reversed.

Both approaches have advantages and disadvantages. The normal ghost particle method is simpler and has no need for artificial interpolation points like the fixed ghost particle method. It potentially needs less ghost particles because they are only created where the fluid actually comes close to the boundary. This might only be a fraction of the actual surface of the boundary. Fixed ghost particles on the other hand has the advantage to add less additional computational work since most of it is done only once at the beginning of the simulation. It also avoids some potential issues that may arise in the normal ghost particle method when dealing with complex boundary shapes. Sharp edges for instance need to be handled carefully with normal ghost particles. Equation -32, which describes how to place ghost particles, might cause problems in such a case. Figure 3 gives an example case. Since ghost particles are always distributed evenly along the boundary such cases are easier to handle.

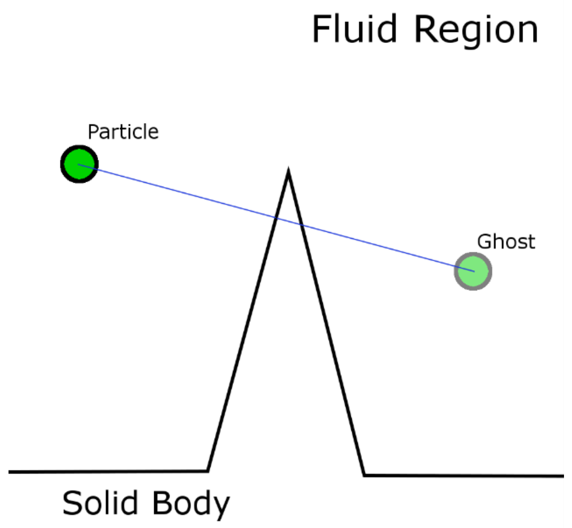


Figure 3: Misplaced ghost particle due to sharp boundary edge

2.4. Open Boundary

Considering that the proposed use case of flight simulation demands for a vast open terrain simulating airflow over the whole terrain using the SPH method appears unfeasible using current desktop computers. To account for this problem the airflow simulation is constrained to a relatively small area around a point of interest (POI). The method used to achieve this is based on the work of Federico et.al. (2012).

To simulate open channels in 2D they use two extra sets of particles, one upstream and one downstream of the actual SPH particles. Those two sets move with the direction of flow, but do not simulate forces in between their particles. The upstream set referred to as inflow particles while the downstream set is referred to as outflow particles. Between the three sets thresholds are defined, referred to as inflow and outflow threshold. Figure 4 shows those particle sets and thresholds. When a particle crosses the inflow threshold between the inflow set and SPH set the particle is removed from the inflow set and added to the SPH set. Consequently this particle is now fully simulated according to SPH. Similarly once a particle crosses the outflow threshold between the SPH and outflow set it moves from the SPH to the outflow set, after which it is no longer simulated as an SPH particle.

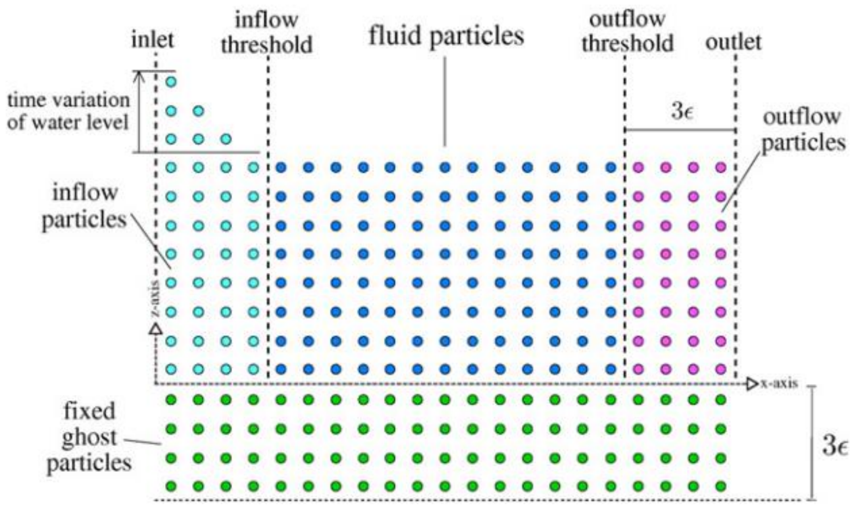


Figure 4: Inflow, SPH and outflow particles with fixed ghost particles underneath, Marrone et.al. 2012

Upstream the inflow particles are initially arranged on a regular grid. Velocity and pressure conditions are assigned to the them and they move according to the assigned velocity. Once they cross over to the SPH set their properties change according to the simulation. After crossing over to the outflow set the properties are frozen, SPH simulation ceases and the particles move along according to their last velocity assigned in the SPH set.

The inflow and outflow sets are necessary to maintain correct hydrostatic pressure at the in- and outflow thresholds. So although the particles in those sets themselves are not simulated they influence the particles within the SPH set. Without the two sets the particles at the left and right boundaries of the SPH set the SPH-particles close to the boundaries would not experience any pressure from the fluid that would be present up- and downstream in a continuous channel. So the main reason for the in- and outflow sets is to get correct density and pressure values at the in- and outflow threshold of the SPH set. Therefore the in- and outflow sets need to expand at least as far as the radius of the SPH kernel support radius from the respective thresholds up- and downstream.

3. Method

The main objective of this work is to simulate airflow over terrain. To achieve this a setup consisting of the a polygonal terrain created from a height map and a fluid simulation to simulate airflow over said terrain was created. The final goal is to derive wind forces acting upon objects in the simulation. The simulation is made with aircrafts as objects in mind.
To get the wind forces acting upon the aircraft a discrete vector field will be extracted from the fluid simulation. The fluid simulation is placed so that the aircraft is in the center of it. Force vectors can be interpolated at positions of interest along the aircraft's fuselage.
The terrain is generated using the marching cubes algorithm introduced by Lorensen and Cline (1987). The algorithm extracts a triangular surface from input data in form of a trilinear function. To do so it splits space into regular sized cubes. To create the vertices of the resulting triangle mesh the algorithm looks for intersection points of cube edges with the input surface. If one end of an edge is above the input surface and the other end is below it an intersection is found and the exact point is interpolated. To connect the so found vertices to triangles a lookup table containing all unique configuration of edge intersections with corresponding vertex connectivities is used. This table was compiled after eliminating rotated and mirrored cases from all 256 possible cases of edge intersection. 256 because there are eight edges that may or may not be intersected. So there are 2^8 distinct cases. Since terrain generation is not the main focus of this work the reader is asked to refer to Lorensen and Cline's paper (1987) for more details on the algorithm.
To simulate air the smoothed particle hydrodynamics (SPH) algorithm is used. The fluid, air in this case, is represented as a set of particles. Those particles represent small blobs of fluid and are used as sample points to calculate the fluids properties, e.g. density and pressure, at their positions. At each simulation step the density at each particle is estimated based on the position of its neighbouring particles within a predefined support radius. The influence of each neighbour is weighed by a kernel function. Based on density the pressure at each particle can be calculated. The velocity of each particle results from the pressure gradient and possibly other forces, like viscosity and external forces, called body forces. Common examples for body forces are gravity or a current. For details on the algorithm see chapter 2.2. In the following implementation details are presented.
The terrain for the use case of this work, desktop flight simulation, is potentially many hundreds of square kilometers large. To keep computation costs reasonable the SPH domain is small in relation to the terrain. Imagine using particles with a diameter of ten meters, to have a simulation resolution that is still reasonably high in relation to the size of a typical aircraft with a wingspan between 10 and 100 meters. It would take 10^8 particles to cover the ground of a 100 square kilometer area with just one layer of particles.

To address this a scheme was developed to keep the SPH domain small in relation to the terrain. Without limiting boundaries the particles would dissipate over the terrain. So boundaries were added. The particles are required to be able to move through their domain freely, and if they reach the boundary they need to be able to leave. At the same time new particles need to be added to keep the domain full. The developed scheme satisfies those requirements. It is based on the work of Federico et.al. (2012) (see chapter 3.3). As discussed earlier this work deals with open channels in 2D, so the approach had to be adapted to 3D. Instead of two extra sets of particles for in- and outflow only one extra set is used that surrounds the SPH domain in all horizontal directions, not above or below though. Below containment is handled by terrain collision and above the particles are free to move to account for the shape of the terrain below. The one extra set in this work is referred to as inflow set and it behaves mostly like the inflow set in the work of Federico et.al. (2012). An important difference is that when leaving the SPH area particles are again added to the inflow set since there is no special outflow set. Also different is that when particles leave the inflow domain they are not deleted. Instead they are moved to the opposite side of the inflow domain and continue traveling downstream from there. For details see chapter 3.3.

3.1. SPH

The calculation of physical properties, like density and pressure, in the smoothed particle hydrodynamics algorithm is based on weighed influence of particles on one another. This influence has the form of forces, e.g. pressure and viscosity. To weigh the influence kernel functions are used. Those kernels have a limited range, the support radius, beyond which the influence of particles is zero. All particles within the support radius of the kernel are considered neighbours. Finding the neighbours is the first action in each simulation time step. In this simulation it is done simply by a brute force search. Each particle's euclidean distance to each other particle is calculated. For each particle a list of indices is created. The indices of other particles that are closer than the kernel support radius are stored in that list. For all subsequent calculations on particles only the particles out of the corresponding neighbour indices list have to be considered.

The next action per time step is to calculate the fluid's density at the positions of all particles. This involves the neighbour particles found previously. The densities are stored in an array with one entry per particle. Initially the self influence of the particles is calculated. This is done using the kernel value at zero multiplied by the particles mass. The resulting value is stored in the aforementioned density array. The influence of the neighbour particles is calculated in a similar manner. The kernel value at the neighbor's distance is multiplied by the neighbours mass and added to the density value stored in the density array. The kernel used for density calculation is the Poly6 kernel, which in 2D has the form

$$W_{poly6}(r,h) = \frac{4}{\pi h^8}(h^2 - r^2)^3 \tag{36}$$

and in 3D the form

$$W_{poly6}(r,h) = \frac{315}{64\pi h^9}(h^2 - r^2)^3 \tag{37}$$

Figure 5 is a plot of the 2D poly 6 kernel.

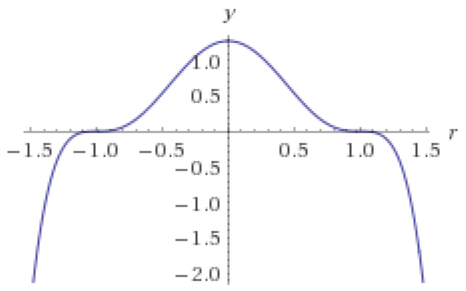


Figure 5: 2D poly6 kernel

The third action is to calculate pressure forces within the body of fluid. This is based on the densities as well as on the fluid's rest density. The rest density is the predefined density the fluid has when it is completely settled and still. Basically the fluid will try to achieve a state where the density equals the rest density at all particles. Where the density is lower the fluid has negative pressure, where it is higher it has positive pressure, over time those forces will move the particles into a state of equilibrium. Pressure is stored in form of force vectors in an array with one entry per particle. To calculate the force vector for a particle again the neighbour particles as well as their corresponding densities are used. The neighbours are iterated. The first step of each iteration is to calculate the direction from the particle to the current neighbour. This will be the direction of the pressure force between the two particles. To finally calculate pressure at the particles the Tait equation is used.

$$\tilde{p} = \frac{k\rho_0}{\gamma}\left(\left(\frac{\tilde{\rho}}{\rho_0}\right)^{\gamma} - 1\right) \tag{38}$$

Where ρ_0 is the fluid’s rest density, $\tilde{\rho}$ is the particle’s density and k is a gas constant. The factor γ is set to 7 according to Solenthaler and Pajarola (2008). $\tilde{\rho}$ is the number density, as used in their work.

$$\tilde{\rho} = m\delta \tag{39}$$

$$\delta = \sum_N W(r - r_N, h) \tag{40}$$

A kernel function is used to weigh pressure according to distance between the particles. Using calculated pressure values and the kernel value the pressure force between the particles is given by

$$f^{pressure} = -\sum_N\left(\frac{\tilde{p}}{\delta^2} + \frac{\tilde{p}_N}{\delta_N^2}\right)\nabla W(r - r_N, h) \tag{41}$$

as presented by Solenthaler and Pajarola (2008).

A kernel different from the poly6 kernel, used in density calculation, comes to use to prevent the problem of clustering. As it was pointed out in earlier by Müller Charypar and Gross (2003) clustering means that particles that come very close to each other will cease to repel each other. Therefore they may get very close to each other, to the extent that they inhabit the same position in space. The reason for this becomes clear when considering that the pressure forces are based on the gradient of the fluid's density, therefore the kernel's first derivative is used in the algorithm. Looking at the poly6 kernel's first derivative, shown in figure 6, one can see that when approaching zero it slopes back to zero.

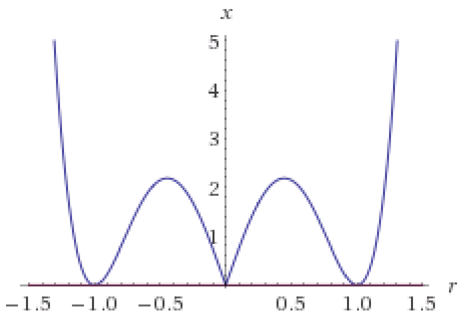


Figure 6: 2D poly 6 kernel gradient

Pressure forces resulting from this kernel would decrease once the particles are closer than the curves inflection point. The kernel used to account for this problem is called spiky kernel, since it has a spike, not a smooth curve at x = 0. As figure 7 shows the gradient of the spike does not decrease towards x = 0, therefore its gradient never reverses.

$$\nabla W_{spiky}(r,h) = \frac{30}{\pi h^5}(h-r)^2 \tag{42}$$

$$\nabla W_{spiky}(r,h) = \frac{15}{\pi h^6}(h-r)^3 \tag{43}$$

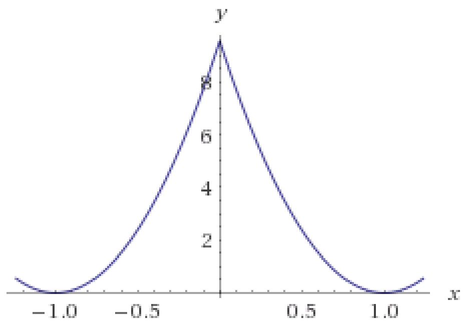


Figure 7: Gradient of 2D spiky kernel

The normalized force direction for the current particle pair multiplied by the pressure force magnitude is added to the particle's entry in the pressure force array.

Viscosity of a fluid describes how strong the fluid's tendency to resist deformation is. Water would be an example for a fluid with low viscosity whereas honey has comparably high viscosity. In SPH viscosity is modeled as the tendency of particles to move along with it's neighbours. To realize this the particle's neighbours are again iterated. The velocity difference is calculated. The influence of the neighbour on the particle of interest is based on the two particles densities and a kernel function to account for increasing distance

$$f^{viscosity} = \frac{1}{\delta}\sum_N \frac{\mu+\mu_N}{2}\frac{1}{\delta_N}(v_N - v)\nabla^2 W(r - r_N, h) \tag{44}$$

$$\nabla^2 W_{viscosity}(r,h) = \frac{20}{\pi h^5}(h - r) \tag{45}$$

$$\nabla^2 W_{viscosity}(r,h) = \frac{45}{\pi h^6}(h - r) \tag{46}$$

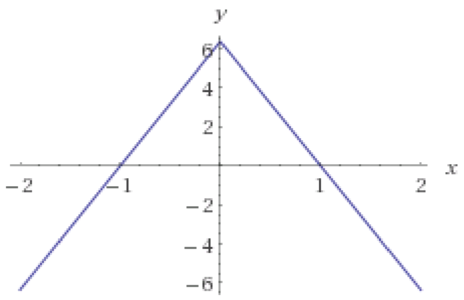


Figure 8: Laplacian of 2D viscosity kernel

The calculated viscosity factor together with the velocity difference results in the viscosity force acting upon the two particles.

Again, not the poly6 kernel is used. Referring to -44 and the explanation for the use of the spiky kernel in the pressure calculation it becomes obvious why. This time the second derivative of the kernel is used. Looking at the Laplacian of the poly6 and the spiky kernel, as displayed in figure 9 and 10, this would result in reversed forces. Instead another spiky kernel, displayd in figure 8, comes to use.

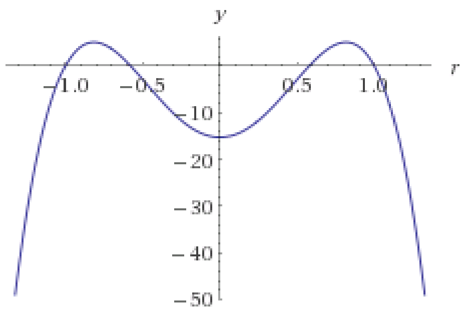


Figure 9: Laplacian of 2D poly6 kernel

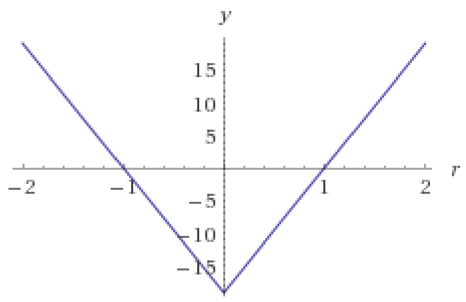


Figure 10: Laplacian of 2D spiky kernel

After calculating the pressure and viscosity forces the particles are ready to evolve accordingly. Some additional external forces are to be considered though. Those forces are commonly referred to as body forces. In this work those forces are gravity and a global wind force. Said global wind force is based on the large scale wind system. Said large scale system is not part of this work, a predefined force vector is used instead. After all those forces are added the particles can be evolved accordingly. This simply involves modifying all particle positions by eachs particle velocity, derived from the acceleration resulting from all forces.
In areas of low density SPH may cause high compression. The particles will try to match the rest density, therefore they have to get close to each other. In extreme cases particles might collapse onto the same position. The upper image in figure 11 shows how particles near the bottom get very close to each other to the extend where the slightly overlap. To prevent this behaviour this work uses a trick to manipulate the density and pressure calculation at close range. Said close range is the radius of the particles. Since each particle has defined mass and density it has a resulting radius. The kernel value at distances closer than the radius is changed by adding a second kernel. The support radius of this kernel is the radius of the particle width. The lower image of figure 11 shows the same scene as the upper image with this change applied.

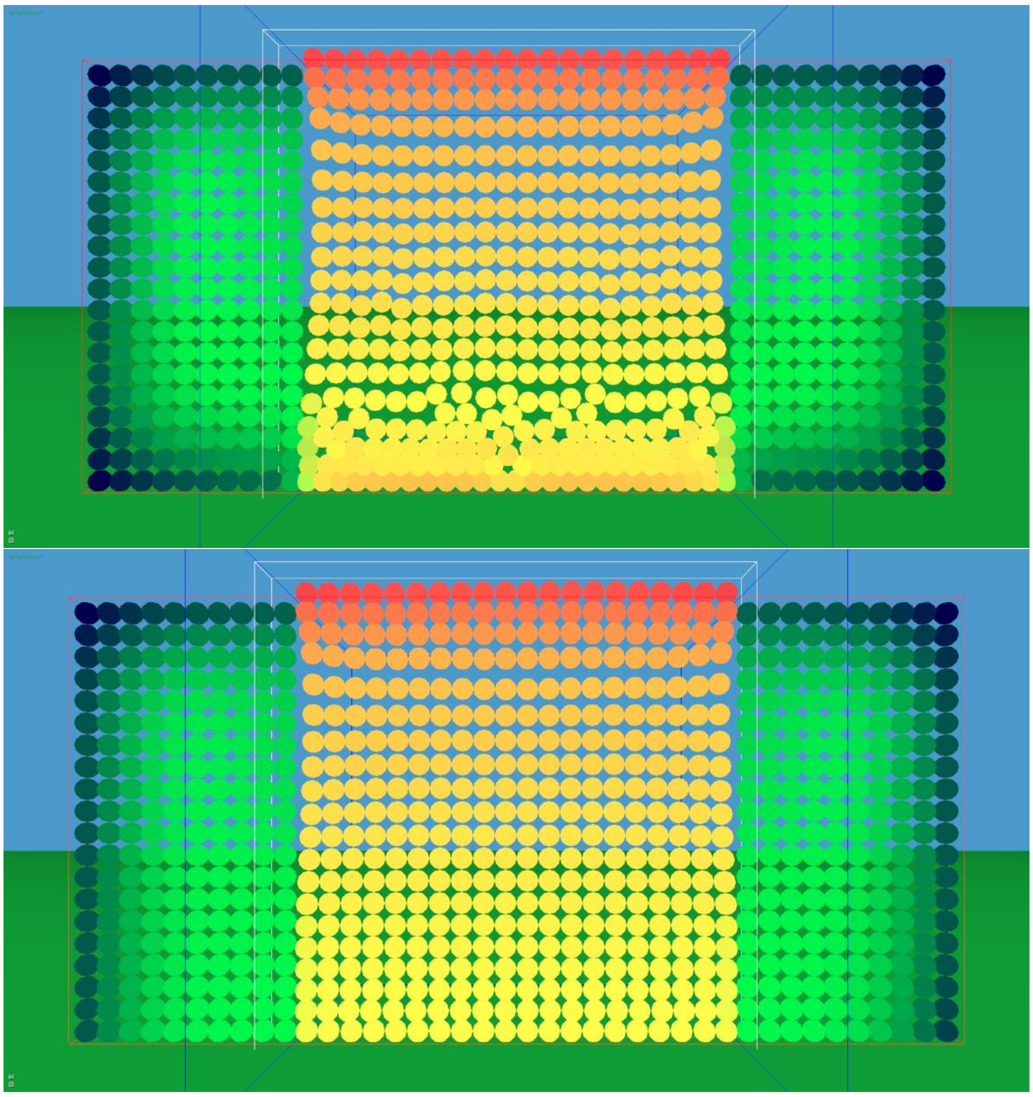


Figure 11: Particle distribution after 270 frames. The green color channel encodes density, the higher the density the higher the green channel value. The first image shows the simulation without ghost particles and near density kernel.

The forces resulting from density and pressure calculation will increase much faster once particles penetrate each other. As figure 11 shows this prevents particles from bunching up and overlapping as described above. Therefore it helps maintaining the fluid’s volume.

3.2. Solid Boundary

As discussed in chapter 2.3 solid boundaries need to be handled carefully. Simple mechanical collision between SPH particles and solid objects would cause erroneous density calculations near those objects.

This issue is accounted for based on the fixed ghost particles method as proposed by Marrone et. al. (2011). For a detailed summary see chapter 2.3. Surfaces are modelled by placing fixed particles, the ghost particles, below the terrain surface. The ghost particles are placed with regular distances in layers along the surface. The first layer is just below the surface. More layers below are added. The lowest layer is as far below as the radius of the support kernel radius of the SPH simulation. This ensures that SPH particles touching the surface will still have a kernel full of neighbours below the surface, therefore will experience pressure from below. Physical properties of those ghost particles are calculated via interpolation points. Those points are positioned at the position of the corresponding ghost particle mirrored by the boundary surface. The properties are calculated at the interpolation point and applied to the corresponding ghost particle.

For this work the ghost particle method was modified to account for potentially vast terrain sizes. To avoid having to create a very large number of ghost particles, only for the area below the inflow- and SPH area ghost particles are created. The area in which ghost particles are created reaches as far as kernel support radius length below the terrain. Initially all particles are created along a regular grid. Upon updating and whenever the inflow- and SPH areas move the ghost particles are moved along. All the ghost particles maintain their relative altitude to the terrain directly above, with the top most layer of ghost particles being directly below the terrain surface. Other than that, like in the work of Marone et.al. (2011) they do not move at all relative to the simulation domain. This way like in the original fixed ghost particles method additional computations after the initial setup are avoided. Some flexibility to change position is added. This flexibility is important because at creation time it is unknown what the shape of the surface for the ghost particles will be.

This method poses a problem when dealing with slopes. The ghost particle layer is not oriented on the boundary surface normal. Therefore gaps tangential to the boundary surface between ghost particles can form. This can results in an unfaithful surface recreation, causing unwanted friction along the surface. Since the ghost particles are created on an axis aligned regular grid those gaps are worst on 45° slopes. Figure 12 shows the ghost particles along a horizontal and a sloped plane, with visibly bigger gaps tangential to the plane underneath the sloped one.

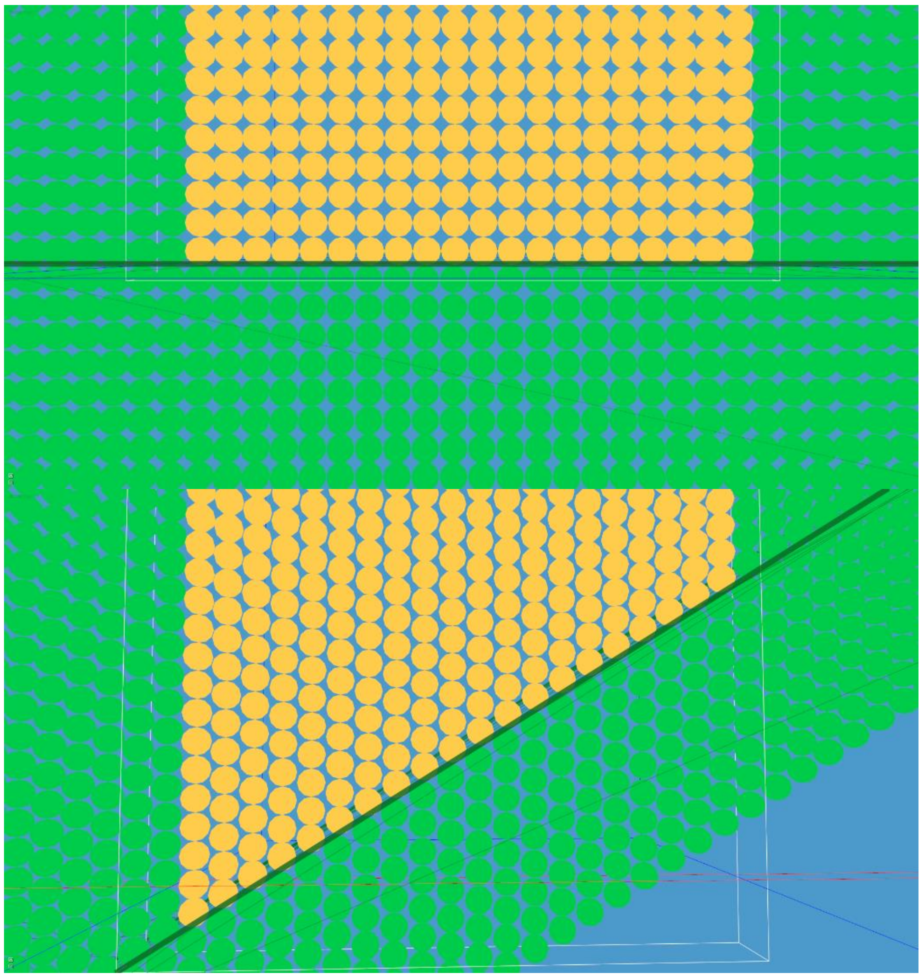


Figure 12: Distribution of ghost particles below the terrain (thick green line). The first image shows a horizontal plane. The second image shows a plane sloping upwards from left to right.

To account for this problem an additional polygonal representation of the boundary surface is used. Simple rigid body collision detection and response between this surface and the SPH boundaries is applied to keep the SPH particles strictly above the surface. The ghost particles are directly below this surface, so the density and pressure calculations are influenced as little as possible by this addition. SPH particles are reliably prevented from moving into a ghost particle gap, therefore canceling out the unwanted friction.

3.3. Open Boundaries

As mentioned earlier the airflow simulation used in this work is contained in a domain relatively small to the size of the terrain. This is done in respect to limited computation resources on current desktop PCs. The method used is based on the work of Federico et.al. (2012) (see chapter 2.4). Their work is limited to 2D, so it had to be expanded to 3D to be used in this work.

The basic concept of having a spatially limited SPH particle set surrounded by non-SPH particles is maintained. Other than having an in- and outflow set though there is only one additional set, referred to as the inflow set. The inflow set fills an area around the SPH simulated area referred to as inflow area. As in proposed by Federico et.al. (2012) the physical properties in the inflow set are frozen and they move along a globally defined wind velocity. The movement of particles in the SPH set are again governed by the SPH simulation. Also just like proposed by Federico et.a. (2012) once a particle crosses the threshold between the SPH- and the inflow area the particle changes to the particle set belonging to the entered area. To avoid possible sudden movement due to changing pressure conditions when a particle enters the SPH area the transition is gradual rather than sudden. Once the particle touches the SPH area, simulation starts, but simulated forces are only applied proportional to the part of the particle that is already within the SPH area. Therefore only when the particle is completely within the SPH-area the calculated forces are fully responsible for its movement.

As discussed earlier (see chapter 2.4) the inflow particles are there to provide correct density and pressure conditions at the boundary of the SPH set, therefore the inflow set expands for the length of the SPH-kernel support radius in any horizontal direction from the respective side of the SPH set. Figure 13 displays the setup from above. Yellow particles are SPH particles while green particles are inflow particles. Encoded in the green channel of the particle color is the calculated density, showing an that the SPH particles all have the same. The inflow particles can move in any direction perpendicular to the ground, therefore enabling any horizontal wind direction. To allow the SPH particles to react to the terrain below, there are no inflow particles above.

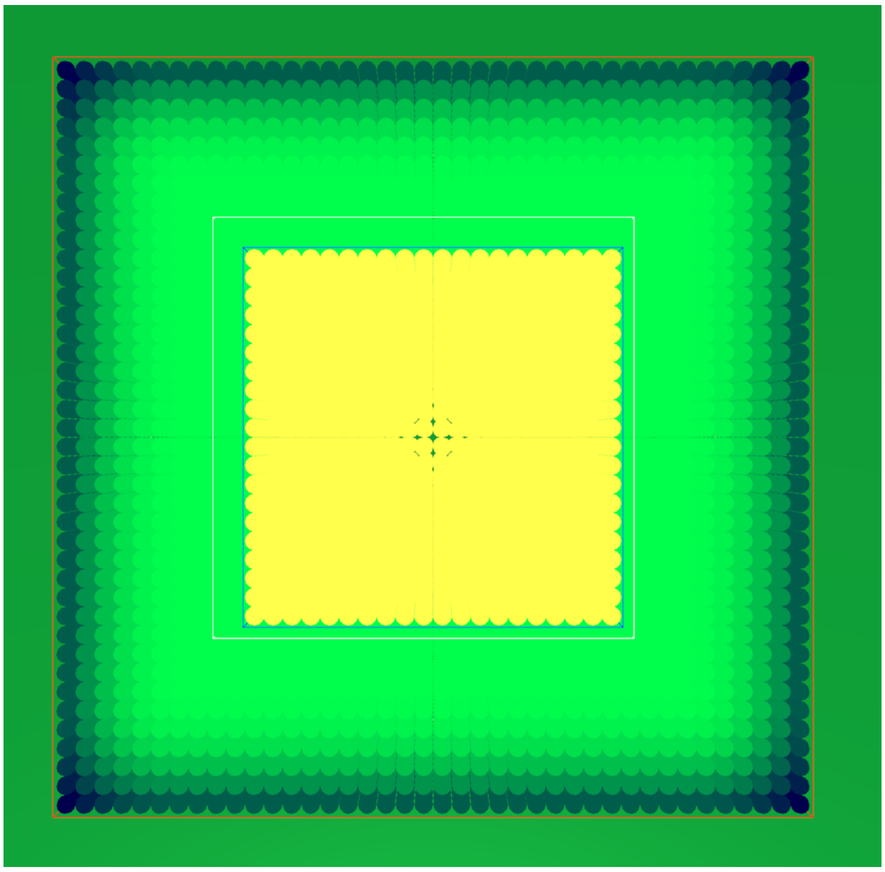


Figure 13: Particle setup from above, yellow particles are SPH particles, green/blue particles are inflow particles

Particles leaving the inflow area are handled by re-inserting them at the opposite side of the inflow area. In other words, if the particle leaves the inflow area in direction of the x-axis it's x coordinate is set to the opposite end of the inflow area. The same is done in z direction. Figure 14 illustrates this in 2D.

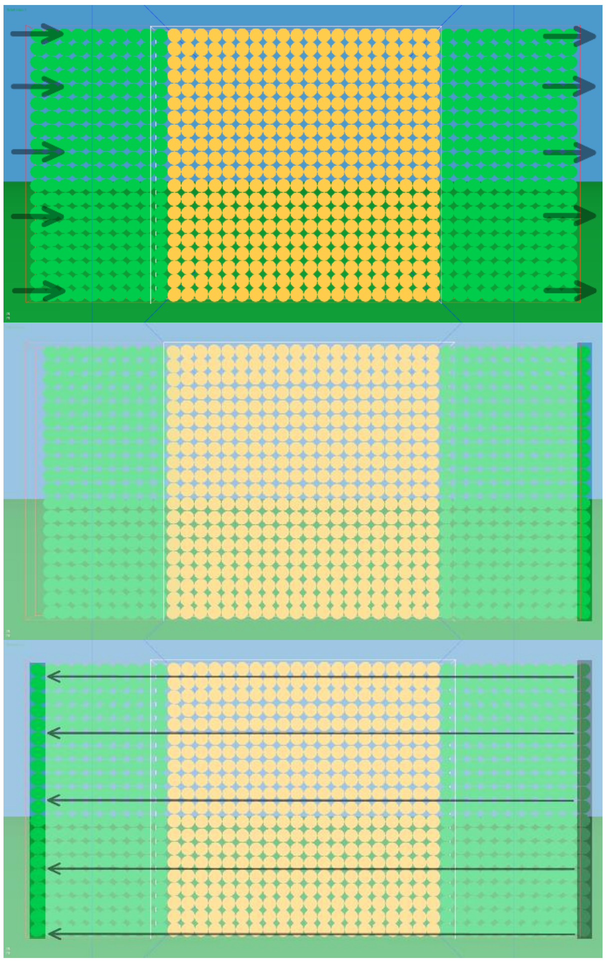


Figure 14: Particles leaving the inflow area downstream are teleported back upstream. Image one shows the initial setup, arrows indicate flow direction. Image 2 shows a set of particles leaving the inflow area. Image 3 shows where they end up after teleportation (old positions are greyed out)

All initial particles are conserved, so the overall mass of the simulation is conserved. Note that the number of particles within the SPH set may vary slightly. Since the inflow particles are incorporated in the density and pressure calculation the overall mass in the SPH simulation does not change.

The posibility of uneven ground had to be considered in this work, while Federico et.al. (2012) assume even ground. Uneven terrain in the SPH area is handled by the rigid body interaction of the SPH simulation, inflow particles on the other hand do not interact with terrain on their own. A simple scheme is used to deal with this problem. Particles store their altitude above ground at initiation as well as at each timestep they spend within the SPH set. When moving as inflow particle the particles altitude above ground is constantly adjusted to this stored altitude. The same altitude is used to vertically resposition the particle once it leave the inflow area and is teleported back upstream.

4. Discussion

To create a number of test setups and simplify interpretation of results the simulation domain was confined to 2D. This enabled quick visual verification of plausible behaviour of the fluid. The first test aimed to assure low compressibility of the fluid setup. As discussed above in chapter 3.1 SPH does not guarantee incompressibility. A modification was implemented to improve this. The test setup consists of flat terrain and a still volume of fluid. Tests are conducted with varying spatial resolution. The resolution will be specified in SPH particles per dimension, e.g. 10 by 10.

After starting the simulation the fluid will start to settle and end up in an equilibrial state. If the fluid is set up correctly it's volume should not change. Since the initial particle configuration is rectangular verifying constant volume is simply done by finding the bounding box around the SPH particles and logging the boxes volume over time. Even though the particles are still 3 dimensional spheres, for the 2D case the area of the bounding box projected onto the xy plane is used instead of the actual volume. The z size of the bounding box solely depends on the particle size. As tests are conducted with varying spatial resolution the particle size changes. Forces and movements are restricted to the xy plane though, therefore no information is lost when ignoring the bouning box's z-expansion.

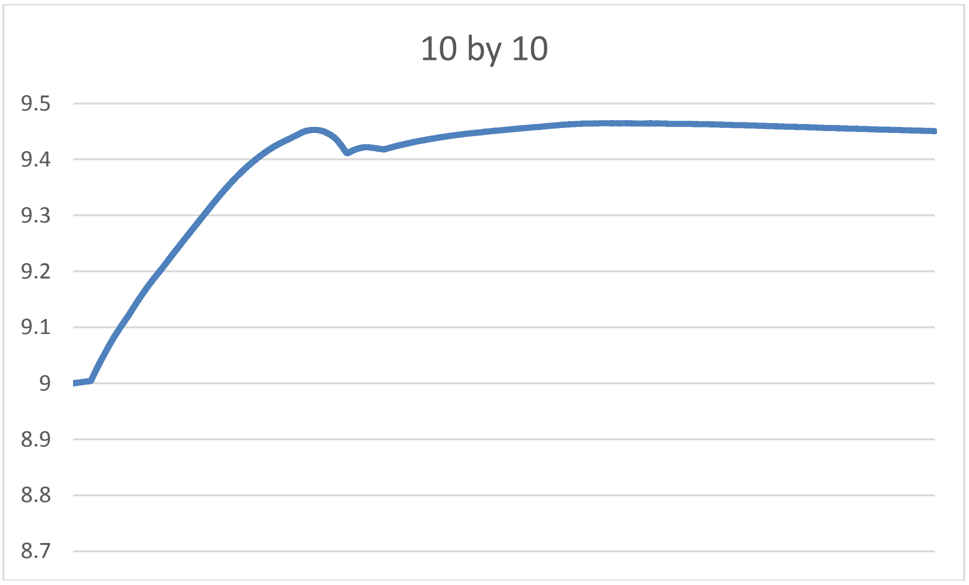


Figure 15: Change of volume for 10 by 10 particles

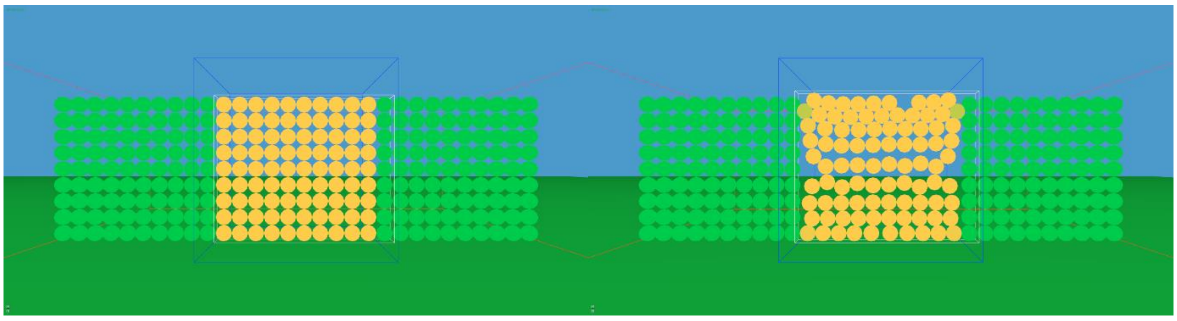


Figure 16: Frame 1 and 8000 of the compressibility test for 10 by 10 particles

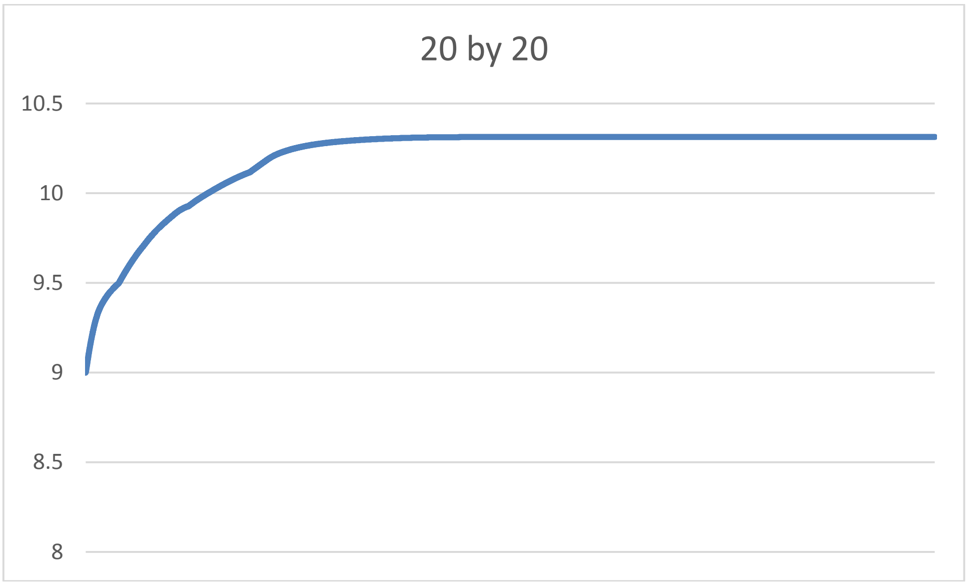


Figure 17: Change of volume for 20 by 20 particles

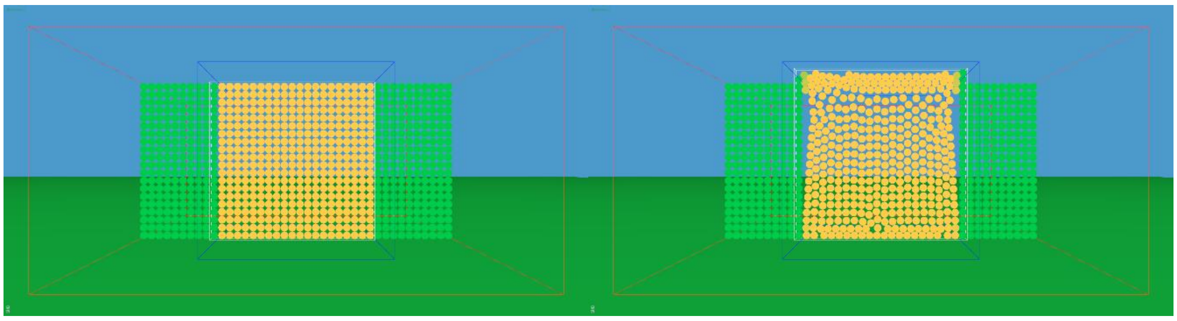


Figure 18 Frame 1 and 8000 of the compressibility test for 20 by 20 particles

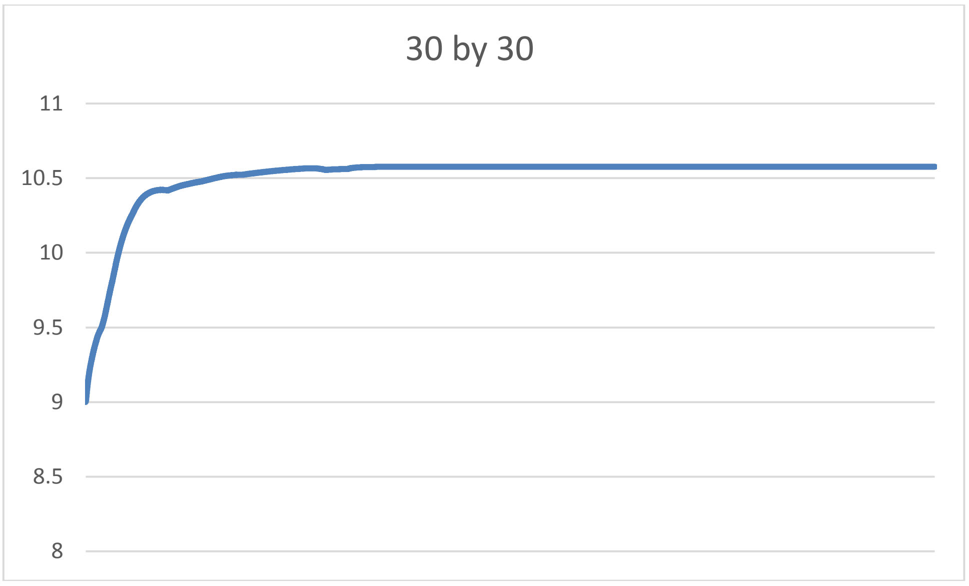


Figure 19: Change of volume for 30 by 30 particles

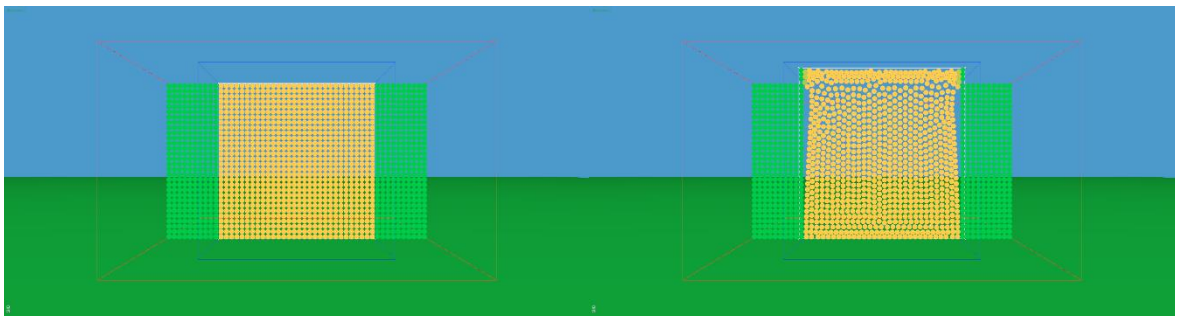


Figure 20: Frame 1 and 8000 of the compressibility test for 30 by 30 particles

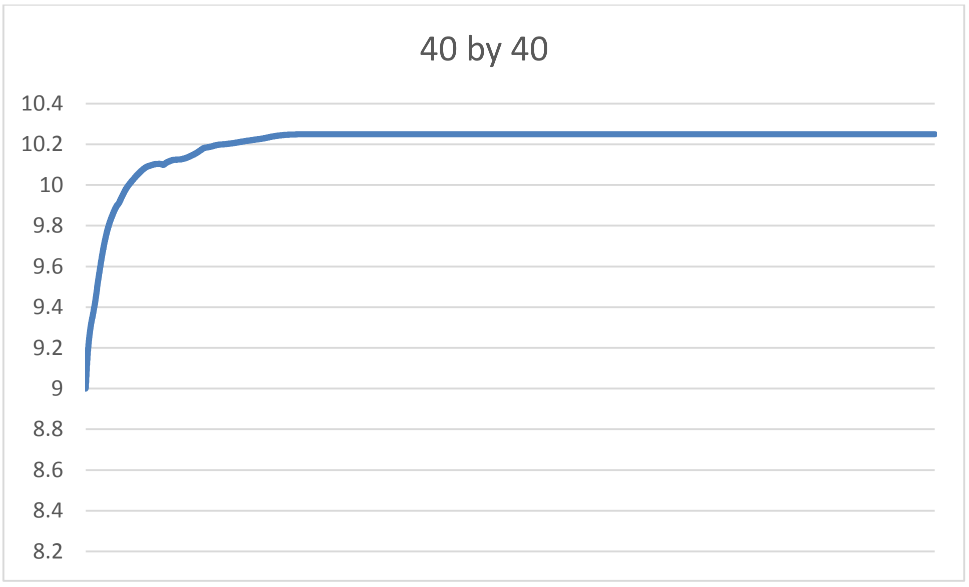


Figure 21: Change of volume for 40 by 40 particles

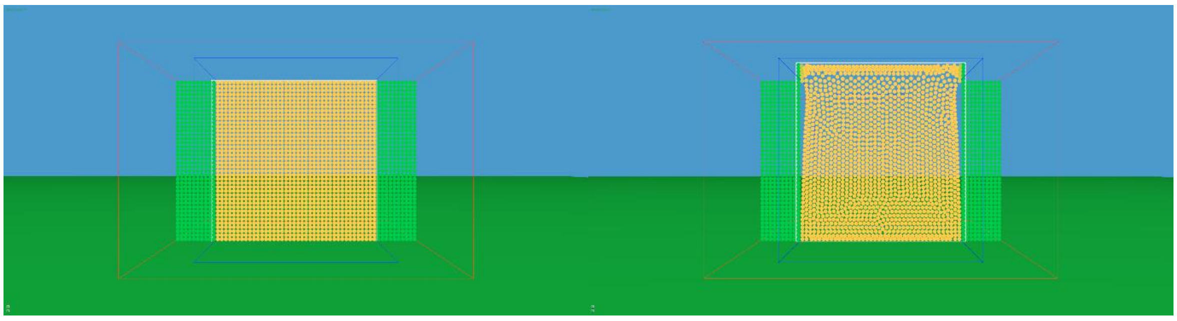


Figure 22: Frame 1 and 8000 of the compressibility test for 40 by 40 particles

The plots in figures 15, 17, 19 and 21 show that as expected after an initial phase of settling the fluids volume stays the same. The plots and images show N by N particle setups, starting with N=10, incrementing in steps of 10 additional particles per dimension. The yellow particles in figures 16, 18, 20 and 22 are the SPH particles, while the green particles are the inflow particles. Recall chapter 3.3, the inflow particles will only move along the global wind velocity, which is zero in this setup. Therefore the green particles are static. The vertical axis of the plots displays the area of the axis aligned bounding box around all SPH particles projected onto the xy axis.

Each plot was recorded over 8000 frames. Simultaneously images of each simulation step were recorded. The images above show frame 1 and 8000. Additinoal frames per test are provided in appendix A 1. Looking at the plots it can be seen that the fluid initially expands for about 2000 frames after which the volume does not change considerably anymore.

Looking at the the frames 8000 in figures 16, 18, 20 and 22, one notices distinctively how the fluid particles closeness increases in the top most rows, decreases somewhat below that and increases again. The increasing closeness towards the bottom is caused by the weight of the fluid above pressing down on the lower particles. This coincides with air or water pressure in real life, which increases downwards, depending on the mass of fluid directly above. This does not explain the closeness of the particles in the top rows. When looking at the fluid's density distribution, visible in figures 25, 27, 29 and 31 it becomes more apparent what happens. Since there are less and less particles on top to fill the kernel disc (or sphere in 3D) the density decreases. Therefore pressure forces decrease as well. Neighbours can get closer to each other before density and pressure is great enough to keep them at distance from each other.

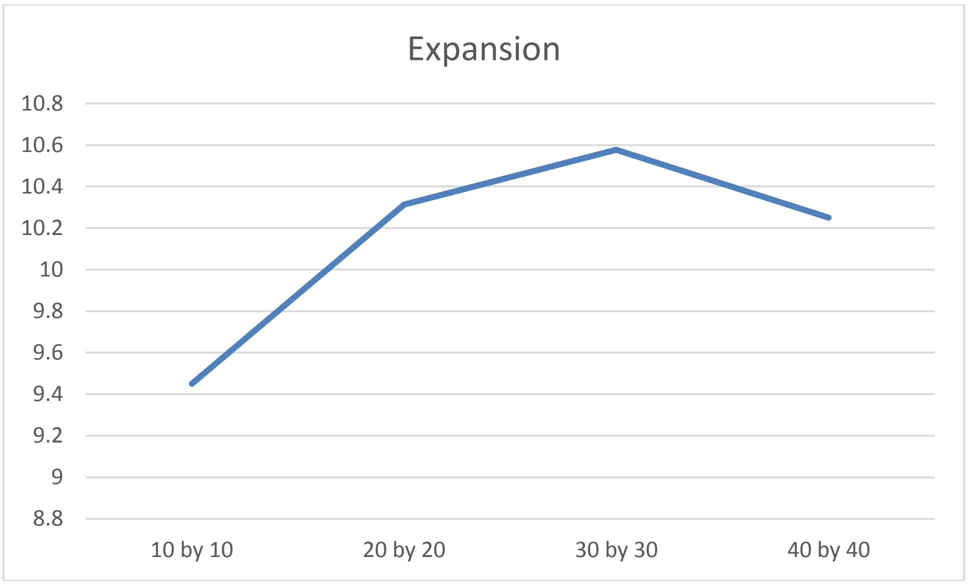


Figure 23: Volumes of 10 by 10, 20 by 20, 30 by 30 and 40 by 40 setups at frame 8000

When comparing the amount of expansion of the different setups one notices that it is not the same for all tests. Figure 23 displays the volume of the fluid at frame 8000 for all test runs. This should not be the case and demands for further investigation.

As described in chapter 2.4 and 3.3 a modified version of the open boundary method as introduced in by Marrone et.al. 2012 is used to enable free flow through the simulation domain. Important for this method is to create enough inflow particles around the SPH domain so that the kernel of the outermost SPH particles is completely filled with inflow particles. Otherwise density and pressure at said outermost SPH particles will be too low. It should be equal to the other SPH particles at the same altitude above ground. To verify this the particles log their calculated fluid density at each timestep. Those densities are translated to color values, which are used to render the particles. So particles with the same color have the same density. Color values from densities are normalized row vise to emphasize density differences in particles at equal altitude.

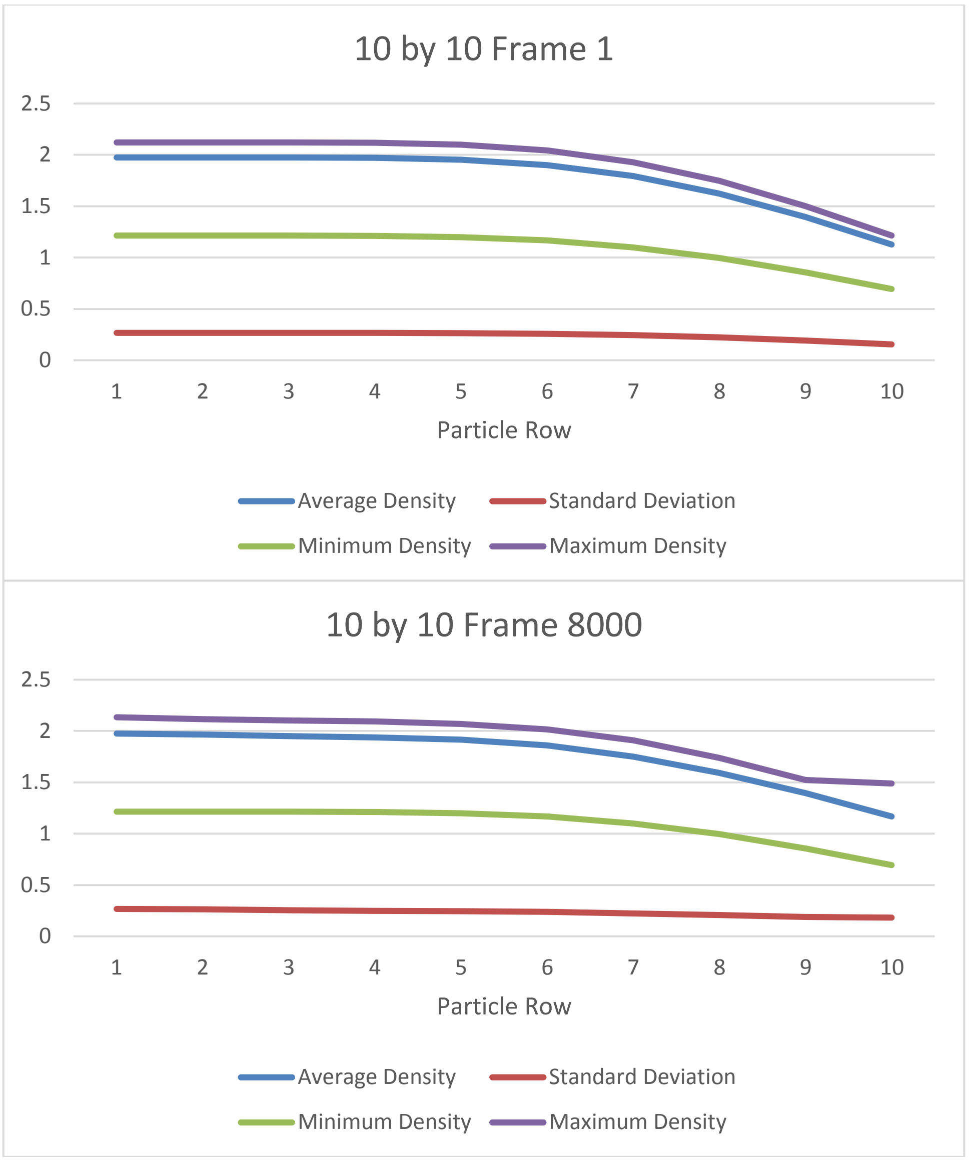


Figure 24: Row wise density distribution for 10 by 10 particles at frames 1 and 8000

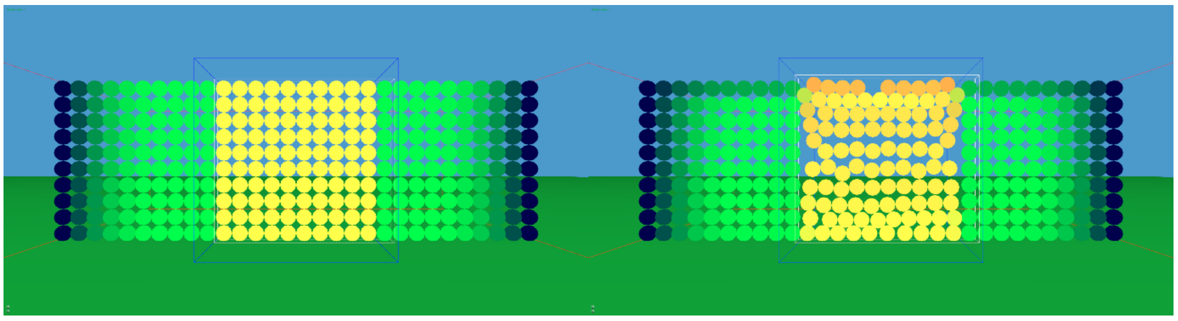


Figure 25: Density distribution for 10 by 10 particles at frame 1 and 8000

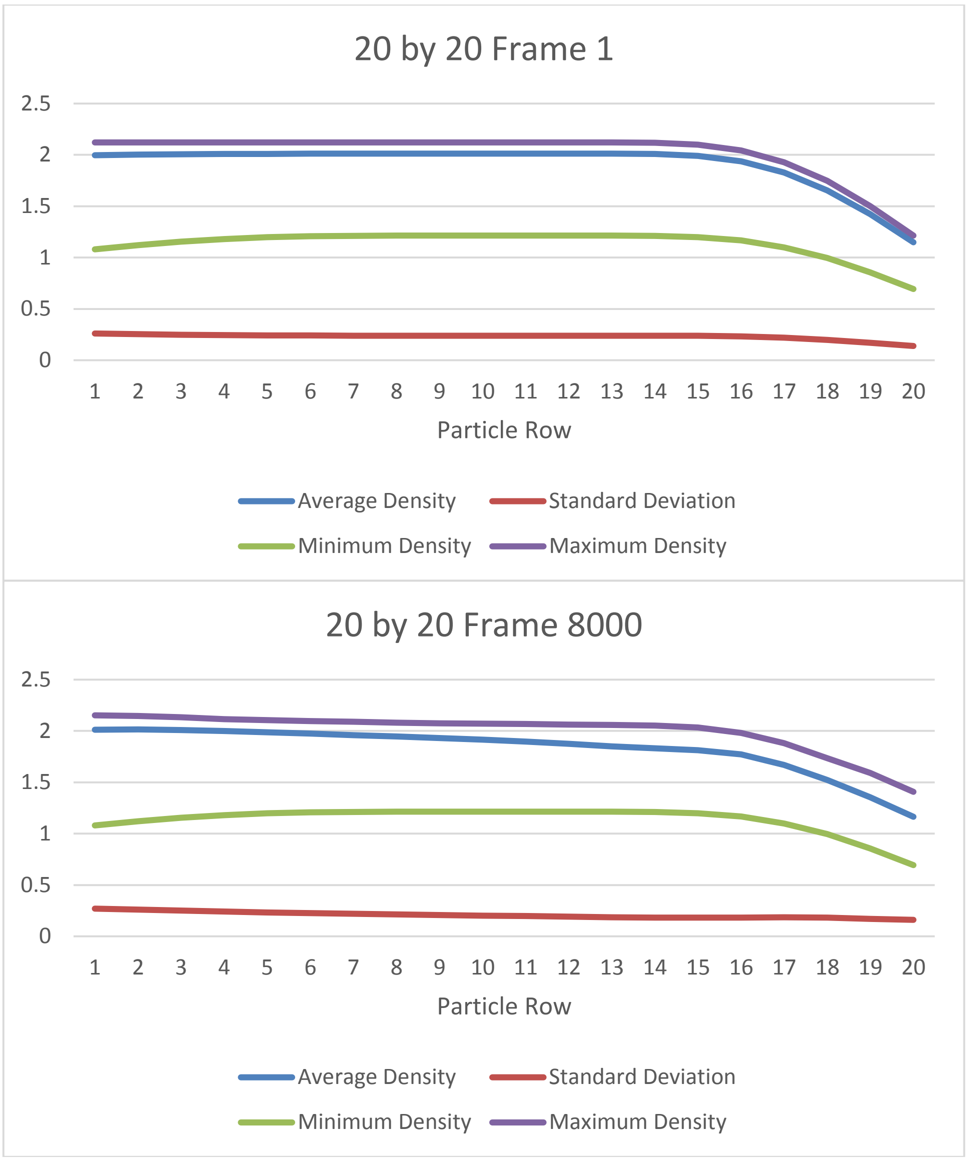


Figure 26: Row wise density distribution for 20 by 20 particles at frames 1 and 8000

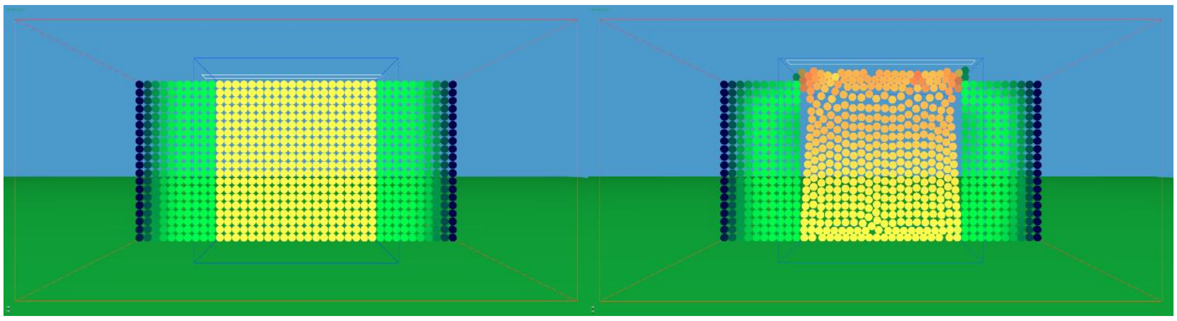


Figure 27: Density distribution for 20 by 20 particles at frame 1 and 8000

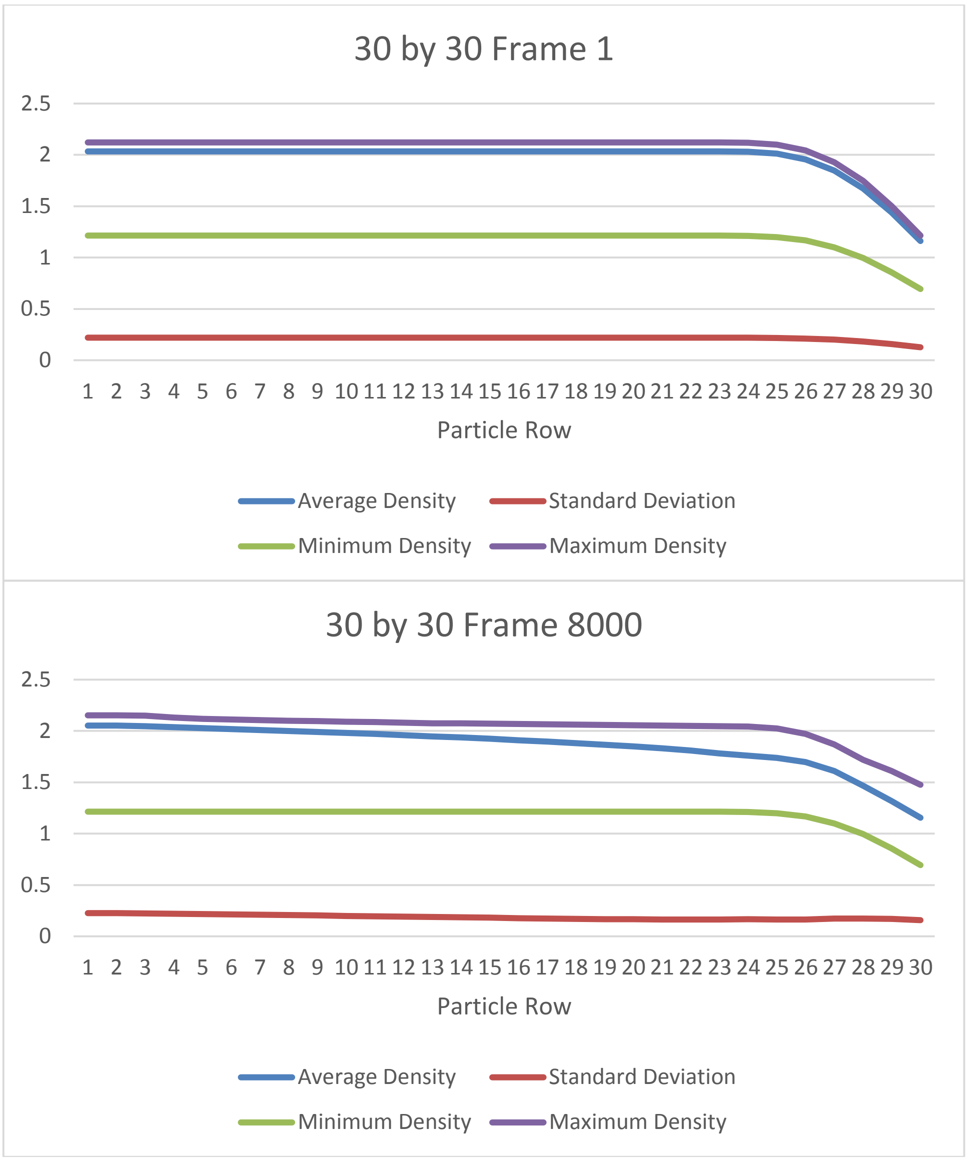


Figure 28: Row wise density distribution for 30 by 30 particles at frames 1 and 8000

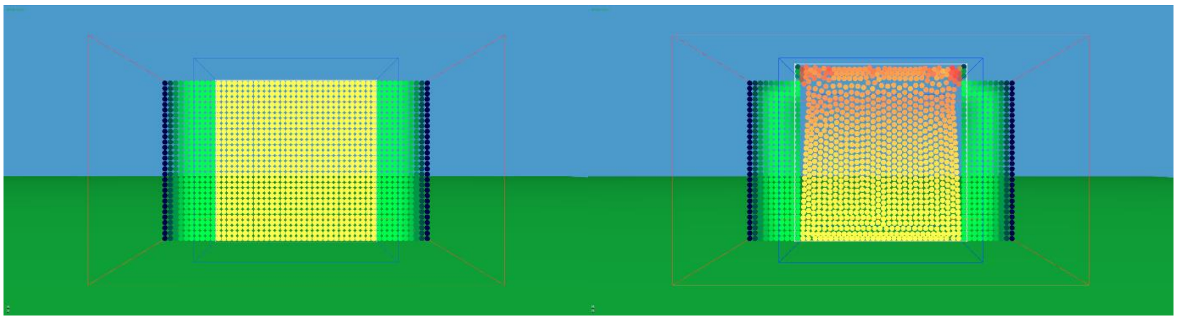


Figure 29: Density distribution for 30 by 30 particles at frame 1 and 8000

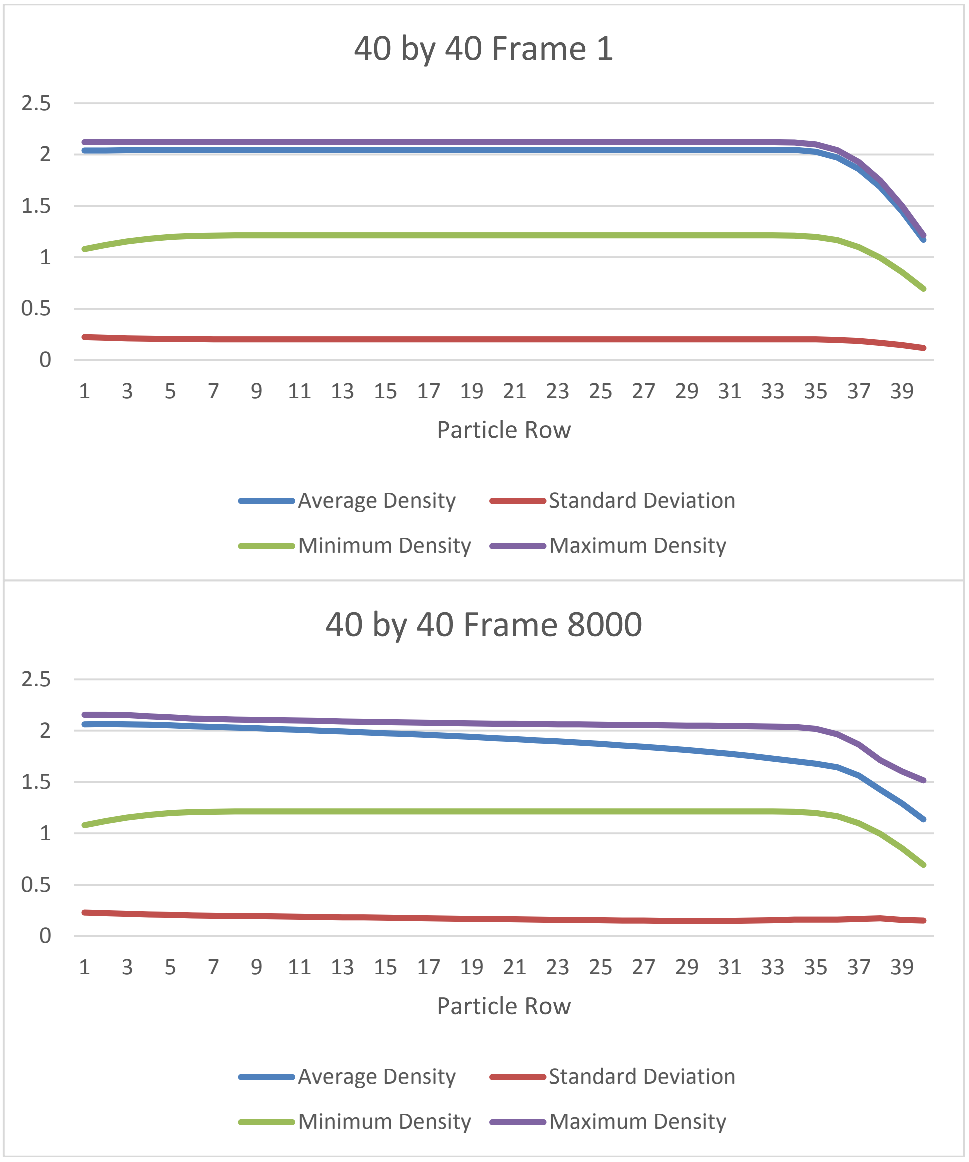


Figure 30: Row wise density distribution for 40 by 40 particles at frames 1 and 8000

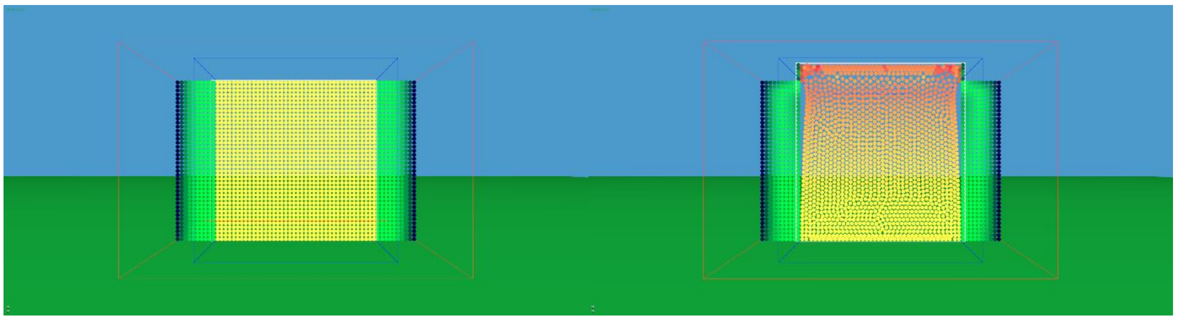


Figure 31: Density distribution for 40 by 40 particles at frame 1 and 8000

The plots in figures 24, 26, 28 and 30 show various data points for each particle row of a single simulation step. They display the maximum and minimum density per frame, as well as the average density and standard deviation. In correspondence to the provided images in figures 25, 27, 29 and 31 one can see that the density is initially mostly uniform throughout the body of SPH particles. More frames per test can be found in appendix A 2. The plots provide some initial insight in the overall distribution as they display that the average, as well as maximum and minimum densities decrease at the top 5 rows. Those row's kernel is decreasingly filled from the top, explaining the decreasing density.

Images and plots from later simulation frames show that the fluid's density decreases nearly linearly until reaching the top five rows, where it drops faster. The images reveal that due to low density the separating forces in the top most rows are so small that particle separation strongly decreases. In the bottom rows, whereas the plots show density is highest, particle separation decreases again. The weight of the SPH particle body presses down on those bottom particles. Those results are constant for all observed spatial resolutions.

Tests with the moving fluid against rugged terrain were conducted. Figure 32 displays two frames out of a test with 20 by 20 particles flowing uphill. The frames are the first and the 1000th one. Flow direction is from the left to the right.

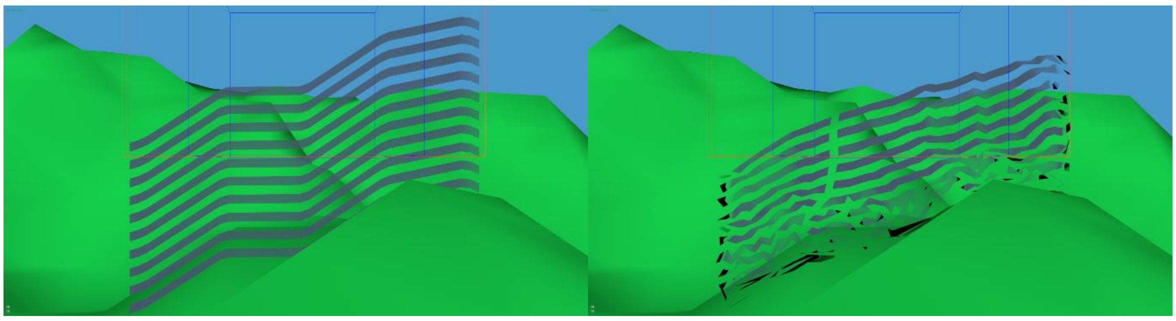


Figure 32: 2D test on rugged terrain. The first image shows the initial state of the simulation, the second image shows the state after 1000 frames.

Instead of rendering the particles directly a technique called smoke surfaces is used. This technique was proposed by von Funck et.al. (2008). It comes to use to emphasize movement and flow rather than the current state of the simulation. As the smoke streaks in the second image show the flow close to the terrain surface gets disturbed over time. The flow further up on the other hand evens out. The initial bend seen in the first image is almost gone in the top most streak of the second image.

The compresibility test and density distribution test was repeated in 3D. Due to the unoptimized nature of the implementation only one test with 10 by 10 by 10 SPH particles could be conducted so far. Also the rendering for both tests was combined. 1400 frames were recorded. Figure 33 shows the first and last frame of the test.

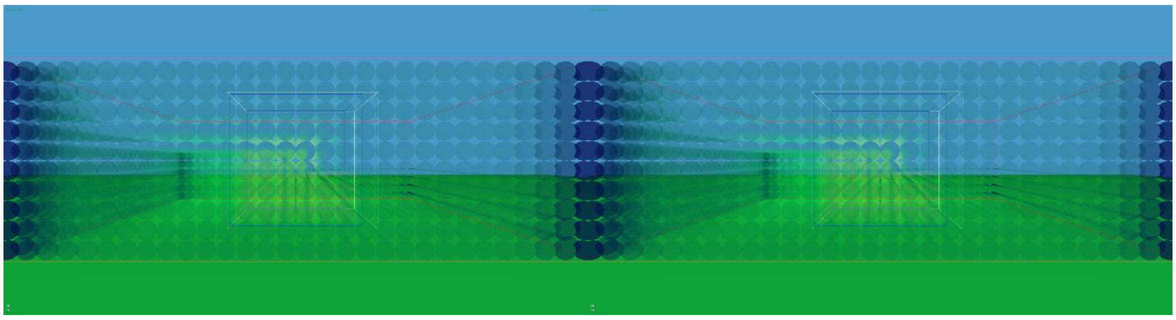


Figure 33: Frame 1 and 1400 of the 3D incompressibility and density distribution test.

The visualization was changed for the 3D test to account for the occlusion of particles further away from the camera. Simple color coding like in the 2D density distribution test would not allow to see details in the inner of the body of fluid. The difference in densities between a particle and its direct neighbours is translated to the particles alpha value in addition to the green color channel. That way particles with greater difference are opaque, while particles with the same density as their neighbours are transparent.
The images show a mostly even distribution of density. The particles at the inflow domain corners are much more opaque. That and the dark blue color indicates a much lower density than in surrounding particles. Given that at those positions much of the kernel’s sphere will remain empty this makes sense. Comparing the first and second image, as well as the two plots in figure 35 hardly if any change in the density distribution is visible. As for a change in volume, not much is to be seen. Figure 34 however shows an increase of 0.31 % in volume.

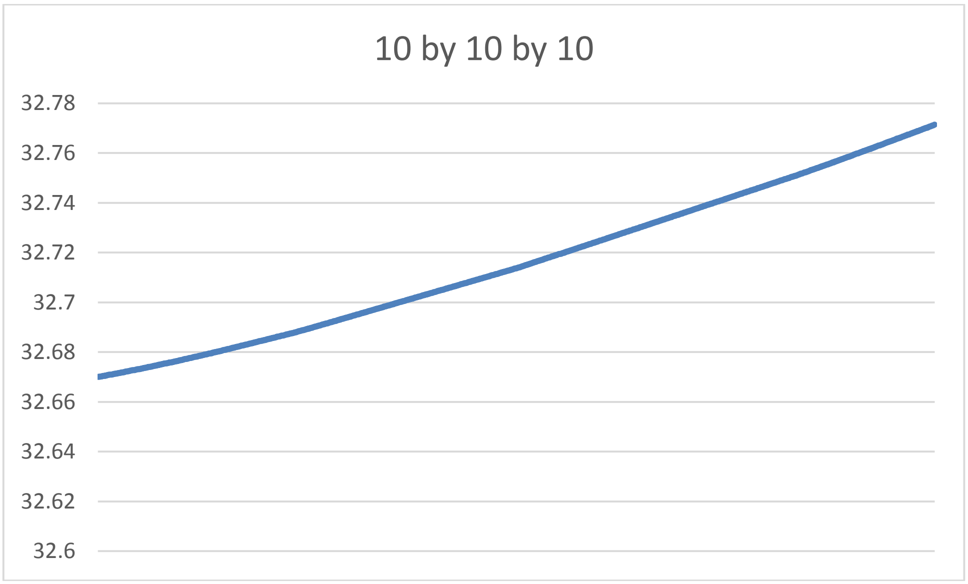


Figure 34: Volume of the 10 by 10 by 10 setup from frame 1 to 1400

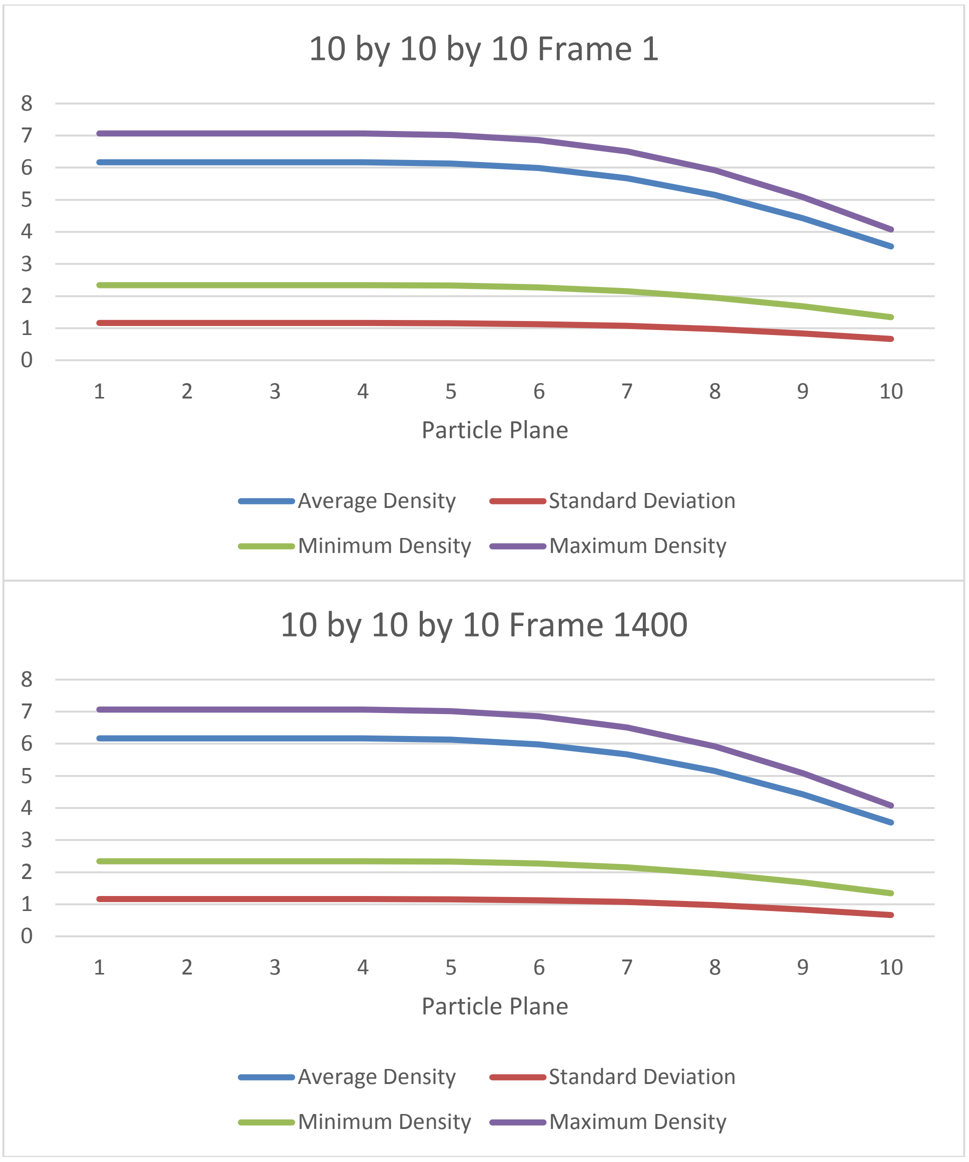


Figure 35: Plane wise density distribution for 10 by 10 by 10 particles at frames 1 and 1400

5. Summary

A general setup was created that can meet the basic requirements of simulating a fluid in a spatially limited domain. To reiterate, the environment of the simulation is a terrain relatively large compared to the fluid simulation domain. A terrain as it would be used in desktop flight simulation is assumed. Such a terrain would typically cover multiple hundreds of square kilometers. Simulating a fluid covering this whole terrain with sufficient detail would have very high computational costs. Therefore only a small region of interest is covered with fluid. This region of interest would be the region surrounding an aircraft in a flight simulator.

The smoothed particle hydrodynamics (SPH) algorithm is used to simulate air flow. SPH is based on the lagrangian viewpoint, it uses particles to simulate fluids. Particles act as blobs of fluid with fixed volume and mass. Based on closeness of other particles the fluid density at a particle's position can change though. Particles follow the pressure gradient resulting from regions of varying density within the fluid. Lastly viscosity, the tendency of a fluid to withstand deformation, is considered. SPH implicitly ensures conservation of mass. It does not however ensure incompressibility. Modifications were added to reduce compression.

SPH can naturally handle free boundaries, however it does not handle flow through scenarios like the one this work deals with. To reiterate the lagrangian viewpoint, particles are released into the fluid and follow its stream. Therefore the SPH particles, and with them the simulation domain, would naturally keep travelling downstream. For this work the simulation domain needs to be placeable freely without following the stream. Federico et.al. (2012) deal with a similar problem in 2D. In addition to the SPH particles two extra sets of particles are used, inflow and outflow particles. Those particles follow the stream but are not simulated using SPH. The inflow particles will cross over to the SPH set upstream once they enter the simulation domain. SPH particles will cease being simulated and cross over to the outflow set once they leave the SPH domain downstream. The extra particle sets ensure correct density calculation at the up- and downstream borders of the SPH domain. This work generalizes the approach to 3D. Instead of two separate sets of additional particles only one set, named inflow set, is used. It surrounds the simulation domain in all horizontal directions. Other than that it behaves like the in- and outflow set of Federico et.al. (2012). Particles leaving the inflow set downstream are teleported back upstream. Therefore all particles in the system are maintained and mass is preserved. Federico et.al. (2012) assume a flat surface below the fluid, this work does not. To deal with uneven surfaces the particles within the simulation domain store their altitude above ground. Particles in the inflow set constantly update their vertical position to maintain the last stored altitude above ground.

As mentioned before the setup needs to be freely moveable relative to the terrain. This is solved implicitly through the inflow set. When the simulation domain and inflow domain are moved, the particles behave just like when leaving their domains due to fluid flow. That means that they are teleported back into their domain.

6. Outlook

The tests conducted so far have revealed that the fluid expands slightly at the beginning of the simulation. The expansion in 2D slightly varies depending on the spatial resolution of the simulation, as shown in chapter 4. The volume of the fluid in a still and equilibrial state shouldn’t change with the resolution though. In 3D on the other hand hardly any change in volume or distribution of density was visible. The next step in development will be to determine what causes the volume change in 2D and why the 3D setup behaves differently. This needs to be addressed and fixed before moving on.

As of now the algorithm is not optimized for performance. This results in very high computation times. Some parts of the SPH algorithm can run in parallel, therefore use of multi-threading will be added.

Once the last issues with the fluid simulation are dealt with a test will be conducted to compare the simulation results to the real world. This test will consist of two phases, data acquisition and simulation comparison. The acquisition will involve a smartphone application to record data during real flight. The kind of data was selected based on availability of sensors on smartphones and usefulness in a test setup. The simulation returns force vectors acting upon the aircraft's fuselage. To get force vectors to compare to the simulation the phone's accelerometer output will be recorded. As long as the airplane flies with neutral controls and unchanged throttle setting, the only acceleration forces experienced by the phone will be caused externally. In other words by changing wind conditions. To recreate the external situation of the airplane the exact position will be needed. Therefore GPS positions will be recorded constantly. The global wind velocity will not be acquired via the application. Weather data, including large scale wind data, is available from local weather services. In Austria this would be the Austro Control GmbH.

With those data points consisting of GPS points and acceleration vectors in addition to the weather data from local weather services a test can be set up. Firstly the terrain beneath the airplane has to be recreated. The simulation setup already has the ability to create 3D terrain from heightmaps, so the correct height map from sources such as the Shuttle Radar Topography Mission (SRTM) by NASA has to be obtained (Ramirez 2015). With the recreated terrain the simulation can be run along the recorded flight path. The calculated force vectors at each GPS coordinate will be stored. Now those stored simulated force vectors can be compared to the force vectors recorded in real flight. This way it will be possible to gain some insight on how close this simulation is to the real world.

Beyond the scope of this work a number of potential improvements to the proposed method will now be discussed. First of all improvements in performance should be addressed. Many aspects of this work are not optimized, therefore there is much potential for further speed up.

It is planned to add the use of multi-threading to the algorithm proposed in this work. However beyond that the use of GPUs to further improve performance has been explored.
Goswami et. al. (2010) and Krog and Elster (2010) for example propose a parallel SPH implementation on the GPU using CUDA. Both works acknowledge the neighbourhood search as computationally costly. A naive implementation has complexity of $O(N^2)$, where N is the number of particles. Each particle's distance to each other particle has to be computed. Storing the neighbourhood removes the need to repeat this for density, pressure and viscosity calculation, as well as for possible additional physical properties. Nevertheless it remains very costly. Using uniform grids is a common optimization scheme for this problem. Grid cells can be created with a side length equal to the kernal support radius. Particles from a cell now only need to check particles in the same cell and the direct neighbour cells. Krog and Elster use it determine neighbourhoods. To optimize subsequent access of particles and neighbours they sort the particle array so that particles and neighbours are close to each other in said array. Goswami et. al. propose an approach also based on a uniform grid. They present a way to calculate neighbourhood efficiently on the GPU.
As Goswami et. al. (2010) point out calculating density, pressure and viscosity cannot be done in parallel since pressure depends on the result of density calculation and viscosity depends on the velocities resulting from pressure calculation. Calculating those properties step by step can however be done particle-wise in parallel.
For future work the use of SPH for large scale air flow can be investigated. This work focuses solely on small scale movement in a volume of several hundreds of meters of side length. Only mechanical interaction with terrain is considered. The general wind direction, caused by thermodynamic phenomena in the atmosphere is not simulated here. Said wind direction is assumed as input from an aditional simulation layer. SPH could be used to create such a layer.
Consider for example land and sea breezes. Generally speaking movements in the atmosphere are caused by regional differences in air pressure. Those air pressure differences are caused by local air temperature differences. Air heats up differently depending on heat radiation from the sun and the ground. Solid ground changes temperature much faster than water, so air over land is warmer than air over water during the day while it is reversed during the night. The warmer body of air rises, causing a low pressure area. Air from the adjacent high pressure area will attempt to fill the low pressure area.
Adding thermal properties to SPH particles is easily done, heat distribution and dissipation can be naturally modeled with use of the kernel functions used in the algorithm. In the same way a number of other properties can be added to simulate even more atmospheric phenomena. Adding humidity and defining a dew point can make the simulation of cloud and

fog formation and dispersal, as well as dissipation possible. When the particles carry information about humidity and temperatur simulating freezing becomes possible as well.
All those are important factors in aviation, especially in flight planing and navigation. The more situations and phenomena that can be simulated faithfully the better pilots can prepare for real flight in a safe and cheap environment.

7. References

Bridson, Robert. 2008. “Fluid Simulation for Computer Graphics.” Wellesley: A K Peters Ltd

Colagrossi, Andrea. Landrini, Mauricio. 2003. “Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics.” Journal of Computational Physics 191 (2): 448-475

Departement of Defense. 2004. “Flying Qualities of piloted Aircraft.” MIL-STD-1797A

Desbrun, Mathieu. Gascuel, Marie-Paule. 1996. “Smoothed particles: a new paradigm for animating highly deformable bodies.” Proceeding of the Eurographics workshop on Computer animation and simulation ‘96: 61-76

Desbrun, Mathieu. Kanso, Eva. Tong, Yiying. 2006. “Discrete differential forms for computational modeling.” ACM SIGGRAPH 2006 Courses: 39-54

Federico, Ivan. Marrone, Salvatore. Colagrossi, Andrea. Aristodemo, Francesco. Antuono, Matteo. 2012. “Simulating 2D open-channel flows through an SPH model.” European Journal of Mechanics – B/Fluids 34: 35-46

FlightGear Wiki. n.d. FlightGear. FlightGear. http://wiki.flightgear.org/FlightGear (accessed March 20, 2014)

Forster-Lewis, Ian. 2007. “Efficient simulation of topograhic lift in Microsoft Flight Simulator.” http://carrier.csi.cam.ac.uk/forsterlewis/soaring/sim/fsx/dev/sim_probe/sim_probe_paper.html (accessed October 25, 2015).

Gingold, Robert. A. Monaghan, Joseph J. 1977. “Smoothed particle hydrodynamics: theory and application to non-spherical stars.” Monthly Notices of the Royal Astronomical Society 181 (3): 375-389

Goswami, Prashant. Schlegel, Philipp. Solenthaler, Barbara. Pajarola, Renato. 2010. “Interactive SPH simulation and rendering on the GPU.” Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation: 55-64

Krog, Øystein E. Elster, Anne C. 2010. “Fast GPU-Based fluid simulation using SPH.” Proceedings of the 10th international conference on Applied Parallel and Scientific Computing (2): 98-109

Lee, Dooyong. Sezer-Uzol, Nilay. Horn, Joseph F. Long, Lyle N. 2005. “Simulation of Helicopter Shipboard Launch and Recovery with Time-Accurate Airwakes.” Journal of Aircraft 42 (2): 448-461

Lorensen, William E. Cline, Harvey E. 1987. “Marching cubes: A high resolution 3D surface construction algorithm.” SIGGRAPH ’87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques: 163-169

Lucy, Leon B. 1977. “A numerical approach to the testing of the fission hypothesis.” Astronomical Journal 82 (12): 1013-1024

Macklin, Miles., Müller, Matthias. 2013. “Position Based Fluids.” ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings 32 (4)

Marrone, Salvatore. Antuono, Matteo. Colagrossi, Andrea. Golicchio, Giuseppina. Le Touzé, David. Graziani, G. 2011. “δ-SPH model for simulating violent impact flows.” Computer Methods in Applied Mechanics and Engineering 200 (13-16): 1526-1542

Monaghan, Joseph J., 2005. “Smoothed Particle Hydrodynamics.” *Rep. Prog. Phys.* 68 (8)

Müller, Matthias. Charypar, David. Gross, Markus. 2003. “Particle-based fluid simulation for interactive applications.” Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation: 154-159

Pedley, Timothy J. 1997. “Introduction to Fluid Dynamics.” Scientia Marina 61(1): 7-24

Ramirez, Eric. 2015. Shuttle Radar Topography Mission. http://www2.jpl.nasa.gov/srtm/ (accessed October 26, 2015)

Robinson, Andrew., Mania, Katerina., Perey Philippe. 2004. “Flight simulation: research challenges and user assessments of fidelity.” Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry: 261-268

Solenthaler, Barbara. Pajarola, Renato B. 2008. “Density contrast SPH interfaces.” Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation: 211-218

von Funck, Wolfram. Weinkauf, Tino. Theisel, Holger. Seidel, Hans-Peter. 2008. “Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments.“ IEEE Transactions on Visualization and Computer Graphics 14 (6): 1396-1403

XFlow CFD. n.d. Unique CFD Approach. Next Limit S.L. http://www.xflowcfd.com/technology/view/cfd (accessed March 20, 2014)

Appendix

A 1. Compression Test 2D

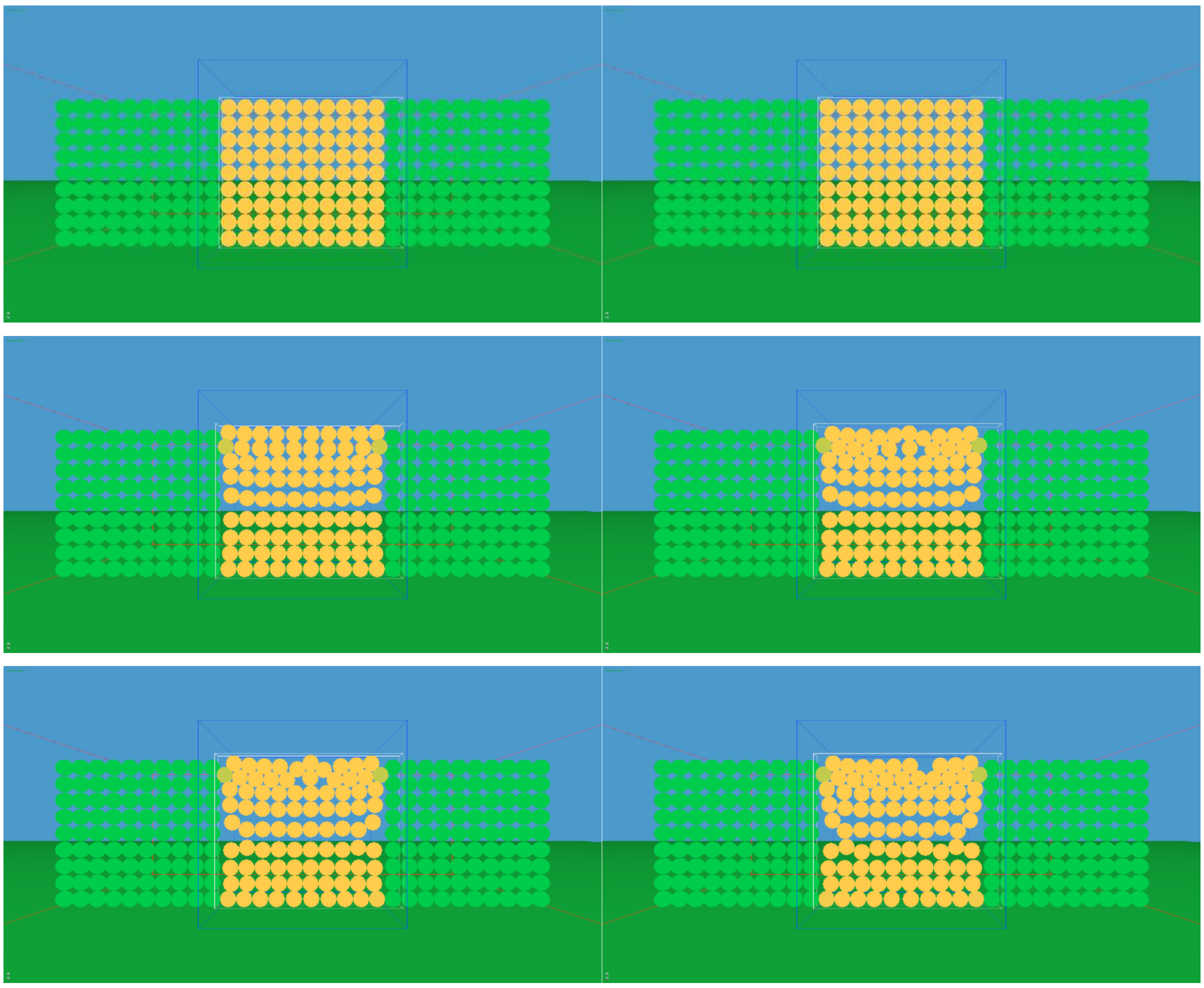
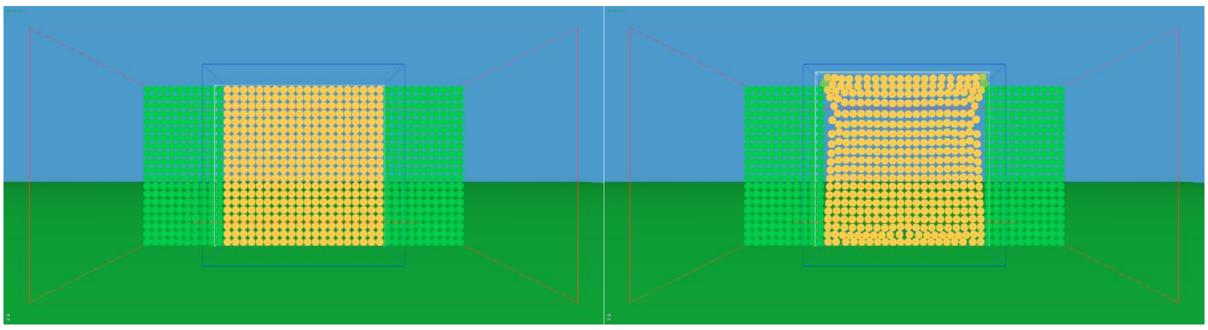


Figure 36: Frame 1, 1000, 2000, 3000, 4000 and 8000 of the compressibility test for 10 by 10 particles



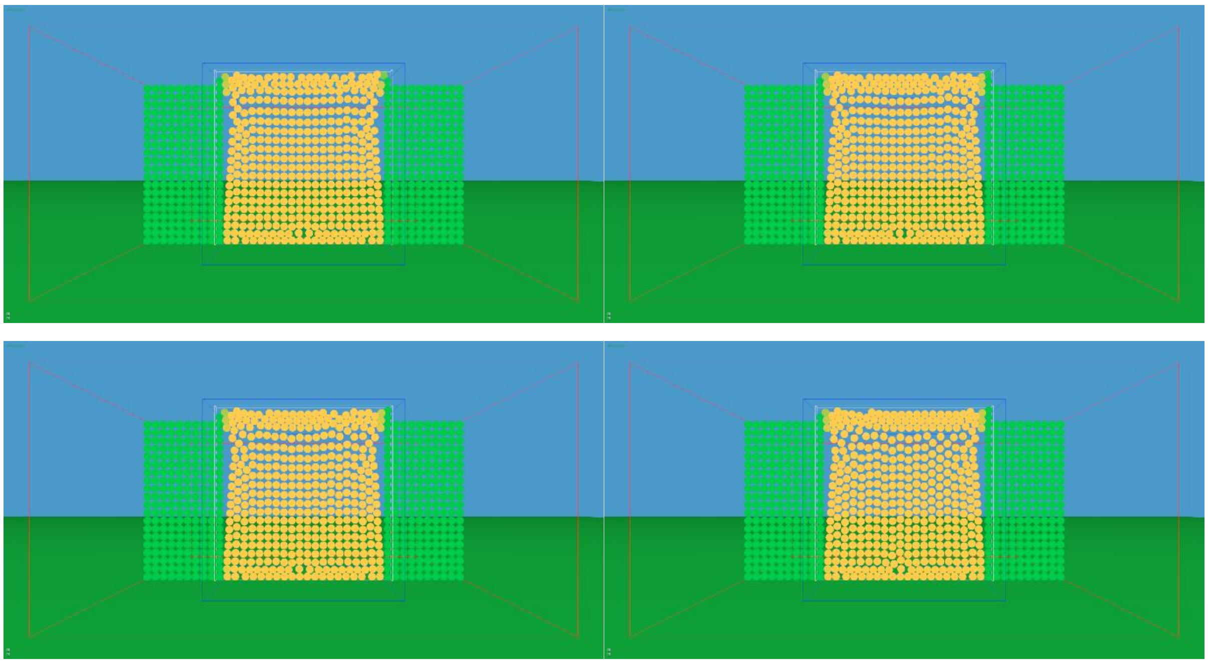


Figure 37: Frame 1, 1000, 2000, 3000, 4000 and 8000 of the compressibility test for 20 by 20 particles

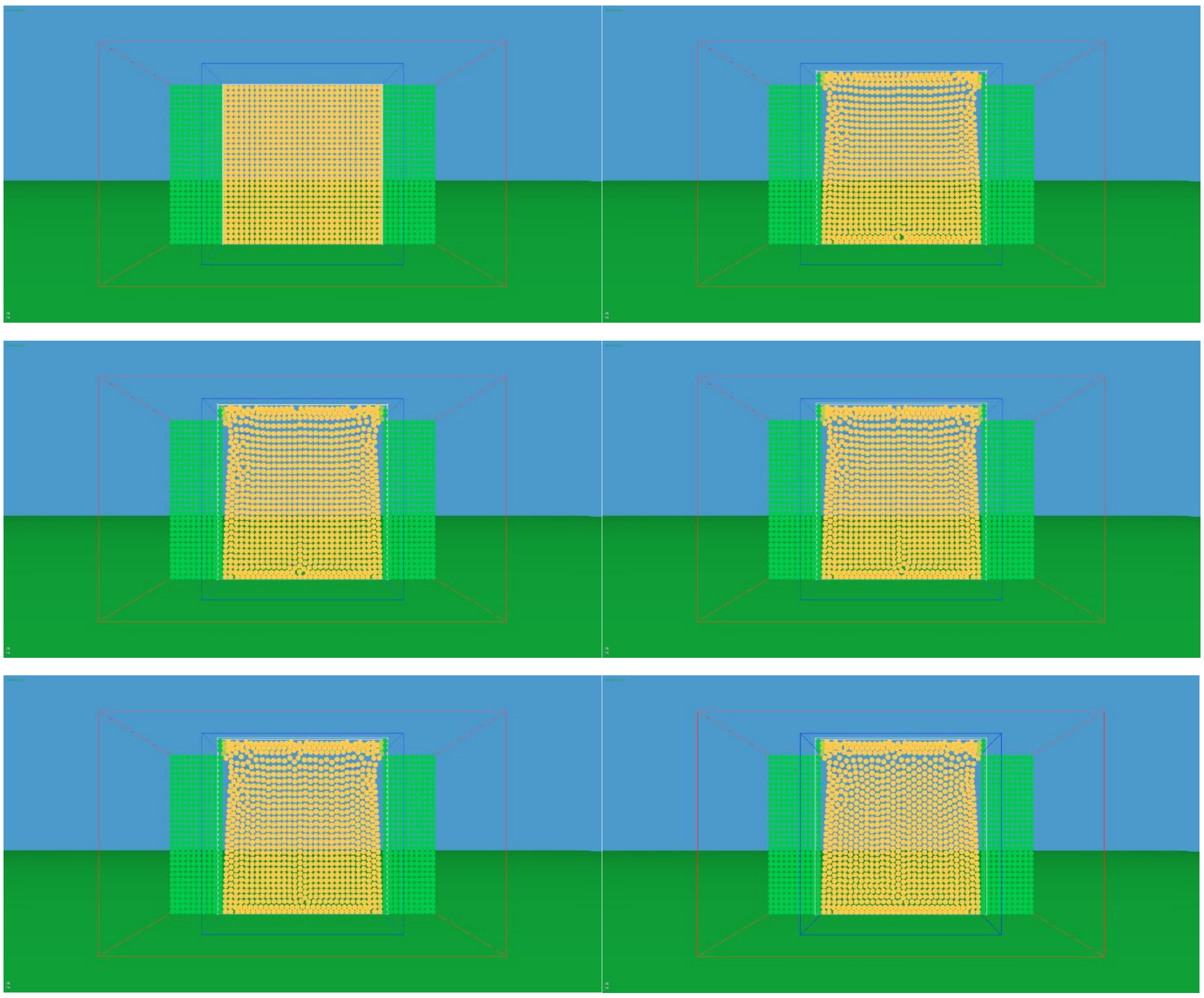


Figure 38: Frame 1, 1000, 2000, 3000, 4000 and 8000 of the compressibility test for 30 by 30 particles

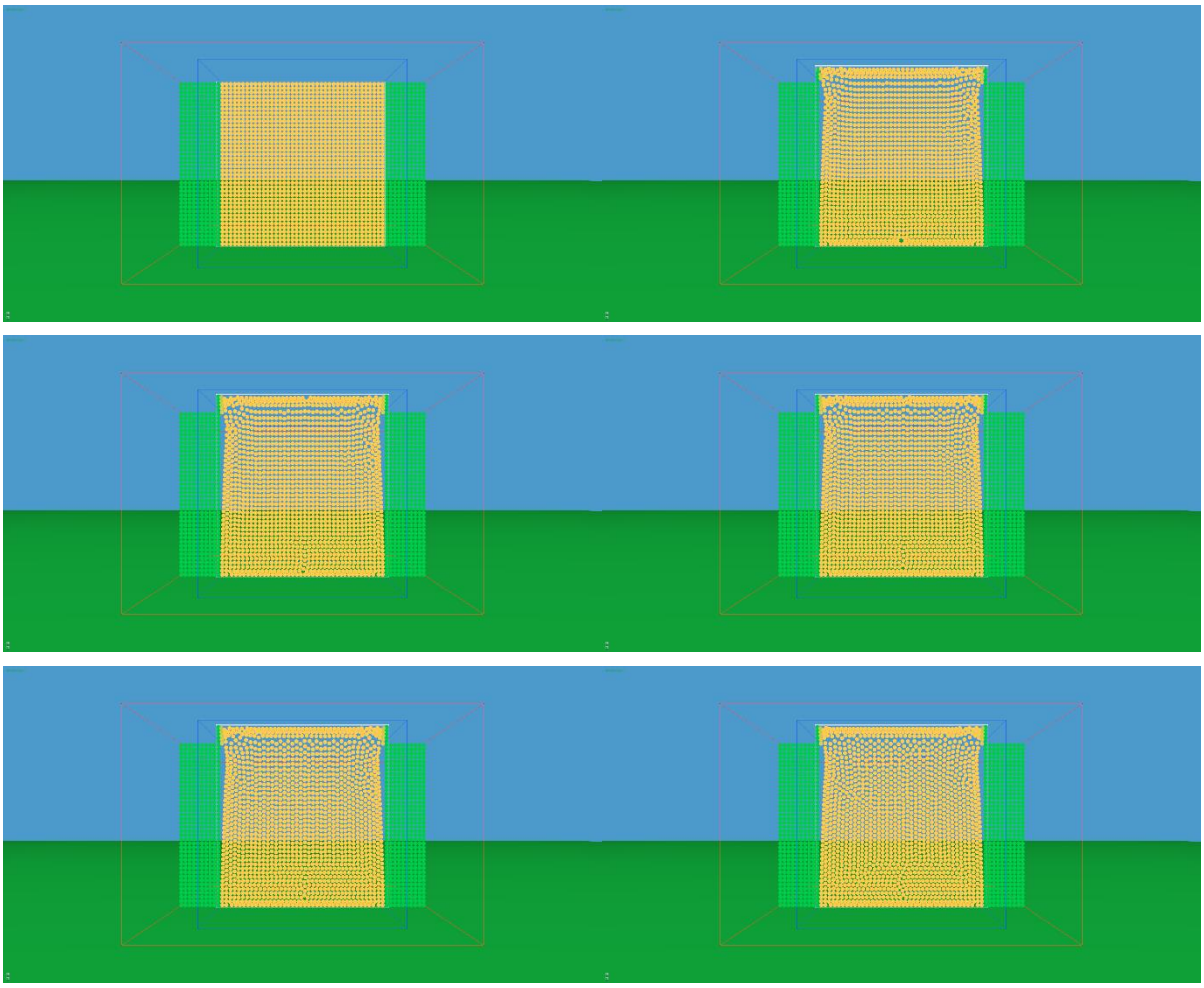
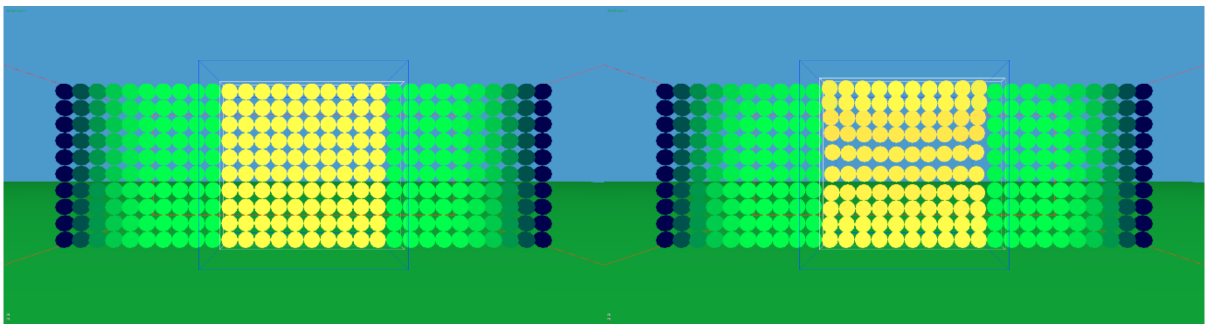


Figure 39: Frame 1, 1000, 2000, 3000, 4000 and 8000 of the compressibility test for 40 by 40 particles

A 2. Density Distribution Test 2D



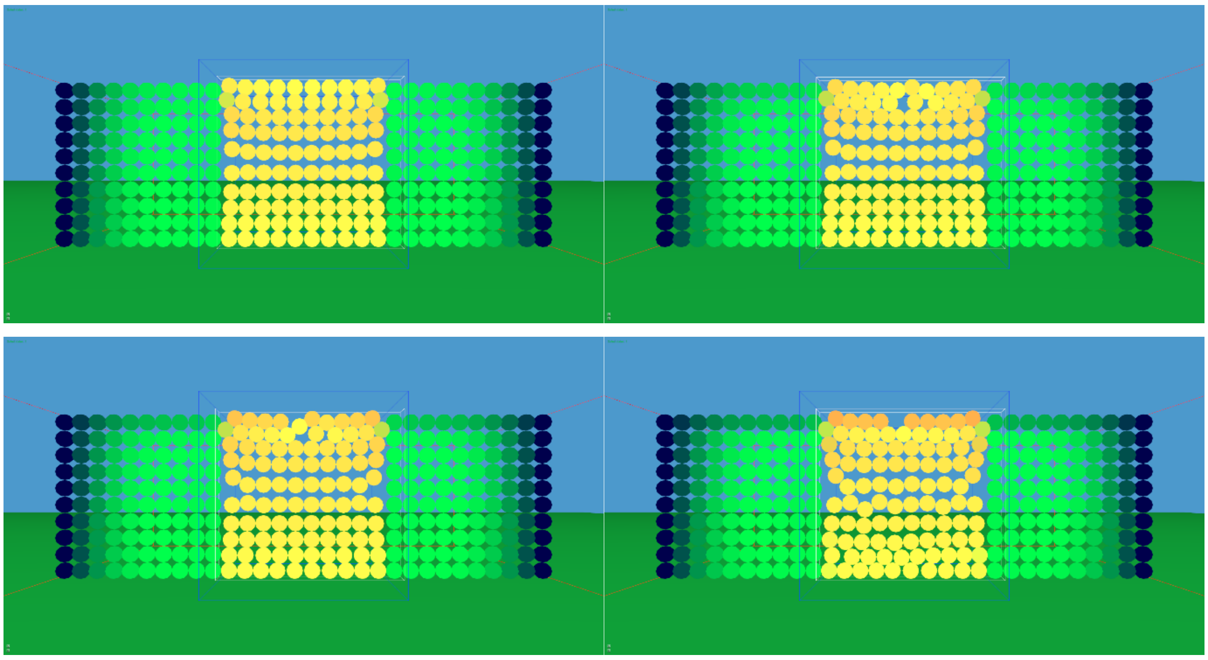


Figure 40: Density distribution for 10 by 10 particles at frame 1, 1000, 2000, 3000, 4000 and 8000

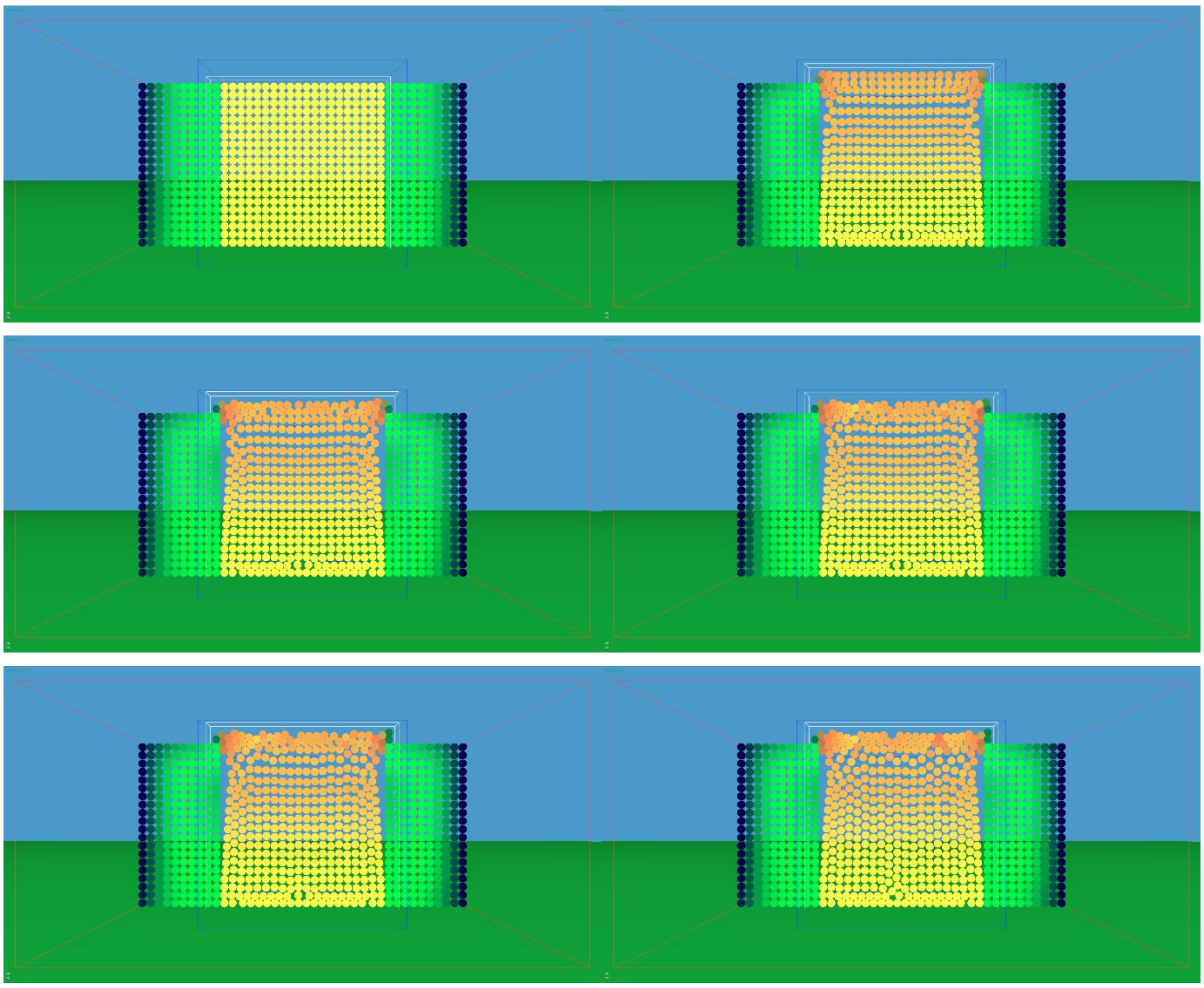


Figure 41: Density distribution for 20 by 20 particles at frame 1, 1000, 2000, 3000, 4000 and 8000

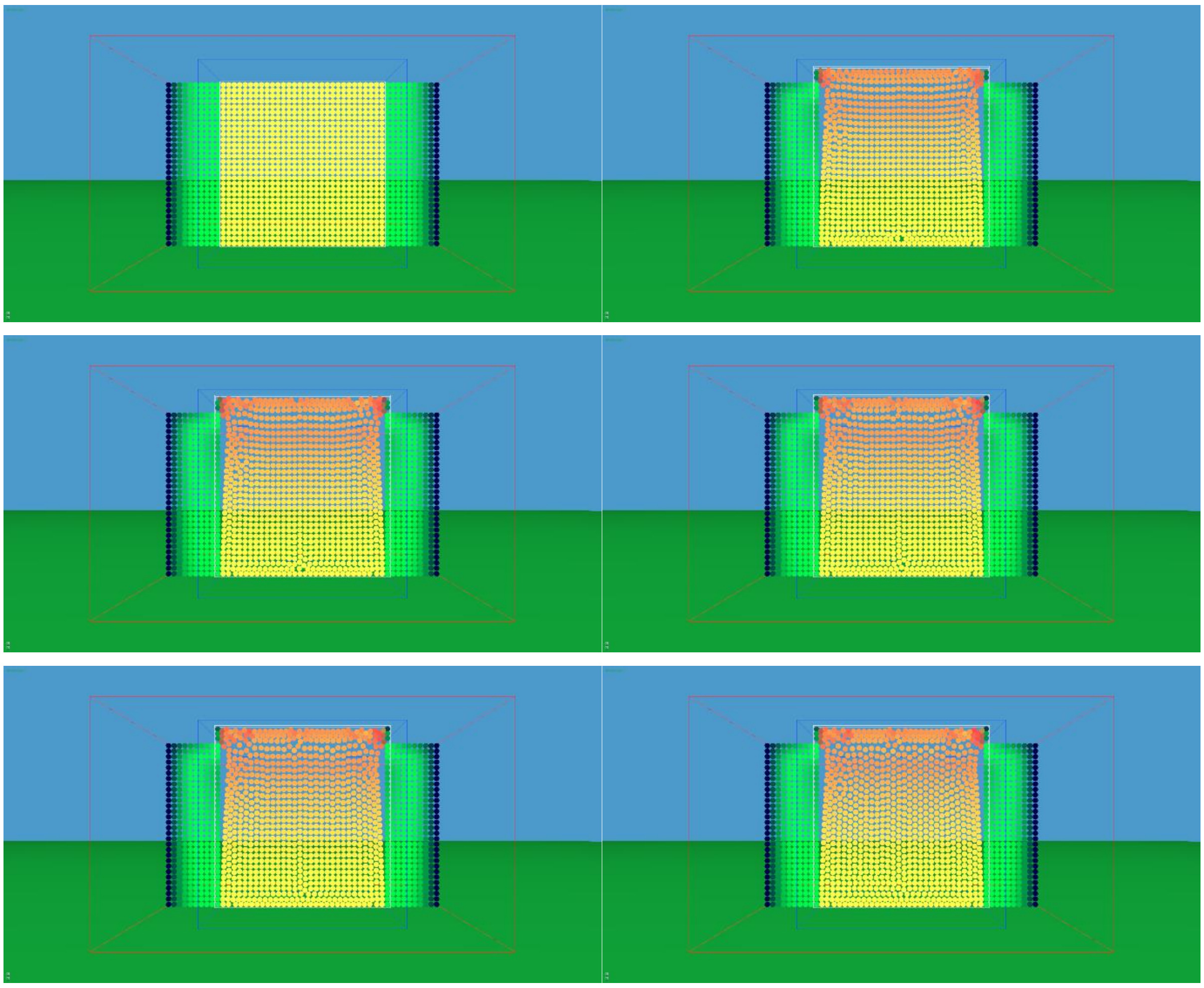
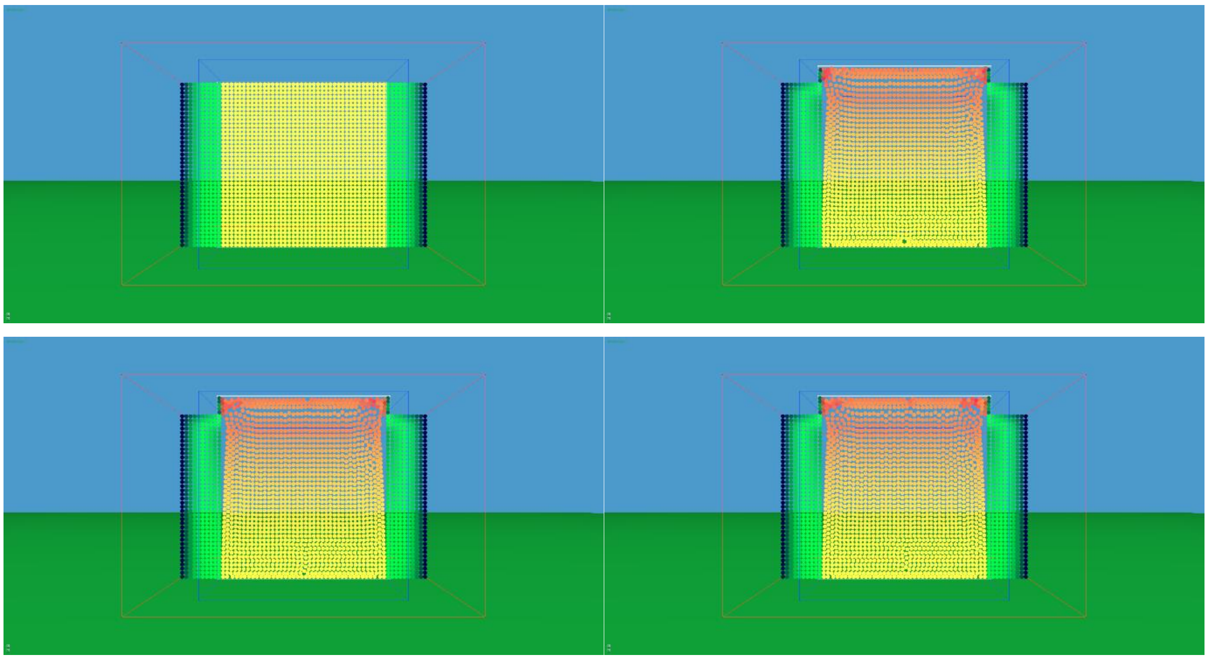


Figure 42: Density distribution for 30 by 30 particles at frame 1, 1000, 2000, 3000, 4000 and 8000



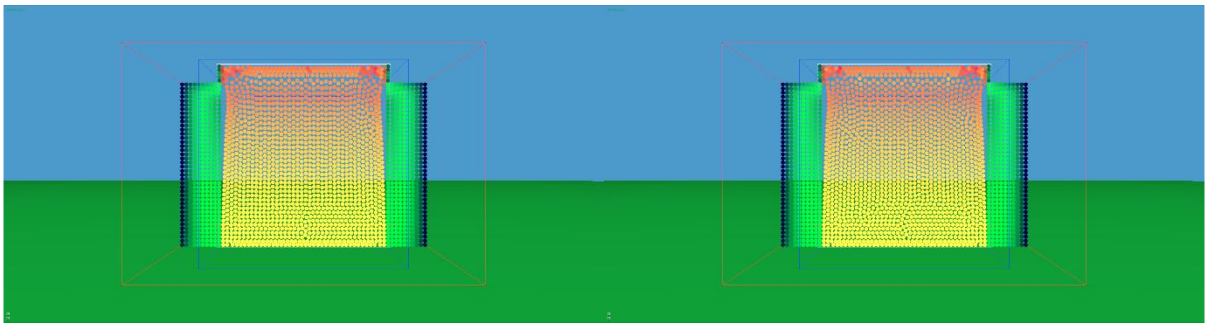


Figure 43: Density distribution for 40 by 40 particles at frame 1, 1000, 2000, 3000, 4000 and 8000