COMP4920 Senior Design Project II, Spring 2020

Advisor: Mehmet Ufuk Çağlayan

POF: Performance Optimized Fluid Requirements Specifications Document

1.5.2020

Revision 3.0

By:

Baran Budak-15070001012

Cihanser Çalışkan-16070001020

İsmail Mekan-15070001048

Revision History

Revision	**Date**	**Explanation**
1.0	03.11.2019	Initial requirements.
2.0	19.12.2019	Requirements model.
2.1	20.4.2020	Grammar errors fixed.
2.2	21.4.2020	Diagrams updated.
2.3	22.4.2020	Explanations added to the mathematical equations.
2.4	22.4.2020	Warning page added.
2.5	23.4.2020	Introduction section writings extended.
2.6	25.4.2020	Surface recognition formulas with explanations added.
3.0	1.5.2020	Non-functional requirements extended.

Table of Contents

List of Tables

List of Tables

WARNING!

Important Note: POF project has hardware-based requirements. Your GPU must have CUDA 8.0.44 or better version and D3D11 support. If you do not have the required components, POF will not work.
We were using the Yaşar university computer lab in the first semester. Since Yaşar University is closed because of the COVID-19, we cannot access the computer laboratory. Therefore, we cannot make any progress in visualization.
The %75 of the project is finished. Implementation of the Marching Cubes algorithm which is the last step about the visualization part of our project could not be completed (We have a working marching cubes code as a prototype. However, we did not implement to the POF system.). For this reason, we have restated our project requirements and goals which will be clarified detailed in the Final Report and Requirements Specifications Document. In brief, the implementation and testing of the surface recognition system is the new goal of our project and some of the requirements are discarded such as Marching Cubes.

1.0 Introduction

Introduction part consists of three parts: Purpose of the POF system, the scope of the POF system and lastly, we give an overview.

1.1. *Purpose*

The purpose of the performance-optimized fluid (POF) system is to research and apply surface reconstruction methods to create a more efficient particle-based simulation system.

The POF system should increase the efficiency of the simulation. We obtain that by analysing situations of particles of each frame and finding the surface particles by using the hash system. Hash system utilizes to lowering the memory consumption.

POF system aims to develop an alternative way to analyze and visualize particles in each frame. We endeavour to research and development in smoothed-particle hydrodynamics (SPH).

In detail, the POF system is reconstructing the surface particles by benefiting from various research papers mentioned. The POF system approaches particles as a continuum and inspects the fluid as a whole object. Herewith, system approaches to fluids as there are no separate particles but rather the fluid is a continuous material. POF system analyses particles in each frame and applies specific algorithms.

1.2. *Scope*

POF system helps to increase performance for particle-based fluid simulation. POF system work as dependent on the NVIDIA Flex and Unity game engine. POF reduces computations for visualization of particles. We increase the computation in the background slightly, but we increase the performance much better.

Initially, there must be a handler class to manage all communications between the sectors of the POF system. When the simulation starts, NVIDIA Flex creates the particles and particle position data is transmitted to the POF system. Hash system receives position data and retrieves the particle id or the cell id of a particle when asked. We created cells in the axis-aligned boundary box (AABB) which is useful to find and analyze particles. POF system must find surface particles by calculating weight function and applying it as mentioned in the research papers. We calculate these functions for a specific range and these values are predetermined intuitionally. POF system computes the colour field quantity of each particle and marks all the surface particles. We find the particles in a cell and calculate how every particle affects the other particles as a scalar value. This is an implementation of Zhu and Bridson method [7]. POF construct the surface vertices as mesh triangles and particles reach their last appearance. However, this last part of the POF system could not be implemented due to the reasons that we have mentioned in warning page.

1.3. *Overview*

This document describes the POF system. Requirements specification document included with user case and sequence diagrams that define user roles in the system and more importantly, explaining how the system operates in the background. Further information exists on the Design Specifications Document revision 3.0 (DSD 3.0) [8].

The document focuses on describing the POF system project requirements. System functions are examined in two separate sections as functional and non-functional requirements. The function of NVIDIA Flex and how it is used in the POF system is described. User characteristics and constraints specify how the POF system can work under which circumstances.

2.0 Diagrams

This section gives use case and sequence diagrams which are design part of our project. Detailed information and explanations are in DSD 3.0 [8].

2.1 *Use Case Diagram*

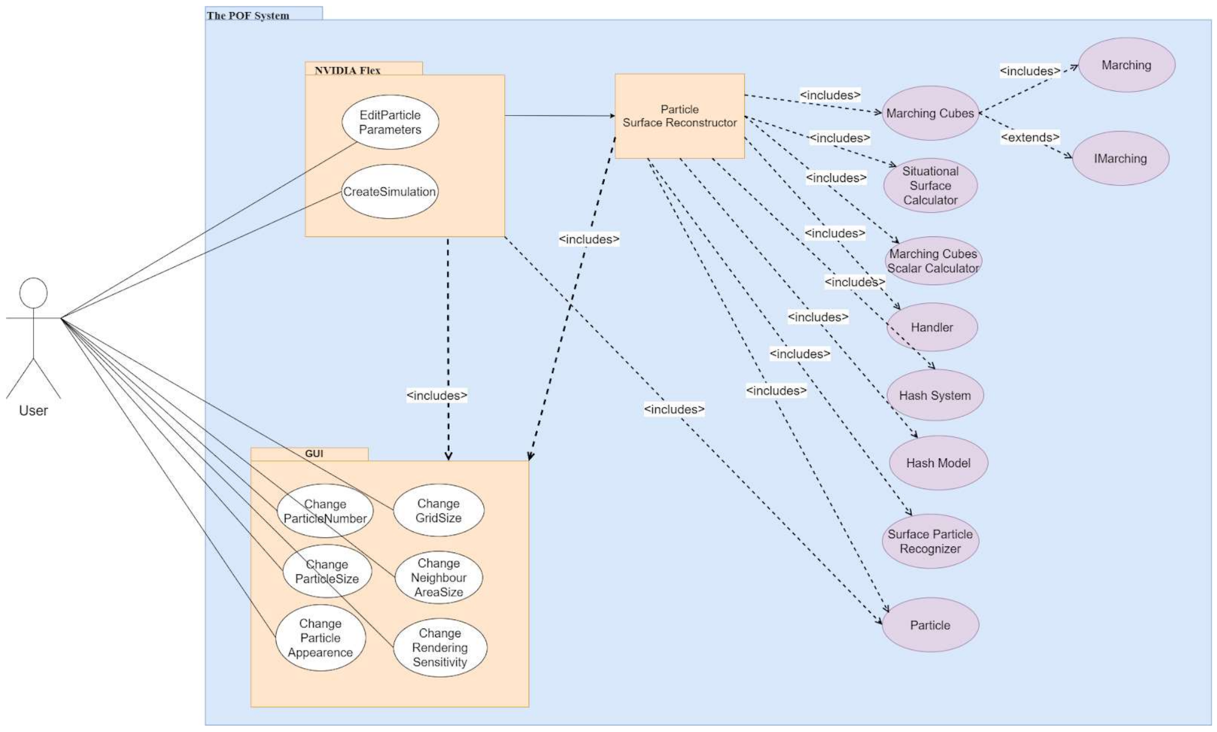

Fig 1: Use Case Diagram

2.2 *Sequence Diagram*

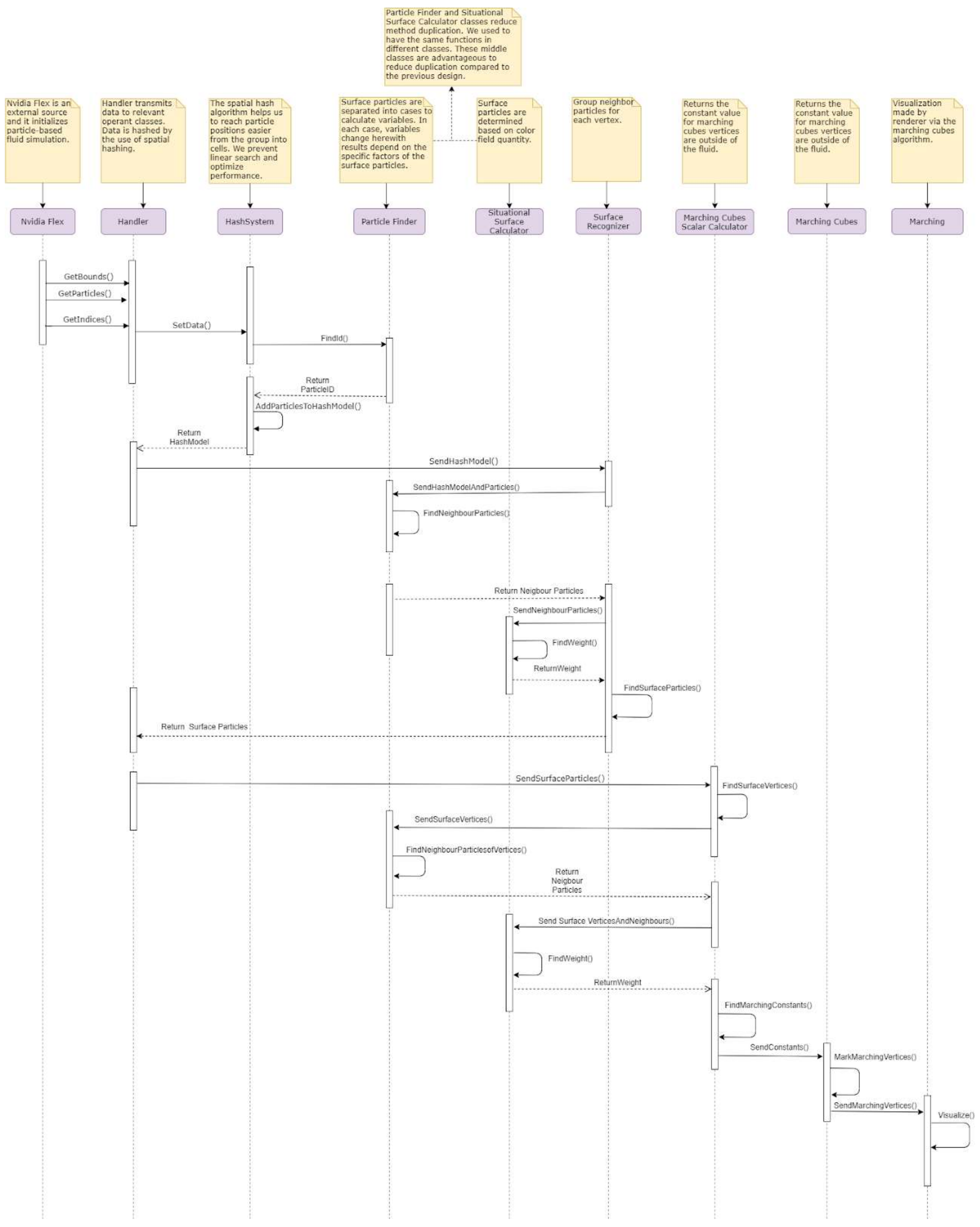
Particle Finder and Situational Surface Calculator classes reduce method duplication. We used to have the same functions in different classes. These middle classes are advantageous to reduce duplication compared to the previous design.
Nvidia Flex is an external source and it initializes particle-based fluid simulation.
Handler transmits data to relevant operant classes. Data is hashed by the use of spatial hashing.
The spatial hash algorithm helps us to reach particle positions easier from the group into cells. We prevent linear search and optimize performance.
Surface particles are separated into cases to calculate variables. In each case, variables change herewith results depend on the specific factors of the surface particles.
Surface particles are determined based on color field quantity.
Group neighbor particles for each vertex.
Returns the constant value for marching cubes vertices are outside of the fluid.
Returns the constant value for marching cubes vertices are outside of the fluid.
Visualization made by renderer via the marching cubes algorithm.
Nvidia Flex
Handler
HashSystem
Particle Finder
Situational Surface Calculator
Surface Recognizer
Marching Cubes Scalar Calculator
Marching Cubes
Marching
GetBounds()
GetParticles()
GetIndices()
SetData()
FindId()
Return ParticleID
AddParticlesToHashModel()
Return HashModel
SendHashModel()
SendHashModelAndParticles()
FindNeighbourParticles()
Return Neigbour Particles
SendNeighbourParticles()
FindWeight()
ReturnWeight
FindSurfaceParticles()
Return Surface Particles
SendSurfaceParticles()
FindSurfaceVertices()
SendSurfaceVertices()
FindNeighbourParticlesofVertices()
Return Neigbour Particles
Send Surface VerticesAndNeighbours()
FindWeight()
ReturnWeight
FindMarchingConstants()
SendConstants()
MarkMarchingVertices()
SendMarchingVertices()
Visualize()

Fig 2: Sequence Diagram

3.0 General Description

The requirement specification document provides a high-level description of the requirements of the POF system. The documentation describes the operation of sections in the POF system and explains its roles.

3.1. System Functions

The POF system shall retrieve the position data of the particles and AABB which is created by the NVIDIA flex particle-based fluid simulation system. The POF system computes the colour field quantity of each particle and marks the surface particles and restore their vertices. The POF system calculates the cell id for each particle and calculates the scalar value of how the particles affect each other by using [7].

3.2. NVIDIA Flex

NVIDIA Flex is a particle-based simulation technique for real-time visual effects. It is an outside source tool for our project. NVIDIA Flex is used for creating particles data to process in our algorithm. NVIDIA Flex is an obligatory external tool since we cannot create fluid simulation from scratch, and it is not our project aim either. The aim of the POF system is enhancing the performance of the already existed particle-based fluid system.

4.0 Functional Requirements

This section describes the functional requirements and stages of the POF system.

4.1 *Retrieve Particle Data*

The POF system takes the particles from NVIDIA Flex which creates the particles and particle attributes such as radius adhesion, damping, and restitution. NVIDIA flex creates an axis-aligned boundary box by looking at the coordinates of particles. AABB is a necessary preliminary step for dividing into cells.

4.2 *Divide into Cells*

The reason for using AABB is to make the search algorithm is more efficient. Axis aligned bounding box is divided into cubic cells to analyze the situation of the particle. Cells are divided by the ratio of one-eight times of radius for the Marching cubes algorithm [6] initialization. Cubes are an easy way to reach vertex information. Instead of holding eight vertex data, the system holds a cube position and it is a memory-efficient way. The POF system uses these cells to calculate the scalar values of the particles inside the cells by using [7].

4.2.1 Zhu and Bridson

The research paper written by Zhu and Bridson [7] offers an alternative to simulate liquids. The paper mentions surface reconstruction from particle and gives the functions and formulas to apply the method. Zhu and Bridson [7] calculate a scalar value of vertices that outside of the fluid to send marching cubes to visualize.

4.2.2 Mathematical Equations

Mathematical equations given below are belongs to the Zhu and Bridson research paper. We must implement these equations as code script in our POF system. Brief explanations are given below for each equation.

$$\varphi(x) = |x - x_0| - r_0 \qquad \textbf{(1)}$$

This equation finds signed distance value of a particle according to its neighbours. X0 is all neighbours average position as one dimension. X is a single particle position. The radius of a particle is subtracted because X values represents the centre of the particles. is how much close to their neighbours.

$$\varphi(x) = |x - \bar{x}| - \bar{r} \qquad \textbf{(2)}$$

This is the previous same formula with a small change. X bar in abstract represents average position data of the neighbour particles of a particle in a specific range. R bar represents the average radii of all particles. In our system all particles have the same value so we can replace it with a constant value.

$$\bar{x} = \sum_i w_i x_i \qquad \textbf{(3)}$$

This equation is used to calculate x bar. For each particle, position data is multiplied with weight data and the sum of these values are equals to the x bar.

$$w_i = \frac{k(|x - x_i|/R)}{\sum_j K(|x - x_j|/R)} \qquad \textbf{(4)}$$

We know the variables in the absolute value. K is a kernel function;

k(s) = max (0 , $(1 - s^2)^3$). R is the radius of the volume around the particle which is x as the previous equations. In Zhu and Bridson [7] writers recommend the R value as twice of the particle spacing. In the denominator, we find the distance as we mentioned in other relevant equations but this time, we add these values. It explains the particle weight which the effect of a particle is compared to other neighbour particles.

4.2.3 What is Kernel Function?

The parameters of a kernel function can be anything such as two integers or two vectors, trees whatever provided that the kernel function knows how to compare them.

4.2.4 What is Weight?

Weight function calculates a single particle how much affected by the summation of every other particle in a specific range in space. Represented as 'w' in mathematical equations.

4.2.5 Particle Classification

Particle classification is one of the main reasons why the POF system offers a better alternative. Grouping particles and by searching and calculating from these bigger parts of the fluid volumes make the system more efficient and faster in terms of computation.

4.3 ***Surface Recognition***

The surface recognizer algorithm detects surface particles and their cells so we can discard inactive cells and focus on the surface particles. This method makes the system more efficient and results with better performance by discarding unnecessary cells. The POF system finds each particle.

$$\varphi(x) = |x - \bar{x}| - \bar{r}f \qquad \textbf{(5)}$$

This equation helps us to decide whether the particle is a surface particle. X is the particle. X bar is the neighbour particles of x in a specific range. If φ(x) = 0, we can say that particle is on the surface. r is the radius. f is a scalar value and there are other calculations mentioned to find f. However, we do not consider the f value while we are implementing in this equation because of the project deadline.

$$\bar{x} = \frac{\Sigma_j x_j k(|x-x_j|/R)}{\Sigma_j k(|x-x_j|/R)} \quad \textbf{(6)}$$

This equation finds the effect of a particle as a weight in a system. Equation finds a particle distance and divides to the other neighbour particles average distance. R is the specific range that we look into neighbours of x. xj is the particles that stay in the range of R. k is the kernel function which is explained in section 4.3.2 Kernel Function.

4.3.1 Colour Field Quantity

Colour field quantity is a mathematical function that calculates a particle that is affected by the other particles. Because of this function, the POF system determines whether a particle is a surface particle.

$$c_S(r) = \sum_J m_J \frac{1}{p_J} W(r - r_J, h) \quad \textbf{(7)}$$

4.3.2 Kernel Function

The kernel function is necessary for kernel and particle approximation of a field function and its derivatives. Kernel function formula:

$$\text{k(s)} = \max(0, (1 - s^2)^3) \quad \textbf{(8)}$$

4.3.3 ***Weight Function***

A gradient is a vector-valued function that computes a particle verge which direction.

4.3.4 Marking Cells and Vertices

We must find the particles in cells and calculate how every particle affects the other neighbour particles. Vertices of these particles are marked as well.

4.4 ***Performance***

This requirement can be accepted as both kinds of requirement types. The project does not give this requirement as mandatory, but to achieve performance has significant importance. This requirement explained in non-functional requirements.

5.0 Non-Functional Requirements

Non-Functional requirements are listed in the table below. These requirements are the main goals of the POF system. Some requirements may not be satisfied because they are not promised to realize since our project is research and development based.

Non-Functional Requirements	**Description**
Efficiency	The aim of the POF system is efficient memory usage. For instance, because of the hash system, we divide 3D space into cells and instead of searching a particle linearly, we only search the particles a cell which is a more efficient way to search particles.
Performance	The system's performance should be increased with the POF system. Due to the POF system, particle simulation has a higher fps rate, or it can be run at lower-end devices. The existed methods will be checked whether it can be developed or not.
Usability	Similar fluid systems are developed on many platforms. However, our project can execute in the Unity game engine which is supported on Windows and macOS. User's Computer must be met with specified software and hardware requirements.
Testability	The POF system is testable with different conditions such as different particle attributes or different scenes. Controllability is not possible fully because NVIDIA Flex inter-communicates with the GPU by using parallel programming methods.

Table 1: Non-Functional Requirements

6.0 Glossary

Term	**Description**
API	Application Programming Interface.
AABB	Acronym for Axis Aligned Boundary Box. Bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set.

Cell	Axis aligned bounding box is divided into small identical cubes.
Colour field quantity	Function that calculates how each particle is affected by all the other particles.
CPU	Central Processing Unit.
GPU	Graphic Processing Unit.
Gradient	The directional derivative of a scalar field gives a vector field directed towards where the increment is most, and its magnitude is equal to the greatest value of the change.
Grid	Series of vertical and horizontal lines that are used to subdivide AABB vertically and horizontally into cells in three-dimensional space.
Iso-surface	An isosurface is a 3D surface representation of points with equal values in a 3D data distribution which is the 3D equivalent of a contour line.
Marching Cubes	Marching cubes is a computer graphics algorithm, published in 1987 for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field.
Mesh	A mesh consists of triangles arranged in 3D space to create the impression of a solid object.
NVIDIA Flex	NVIDIA Flex is a particle-based simulation technique for real-time visual effects.
OPENGL	Open Graphics Library is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics.
POF	Performance-Optimized fluids.
Polygonal Mesh	A polygon mesh is the collection of vertices, edges, and faces that make up a 3D object.
SPH	Smoothed-Particle Hydrodynamics
Unity 3D	Unity is a cross-platform game engine developed by Unity Technologies. Unity is used for developing video games and simulations for consoles and mobile devices.
Visual Studio	Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft.

Table 2: Glossary

7.0 User Characteristics

The performance-optimized fluid system can be used by anyone who has an interest in particle-based fluid simulation.

8.0 General Constraints

A D3D11 capable graphics card with the following driver versions:

NVIDIA: GeForce Game Ready Driver 372.90 or above.

AMD: Radeon Software Version 16.9.1 or above.

To build the demo at least one of the following is required:

Microsoft Visual Studio 2013 or above.

G++ 4.6.3 or higher

CUDA 8.0.44 or higher

DirectX 11/12 SDK

References

1. **[AIA12]** Akinci, G., Ihmsen, M., Akinci, N. and Teschner, M. (2012). Parallel Surface Reconstruction for Particle-Based Fluids. Computer Graphics Forum, 31, 1797-1809.

2. **[BP94]** Paul Bourke 1994, Marching Cubes, viewed 1 May 2020, <**http://paulbourke.net/geometry/polygonise/**>

3. **[MCG03]** M. Müller, D. Charypar, and M. Gross (2003). Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03), 154–159.

4. **[PTB03]** Premžoe, S. , Tasdizen, T. , Bigler, J. , Lefohn, A. and Whitaker, R. T. (2003). Particle-Based Simulation of Fluids. Computer Graphics Forum, 22, 401-410.

5. **[TH03]** Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M.H. (2003). Optimized Spatial Hashing for Collision Detection of Deformable Objects. VMV.

6. **[WH87]** William E. Lorensen and Harvey E. Cline. (1987). Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphics. 21, 163-169.

7. **[ZB05]** Zhu, Y., & Bridson, R. (2005). Animating sand as a fluid. (New York, NY, USA, 2005) *ACM Trans. Graph., 24*, 965-972.

8. Design Specification Document revision 3.0 (DSD 3.0)