

LEAVEN - Lightweight Surface and Volume Mesh Sampling Application for Particle-based Simulations

Alexander Sommer¹
alexander.sommer@hs-rm.de

Ulrich Schwanecke¹
ulrich.schwanecke@hs-rm.de

¹ Computer Vision and Mixed Reality Group, RheinMain University of Applied Sciences
Wiesbaden Rüsselsheim, Germany

ABSTRACT

We present an easy-to-use and lightweight surface and volume mesh sampling standalone application tailored for the needs of particle-based simulation. We describe the surface and volume sampling algorithms used in LEAVEN in a beginner-friendly fashion. Furthermore, we describe a novel method of generating random volume samples that satisfy blue noise criteria by modifying a surface sampling algorithm. We aim to lower one entry barrier for starting with particle-based simulations while still pose a benefit to advanced users. The goal is to provide a useful tool to the community and lowering the need for heavyweight third-party applications, especially for starters.

Keywords: mesh sampling, volume representation, surface representation, particle-based simulation

1 INTRODUCTION

In many fields of computer graphics, the plausible simulation of different physical phenomena is an important topic. As simulations hit interactive framerates, the urge for a unified simulation solver allowing to simulate various phenomena in one framework grew. A lot of methods like position-based dynamics [16] and their successors unified particle physics [14] and projective dynamics [3][19] use particles to represent all various simulation objects, like rigid-bodies, cloth, fluid, granular material, gases, and deformable solids. Further, in other non-real-time simulation methods, like Smoothed Particle Hydrodynamics (SPH) [18] or material point methods [9], particle representations play an important role. They are not only used for characterizing parts of a continuum but also in the efficient and correct handling of object- and domain boundaries [2].

We experienced that especially for beginners generating a particle representation of different geometrical objects is an entry barrier for getting started with particle-based simulations. There are a certain number of well-described algorithms for generating surface representations, as well as some algorithms for generating suitable volume representations (see Section 2).

Beginners often struggle to find a convenient and ready-to-use implementation or tool that is suitable for the requirements of particle-based simulation. The use of heavy-weight third-party software for sampling can also introduce new issues. These programs can be

hard to access and learning how to work around the vast amount of functions can be unnecessarily time-consuming. Furthermore, data-export is often limited to undocumented binary data formats.

Therefore, we introduce LEAVEN, a Lightweight surface And Volume Mesh sampling application tailored to the needs of particle-based simulations. LEAVEN is an easy-to-use graphical application that enables beginners to sample a surface or volume of a 3D triangle mesh with only a few straightforward clicks. Furthermore, it leaves enough customization options to be valuable for experienced simulators. LEAVEN is made available as open-source and can be used as a library in other projects.

Our paper is organized as follows: first, we give a brief overview of sampling algorithms (Section 2), then we describe the algorithms used for sampling in our application (Section 3), followed by a detailed description of the developed application (Section 4) and its usage (Section 5). Our main contributions are:

- Giving a beginner-friendly introduction to particle sampling techniques
- Lowering the entry barrier for particle-based simulations
- Modifying a surface sampling method to generate volume samples that satisfy blue noise criteria
- Contributing an easy-to-use, lightweight, open-source standalone application for sampling the surface and volume of arbitrary closed triangle meshes for use in particle-based simulations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

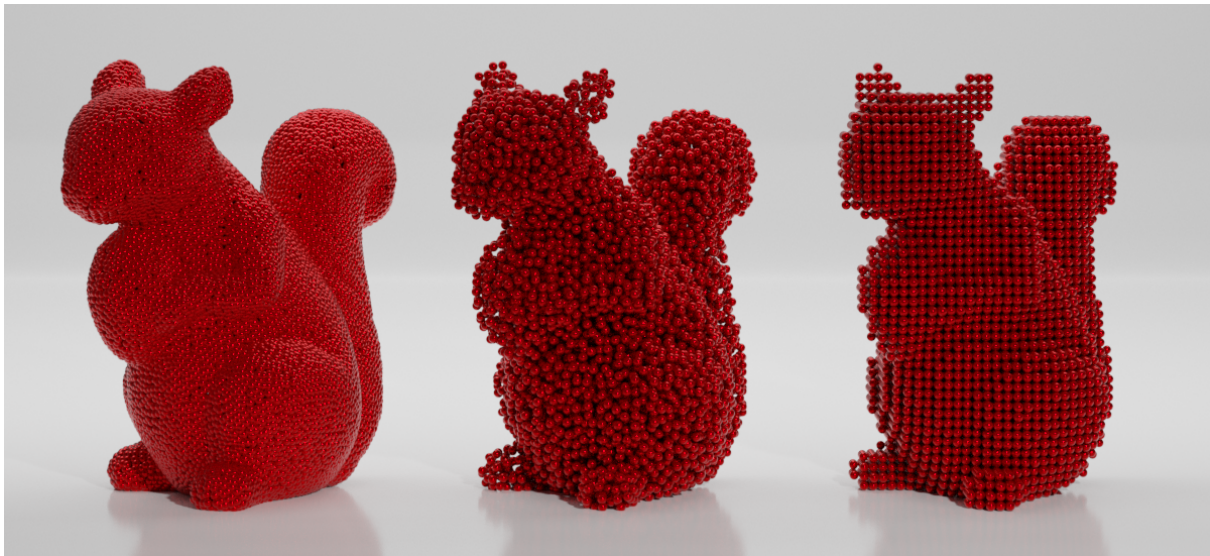


Figure 1: Different samplings in comparison. Surface sampling (left), volume sampling with randomly distributed particles (middle), and volume sampling with grid-based packing (right).

2 RELATED WORK

Various fields of computer graphics, such as texturing, remeshing, rendering, modeling, or animating fractures, have methods for sampling surfaces or volumes that are tailored to their specific needs. An important sampling criterion that many of these methods have in common is that the sampling behaves like blue noise. Blue noise distributions have been an important research topic [8][17][6]. A sampling that has blue noise characteristics is uniform and unbiased distributed in the spatial domain. Furthermore, when looking at the frequency domain with the Fourier transformed spatial information, there should be a lack of low-frequency noise and structural bias. One popular method for generating samplings that satisfy blue noise criteria is Poisson disk sampling [15][5]. In Poisson disk sampling, the samples are placed randomly on the valid spatial domain, which is the surface of an object for surface sampling or its volume for volume sampling. After placing the samples, a validation process guarantees that no two samples are too close to each other. Cline et. al. [7] introduced an optimized dart-throwing algorithm for generating Poisson disk point sets as a sampling of the surface of a 3D triangle mesh. Their algorithm needs sequential processing and has its applications in modeling and polygon remeshing. Wei et. al. [23] present a parallel dart-throwing algorithm by drawing suitable samples from a dense point cloud on the object's surface. This parallel version can achieve interactive frame-rates even for a large number of samples and is therefore well suited for real-time applications. Bowers et. al. [4] extended this algorithm to work not only with the Euclidean but also with a fast approximation of the Geodesic norm, as a measurement of sample point distances. This method is well suited for interactive tex-

turing applications or other tasks where quick sampling is needed. Yuksel [24] describes an approach of generating surface samplings without using a form of dart-throwing but rather assigning weights to possible samples. He uses a greedy algorithm to eliminate the points with the highest weight. Wang, T. et. al. [21] propose another rapid parallel surface sampling method. They use blue-noise pattern planes that they progressively project on the mesh surface using ray tracing. This method has its applications especially when fast sampling is required. A novel data-driven approach by Wang, Z. et. al. [22] promises to achieve even better performance than the classic geometrical approach on triangle meshes.

In contrast to the previously mentioned techniques, Jiang et. al. [11] use different methods inspired by SPH simulations. They are not only generating surface samplings but also volume samplings in that way. Their idea is to treat each sampling point as an SPH particle and establish a constant density in SPH kernel estimation functions by applying correction forces. Adams et. al. [1] describe an adaptive volume sampling method for particle-based fluid simulation. They use a sampling condition to put more focus on geometrically complex simulation regions.

3 ALGORITHMS

The main algorithms used in LEAVEN can be divided into surface- and volume sampling algorithms. In particle-based simulations, surface representations are typically used for boundary handling and sometimes for two-way coupling, while volume samplings are used for initial particle positions or rigid-body representations in unified solvers. They both have in common that they need a discrete 3D representation

Model	BBox Scaling	Surface Sampling		Random Volume Sampling		Grid Volume Sampling	
		time	particles	time	particles	time	particles
Bunny	2.0	13ms	9122	293ms	14875	100ms	22206
Squirrel	3.0	30ms	14889	573ms	31795	109ms	48012
Armadillo	6.0	87ms	53921	5096ms	145403	2559ms	215141
Dragon	12.0	595ms	233355	23444ms	943703	6710ms	1445523

Table 1: Timing results for different meshes and varying particle numbers resulting from different scales of the bounding box and a fixed particle radius of 0.02m for volume sampling / minimum distance for surface sampling.

as an input. 3D geometry in computer graphics is often represented as polyhedrons, with a polygon mesh defining the surface of the object. We use triangle meshes as input since these meshes are most common, easy to work with, and accessible. For volume sampling, the input mesh describing the surface of an object must be closed to have a distinguishable inside and outside. Figure 1 shows different samplings of the same 3D object. On the left the surface sampling algorithm is applied, where particles can get as close as half the particle diameter. Shown in the middle is our approach to generating random blue noise samples within a volume, with a minimum particle spacing equal to the particle diameter. A grid-based volume sampling is shown on the right. Table 1 shows timing results for the three algorithms available in LEAVEN on different standard meshes.

3.1 Surface Sampling

Akinci et. al. [2] describe how boundary particles can be a suitable boundary representation in SPH fluid simulation and how to calculate a two-way fluid-rigid coupling. Later this practice has been used for boundary handling in position-based methods like position-based fluids [13], or for the simulation of gases and granular material [10].

There are some basic requirements for a good surface representation with particles. First of all, the sampling should be dense enough that no simulation particle can tunnel through holes between boundary particles. Furthermore, the particles should be spread out uniformly on the object’s surface to guarantee an even force distribution to prevent artifacts.

To generate such a sampling of the surface of an object, we use a uniform sampling algorithm on arbitrary triangle meshes introduced by Bowers et. al. [4]. The method builds upon a grid cell sampling approach to generate Poisson disk samples on arbitrary manifold surfaces [23]. The goal of the sampling process is to find a set of sample points (particles) \mathbb{S} that is randomly but uniformly distributed on the surface ∂V of a 3D Volume V . Thereby all particles should have a minimum distance d to each other, i.e.

$$\text{dist}(\mathbf{s}_i, \mathbf{s}_j) \geq d, \quad \forall \mathbf{s}_i, \mathbf{s}_j \in \mathbb{S}$$

For most cases in particle-based simulation, d equals the radius r of the simulation particles. The distance

metric $\text{dist}(\cdot, \cdot)$ measuring the spatial separation can be Euclidean or Geodesic. The Euclidean distance measures the shortest line segment between two points in 3D space. The Geodesic distance measures spatial separation with respect to the object’s surface as a sub-manifold. This can be beneficial for achieving a good sampling of complex objects with thin features. For a detailed explanation of Geodesic distances and their fast approximation, we refer the reader to [4].

The algorithm starts initially by calculating a much larger set \mathbb{P} with candidate positions on the surface ∂V without bothering about the distances between them. The candidate samples are found by picking a random triangle from the triangle mesh with a probability proportional to the triangle’s area. On this triangle, an arbitrary position \mathbf{p} is generated by choosing two random values $\tau_1, \tau_2 \in [0, 1]$ for its barycentric coordinates

$$u = 1 - \sqrt{\tau_1}, \quad v = \tau_2 \sqrt{\tau_1}, \quad w = 1 - u - v$$

inside the triangle, which leads to a random position

$$\mathbf{p} = u\mathbf{a} + v\mathbf{b} + w\mathbf{c},$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are the three vertices of the triangle. In this manner, the initial set \mathbb{P} is filled with a large number of candidate positions. In the application LEAVEN, we generate $\rho_S \cdot \frac{A(\partial V)}{\pi r^2}$ positions in this way, where $A(\partial V)$ is the surface area, which is the sum of all triangle areas and ρ_S is a controllable density parameter. We use a default value of $\rho_S = 40$. After this, the bounding box containing V is sliced into grid cells with a uniform side length of $\frac{d}{\sqrt{3}}$. This leads to a cell diagonal of d . For each position, $\mathbf{p} \in \mathbb{P}$ the corresponding grid cell id is calculated and the whole data array containing \mathbb{P} is sorted by the grid cell id. Each cell that contains at least one position \mathbf{p} is a valid surface cell. Cells without candidate positions are much likely located outside or inside the volume.

Since the majority of cells is empty a hash table is a suitable lookup structure for finding valid cells and their positions. The cell id is used as the hash key. In contrast to [4], we use a spatial hashing function with bit-wise XOR:

$$(i \cdot p_1 \text{ XOR } j \cdot p_2 \text{ XOR } k \cdot p_3) \% \text{size}(\text{hash_table})$$

where

$$p_1 = 73856093, \quad p_2 = 19349663, \quad p_3 = 83492791$$

as described in [20], where i, j, k are the cell indices, to prevent hash collisions. Each hash bucket contains the cell id, a pointer to the first position \mathbf{p} that is associated with this cell id in the concurrent sorted data array, and if already chosen, the sampled position. In the end, each valid surface cell will contain at most one sample.

Algorithm 1 Uniform surface sampling

```

1: generateCandidateSetP()
2: sortSetPByCellId(setP)
3: insertSetPInHashMap(setP)
4: for trial  $t = 1, \dots, n$  do
5:   for all valid cells  $i$  do
6:     if cell  $i$  is already sampled then
7:       break
8:     if  $t$ -th candidate point  $\mathbf{p}_{it}$  doesn't exist then
9:       break
10:    conflict = false
11:    for all neighbor cells  $j$  do
12:      if  $j$  not a valid cell then
13:        break
14:      if  $j$  has no sample  $\mathbf{s}_j$  yet then
15:        break
16:      if  $\text{dist}(\mathbf{p}_{it}, \mathbf{s}_j) < d$  then
17:        conflict = true
18:      if conflict == false then
19:        sample for cell  $i$ :  $\mathbf{s}_i = \mathbf{p}_{it}$ 
20:        put  $\mathbf{s}_i$  in  $\mathbb{S}$ 

```

The main sampling algorithm consists of several trials n , which is another controllable parameter. A higher number of trials n makes it more likely to find a sample for each valid cell but needs more computation time. We employ a default value of $n = 10$. In each trial $t = 1, \dots, n$, each valid cell is processed. If the cell already contains a valid sample it is skipped. If there are no more candidate positions in \mathbb{P} , the cell can also be neglected. Otherwise, neighboring cells are checked to see if any already sampled position is too close to the current candidate position \mathbf{p}_{it} . If this is not the case \mathbf{p}_{it} is added to the set of samples \mathbb{S} . The whole sampling algorithm is shown in Algorithm 1. For a parallel version and further details, we refer the reader to [4].

3.2 Volume Sampling

In contrast to surface sampling, volume sampling is used to define initial positions of simulation particles, whether it may be free-moving particles like fluid, gas, or granular material, or a rigid-body representation in a unified solver like unified particle physics [14]. Therefore the requirements for the sampling are different. For initial positions the particles mustn't be primarily in an invalid state, meaning that the particles are not allowed to overlap. Overlapping initial positions can lead to high correction forces/position changes, causing the simulation to explode.

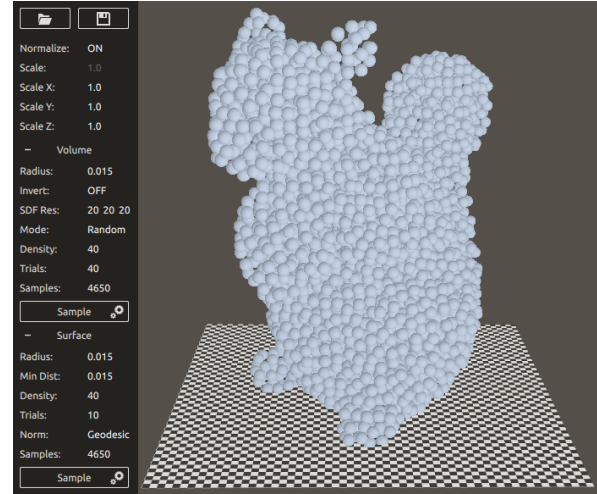


Figure 2: LEAVEN's application window with a triangle mesh sampled with 4650 randomly distributed particles through the volume.

Particles that represent a rigid-body should maintain a constant distance from each other during simulation. Therefore it makes sense to sample them closely and uniformly inside the volume V . The goal is to find a set of sample positions \mathbb{S} , that is spaced out on an equidistant grid with a minimal side length of $2 \cdot r$, where r is the particle radius. For deciding which positions are inside the Volume V we use a signed distance field (SDF). It is defined as a signed distance function $\Psi : \mathbb{R}^3 \rightarrow \mathbb{R}$,

$$\Psi(\mathbf{p}) = \text{sgn}(\mathbf{p}) \inf_{\mathbf{p}^* \in \partial V} \|\mathbf{p} - \mathbf{p}^*\| \quad \text{with}$$

$$\text{sgn}(\mathbf{p}) = \begin{cases} -1 & \mathbf{p} \in V \\ 1 & \text{otherwise} \end{cases}$$

measuring the Euclidean distance of a point $\mathbf{p} \in \mathbb{R}^3$ to the nearest point \mathbf{p}^* on the object surface ∂V . Generating such a parametric SDF representation of an arbitrary closed Volume V is a topic on its own and beyond the scope of this paper. In LEAVEN we use a grid-based SDF with hierarchical cell size and polynomial degree refinement based on polynomial fitting as described in [12]. The general idea behind this is to fit polynomial functions in cells of axis-aligned hexahedral grids defined by the input meshes. The algorithm iteratively refines the SDF either by spatial subdivision or cell-by-cell adjustment of the degree of the polynomial.

After generating the SDF, the bounding box containing V is divided into grid cells similar as described in Section 3.1 but with a cell size of $2 \cdot r$. For each cell i one sampling particle is added to the cell center at position \mathbf{p}_i . Then the signed distance function is evaluated as position \mathbf{p}_i . When $\Psi(\mathbf{p}_i) < 0$ the position is inside the Volume V and therefore it is a valid sampling position \mathbf{s}_i , that can be added to the sampling set \mathbb{S} (see Algorithm 2).



Figure 3: Volume and surface samplings used in a unified simulation framework simulating a granular material and a rigid-body (red) moved by an excavator.

Algorithm 2 Grid-based volume sampling

```

1: generateSDF()
2: for all cells  $i$  do
3:     position  $\mathbf{p}_i = \text{cellCenter}$ 
4:     if  $\Psi(\mathbf{p}_i) < 0$  then
5:         sample  $\mathbf{s}_i = \mathbf{p}_i$ 
6:         put  $\mathbf{s}_i$  in  $\mathbb{S}$ 
    
```

Particles that represent parts of fluids, gases, or granular materials, in general, change their distance to each other during the simulation. Sampling them in regular patterns, like previously, can introduce disturbing visual artifacts at the beginning of a simulation. Therefore we modify the concept of Bowers et. al. for surface sampling [4] explained in Section 3.1 to present a novel approach for generating random samples inside a 3D volume that satisfies blue-noise criteria.

Our modified sampling method divides the bounding box of V into uniform grid cells with a side length of $\frac{2 \cdot r}{\sqrt{3}}$ resulting in N cells. We fill the set of candidate positions \mathbb{P} by picking uniformly distributed positions within the whole bounding box and only adding the positions \mathbf{p} to \mathbb{P} that lie inside the volume (i.e. $\Psi(\mathbf{p}_i) < 0$). In this way, we generate $\rho_V \cdot N$ samples, where the number of samples generated is proportional to the number of grid cells in the bounding box. The proportionality constant ρ_V is a controllable density parameter. We recommend this parameter to be equal to the number of trials n . The rest of the sampling process is the same as for the surface sampling algorithm with Euclidean distance norm (see Algorithm 1).

4 APPLICATION

As mentioned before, the main goal with LEAVEN is to provide an easy-to-use tool suitable for beginners and advanced users in particle-based simulation. In the following, we will briefly explain the settings the user can manipulate in the application¹.

Figure 2 shows the LEAVEN user interface (UI). On the righthand side of the UI is the viewer. It displays the result of the sampling process. The depicted 3D model can be manipulated by arcball rotation, zooming, and panning. The lefthand side of the UI contains the settings for the sampling process. When mesh normalization is turned on, the mesh is uniformly scaled to fit inside a bounding box with a side length of 1. The mesh can be scaled further up or down by a factor. It is also possible to scale each axis individually, for example, to easily generate rectangular boxes from a unit cube for non-cubic simulation domains.

For volume sampling, the radius of the sampling particles can be specified. When sampling is inverted, the volume between the bounding box and the mesh outside is sampled. The SDF Resolution defines the accuracy of the parametric volume representation. For objects with finer details, a higher resolution is needed. The grid mode samples the particles uniformly and aligned on a grid inside the volume. Random mode generates randomized sampling inside the volume that satisfies blue noise criteria as described in Section 3.2. In random mode the controllable density parameter ρ_V and the number of trials n can be set.

¹ <https://github.com/a1ex90/Leaven>

For surface sampling, the radius of the displayed sampling particles can also be specified. This only affects the visual appearance of the particles in the application's preview display and should not be confused with the *Min Dist* parameter, which controls the minimum distance d between particles that influences the sampling (see right side of Figure 2). For boundary handling, a good rule of thumb for choosing d is to use the same size as the simulation particle radius r . Furthermore, the controllable density parameter ρ_S and the number of trials n can be chosen (see Section 3.1). It is also possible to switch between Geodesic or Euclidean distance norm for sampling particle distances. For more fine details in the mesh, the geodesic norm should be the one in favor.

5 USAGE EXAMPLE

Figure 3 shows a simulation scene where our sampling has been applied to many various objects. The scene has been simulated with the unified particle physics [14] variant of position-based dynamics [16]. A granular material with friction, behaving like sand, is simulated. Initial particle positions (sand-colored) are sampled with LEAVEN's volume sampling random mode on. The rigid-body, a squirrel, is simulated within the same framework with a two-way coupling with the granular material. It is sampled with grid-based volume sampling (red). The excavator (yellow dots), as well as the domain boundaries (blue dots), are surface sampled with a minimal distance between samples equal to the particle radius.

6 CONCLUSION

With this paper we contribute an easy-to-use application for representing surfaces and volumes by a set of particles. We used a state-of-the-art surface sampling algorithm [4] and present a novel approach for generating volume samples that satisfy blue noise criteria by using an SDF representation of the volume. Furthermore, we give a beginner-friendly introduction to the field of sampling techniques tailored for the needs of particle-based simulation.

REFERENCES

- [1] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):48–es, July 2007.
- [2] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4), July 2012.
- [3] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, July 2014.
- [4] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6), December 2010.
- [5] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, page 22, New York, NY, USA, 2007. ACM.
- [6] Jiating Chen, Xiaoyin Ge, Li-Yi Wei, Bin Wang, Yusu Wang, Huamin Wang, Yun Fei, Kang-Lai Qian, Jun-Hai Yong, and Wenping Wang. Bilateral blue noise sampling. *ACM Trans. Graph.*, 32(6), November 2013.
- [7] David Cline, Stefan Jeschke, K. White, Anshuman Razdan, and Peter Wonka. Dart throwing on surfaces. *Computer Graphics Forum*, 28:1217 – 1226, 08 2009.
- [8] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, January 1986.
- [9] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.*, 37(4), July 2018.
- [10] Markus Ihmsen, Arthur Wahl, and Matthias Teschner. High-Resolution Simulation of Granular Material with SPH. In Jan Bender, Arjan Kuijper, Dieter W. Fellner, and Eric Guerin, editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association, 2012.
- [11] Min Jiang, Yahan Zhou, Rui Wang, Richard Southern, and Jian Jun Zhang. Blue noise sampling using an sph-based method. 34(6), October 2015.
- [12] Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender. An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 07 2017.
- [13] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4), July 2013.
- [14] Miles Macklin, Matthias Müller, Nuttapon Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014.
- [15] Michael Mccool and Eugene Fiume. Hierarchical poisson disk sampling distributions. *Proceedings - Graphics Interface*, 05 1992.
- [16] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118, 2007.
- [17] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, 23:488–495, 08 2004.
- [18] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009. ACM.
- [19] Alexander Sommer, Ulrich Schwanecke, and Elmar Schoemer. Chebyshev's method on projective fluids. *Journal of WSCG*, 28:132–136, 2020.
- [20] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. *VMV'03: Proceedings of the Vision, Modeling, Visualization*, 3, 12 2003.
- [21] Tong Wang and Reiji Suda. Fast generation of poisson-disk samples on mesh surfaces by progressive sample projection. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(2), August 2018.
- [22] Zong-Sheng Wang, Jung Lee, Chang Geun Song, and Sun-Jeong Kim. Data-driven point sampling with blue-noise properties for triangular meshes. In *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, CSSE 2020, pages 77–82, New York, NY, USA, 2020. Association for Computing Machinery.
- [23] Li-Yi Wei. Parallel poisson disk sampling. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, New York, USA, 2008. ACM.
- [24] Cem Yuksel. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, 34, 05 2015.