A Generic Particle Modeling Library for Fluid Simulation

An Object Oriented Approach Based on the Physolator Framework

Dirk Eisenbiegler
University of Furtwangen, Germany

Abstract—**Particle Modeling is a frequently used technique for fluid simulation. Different kinds of implementations of such simulations have been presented in the last decades. In order to achieve good results as to accuracy and computing performance, one has to find appropriate formulas and parameters for the particles and one has to find efficient computer programs implementing the simulation. This paper presents a generic and reusable particle modeling software library for such simulations. The approach is based on the Physolator – an object oriented physical simulation framework.**

Keywords-particle modeling; fluid simulation; Physolator; Java framework; object oriented modeling

I. Introduction

Particle modeling is sometimes also referred to as quasi-molecular simulation. Simulating the behavior of a fluid molecule by molecule would be too costly as to the calculation time. Therefore, particles are used as a substitute for molecules. These particles are virtual. They do not exist in real world. Each particle represents a small piece of fluid. Also a molecule inside a fluid represents a small piece of a fluid. In this sense particles resemble molecules. By definition, particles shall be far bigger portions of fluids than molecules. For a given amount of fluid, the number of particles is far smaller than the number of molecules. The size of the particles can be arbitrarily chosen. Smaller particles lead to more precise simulation results, but significantly increase the computing effort. In a typical simulation scenario a fluid is represented by some thousand particles. With such a number of elements, a numerical simulation can still executed in a reasonable amount of time.

In the domain of fluid simulation, there are different kinds of applications and lots of different simulation programs have already been developed. However, there are quite some recurring tasks: defining the particle behavior via appropriate formulas, finding the right particle parameters especially the particle size, computing the equilibrium particle distance in a grid, defining the initial state of a fluid by placing particles in a grid with equilibrium distances, finding efficient programs for simulation, implementing visual components for representing the simulation results.

II. Generic Library for Fluid Simulation

The work presented in this paper aims at building a generic and reusable particle modeling library. The library is based on the Physolator framework [5,6]. Physolator is a physical simulation framework based on the Java programming language. Physolator supports an object oriented development style. Physical components, graphical components and numerical procedures can be developed independently and can be connected during run time. This paper presents a library of components for fluid simulation consisting of physical and graphical components. Building your own physical simulation based on this library is easy: define your own particle behavior by overwriting the appropriate methods, define your own particle parameters, use predefined programs for placing the particles, run the simulation inside the Physolator. This paper does not list all the classes and interfaces of the particle modeling library. Instead it presents the core concepts and explains the main features of the library by an example.

Particle modeling techniques have to consistent and effective at the same time. Consistent means, that the particle based fluid simulation produces results that are consistent with macroscopic fluid physics. Fluid simulation requires a lot of computing effort. An important parameter of every particle simulation is the "size" of the particle – in other words: its mass. For sake of smaller amounts of particles, fluid simulation is often performed in a two dimensional world rather in a three dimensional world (see [9,10]). The program code of the particle modeling library for fluid simulation so far only supports 2D simulations. The concepts used for two dimensional 2D and 3D simulations are pretty much the same. The particle modeling library shall also support 3D simulations in a future version.

III. Attraction and Repulsion between Particles

Particles shall behave like molecules: repulsion, when two molecules are too close to one another, attraction at bigger distances and the attraction converging to 0 as the distance approaches infinity. The attraction and the repulsion between molecules can approximately be described by the Lennard-Jones-12-6 potential. In particle modeling usually uses formulas that are similar to the Lennard-Jones-12-6 formula.

Given two particles with distance *r*. $\phi(r)$shall describe the energy potential between the two particles. In this library, the following formula shall be used to define $\phi(r)$.

$$\phi(r) = -4\epsilon\frac{\sigma^2}{r^2} + 4\epsilon\frac{\sigma^4}{r^4}$$

This formula has been introduced by Greenspan [1] and has been adapted by many others. The Greenspan formula has two parameters $\sigma$and ϵ. In order to give a fluid of some material the right physical behavior, one has to find appropriate values for $\sigma$and ϵ. The following sections will explain, how one finds appropriate values for $\sigma$and ϵ.

It should be noted, that this formula was defined in a pragmatic way. The formula is designed for simulation. It has not been derived from any real world physical formula. The formula just tries to make sure, that the particles act similar to molecules.

Besides the formula from Greenspan, we will also need its first and its second derivative with respect to r. $F(r)$ represents the force between two particles and $D(r)$ represents the stiffness

$$.\ F(r) = \frac{d\phi(r)}{dr} = 8\epsilon\frac{\sigma^2}{r^3} - 16\epsilon\frac{\sigma^4}{r^5}$$

$$D(r) = \frac{dF(r)}{dr} = -24\epsilon\frac{\sigma^2}{r^4} + 80\epsilon\frac{\sigma^4}{r^6}$$

IV. Particle Grids, Mass Density and σ

With the help of σone defines the size of the particles. $\sigma$is arbitrarily chosen. Given a physical system with two particles. Then $r_0 = \sqrt{2}\sigma$is the equilibrium distance between the particles. With parameter σone defines, how fine-grained the physical system shall be. For a given amount of fluid, a smaller value for σresults in a bigger number of particles and thus the computing effort during simulation is increased.

In particle modeling a certain amount of a fluid is represented by a certain number of particles. A big number of particles in an equilibrium state builds a grid. In a two dimensional world, an infinite number of particles automatically builds a hexagonal grid. Let $r_2$be the distance of two neighboring particles in a two dimensional hexagonal grid in an equilibrium state. The particle modeling library uses interval switching to compute $r_2$for given values of $\sigma$and ϵ.

Let us assume, that the mass density $\rho$of the material is well known. Let us assume, that the hexagonal grid is one layer of a three dimensional hexagonal highest density close packing. Then the mass for a single particle can be derived from $\rho$and $r_2$as follows.

$$m = \frac{\rho {r_2}^3}{\sqrt{2}}$$

V. Finding an Appropriate Value for ϵ

$\epsilon$is a material specific parameter that defines the stiffness. Let us assume, that the Lennard-Jones-12-6 potential of the material is already well known. In this case one can postulate that for a given amount of fluid the binding energy in the particle world shall be as big as the binding energy in the molecular world. More precisely: the binding energy of two particles in equilibrium state shall be as big as the binding energy of n pairs of molecules in equilibrium state, where the mass of the n pairs of molecules equals the mass of the pair of particles. This defines ϵin an unambiguous manner.

VI. Particles with Different Materials

The formulas presented so far make the assumption that all particles are of the same kind. For simulations with different kinds of material, one has to provide different kinds of particles with different parameters σ, ϵ and m. In such a physical model one also has to consider forces between different kinds of particles. Given a particle A with parameters $\sigma_A$and $\epsilon_A$and a particle B with parameters $\sigma_B$and ϵ_B. The force $F(r)$ between these two particles and as well as $\phi(r)$ and $D(r)$ are computed with the given equations using the following parameters $\sigma_{AB}$and ϵ_{AB}.

$$\epsilon_{AB} = \sqrt{\epsilon_A\epsilon_B}$$

VII. Damping

A fluid internal movement results in damping. As soon as a particle is moving with respect to the surrounding particles this leads to a resistance force F_w. $F_w$is directed opposite to this movement and depends on the fluid's mass density $\rho$and on the fluid's viscosity η.

Let $v_f$be the velocity of some particle with respect to the surrounding fluid. $v_f$is the difference between the total particle velocity of the particle and the total velocity of the surrounding liquid. The total speed of the surrounding fluid is approximated by the weighted average of the neighboring particles. Neighboring means, that one defines a reasonable radius and all particles with a distance less than this radius considered to be neighbors. $2r_2$is a good choice for such a radius. In this approach it is assumed that the particle behaves like a ball with diameter of $r_2$that is move through the fluid with a velocity of v_f. For such a ball the fluid resistance force is$F_w = \frac{1}{2}c_w A\rho {v_f}^2$

with $A = \frac{1}{4}{r_2}^2\pi$. Variable $c_w$is the drag coefficient. In a simple approximation one may consider $c_w$to be a constant with $c_w = 0.45$. More precisely, $c_w$depends on the viscosity η.

VIII. External Forces

The particle-particle forces described by the Greenspan formula and the viscosity are fluid internal forces. Besides, there may also be different kinds of external forces applying to the particles inside the fluid.

Here are some examples for external forces. The particles inside a fluid may be attracted by the gravitational forces from external bodies. Just like a water drops dropping down. The particles inside the water drop are attracted by earth. If a liquid is in touch with rigid bodies, then the liquid is repulsed by the right body. Example: A water drop rests on top a rigid surface. The rigid surface exerts an upward force to the liquid hindering it from falling. If the liquid is electrically loaded, then forces from electrical fields apply. These are just some examples for external force. Many other kinds of external forces may have to be considered.

IX. Rigid Body Particles

In the particle modeling library one can use particles not only for modeling fluids but also for modeling rigid bodies. The following picture shows a snapshot of a simulation with a water drop falling from a ceiling plate. The plate is a rigid body. It is represented by a set of graphite particles. Just like fluid particles, rigid body particles do apply attractive and repulsive forces to their neighboring particles. The major difference: other particles do not apply forces to rigid body particles.

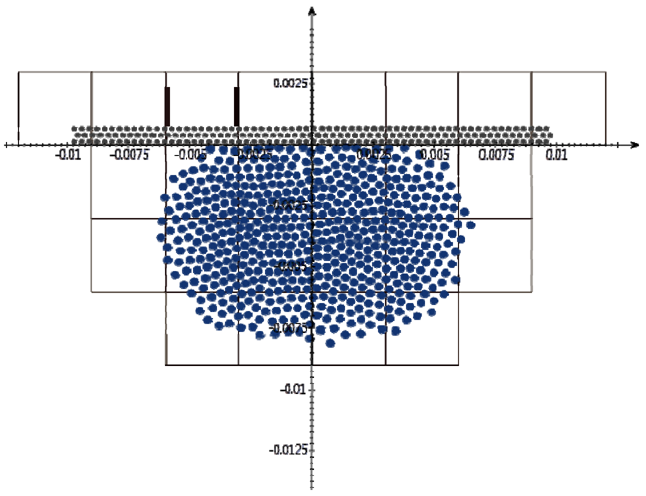

FIGURE I.

X. THE GRID OF BOXES

The particle modeling library supports a mechanism for building an initial setting of a particle system. The physical system assigns a set of particle – fluid particles and rigid body particles – to their initial state. Then the simulation is run. During every simulation step the forces between all pairs of particle has to be computed. With one exception: The forces between rigid body particles need not be computed.

With an increasing number of fluid particles n the computation effort for computing the particle-particle forces grows with n^2. To reduce the computing cost, a limit r_{max}shall be defined. Forces between particles with a distance greater than r_{max}shall be skipped. Only the forces of particles that are close to one another have a significant influence on the simulation result. A well chosen value of r_{max}can very much save computation effort during simulation without significantly reducing accuracy.

The fluid library internally uses a grid of boxes (see figure I). The grid of boxes is a data structure based on hash table technique. This data structure is used for efficiently determining the neighbors for every particle, i. e. the particles with a distances smaller than or equal to r_{max}. Each box is a squared area with an edge length of r_{max}. In the beginning of every simulation step, every particle is assigned to one of the boxes. The particles with a distance smaller or greater r_{max}can be found in the same box or in one of the 8 boxes at the rim of this box.

The grid of boxes is also used to determine the fluid resistance force due to viscosity. For this purpose all particles with a distance smaller or equal to $2r_2$shall be considered. To find these neighboring particles with a distance smaller or equal to $2r_2$, the grid of boxes shall be used. One has to make sure, that the $r_{max}$is greater than $2r_2$. Usually, $r_{max}$is far greater than $2r_2$ to achieve a reasonable accuracy when computing the particle-particle forces. In the example of figure I, r_{max} is roughly ten times the size of $2r_2$.

XI. SIMULATION STEP WIDTH

Before starting a simulation run one has to define a reasonable step width. A smaller step width leads to more precise results. However, for a simulation run with a given time span smaller step widths lead to a bigger number of simulation steps.

According to the sampling theorem, the frequency of simulation steps must be at least $2f$ where f is the natural frequency. The frequencies occurring inside a particle system depend on the particle sizes and the stiffness. Let us consider a simple physical system with two particles of the same kind. Let us assume that the distance between the two particles is a little bit more than the equilibrium distance $r_0 = \sqrt{2}\sigma$. This system would oscillate with the following frequency.

$$f_0 = \sqrt{\frac{D(r_o)}{8\pi^2 m}}$$

This formula can be used to a get a rough approximation for the frequencies occurring inside a particle model with several particles. In a physical system with different materials one has to deal with particles of different kinds. In this case it is recommended to compute the f_0 value for every material and choose the maximum value. In a grid with a lot of particles, the particles are denser and there are stronger forces than in a system with just two particles. This is why one should better use a simulation frequency of at least $4f_0$rather than $2f_0$.

XII. USING THE PARTICLE MODELING LIBRARY

The particle modeling library is a set of Java classes. It is based on the concepts presented in this paper. The core of this library is a class named *Particle2DSystem*. This class represents a generic physical system consisting of a set of particles. To build your own physical system you have to build a son class of *Particle2DSystem.*

The example program code below defines such a son class named *WaterDropFromCeiling*. This class can be loaded and executed inside the Physolator framework [5,6]. *Particle2DSystem* comes with a built in graphics component for visually representing particle systems (see figure I).

In this physical system there are two kinds of particles: water particles and graphite particles. In the program code the variables *water* and *graphite* are objects of type *Material*. They contain the relevant physical data of the materials water and graphite such as the materials mass density and the materials viscosity.

The variables *waterPS* and *graphitePS* are *Particle2DSchema* values. Objects of Type *Particle2DSchema* are built on top of *Material* objects. They define the behavior of a particle of some material. They contain a link the *Material* object and they contain the particles core parameters: σ, ϵand m. During the initialization of *waterPS* and *graphitePS* these core particle parameters are handed over to the constructor of *Particle2DSchema.* The particle constants used in this program code have been adapted from [10] and converted to SI units. In the program code below, all variables values are expressed in SI units.

Besides the core parameters σ, ϵ and m, *Particle2DSchema* objects also contain some derived parameters such as r_2 and f_2 and they also contain methods implementing the physical formulas $\phi(r)$, $F(r)$ and $D(r)$ as well as a method for computing the friction force due to viscosity.

```
import static java.lang.Math.sqrt;
import static particles.components2d.Particle2DGridGenerator.*;
import static particles.components2d.MaterialCatalogue.*;
import particles.components2d.*;
importde.physolator.usr.SimulationParameters;
importde.physolator.usr.num.IterativeMethodCatalogue;

public class WaterDropFromCeiling extends Particle2DSystem {

   public final double rmax = 0.003;

   public Particle2DSchema waterPS = new Particle2DSchema(
      water, 4.325669e-4, 2.810376e-2, 1.722469e-7, rmax) {
         public void addExternalAcceleration(Particle2D p) {
            p.a.y = p.a.y - 9.81;
         }};

   public Particle2DSchema graphitePS = new Particle2DSchema(
      graphite, 2.711047e-4, 2.96284e-4, 1.5544282e-8, rmax);

   publicWaterDropFromCeiling() {
      initSystem(rmax);
      final double radius = 2.1e-3;
      addRectangle(containerFixed, graphitePS,
         -radius, 0, radius, 0.5 * sqrt(3) * graphitePS.r2);
      addCircle(containerMovable, waterPS,
         0, -0.25 * 0.25 * sqrt(3) * waterPS.r2, radius,
         (x, y) -> y <= -0.5 * waterPS.r2);
      finishAddingParticles();
   }

   public void initSimulationParameters(SimulationParameters s) {
      s.realTime = false;
      s.fastMotionFactor = 0.002;
      s.iterationsPerFrame = 1000;
      s.iterativeMethod = IterativeMethodCatalogue.adamsBashforth;
   }
}
```

FIGURE II. EXAMPLE JAVA PROGRAM CODE

In the *waterPS* object the method *addExternalAcceleration* is overwritten. Overwriting this method is useful when you have to consider external forces from outside the fluid. In our example the water particles shall be attracted by earth with an acceleration of $9.81\frac{m}{s^2}$.

The class Particle2DSystem has two particle containers: *containerMovable* and *containerFixed*. They are used for fluid particles and rigid body particles, respectively. In the constructor of the program code, water particles are added to *containerMovable* and graphite particles are added to *containerFixed*. Both containers internally organize their particles in a grid of boxes. The edge length of these boxes is *rmax*. The particles and their initial positions are defined in the constructor. You can assign particles one by one or you can use predefined methods that fill a given area with a hexagonal grid of particles of some kind. *addRectangle* and *addCircle* are such methods. In this example the invocation of *addCircle* produces a semi-circle. This is due to the effect that the last parameter is a filter (a Java lambda term) that restricts the particle locations inside the circle a suitable way – turning the circle into a semi-circle.

Physolator provides different kinds of simulation parameters for controlling the simulation run (see [5,6]). The program code overwrites the *initSimulationParameter* method and there assigns constants to the simulation parameters that are appropriate for this physical system.

XIII. CONCLUSION

This paper has introduced an object oriented and easy to use particle modeling library for simulating fluids. In a future version, the library shall also support three dimensional particle systems. The material library shall be extended. Furthermore, the particle modeling library shall also support an automatic scaling mechanism. In this scenario σ shall be used as the scaling factor. The other two core particle particle parameters ϵ and m shall automatically be derived from σ and the physical parameters provided by the material object.

REFERENCES

[1] D. Greenspan, "Particle Modeling", Birkhäuser Boston, Basel, Berlin, 1997.

[2] D. Greenspan, "N-Body Problems and Models", World Scientific Publishing Co. Pte. Ltd., 2004.

[3] D. Greenspan, "Computer Studies in Particle Modeling of Fluid Phenomena", Mathematical Modeling, Vol. 6, pp 273-294, Pergamon Press Ltd., 1985.

[4] M. J. P. Nijmeijer et al., "Molecular Dynamics of the Surface Tension of a Drop", The Journal of Chemical Physics, vol. 96, no. 1, pp. 565-576, 1992

[5] D. Eisenbiegler, "The Software Architecture of the Physolator–a Physical Simulation Framework", MSAM 2015, Atlantis Press, pp. 61-64.

[6] D. Eisenbiegler, "Objektorientierte Modellierung und Simulation physikalischer Systeme mit dem Physolator", BoD Norderstedt, 2015.

[7] D. Eisenbiegler, "Physolator – Getting Started", Video Tutorial, http://www.physolator.de/joomla/index.php/en/manual#GettingStarted.

[8] D. Eisenbiegler, "Physolator Programming", Video Tutorial, http://www.physolator.de/joomla/index.php/en/manual#PhysolatorProgramming.

[9] M. Nagel, "Quasi-molecular Modeling of Waterdrops", Bachelor Thesis, University of Greifswald, Germany.

[10] M. S. Korlie, "Particle Modeling of Liquid Drop Formation on a Solid Surface in 3-D", Elsevier Science Ltd, Computers Math. Applic. Vol. 33, No. 9, pp. 97-114, 1997.

[11] M. S. Korlie, "Three-Dimensional Computer Simulation of Liquid Drop Evaporation", Computers & Mathematics with applications, Elsevier, 1999