Volume 1 (2021), Number 1, pp. 1-7

Animating Viscous Fluid using a Globular Particle System

Leevon Levasseur

V00868326

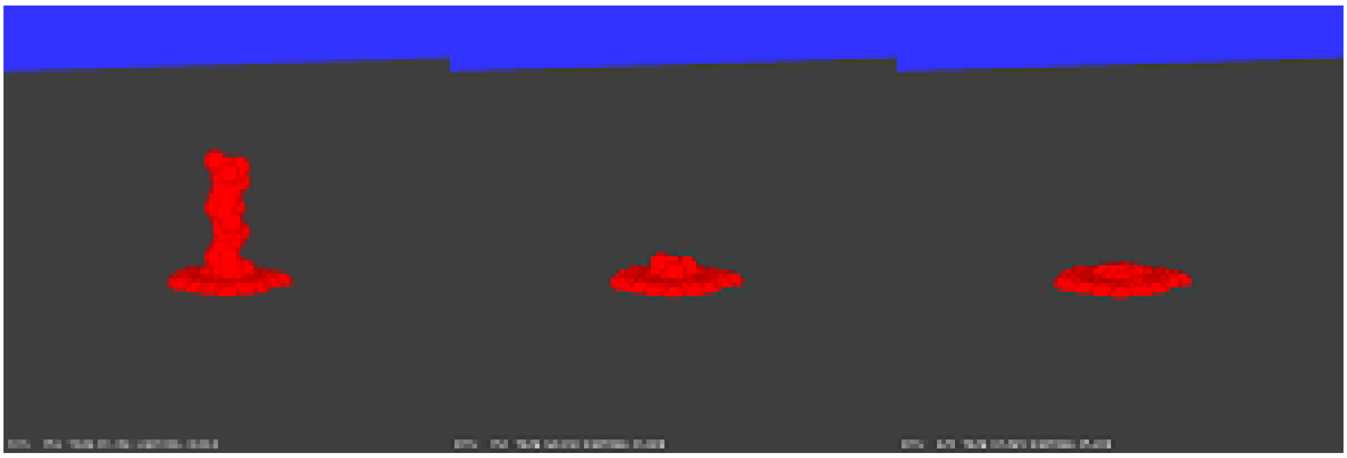

Figure 1: *Frames from liquid animation with best results (material constant $C_d = 1.5r_0$).*

Abstract

This paper describes a method for modelling and animating viscous fluid using a physically-based particle simulator and soft collisions. This paper was inspired by the work presented by Gavin Miller & Andrew Pearse in Globular Dynamics: A connected particle system for animating viscous fluids. Using the same fundamental theory, this paper delves into the decisions for variable values and results that Miller & Pearse neglect. Results are expanded to show how each variable influences the animation. Specifically, how to make certain materials related to fluid. The simulation was constructed using C++ and AnimTcl, and rendered using OpenGL. A Forward Euler approach with a small time step and elastic penalty collisions were used to determine the next position of each particle. The results of this paper are customizable by the user to form material that looks like powder, liquid, or solid.

1. Introduction

Fluid animation is a complex field of animation that demands extensive and expensive computations, including the use of the Naiver Stokes equation and field flow. Viscous fluid naturally falls in this category of research, but has interesting properties that allow it to be modelled efficiently without such complex solutions. This paper delves into the process of creating different types of viscous fluid-like material using globular dynamics. This paper presents a customizable, physically-based particle system to model viscous fluid like lava, mud, ketchup, putty, and even powder-like materials like salt or sand. By using non-rigid "blobby" molecules as the particles, soft collisions are used to stick or shear particles based on inter-globule force calculations. Repulsion, attraction, and drag combined to give the material a sense of flow. The simulation uses a Forward Euler approach to calculate particle positions and penalty force calculations for collisions with an effector. The system and simulator was developed in C++ and Animtcl. OpenGL was used to render the objects. Trials were performed on a machine with an Nvidia RTX 2070 Super GPU, Ryzen 5 3600xt CPU @ 3.79Hz, 16 gigabytes of RAM, and Windows 10 OS.

1.1. Related Work

This paper was inspired by the paper titled Globular Dynamics: A Connected Particle System for Animating Viscous Fluids by Gavin Miller and Andrew Pearce, written in 1989. Their paper provides fundamental theory regarding globular dynamics and what results to expect. This paper differs from theirs by focusing on the results of customization when performing inter-globule calculations, and explaining aspects that are not clear for users wishing to implement themselves.

The idea of using blobby particles to simulate viscous fluid stems from the idea of soft body deformation over time [B. Wyvill et al, 1986] and inter-particle reactions based on the distance, such as discussed in *flocks* [C. W. Reynolds, 1987]. The influence each particle experiences from other particles is the primary force that dictates the animation.

2. Overview

Globular dynamics defines the relationship between particles existing in a multi-particle system. A particle in this system is referred to as a "globule" or "glob" throughout this paper. Globules are represented as soft iso-spheres with a center of mass in Cartiesian space, an exterior defined by its radius, and an inital velocity. Globules are soft bodies, in other words: non-rigid. Because of this, globules experience soft collisions, meaning they can overlap with each other. Constraints can be set to customize collisions so globules can repel away from each other or stick together. Clusters of globules that stick together appear as one substance or material. This is crucial to dictate liquid flowing over itself or a solid holding its shape.

In this paper, colour and temperature are omitted from the calculation and render but could be set to create more customizable render outcomes. For example, the user might want to simulate foam or spray from a liquid, or change the type of material over time due to temperature (freezing or smelting). This is past the scope of this paper but will be investigated in future work.

2.1. Inter-Globule Force

Globules who collide with one another must react correctly based on the type of material the user defines. There is a relationship between globules based on their distance from one another. This inter-globule force can be summed over all particles in the system and used with Newton's Second Law to determine the center position of each individual globule. Over a time step t, a double integration of the sum of all forces exerted on a particle **p** can be computed to determine where the next position will be:

$$x_p = \frac{1}{M_p} \int\int f_{ext} + \sum_{i=1}^{N} f_{ip}\, dt\, dt \qquad (1)$$

where M_p is the mass of particle **p**, f_{ext} is the sum of any external force being acted on particle **p**, and f_{ip} is the inter-globule force particle **i** exerts on particle **p**. It can be expressed as:

$$f_{ip} = \begin{cases} \overline{P}_\Delta [S_r(\frac{b_1}{D^m} - \frac{b_2}{D^n}) - S_d(\frac{\overline{V}_\Delta \cdot \overline{P}_\Delta}{D^2})] & if\ i \neq p \\ 0 & if\ i = p \end{cases} \qquad (2)$$

where $\overline{P}_\Delta$ is the vector difference between the center of particle **p** and particle **i**, $\overline{V}_\Delta$ is the vector difference in their velocities, and D is the scalar distance between them. Within the square brackets in (2) are two terms: the repulsion/attraction term, and the drag term. Both terms depend on the scalar distance between the particles, D. The repulsion/attraction term is defined by constants b_1, m, n, which are defined by the user, and b_2 defined as:

$$b_2 = b_1\, r_0^{n-m} \qquad (3)$$

where r_0 is the distance between globules where the repulsion and attraction cancel out exactly. This is also defined by the user. It is recommended to choose an r_0 that is reasonable based on the globules that are interacting. For example, if the globules in the animation have a radius of 1m, to allow overlap through their soft exterior walls when they rest, set $r_0 \in [1,2)$. Terms b_1 scales how strong the repulsion and attraction is, while m and n control how extreme the repulsion and attraction affect the globule. Set $b_1 \in [1,3]$ for stable animations and always set $m > n$ for realistic results. Miller & Pearce use $m = 5$ and $n = 3$ in their paper, which work best for the examples they produce. This paper compares results with different values of m and n. As m increases, the slope of the repulsion diverges faster to infinity as the particles get very close, but increases the overall attraction until this "spike" is reached (Figure 2B). The term n controls the rate of attraction, determining how graduale the attraction influences the globule. If n is small, the attraction slowly pulls in from a larger distance (Figure 2C). If n is large, globules experience attraction quickly or not at all (Figure 2D). The difference between m and n controls how deep the dip from strong attraction to strong repulsion is.

2.2. Repulsion/Attraction and Drag Scaling Factors

The terms S_r and S_d in (2) are the scaling factors for the radial and drag force. They control how much of the specific force globules will experience based on their distance from each other. They are used to define what type of material the user wants. These factors describe the radius of influence globules have on one another. Their values are given by:

$$S_r = \begin{cases} 1 - \frac{D^2}{C_r^2\,(r_i + r_p)^2} & D^2 < C_r^2\,(r_i + r_p)^2 \\ 0 & D^2 \geq C_r^2\,(r_i + r_p)^2 \end{cases} \qquad (4)$$

$$S_d = \begin{cases} 1 - \frac{D^2}{C_d^2\,(r_i + r_p)^2} & D^2 < C_d^2\,(r_i + r_p)^2 \\ 0 & D^2 \geq C_d^2\,(r_i + r_p)^2 \end{cases} \qquad (5)$$

where r_i and r_p are the radii of particles **i** and **p** respectively and C_r and C_d are defined as the material constants. Notice how these scaling factors result in a positive value between [0, 1]. The amount of computations needed for (2) is reduced when one of these scaling factors equates to 0, meaning two globules are sufficiently distant from each other. This radius of influence is defined by the conditional statements depending on terms C_r and C_d. These terms define how "sticky" the globules are and are based on the user provided r_0. When $D^2 = C_r^2\,(r_i + r_p)^2$ the force in the radial force equals 0. The same is true for the drag force with C_d^2.

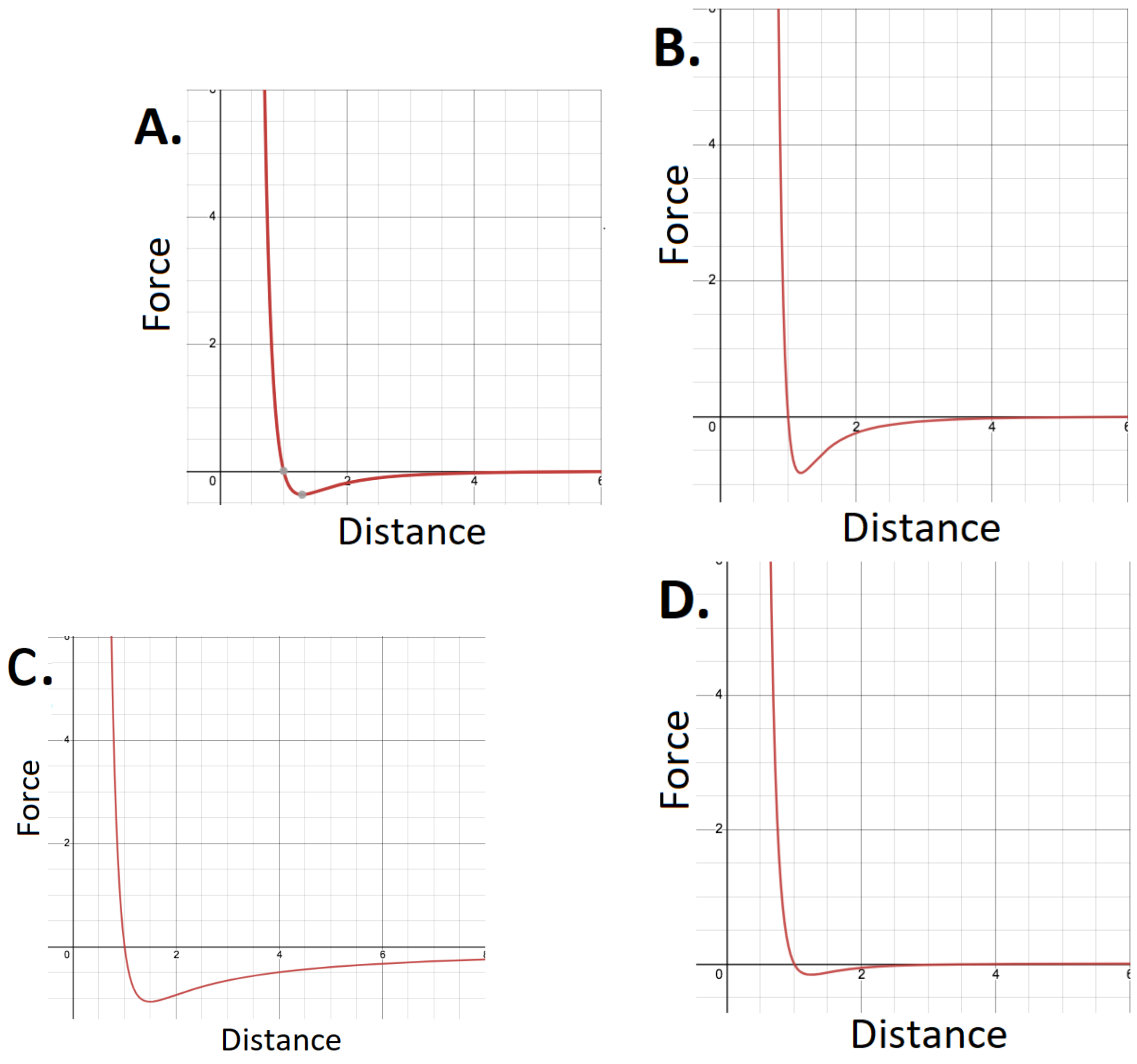

Figure 2: *Force distribution for replusion/attraction with* ***A:*** *(Recommended)* $m = 5$, $n = 3$ ***B:*** $m = 10$, $n = 3$ ***C:*** $m = 5$, $n = 1$ ***D:*** $m = 5$, $n = 4$

2.2.1. Powder:

To model powder-like motion, the radial and damping forces are equally attenuated so damping only occurs when the globules are under compression. Terms C_r and C_d are set to:

$$C_r = C_d = r_0 \tag{6}$$

An inelastic collision occurs when globules collide, causing the globules to bounce off each other and spill away. This result models the effect of pouring sand or salt, as they do not clump while there is available space.

2.2.2. Liquid

To model viscous-fluid, the globules must continue to spill away but the motion must be damped, while also allowing external forces to pull apart the globules easily. Terms C_r and C_d are set to:

$$C_r = r_0 \qquad C_d = \alpha r_0 \tag{7}$$

Where $\alpha \in [1.5, 2]$. Miller & Pearse use $\alpha = 2$, but by changing this value between 1.5 and 2, results demonstrate differing viscosity. $\alpha = 2$ results in quite viscous fluid that clumps easily, like ketchup, while $\alpha = 1.5$ creates liquid that spills out faster and settles in the center, like cooking oil.

2.2.3. Solid

To model a solid-like substance, the attraction term is utilized by increasing the radius of influence for the radial force, while also having a strong damping force. As discussed when explaining the repulsion/attraction term in (2), as the distance increases between particles, the attraction term pulls the globules together if m and n

are set to a ratio that accommodates that. Terms C_r and C_d are set to:

$$C_r = C_d = 2r_0 \tag{8}$$

With a larger C_r term, the globules will pull closer to each other from farther away and converge to a solid with little relative movement. The solid can shear or tear if an external force is applied that outweighs the strength of attraction.

2.3. External Force

Any external force can be applied to a globule to force it away from others. This paper tests with an external force of gravity and an elastic penalty collision with the ground plane, applied only in the y-direction. The penalty force applied from the frictionless-ground plane is:

$$F_N = -k_s\,[\,(x_p - p_g\,)\cdot N_g\,]\,N_g - k_d\,(\,V_p \cdot N_g\,)\,N_g \tag{9}$$

Where k_s and k_d are the spring constants for stiffness and damping for the ground, set by the user. p_g is the point on the ground plane where the collision occurs, V_p is the velocity of the particle $\mathbf{p}$ at that time t, and N_g is the normal of the ground plane, set to the y-direction for this example. In future work, external forces like collisions from projectiles and down force from an effector, like a knife, will be added to show more detail regarding characterizations of the material.

3. Evaluation (Simulation)

The animation that was compared for each material was simply pouring globules onto a frictionless-ground plane and seeing how they react. It was expected that powder-like globules collide and bounce away from each other, liquid globules spill with a damping force, and solids cluster and stabilize. Experiments with tightening the pour were performed by altering the recommended variable values. Viscosity for liquid was tested using differing values for C_d from (5). A Forward Euler method was used to calculate the next position of a particle, but a very small timestep was required to avoid exploding animations. The performance of the simulation depended on the number of particles used, decreasing at an exponential rate as particles increased.

3.1. Initialization

Liquid is dense and should require many globules to model it properly. A cluster of globules pouring from a spout in a flow due to gravity was modelled. To initialize the particles, an AnimTcl script controls the number of particles, N, in the system, and stacks these particles in a helix formation (Figure 3). This helix formation was used to initialize particles in a semi-random and clustered state. It is important to not initialize globules that experience inter-globule force too close to each other as they will experience a sharp repulsion force immediately as the simulation begins and explode away from each other. It is also important to initialize globules with noise in their x/z components if they are only being influenced by the force in the y-direction. Otherwise, when two globules fall on top of each other who have the same x/z components, they will perfectly bounce off or rest upon each other, rather than spill down a side. Their y position was set depending on their index in the list of particles, and their x/z position was set using sine/cosine functions on their index in the list of particles, to form a unit circle. To clump the particles together, the particles were split into four groups that all spiraled around the unit circle. Their y component was offset so all groups would start on the same plane, and their x/z components were offset by a factor of $\frac{\pi}{2}$ to start at different locations around the unit circle. Finally, it is important to note that the initial distance between particles should depend on what r_0, m, and n are initialized to. With unreasonable r_0 values or extreme m or n values, the helix must be expanded to avoid exploding particles.

The simulator is initialized with initial values for r_0, m, n, and

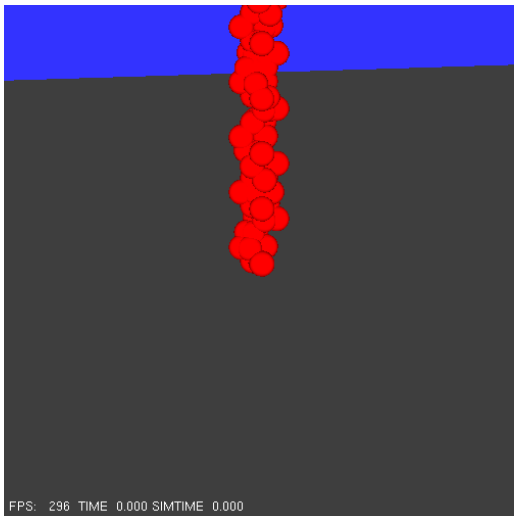

Figure 3: *100 Globules initialized in a helix formation.*

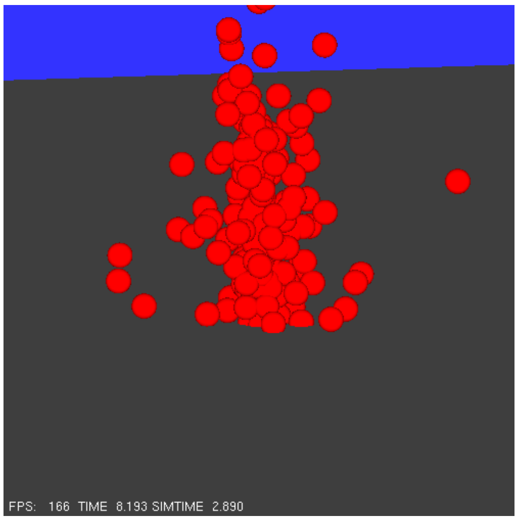

Figure 4: *Globules explode away from each other if the timestep is too large. This figure shows 200 globules that should fall as a powder, but recoil away from each other due to a timestep of 0.01.*

Variable Values	Results compared to Best Case
$r_0 = 2.0$, $b_1 = 2.0$, $m = 5$, $n = 3$	Solid-like model. Not enough repulsion due to inability to get close to each other, resulting in clumping.
$r_0 = 1.0$, $b_1 = 1.0$, $m = 5$, $n = 3$	Solid-like model. Strength of radial force diminished, causing globules to clump.
$r_0 = 1.0$, $b_1 = 2.0$, $m = 7$, $n = 5$	Explode apart. Globules experience extreme repulsion forces from farther apart, less attraction.
$r_0 = 1.0$, $b_1 = 2.0$, $m = 5$, $n = 1$	Liquid-like model. Globules tighten together as they drop and spill out slower than best case.

Table 1: *Comparing Powder Animations*

Variable Values	Results compared to Best Case
$r_0 = 2.0$, $b_1 = 1.0$, $m = 5$, $n = 3$	Large non-rigid body. Collisions reach equilibrium too quickly and globules settle on each other's boundaries.
$r_0 = 1.0$, $b_1 = 1.0$, $m = 7$, $n = 5$	Globules spread apart to form a disc. Animation begins the same as the best case, but spills out by the end.
$r_0 = 1.0$, $b_1 = 1.0$, $m = 7$, $n = 1.5$	Tighter pour, but the model collapses on itself. Due to large radius of influence, smaller n increase total attraction.
$r_0 = 1.0$, $b_1 = 3.0$, $m = 5$, $n = 1$	Tighter pour with less collapse. Model is more compressible. It is understood increasing b_1 has this effect.

Table 2: *Comparing Solid Animations*

b_1 which are customizable. As well, the material, timestep, ground spring constants, and the force of gravity are defined for the simulator. For all experiments, the timestep $\Delta t = 0.001$, ground spring constants $k_s = 5000$ and $k_d = 300$. A larger timestep produced exploding animations due to particles moving too fast towards each other to compute the repulsion force correctly (Figure 4). The ground spring constants were set to these values from previous experiments with Spring Mass Systems.

3.2. Results - Powder:

For the animation of powder pouring onto the ground plane it was essential to have particles avoid clumping. Rather, they should collide, bounce, and spill away from each other. Figure 5 shows frames from an animation with parameters $r_0 = 1.0$, $C_r = C_d = r_0$, $b_1 = 2.0$, $m = 5$, $n = 3$, which gave the best result. The animation used 200 globules and operated at an average of 120 fps. Experimental animations for powder-like material were tested against this best case (Table 1).

3.3. Results - Solid:

The animation for a solid material must make the globules clump together due to a large radius of influence for attraction and a strong damping force. It was important to get a result that converged to a stable state, meaning the globules would not spill away or collapse into the middle. To achieve the best result, parameters were set to $r_0 = 1.0$, $C_r = C_d = 2r_0$, $b_1 = 1.0$, $m = 5$, $n = 3$, as shown by frames of the animation in Figure 6. The animation used 200 globules and operated at an average of 120 fps. Additional animation trials were conducted with different parameters and compared to the best result (Table 2).

3.4. Results - Liquid:

The goal of this paper is to animate a realistic viscous fluid. For liquid material, globules must spill to fill open space while also staying close enough together to simulate a viscous flow. It was discovered that the best parameters for liquid depended on how viscous the user intended the liquid to be. Using $r_0 = 1.0$, $C_r = r_0$, $b_1 = 2.0$, $m = 5$, $n = 3$ were the best values for liquid, and the viscosity was controlled by C_d, where $C_d = 1.5r_0$ was less viscous than $C_d = 2r_0$, as shown by frames of the animation in Figure 1 and Figure 7. When the animation used 400 globules it operated at an average of 60 fps. Further experiments with different parameter values were tested for liquid material and are compared to the best results (Table 3).

4. Conclusion:

Viscous fluid, powder, and solid-like materials were simulated using a C++ physically-based particle simulator with non-rigid particles modelled using iso-spheres in OpenGL. The animation, performed using AnimTcl, initialized a column of globules falling due to gravity in a helix formation. A Forward Euler method was used to calculate the next position of a particle in the system and penalty force was applied when the globules collided with the frictionless ground plane. The globules would interact with each other differently based on what parameters were given for the inter-globule

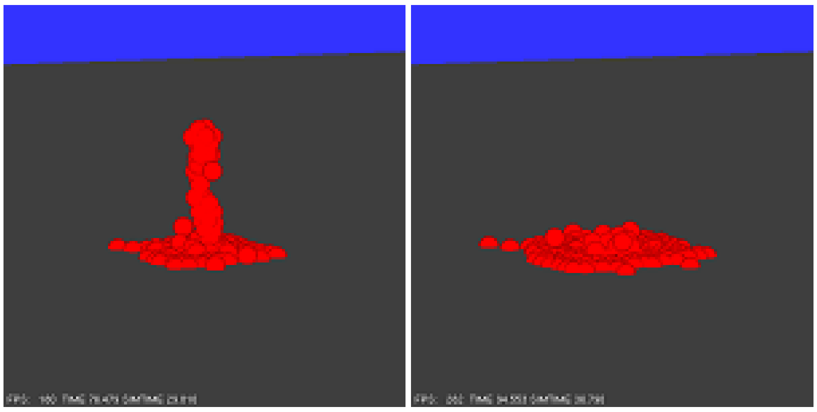

Figure 5: *Frames from powder animation with best results.*

Variable Values	Results compared to Best Case
$r_0 = 2.0$, $b_1 = 2.0$, $m = 5$, $n = 3$	Solid-like model. Little to no spill. Globules conglomerate in center.
$r_0 = 1.0$, $b_1 = 2.0$, $m = 7$, $n = 5$	Model has more splash. Globules slide past easier.
$r_0 = 1.0$, $b_1 = 2.0$, $m = 7$, $n = 1.5$	Tighter pour, but less spill. The flow ends rather abruptly. The model looks like a melted solid.
$r_0 = 1.0$, $b_1 = 3.0$, $m = 5$, $n = 1$	Tighter pour with only slightly less spill. Larger spill than $< n$.

Table 3: *Comparing Liquid Animations*

force equation. This allowed customizable materials for powder, solid, and liquid. The best values for each material are listed in Table 4, and the results are shown using frames from the animations in Figures 1, 5, 6. For each material, interesting customizable attributes were discovered to control different aspects of the model. It was discovered that increasing both m and n resulted in more extreme collisions which could simulate a splash effect for liquid, but caused volatile radial force for powder. By reducing n, the model will experience more attraction from nearby globules. This can be used to tighten the pour but causes the model to collapse on itself. A similar effect can be achieved with a larger b_1 value to make the model more compressible, while also not collapsing on itself. Finally, it was discovered that the viscosity of a liquid could be altered by changing the value of C_d in (5). As the value approaches r_0, the material becomes more like a powder and spills away easier. As the value approaches $2r_0$, the material becomes more like a solid.

Material	Variable Values	Results
Powder	$r_0 = 1.0$, $b_1 = 2.0$, $m = 5$, $n = 3$, $C_r = C_d = r_0$	Natural bounce and spill. No clumps.
Solid	$r_0 = 1.0$, $b_1 = 1.0$, $m = 5$, $n = 3$, $C_r = C_d = 2r_0$	Clump together and stabilize.
Liquid	$r_0 = 1.0$, $b_1 = 2.0$, $m = 5$, $n = 3$, $C_d = 1.5r_0$	Clump by spill to fill space. Flow.

Table 4: *Best Results for All Materials*

Animations were performed on a machine with an Nvidia RTX 2070 Super GPU, Ryzen 5 3600xt CPU @ 3.79Hz, 16 gigabytes of RAM, and Windows 10 OS. The FPS of the animation depended on how many particles were in the system. It was found that simulating 400 globules gave an average of 60 FPS, 200 globules gave an average 120 FPS, and 100 globules gave an average of 300 FPS.

5. Challenges and Limitations:

As Miller and Pearse discuss in their paper, creating fluid using soft iso-sphere globules is great for creating liquid that collides with the ground but not for modelling liquid in a container. Because the globules are non-rigid, half of their surface will intersect with a rigid effector when they collide. This is okay if the animation does not view under a surface, as implemented for this paper. However if the user wished to model liquid in a water bottle for example, an offset must be applied to the penalty collision, or the container walls must be scaled.

By using an Euler approach to solve for the next position, a small time step is required. Simulations for this paper were first tested with a timestep of 0.01 and the result had particles exploding everywhere. It was determined that a timestep of 0.001 was necessary, but this made the animation take quite a bit of time. To visualize the animation closer to "real time", captures had to be sped up 16 times faster.

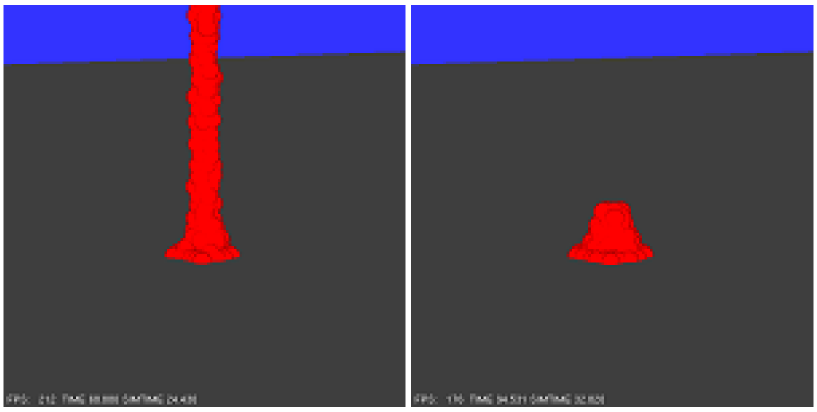

Figure 6: *Frames from solid animation with best results.*

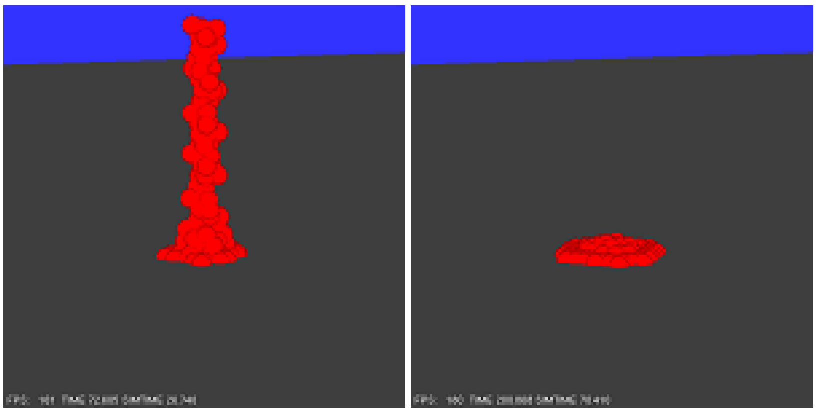

Figure 7: *Frames from liquid animation with material constant* $C_d = 2r_0$.

The simulator is limited to 400 globules for animations with consistently higher frame rate than 60 FPS.

6. Future Work and Implementations:

It is intended to test globules made with this simulator with more effectors than just the frictionless ground plane. These effectors could include but are not limited to: a projectile, a knife cut, a track of some sort. Correct collision detection must be implemented for such effectors which could slow the simulation, so further optimization will be tested to speed up calculations. Offloading computations from the CPU to GPU will be experimented. The simulation could be sped up using a semi-lagrangian method for calculating the next position of a particle [N. Foster et al, 2001]. Finally, adding temperature to the simulation can add a time variance to the material type, which could simulate freezing or melting of a material. This can be used to simulate something like a lava lamp.

7. References:

1. Miller, Gavin, and Andrew Pearce. “Globular Dynamics: A Connected Particle System for Animating Viscous Fluids.” Computers & Graphics, Pergamon, 1989.

2. C.W. Reynolds. “Flocks, herds, and schools: A distributed behavioral model.” SIGGRAPH 87 Comp. Graphics21 (4), 25-34 (1987).

3. PDI/DreamWorks, Nick Foster, et al. “Practical Animation of Liquids.” Practical Animation of Liquids | Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 1 Aug. 2001.

4. Wyvill, Brian. “20-Globular Dynamics.” CSC 473: Fundamentals of Animation. CSC 473: Fundamentals of Animation, Victoria, B.C, University of Victoria.