Real-time Simulation of Granular Matter using Smoothed Particle Hydrodynamics

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
André Neder

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Alexander Maximilian Nilles, M.Sc. (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Januar 2024

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

1 Introduction

Granular matter consists of discrete, macroscopic particles that interact through contact forces, exhibiting unique behaviors such as flowing, layering or piling. Simulating such behaviors in realistic scenarios is especially challenging, due to the necessity of employing a very high number of discrete elements to achieve visually plausible simulations. Granular materials find diverse applications across geophysics, civil engineering, pharmaceuticals, and other industries, playing crucial roles in phenomena such as landslides, construction, powder mixing, storage and transportation. A powerful technique for simulating such materials is Smoothed Particle Hydrodynamics (SPH). SPH is a meshless Lagrangian method that has been employed a lot in fluid dynamics and can also be used to simulate granular behavior. By combining well-established SPH techniques for simulating granular materials with the latest developments in SPH research, real-time simulations might become achievable. In the pages ahead, I will assess the achievability of real-time simulations for granular matter, by combining established methods and recent advancements to achieve a dynamic and interactive implementation. On the upcoming pages an algorithm, composed of different techniques from the field of Smoothed Particle Hydrodynamics, is presented. Firsly, in section 3 some basic concepts of some used techniques are explained. Then in section 4 the idea of Smoothed Particle Hydrodynamics, and some of its different variations are presented, aswell as the utilized programming framework. Then, in section 5 the actual implementations are explained. In the final chapter 6, a conclusion on what has been achived and an outlook over possible improvements and expansions is given.

2 Related Work

There have been a lot of publications about the simulation of granular matter. Some using grid based approaches [7], others using particle based methods like Smoothed Particle Hydrodynamics, which have first been introduces by Müller et al. [28]. One of the most interesting articles, that tries to simulate granular materials, have been the works of Narain et al. [7]. In their paper „Free-Flowing Granular Materials with Two-Way Solid Coupling“ they describe a hybrid algorithm, which transfers particle data on to an eulerian grid, where the pressure and friction calculations are performed. The resulting forces are then applied to the particles. Two-Way Coupling beeing the combination of the particle simulation influencing rigidbodies and the other way around. Later, in the works by Alduán et al., titled „SPH Granular Flow with Friction and Cohesion“ [5], the ideas presented by Narain et al. [7] are picked up and transfered to the Predictive-Corrective Smoothed Particle Hydrodynamics method by Solenthaler and Pajarola [6]. Later on, Ihmsen et al. [1] improved the alogithm, by refining the concept of upsampling the simulation, by blending between the simulation and external forces, which has been proposed by Alduán el al. [2] in their paper „Simulation of High-Resolution Granular Media“. In contrast to the mentioned works, which all simulate the behavior of granular matter in a non realtime fashion, I want to run the simulation with interactable framerates.

3 Basics

Some of the basic concepts of the utilized methods are the Navier-Stokes equations, Signed Distance Functions and Bitonic Sort, which will be briefly explained in this chapter.

3.1 Navier-Stokes equations

The basic concept behind most fluid simulations lies in the Navier-Stokes equations, which describe the motion of fluids and deformable solids. As granular flow can be interpreted as a kind of fluid they will be employed here aswell. One of the most important parts is the continuity of mass equation, which „describes the evolution of an object's mass density ρ over time t, i.e.," [10]

$$\frac{D\rho}{Dt} = -\rho(\nabla v) \tag{1}$$

with v being the velocity and ∇ the gradient operator. It states that the mass of fluid entering a given region must be equal to the mass leaving that region, accounting for any change in density within the fluid. The second part is the incompressible Navier-Stokes equation

$$\rho\frac{Dv}{Dt} = -\nabla p + \mu\nabla^2 v + f_{ext} \tag{2}$$

which describes the conservation of momentum. Here μ denotes the dynamic viscosity of the fluid and f_{ext} the influence of external forces. In this context, the pressure p serves as a Lagrange multiplier. Its selection is crucial in ensuring that the conservation of mass is met. Overall the formula describes the interplay of inertial forces, pressure, and viscosity, maintaining the conservation of momentum.

3.2 Signed Distance Functions

Signed distance functions (SDF) are a versatile tool in computer graphics. They determine the shortest distance of a point in space to the surface of a geometric object, while preserving information about if the point is inside or outside the object. This allows for precise implicit reconstruction of surfaces. A 2D example can be found in illustration 1. It shows the signed distance for a circle and a rectangle, where the white lines are the outlines of the objects and the blue and orange lines indicate areas of same negative and positive distance respectivly. A greate source for understanding and utilizing SDFs has been an article by Inigo Quilez, as it gives easily understandable examples. [21]

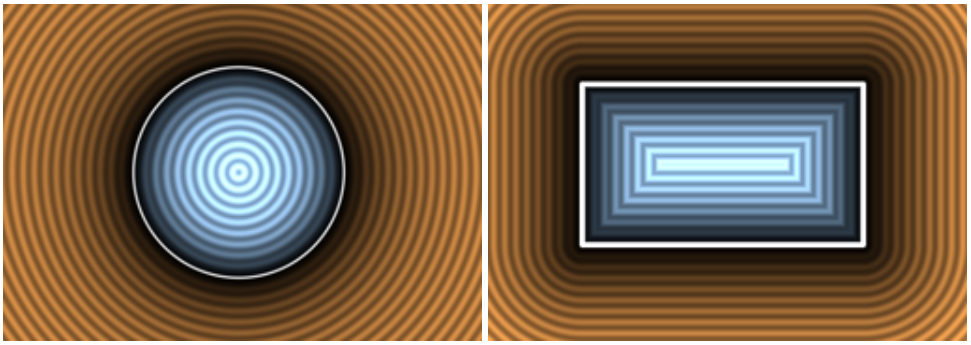

Abbildung 1: SDF of a circle and a box in 2D

3.3 Bitonic Sort

Bitonic merge sort is a sorting algorithm with a performance complexity of $O(\log^2(n))$. The algorithm can be implmented on the GPU to achieve a massive performance increase by parallelization. In this work, a modified version of the implementation by @tgfrerer has been used, as shown in the article on his website [23]. The algorithm uses a pattern of comparisons to sort a list of elements in parallel. An example for such a pattern can be seen in illustration 2. The algorithm has been modified to use Vulkan's Push Constants to select the type of comparison inside of the shader provided by @tgfrerer [23]. Another change is the use of Specialization Constants to set the local workgroup size of the compute shader invocation.

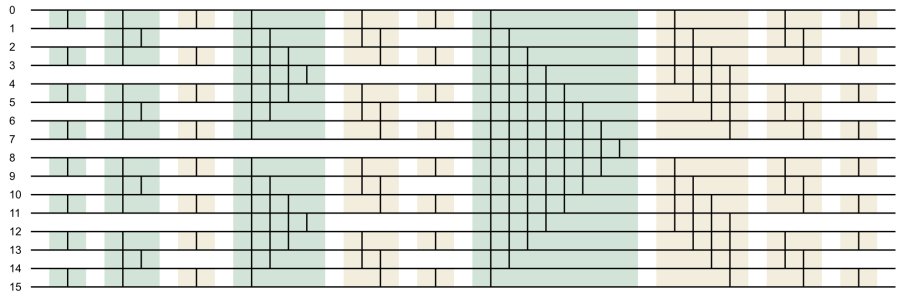

Abbildung 2: Sorting diagram for 16 sortable elements. Horizontal lines represent array indices into our array of sortable elements, each vertical line represents one worker thread performing a compare-and-swap operation.

4 Method

To understand the implemented algorithms it is neccessary to be familiar with Smoothed Particle Hydrodynamics, which will be explained in this chapter. I will also list the used libraries and APIs and explain their purpose and why they were employed.

4.1 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a computational technique widely employed in fluid dynamics and other fields for simulating complex physical phenomena.SPH represents the fluid or material as a collection of discrete particles. Each particle has physical properties such as density, pressure and velocity.

These interact through a kernel function W with radius h, which smoothes these properties over neighboring particles, allowing for a continuous representation of the material and facilitating the simulation of intricate behaviors, including fluid flow, collisions, and deformations. An arbitrary field A can be approximated as following, with m being the mass and ρ the density of a particle. [27]

$$A(x) = \sum_j m_j \frac{A_j}{\rho_j} W(|x - x_j|, h) \tag{3}$$

This concept is then applied to calculate the properties of a given particle. SPH's versatility extends beyond fluids, making it applicable to problems involving granular materials (e.g. [2]), astrophysics [30], and even solid mechanics [31].

A typical implementation for a kernel function is the cubic spline kernel by Monaghan [19]. I this work a spiky kernel as implemented by Sebastian Lague [24] is used as it provided better results for granular matter.

Weakly Compressible SPH

Weakly Compressible Smoothed Particle Hydrodynamics (WCSPH) is a variant of the traditional SPH method, where the incompressibility of the Navier-Stokes equation is not as highly enforced, allowing for a more computationally efficient simulation while still capturing the essential fluid behaviors. This approach is particularly useful in scenarios where the fully incompressible SPH formulation might be computationally expensive and it finds a compromise between precision and computational efficiency. This is especially usefull in the context of large-scale simulations or real-time applications.[27]

Predictive-Corrective Incompressble SPH

Predictive-Corrective Incompressible Smoothed Particle Hydrodynamics (PCISPH) is an advancement in the SPH method, designed to address issues related to incompressibility in fluid simulations. It employs a two-step process where a predictive step estimates particle velocities and a corrective step, which iteratively adjusts these velocities to enforce incompressibility. This technique is particularly useful in scenarios where traditional SPH methods such as WCSPH struggle with maintaining fluid volume conservation, providing more accurate and stable simulations, especially for scenarios involving complex fluid interactions, like fluid splashes or turbulent flows. [6]

Implicity Incompressble SPH

Implicit Incompressible Smoothed Particle Hydrodynamics (IISPH) is a variant of the SPH method that also addresses the challenge of maintaining incompressibility in fluid simulations. Unlike traditional SPH methods, IISPH employs an implicit formulation for the pressure term, this will be described in section 5.8, allowing for larger time steps and improved stability.[20]

4.2 Libraries & Technologies

C++ & Vulkan

When choosing a framework for a fluid simulation using SPH most implementations (for example: [25] [26]) use C++ in combination with OpenMP to achieve parallelization, other implementations involve Cuda to use the massive parallel compute performance of GPUs [26]. For rendering these simulations, dedicated renderers are used. In this implementation Vulkan is used to simulate and visualize the granular material on the GPU using a combination of compute shaders and the rasterization pipeline. This allows for massive parallel computation of the simulation togehter with the visualization in real time, giving maximum flexibility and extensibility by for example a raytraced visualisation. Vulkan uses SPIRV Shaders, which allows the use of any shading language, that can be compiled to it. I opted for GLSL for its well known C like structure. `https://www.vulkan.org/`

CMake

CMake is a popular choice in software development for its role as a cross-platform build system and project configuration tool, streamlining the build process, enhancing code portability, and offering flexibility in managing diverse project structures and dependencies. `https://cmake.org/`

Vulkan Memory Allocator

In difference to other Graphics APIs Vulkan requires the programmer to do the memory management on the GPU. Vulkan Memory Allocator (VMA) is employed to achive efficient management and optimization of GPU memory, simplifying memory allocation and deallocation processes and improving overall performance. `https://gpuopen.com/vulkan-memory-allocator/`

shaderc

Shaderc is utilized for the compilation and linking of shaders due to its user-friendly interface and simplicity in handling these processes. This made is straight foward to compile the used GLSL shaders to SPIRV for Vulkan to use. `https://github.com/google/shaderc`

TinyGLTF & STB Image

In computer graphics, TinyGLTF is often used for seamless glTF asset loading, and STB Image is favored for its straightforward approach to handling and processing images, both known for their efficiency and ease of use across different applications. `https://github.com/syoyo/tinygltf`

Dear ImGui

Dear ImGui is a popular immediate mode graphical user interface (GUI) library. It simplifies the process of creating and integrating user interfaces into applications by providing an intuitive and efficient framework and has implementations for nearly every common graphics API and windowing library. It also supports features like multiple viewports and docking elements to them. `https://github.com/ocornut/imgui`

GLFW

GLFW is a widely-used windowing library that simplifies the creation and management of windows, handling user input and facilitating cross-platform development for OpenGL- and Vulkan-based applications. `https://www.glfw.org/`

glm

GLM is a versatile C++ mathematics library designed for graphics programming, providing a comprehensive set of functions and classes for vector and matrix operations commonly used in computer graphics. `https://github.com/g-truc/glm`

TriangleMeshDistance

TriangleMeshDistance offers a user-friendly programming interface for effortlessly creating signed distance fields of adjustible resolution from 3D meshes.
`https://github.com/InteractiveComputerGraphics/TriangleMeshDistance`

5 Implementation

The whole implementation consists of a simple abstraction over the Vulkan API, I created, to speed up the creation of all required objects, such as buffers, descriptor sets and pipelines. I first create all the particles on the CPU side and store them as a storage buffer on the GPU. This buffer can later be bound as vertex buffer to be used in the rendering. The rigid bodies are loaded from the glTF files and processed into buffers and textures to be used by a dedicated renderpass. To incorporate them into the simulation, they are converted into a signed distance representation stored as 3D textures. With this, all the major data of the simulation is accessable on the GPU. The actual algorithm 1 consits of 13 different compute shaders, which execute the different steps. First, the particles are grouped into their grid cells and the external force for each particle is set to the gravity. Then the resulting grid entries are sorted and then the start indices of all cells are determined. After that, ρ, F^{drag}, s, v_{adv} and ρ_{adv} will be calulated in separat shaders. Their purpose will be explained in detail later on. All steps of the simulation are separated by pipeline barriers to ensure a property has been calculated for all particles before accessing them in the next step. After the computation of ρ_{adv} has been issued, the first command buffer is send to the gpu for processing and its completion is awaited, as the next steps will need to be executed for a dynamic amount of iterations. These iterations consist of computing $d_{ij}p_j$, p^{l+1}, $p(t)$ and applying the yield criterion for the stress. After these steps have been executed the average density error determines if another iteration is required. For this, its value needs to be transfered from the GPU to the CPU. All other computations happen exclusivly on the GPU by accessing the particles stored in the storage buffer. After the IISPH iterations are done, the resulting forces are computed and integrated over the elapsed time. In the last step the simulation is visually enhanced by advecting a set of high resolution particles based on the flow of the before computed low resolution behavior and external forces. After this step the storage buffer containing the high resolution particles is bound to a renderpass for drawing and the algorithm starts from the beginning. All steps will now be explained in detail.

Algorithm 1 Full Simulation Frame

find neighbors
compute ρ and F^{drag}
compute s
compute v_{adv}
compute ρ_{adv}
$l = 0$
while $l < 2 \;||\; \rho_{avg} - \rho_0 < \eta$ **do**
    compute $d_{ij}p_j$
    compute p^{l+1}
    $p(t) = p^{l+1}$
    apply yield criterion
    $l = l + 1$
end while
compute F^p and F^f
integrate
advect HR particles

5.1 Neighborhood Search

Using the SPH approximation involves finding all neighboring particles within the smooting length h. In a naive implamentation this means iterating all other particles and check their distance to each other. With a runntime of $O(n^2)$ this is not an option for a realtime scenario. To improve the neighborhood search a number of algorithms could be employed. In this implementation a grid based approach has been used, where every particle is first segmented into a spatial grid with cell size of h. This provides the possibility to find every neighbor of a given particle, by finding its cell and iterating over its members and all adjacent cells. This dramatically decreses the amount of particles that have to be checked every frame. To get a list of particles for each cell, all grid entries, consisting of a cell key and a particle ID, are first sorted by the cell key. As this sorting has to be done at the beginning of every frame, this has to be done as fast as possible. To achive this, a parallel sorting algorithm, such as radix sort or in this case bitonic merge sort is used. After the list of grid entries has been sorted, the start of every cells entries has to be found, by checking if the cell key of an entry and its predecessor are the same. The starting index of the cell is then stored in an array of length of n, being the particle count, at the position of the cellkey. This implies that all cell keys have to be within the range 0 to $(n-1)$. When iterating over a given cells entries, the entries particle ID is used to check if the particle is within the smoothing length. A cell is found by calculating a particles cell key an the keys of its neighboring cells, and looking up the starting index of that cells entries. The entries can then be iterated until

the cell key changes. In the below examples a list of 8 grid entries is shown, first in their unsorted state, then after they have been sorted and lastly the list of starting indices of each cell, where -1 denotes entries where no cell starts. In the actual implementation unsigned integers are used to represent the starting indices and the maximum unsigned integer is used for empty cells. [29] [24] [23]

Tabelle 1: Example of unsorted list of grid entries

Cell key	2	7	5	3	7	7	5	2
Particle ID	0	1	2	3	4	5	6	7

Tabelle 2: Example of sorted list of grid entries

Cell key	2	2	3	5	5	7	7	7
Particle ID	0	7	3	2	6	1	4	5

Tabelle 3: Example of list of starting indices

Starting indices	-1	-1	0	2	-1	3	-1	4

This approach allows the grid to have an undefined size as only the amount of particles matters for the creation of it.

5.2 Density

When implementing a SPH simulation the first thing to do is calculate the density ρ of a particle. It is determined by a sum over the neighboring particles. The volume V of each particle within the smoothing length h is multiplied with the density of it and the weighting kernel $W_{ij} \equiv W(x_i - x_j, h)$. As the volume can be expressed as $V = \frac{m}{\rho}$ we can simplify the equation to only the sum of weighted masses. [27]

$$\rho_i = \sum_j m_j W_{ij} \tag{4}$$

5.3 Unilateral incompressibility

The next step of the simulation is to compute a pressure that compensates the difference in density between particles. The density and pressure underlie the principle of unilaterial incompressibility, which can be expressed in the form:

$$\rho_i \leq \rho_0 \perp p \geq 0 \tag{5}$$

This says that the density of a given particle i can not be greater than the rest density of the material and the pressure can not become negative. [5]

5.4 External forces

A change in density can be achived by applying external forces to the particles. In this simulation a gravitational force and an approximated drag force are employed.

5.5 Gravity

Gravity is given as a fixed acceleration in a given direction in 3D space. Multiplied with the weight of a given particle we get an actual force.

5.6 Drag

As an additional force that approximated the interaction with air a drag force F^{drag} was applied. Which uses a simplified implementation of the drag by Gissler et al.. For this the relative velocity difference of a particle to the surrounding air is computed and its length squared. [8]

$$v_{i,rel}^2 = |v_a - v_i|^2 \ \frac{v_a - v_i}{|v_a - v_i|} \tag{6}$$

The velocity of the air v_a can be set to 0 or could be taken from a velocity field of some kind. Computing the drag force, to apply to each particle, involves determining the crosssecional area A_i of a particle. This can be represented by the area of a circle. Then the actual area exposed to oncomming air needs to be determined. For this a cone which is defined by the direction of the relative velocity differnce and a set angle is used to check if there are any other particles that shield the current particle form the air. This is done by a dot product of the relative velocity difference and the vector to the neighboring particles, which returns the angle between them. To not only get a boolean value for the shielding, an occlusion value is introduced, which is set to the smallest angle between neighboring particles in the cone. This value is then subtracted from 1 and added to the drag formulation as weight ω. Further parameters for the equation are ρ_a the density of the air or the medium that surrounds the particles and $C_{D,i}$ the drag coefficient for shape of the particle. [8] The whole formula is then composed as :

$$F_i^{drag} = \frac{1}{2}\rho_a v_{i,rel}^2 C_{D,i} \omega A_i \tag{7}$$

The deformation of a particle by the surrounding air is left out in this implementation.

5.7 Strain & Stress

To simulate friction between the particles of the granular material we first need to calculate the strain ε, which measures the deformation of a material

under external forces and is represented as a tensor of rank 3. This involves determining the velocity gradient ∇u_i by an outer product with the kernel gradient ∇W_{ij}.

$$\nabla u_i = \sum_j V_j \nabla W_{ij} u_j^T \tag{8}$$

The strain is then computed as:

$$\varepsilon = \frac{1}{2}(\nabla u_i + \nabla u_i^T) \tag{9}$$

What is stress... For the computation of the frictional stress a parameter D, which relates the frictional stress s to the dissipation of strain, is calculated as:

$$D_i = \frac{2m_i^2 \Delta t}{\rho_i^2} \sum_j \frac{1}{\rho_j} \nabla W_{ij} \nabla W_{ij}^T \tag{10}$$

The paramter can be precomputed for a prototype particle with a filled neighborhood, to make the compution more efficient. The stress is obtained by multiplying the strain tensor with the inverse of D.

$$s_i = D^{-1}\varepsilon \tag{11}$$

For most dry granular materials there is no or neglectable cohesion (The tendency of particles to stick to each other). This means that we can subtract the mean hydrostatic stress $s_{i,hydrostatic}$ from the stress computed above, enforcing a traceless deviatoric stress $s_{i,deviatoric}$, which will further be used as s_i.

$$s_{i,hydrostatic} = \frac{1}{2} \operatorname{Tr} s_i \tag{12}$$

$$s_{i,deviatoric} = s_i - s_{i,hydrostatic} \tag{13}$$

Cohesion can be handled similarly to friction. (...)

The amount of friction that can be applied is limited by the pressure. This is expressed by the Drucker-Prager yield criterion:

$$||s_i|| \leq p_i \alpha \tag{14}$$

$$\alpha = \sqrt{2} \sin \Theta \tag{15}$$

where α is the frictional coefficient for an angle of repose θ (the maximum angle a material can pile before slipping) and $||s_i||$ the frobenius norm of s_i.The yield constraint can be approximated in a piecewise linear manner by applying it to each component of s. [5] [1] [2] [7]

5.8 Pressure

In traditional SPH methods the pressure p is calculated by an equation of state like the ideal gas equation [27]

$$p_i = k * (\rho_i - \rho_0) \tag{16}$$

or Tait's equation [27]

$$p_i = B((\frac{\rho_i}{\rho_0})^\gamma - 1) \tag{17}$$

where k, B and γ are parameters controlling the stiffness of the pressure calculation, which means the scaling from the density error to the resulting pressure. PCISPH then tries to minimize the mean density error by doing multiple iterations, correcting the pressure to apply. As WCSPH does not provide a good enough incompressibility for stable friction calculations and PCISPH can take a lot of iterations to converge, I opted for IISPH as it provides pressures that result in a very small mean density error, after only few iterations and supports larger timesteps than other variants of SPH.

The IISPH implemetation consist of two parts. The first being the advection of the velocities and densities and the second, the iterations to solve for the pressure. For the advection all external forces are summed up and the particle velocities are advanced in time to get an intermediate velocity v_i^{adv}. [20]

$$F_i^{adv} = F^g + F_i^{drag} \tag{18}$$

$$v_i^{adv} = v_i + \Delta t \frac{F_i^{adv}}{m_i} \tag{19}$$

The intermediate velocity is then used to calculate predicted densities ρ_i^{adv} for all particles. [20]

$$\rho_i^{adv} = \rho_i + \Delta t \sum_j m_j v_{ij}^{adv} \nabla W_{ij} \tag{20}$$

At this point the base pressure to use for the upcoming steps is set to $p_{last} = 0.5p$ [20]. The following equation is then used to determine the pressure values.

$$p_i^{l+1} = (1-\omega)p_i^l + \omega \frac{1}{a_{ii} * \Delta t^2}(\rho_0 - \rho_i^{adv} - \Delta t^2 \psi) \tag{21}$$

Here l is the index of the current iteration of the algorithm. The factor ω is called the relaxation factor and is set ot 0.5 as this value yields the best results [20]. a_{ii} is a coefficient that is computed per particle as:

$$a_{ii} = \sum_j m_j (d_{ii} - d_{ji}) \nabla W_{ij} \tag{22}$$

, with d_{ii} and d_{ji} being the same parameter for particle i and j respectivly, which is computed as:

$$d_{ii} = \Delta t^2 \sum_j -\frac{m_j}{\rho_i^2} \nabla W_{ij} \tag{23}$$

The before used value ψ is calculated as shown below:

$$\psi = \sum_j m_j (\sum_j d_{ij} p_j^l - d_{jj} p_j^l - \sum_{k \neq i} d_{jk} p_k^l) \nabla W_{ij} \tag{24}$$

$$\sum_{k \neq i} d_{jk} p_k^l = \sum_k d_{jk} p_k^l - d_{ji} p_i^l \tag{25}$$

, where $\sum_j d_{ij} p_j^l$ is computed every iteration as:

$$\sum_j d_{ij} p_j^l = \Delta t^2 \sum_k -\frac{m_j}{\rho_j^2} p_j^l \nabla W_{ij} \tag{26}$$

At this point, the yield can be testet with the new pressure and a the new density is predicted as:

$$\rho_i^{l+1} = |p * a_{ii} * \Delta t^2 - (\rho_0 - \rho_i^{adv} - \Delta t^2 \psi)| + \rho_0 \tag{27}$$

This is then being used to determine the overall average density error of all particles. For this the Vulkan Shader Atomic Float Extension is utilized to sum up the density error over all particles on the GPU and then divide it by the number of particles on the CPU. This value then determines if another iteration is required to achieve the desired threshold. Always at least two iterations are performed before using the average density error as a termination criterion. The implementation of the whole pressure calculation was guided by the open source library SPlisHSPlasH [25]. For more detailes on how the IISPH algorithm works, see the paper, Implicit Incompressible SPH, by Ihmsen et al. [20].

5.9 Internal Forces

For the calculation of the pressure force of a given particle the SPH concept is applied [1].

$$F_i^p = -m_i \sum_{j \neq i} m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}) \nabla W_{ij} \tag{28}$$

A similar formulation can be used to determine the frictional force of that particle [1].

$$F_i^f = -m_i \sum_{j \neq i} m_j (\frac{s_i}{\rho_i^2} + \frac{s_j}{\rho_j^2}) \nabla W_{ij} \tag{29}$$

5.10 Integration

For the integration of the simulation a simple Euler integration has been used. It could be exchanged for a more advanced integrations scheme such as Leap-Frog or Verlet.

$$v_i(t + \Delta t) = v_i^{adv} + \Delta t \frac{F_i^p}{m_i} \tag{30}$$

$$x_i(t + \Delta t) = x_i + \Delta t v_i(t + \Delta t) \tag{31}$$

5.11 Boundary Handling

Particle-Based Approaches

Most implementations of SPH use the concept of Akinci et al. [3] when implementing boundary handling and rigidbody interactions. This first implementation of this work also used this method, but later the concept of Volume maps was implemented due to multiple advantages. The idea is to represent boundaries and rigidbodys by sampling their surface or volume with a dense mesh of particles, which can then be incoporated into the normal simulation. For static objects the boundary particles are not integrated in time. For dynamic rigidbodies the particles representing them, collect the counteracting forces, they excert onto the fluid. The collected forces can then be summed up and integrated by a rigidbody physics solver. This method presents some drawbacks. One of them being the amount of particles required to sample the 3D models in the scene, which then have to be checked for each neighboring fluid particle. [9] [1] [3] [18]

Volume Maps

In this implementation the concept of Volume Maps is employed. They are an implicit boundary representation, inspired by the density maps concept. They use signed distance functions to determine the volume of the intersection of the kernel function around a particle and the boundary. For this, the boundary or bounding box of any arbitrary mesh is extended by the kernel radius and then sampled on a grid with a set resolution. Here the TriangleMeshDistance library is used to create a signed distance representation of any triangle based mesh. For each sample a vector to the nearest point on the surface of the object is stored aswell as the intersection volume V_B, which is calculated by the signed distance Φ, treated by a cubic extension function γ^*, to get a continuos differentiable function, which results in a smooth transition at $x = h$. [9]

$$V_b(x) = \int_{N(x)} \gamma^*(\Phi(x'))dx' \tag{32}$$

$$\gamma^*(x) = \begin{cases} \frac{C(x)}{C(0)}, & \text{if } 0 < x < h \\ 1, & \text{if } x \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

For this cubic extension function the cubic spline kernel $C(x)$ [19] is used. The use of this function does not enforce the use of the same function for the SPH approximation, where any kind of kernel function can be used. [9] The grid is then stored as a 32-bit 4-channel floating point 3-dimensional image and uploaded to the GPU. As Vulkan does not support arrays of 3D images, as it does for 2D, the descriptor indexing extension has to be used to allow for multiple 3D textures being bound dynamically. When sampling the texture in the shader, the nearest position on the surface and the intersection volume can be determined using only one texture read access instead of iterating all neighboring boundary particles as done in the particle-based approach. After transforming the volume map from its texture space into world space inside of the shader, a smooth value for both required properties is can be determined by the sampler. These values can the be used in any of the above mentioned equations, by subsituting the mass for the determined volume V_b and the reference density of the simulated material ρ_0.

$$\rho_i = \sum_b V_b \rho_0 W_{ib} \tag{34}$$

$$F_i^f = -m_i \sum_b V_b \rho_0 (\frac{s_i}{\rho_i^2}) \nabla W_{ib} \tag{35}$$

$$F_i^p = -m_i \sum_b V_b \rho_0 (\frac{p_i}{\rho_i^2}) \nabla W_{ib} \tag{36}$$

For the calculation of the resulting boundary forces only the pressure and stress of the particle is used, other than in the calculation of the forces between particles, where both particles contribute to the force. At this point the boundary forces can be applied to the actual rigidbody the volume map represents, to allow for two way coupling of the simulation and the boundaries. In the following illustration a slice of a volume map can be seen.

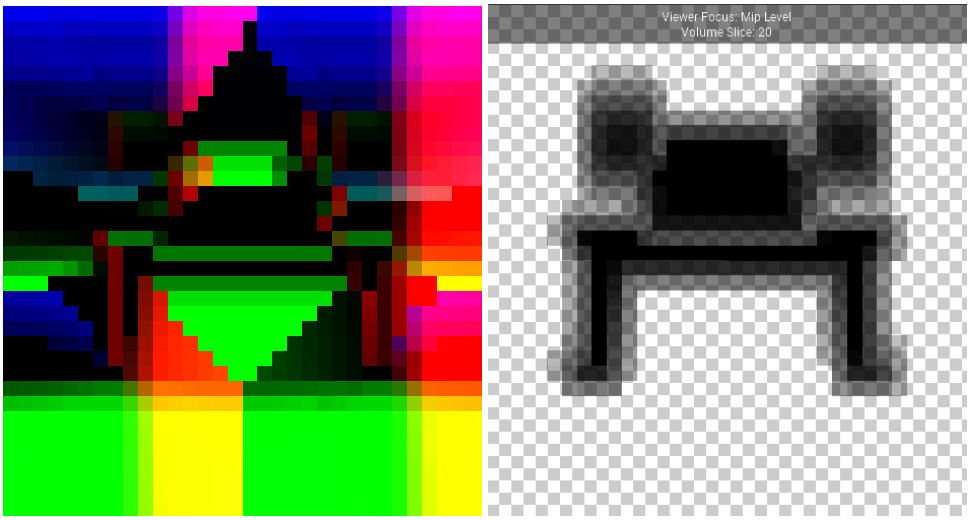

Abbildung 3: Slice of a volume map for a dump truck model, showing the nearest point and volume. Extracted using Nvidia Nsight Graphics.

5.12 Upsampling

As granular matter consists of vast number of grains, simulating them all as individual particles, would not only make the computational cost and memory consuption incredibly high, but would also result in a very restricted time step due to the CFL condition. To achive real time results, the simulation is done for a set of low-resolution (LR) particles which approximate the granular flow, which are then upsampled to another set of high-resolution (HR) particles. [1]
// TODO: Sampling

After the above described algorithm is executed on the low-resoltion particles, the high resolution particles are advected based on the velocity field of the low-resolution simulation. This is again done by the SPH concept, but with a different smoothing length h_{HR} which is set to $h_{HR} = 3h_{LR}$. To find the velocity of a HR particle a distance based weight w is introduced.

$$w(d_{ij}) = \max(0, (1 - \frac{d_{ij}^2}{h_{HR}^2})^3) \tag{37}$$

In this equation i refers to a high-resolution particle and j to a low-resoltion particle or a boundary and d_{ij} to the distance between them. The velocity is the computed as the weighted average velocity over the neighboring LR particles.[1]

$$v_i^*(t + \Delta t) = \frac{1}{\sum_j w(d_{ij})} \sum_j w(d_{ij}) v_j \tag{38}$$

As the LR particles should be able to not only follow the flow of the granular material, but react to gravity when there are no LR particles in its range, a blending weight α is introduced, to define the contributions of the simulation and external forces. The final velocity of each HR particle is then computed as:

$$v_i(t + \Delta t) = (1 - \alpha)v_i^*(t + \Delta t) + \alpha(v(t) + \Delta t\frac{F^g}{m}) \tag{39}$$

where α is:

$$\alpha = \begin{cases} 1 - \arg\max_j w(d_{ij}), & \text{if } \frac{\arg\max_j w(d_{ij})}{\sum_j w(d_{ij})} \geq 0.6 \\ 1 - \arg\max_j w(d_{ij}), & \text{if } \arg\max_j w(d_{ij}) \leq w(r_{LR}) \\ 0, & \text{otherwise} \end{cases} \tag{40}$$

The constant 0.6 is an empirically tested value for the above defined high-resolution smoothing length. [1] I then additionally modified the alpha value, by multiplying it with 1 minus the dot product of the external force and the vector to the boundary, clamped between 0 and 1, to prevent hr particles falling through the volume map based boundaries.

5.13 Visualization

As the visualisation is not the main focus of this work, it has been kept simple. To visualize the simulation, all particles are rendered as instanced meshes, as the number of particles to draw can become quite large and in this way all of them can be rendered in just one draw call. A detailed explanation of instanced rendering can be found on `https://learnopengl.com/Advanced-OpenGL/Instancing`. The rigidbodies are rendered by drawing each node of the model with its associated material, as its own draw call and are shaded with a simple diffuse lighting model.

5.14 User interaction

For the user to experience the simulation in 3D space, a camera controller has been implemented, that allows to explore the simulation from every angle. The simulation can be paused and reset to its initial state by buttons provided via the user interface. It also allows to modify some of the simulations parameters, such as the density, gravity or air velocity. Another purpose the UI serves is to provide infomation about the perfomance, such as timing, iteration count or framerate. The UI has been implemented in a way that allows the user to rearange all the windows, inside and outside of the main window. There for the docking and multi-viewport features of Dear ImGui have been utilized.

6 Conclusion & Future Work

When comparing this work with some of the related works presented in section 2, the major difference lies in the time a frame takes to compute, as they mostly run on the CPU. My implementation combines the basic concepts, of simulating granular matter using SPH, from the works of Ihmsen et al. [1] and Alduan el al. [5] with the boundary handling introduced by Bender el al. [9], the IISPH algorithm by Ihmsen et al. [20] for the pressure calculation and the approximation of drag forces by Gissler et al. [8]. All of this results in a simulation, that achieves the same effects as in the mentioned papers, while running with interactable framerates. The simulation achieves the accumulation of stable piles with different angles of repose, while keeping them stable for an extended amount of time, as shown in figure 4.

Abbildung 4: Three piles with different angles of repose, simulated for x frames. TODO

After a longer simulation time the piles will eventually melt down, due to a which results in a „bubbling“ effect, which is caused by minor inaccuracy in the pressure calculation. This problem can be improved by lowering the default maximum allowed compression of 0.5%, which I found being a good compromise between performance and accuracy. The maximum timestep was set to 16ms as this results in a framerate of 60 frames per second, while keeping the iterations required in a order of magniture that can be computed within that timeframe. These values were determined for a simulation with 8192 low resolution and about half a million high resolution particles for the piling scenario. For other scenarios the maximum compression needs to be even higher, at 1% to maintain a stable framerate. These large timesteps are only possible because of the IISPH algorithm, as the CFL condition, which determines the maximum viable timestep based on the simulation parameters, would only allow few or even sub millisecond timesteps with WCSPH or PCISPH. (Todo: Improvements?) The above mentioned „bubbling“ effect gets even worse when combining a large compression of 1% with low resolution volume maps, as they can not handle complex objects. Creating higher resolution volume maps improves this by some margin, but introduces longer loading times and higher memory consumption. The longer loading times could be countered by parallelization or precomputing the volume maps and loading them from files. To improve the memory consumption the rigid body

model could be separated into multiple volume maps with resolutions according to the required complexity of the part or to work around large empty areas for concave objects. Combined with some kind of hirarchical respresentation of the object, the sample time for one rigidbody should not increase by too much. Throught the upsampling of the low resolution simulation a visually plausible representation of vast amounts of granular material, comparable to other papers, is achieved. A major problem that Ihmsen et al. [1] already tried to solve by their upsampling approach, is the clumping of high resolution particles, which is still visible. Another problem is the sticking of particles to some shapes. This seems to come from the use of the volume maps, as the kernel used for the upsampling does not capture the contribution of the boundary quite right. One feature missing from my implementation compared to other implementations is two-way coupling, as the simulation only reacts to static boundaries and does not influence them. This could be implemented in the future by collecting the forces for each rigidbody and feeding them into a dedicated solver.

Abbildungsverzeichnis

Literatur

[1] M. Ihmsen, A. Wahl und M. Teschner, „High-Resolution Simulation of Granular Material with SPH," in *Workshop on Virtual Reality Interaction and Physical Simulation*, J. Bender, A. Kuijper, D. W. Fellner und E. Guerin, Hrsg., The Eurographics Association, 2012, ISBN: 978-3-905673-96-8. DOI: 10.2312/PE/vriphys/vriphys12/053-060.

[2] I. Alduan, Á. Tena und M. A. Otaduy, „Simulation of High-Resolution Granular Media," in *CEIG 09 - Congreso Espanol de Informatica Grafica*, C. Andujar und J. Lluch, Hrsg., The Eurographics Association, 2009, ISBN: 978-3-905673-72-2. DOI: 10.2312/LocalChapterEvents/CEIG/CEIG09/011-018.

[3] M. Ihmsen, N. Akinci, M. Gissler und M. Teschner, „Boundary Handling and Adaptive Time-stepping for PCISPH," Jan. 2010, S. 79–88. DOI: 10.2312/PE/vriphys/vriphys10/079-088.

[4] N. Bell, Y. Yu und P. J. Mucha, „Particle-Based Simulation of Granular Materials," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Ser. SCA '05, Los Angeles, California: Association for Computing Machinery, 2005, S. 77–86, ISBN: 1595931988. DOI: 10.1145/1073368.1073379. Adresse: https://doi.org/10.1145/1073368.1073379.

[5] I. Alduán und M. A. Otaduy, „SPH Granular Flow with Friction and Cohesion," Ser. SCA '11, Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011, S. 25–32, ISBN: 9781450309233. DOI: 10.1145/2019406.2019410. Adresse: https://doi.org/10.1145/2019406.2019410.

[6] B. Solenthaler und R. Pajarola, „Predictive-Corrective Incompressible SPH," in *ACM SIGGRAPH 2009 Papers*, Ser. SIGGRAPH '09, New Orleans, Louisiana: Association for Computing Machinery, 2009, ISBN: 9781605587264. DOI: 10.1145/1576246.1531346. Adresse: https://doi.org/10.1145/1576246.1531346.

[7] R. Narain, A. Golas und M. C. Lin, „Free-Flowing Granular Materials with Two-Way Solid Coupling," in *ACM SIGGRAPH Asia 2010 Papers*, Ser. SIGGRAPH ASIA '10, Seoul, South Korea: Association for Computing Machinery, 2010, ISBN: 9781450304399. DOI: 10.1145/1866158.1866195. Adresse: https://doi.org/10.1145/1866158.1866195.

[8] C. Gissler, S. Band, A. Peer, M. Ihmsen und M. Teschner, „Generalized Drag Force for Particle-Based Simulations," *Comput. Graph.*, Jg. 69, Nr. C, S. 1–11, Dez. 2017, ISSN: 0097-8493. DOI: 10.1016/j.cag.2017.09.002. Adresse: https://doi.org/10.1016/j.cag.2017.09.002.

[9] J. Bender, T. Kugelstadt, M. Weiler und D. Koschier, „Volume Maps: An Implicit Boundary Representation for SPH," in *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*, Ser. MIG '19, Newcastle upon Tyne, United Kingdom: Association for Computing Machinery, 2019, ISBN: 9781450369947. DOI: 10.1145/3359566.3360077. Adresse: https://doi.org/10.1145/3359566.3360077.

[10] D. Koschier, J. Bender, B. Solenthaler und M. Teschner, „A Survey on SPH Methods in Computer Graphics," *Computer Graphics Forum*, Jg. 41, S. 737–760, Mai 2022. DOI: 10.1111/cgf.14508.

[11] Z.-B. Wang, R. Chen, H. Wang, Q. Liao, X. Zhu und S.-Z. Li, „An overview of smoothed particle hydrodynamics for simulating multiphase flow," *Applied Mathematical Modelling*, Jg. 40, Nr. 23, S. 9625–9655, 2016, ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2016.06.030. Adresse: https://www.sciencedirect.com/science/article/pii/S0307904X16303419.

[12] S. Band, C. Gissler, M. Ihmsen, J. Cornelis, A. Peer und M. Teschner, „Pressure Boundaries for Implicit Incompressible SPH," *ACM Trans. Graph.*, Jg. 37, Nr. 2, Feb. 2018, ISSN: 0730-0301. DOI: 10.1145/3180486. Adresse: https://doi.org/10.1145/3180486.

[13] M. Fujisawa und K. T. Miura, „An Efficient Boundary Handling with a Modified Density Calculation for SPH," *Computer Graphics Forum*, Jg. 34, Nr. 7, S. 155–162, 2015. DOI: https://doi.org/10.1111/cgf.12754. Adresse: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12754.

[14] T. Takahashi, Y. Dobashi, T. Nishita und M. C. Lin, „An Efficient Hybrid Incompressible SPH Solver with Interface Handling for Boundary Conditions," *Computer Graphics Forum*, 2018, ISSN: 1467-8659. DOI: 10.1111/cgf.13292.

[15] M. Macklin, M. Müller, N. Chentanez und T.-Y. Kim, „Unified Particle Physics for Real-Time Applications," *ACM Trans. Graph.*, Jg. 33, Nr. 4, Juli 2014, ISSN: 0730-0301. DOI: 10.1145/2601097.2601152. Adresse: https://doi.org/10.1145/2601097.2601152.

[16] M. Weiler, D. Koschier, M. Brand und J. Bender, „A Physically Consistent Implicit Viscosity Solver for SPH Fluids," *Computer Graphics Forum*, Jg. 37, Nr. 2, S. 145–155, 2018. DOI: https://doi.org/10.1111/cgf.13349. Adresse: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13349.

[17] A. Sommer, U. Schwanecke und E. Schoemer, „Interactive High-Resolution Simulation of Granular Material," *Journal of WSCG*, Jg. 30, Nr. 1-2, S. 9–15, 2022, ISSN: 1213-6964. DOI: 10.24132/jwscg.2022.2. Adresse: http://dx.doi.org/10.24132/JWSCG.2022.2.

[18] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler und M. Teschner, „Versatile Rigid-Fluid Coupling for Incompressible SPH," *ACM Trans. Graph.*, Jg. 31, Nr. 4, Juli 2012, ISSN: 0730-0301. DOI: 10.1145/2185520.2185558. Adresse: https://doi.org/10.1145/2185520.2185558.

[19] J. J. Monaghan, „Smoothed Particle Hydrodynamics," *Annual Review of Astronomy and Astrophysics*, Jg. 30, Nr. 1, S. 543–574, 1992. DOI: 10.1146/annurev.aa.30.090192.002551. Adresse: https://doi.org/10.1146/annurev.aa.30.090192.002551.

[20] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath und M. Teschner, „Implicit Incompressible SPH," *IEEE Transactions on Visualization and Computer Graphics*, Jg. 20, Nr. 3, S. 426–435, 2014. DOI: 10.1109/TVCG.2013.105.

[21] I. Quilez, „Inigo Quilez :: computer graphics, mathematics, shaders, fractals, demoscene and more," Adresse: https://iquilezles.org/articles/distfunctions/.

[22] I. Quilez, „Inigo Quilez :: computer graphics, mathematics, shaders, fractals, demoscene and more," Adresse: https://iquilezles.org/articles/distfunctions2d/.

[23] @tgfrerer, „Implementing Bitonic Merge Sort in Vulkan Compute," Adresse: https://poniesandlight.co.uk/reflect/bitonic_merge_sort/.

[24] S. Lague, „Fluid-Sim," Adresse: https://github.com/SebLague/Fluid-Sim/tree/main.

[25] J. Bender u. a., „SPlisHSPlasH," Adresse: https://splishsplash.physics-simulation.org/.

[26] „DualSPHysics," Adresse: https://dual.sphysics.org/features/.

[27] M. Becker und M. Teschner, „Weakly Compressible SPH for Free Surface Flows," Bd. 9, Jan. 2007, S. 209–217. DOI: 10.1145/1272690.1272719.

[28] M. Müller, D. Charypar und M. Gross, „Particle-Based Fluid Simulation for Interactive Applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Ser. SCA '03, San Diego, California: Eurographics Association, 2003, S. 154–159, ISBN: 1581136595.

[29] S. Green, „Particle Simulation using CUDA,“ Adresse: https://web.archive.org/web/20140725014123/https://docs.nvidia.com/cuda/samples/5_Simulations/particles/doc/particles.pdf.

[30] V. Springel, „Smoothed Particle Hydrodynamics in Astrophysics,“ *Annual Review of Astronomy and Astrophysics*, Jg. 48, Nr. 1, S. 391–430, Aug. 2010, ISSN: 1545-4282. DOI: 10.1146/annurev-astro-081309-130914. Adresse: http://dx.doi.org/10.1146/annurev-astro-081309-130914.

[31] A. Shutov und V. Klyuchantsev, „On the application of SPH to solid mechanics,“ *Journal of Physics: Conference Series*, Jg. 1268, S. 012 077, Juli 2019. DOI: 10.1088/1742-6596/1268/1/012077.