**A picture containing clipart

Description automatically generated COMP4910 Senior Design Project 1, Fall 2019**

**Advisor: Gizem Kayar**

**POF: Performance Optimized Fluid Requirements Specification Document**

**4.12.2019**

**Revision 2.0**

**By:**

**Baran Budak-15070001012**

**Cihanser Çalışkan-16070001020**

**İsmail Mekan-15070001048**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Explanation** |
| **1.0** | **03.11.2019** | **Initial requirements** |
| **2.0** | **16.11.2019** | **Requirements Model** |

**Contents**

Revision History2

Contents3

1.0 Introduction4

1.1 Purpose4

1.2 Scope4

1.3 Overview4

2.0 Diagrams **5**

2.1 Sequence Diagram5

2.2 Use Case Diagram6

2.2 Activity Diagram6

3.0 General Description**7**

3.1 System Functions7

3.2 NVIDIA Flex7

4.0 Functional Requirements **7**

4.1 Non-Functional Requirements8

5.0 Glossary9

6.0 User Characteristics 9

7.0 General Constraints **9**

8.0 References **9**

**1.0 Introduction**

**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 simulation utilizing running it faster while occupying less memory of the computer. 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.

**1.2. *Scope***

The POF system shall help to increase performance for simulating fluids. The system reduces the necessary computation operation for particles during the simulation. Initially, the POF system runs the NVIDIA flex because the POF system needs particle position data. The POF system works with the Unity engine for visualization but another program can be used for simulation.

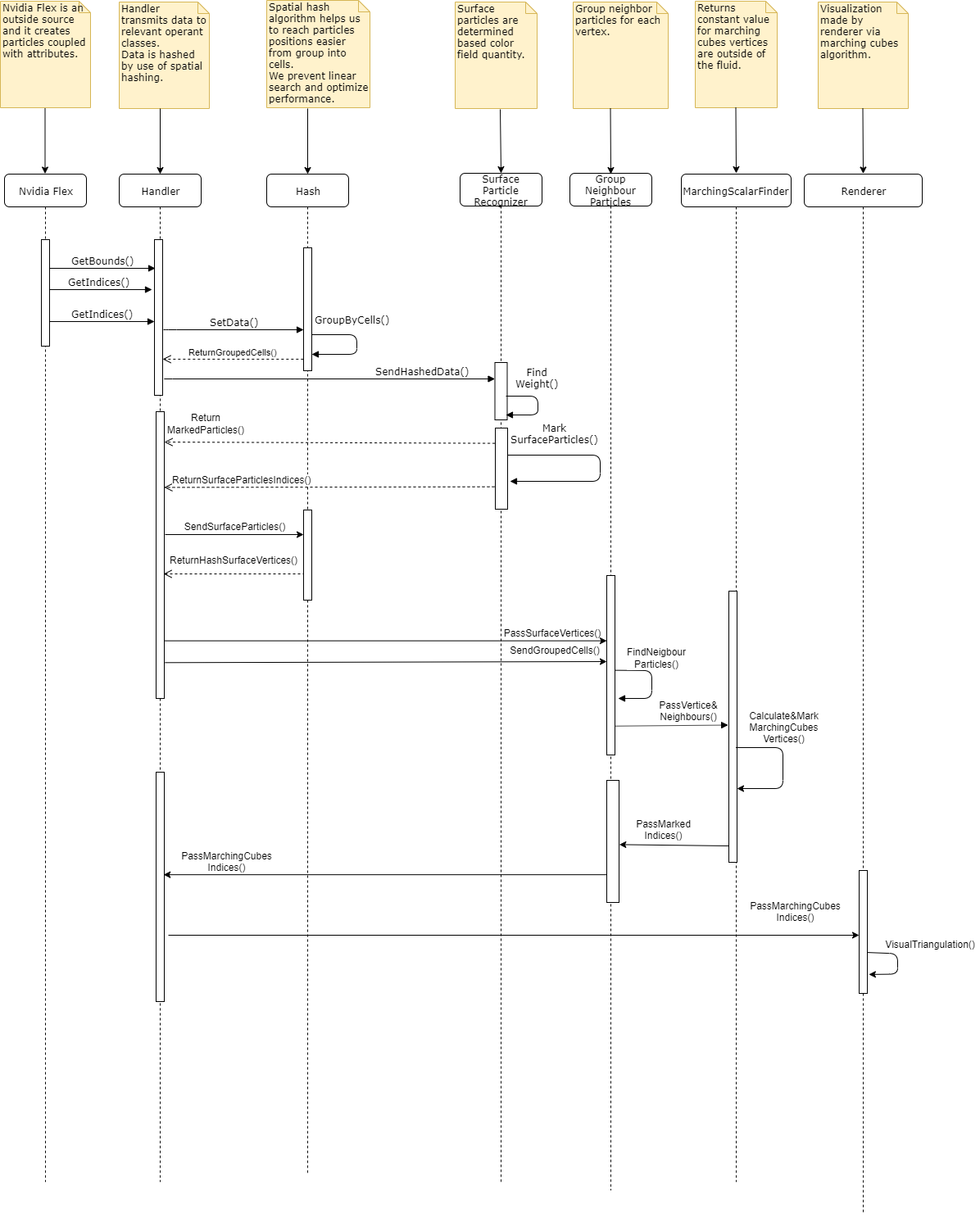
The POF system computes the color field quantity of each particle and marks all the surface particles. Surface particles are calculated and marked for the 2r distance which is the two times of a particle radius and each particle is the same. Marked vertices control by the handler in our code. For every surface vertex, we compute every small cell in the 4 times of radius area in the axis-aligned bounding box (AABB). After we find the cells, we find the particles in those cells and calculate how every particle affects the other particles as a scalar value of the vertex which the method is defined in [ZB05]. Lastly, for all the vertex data, we draw a cube and check those eight vertices of a cube and if the cell is on the surface, we pass the information of the vertices for the triangulation stage.

**1.3. *Overview***

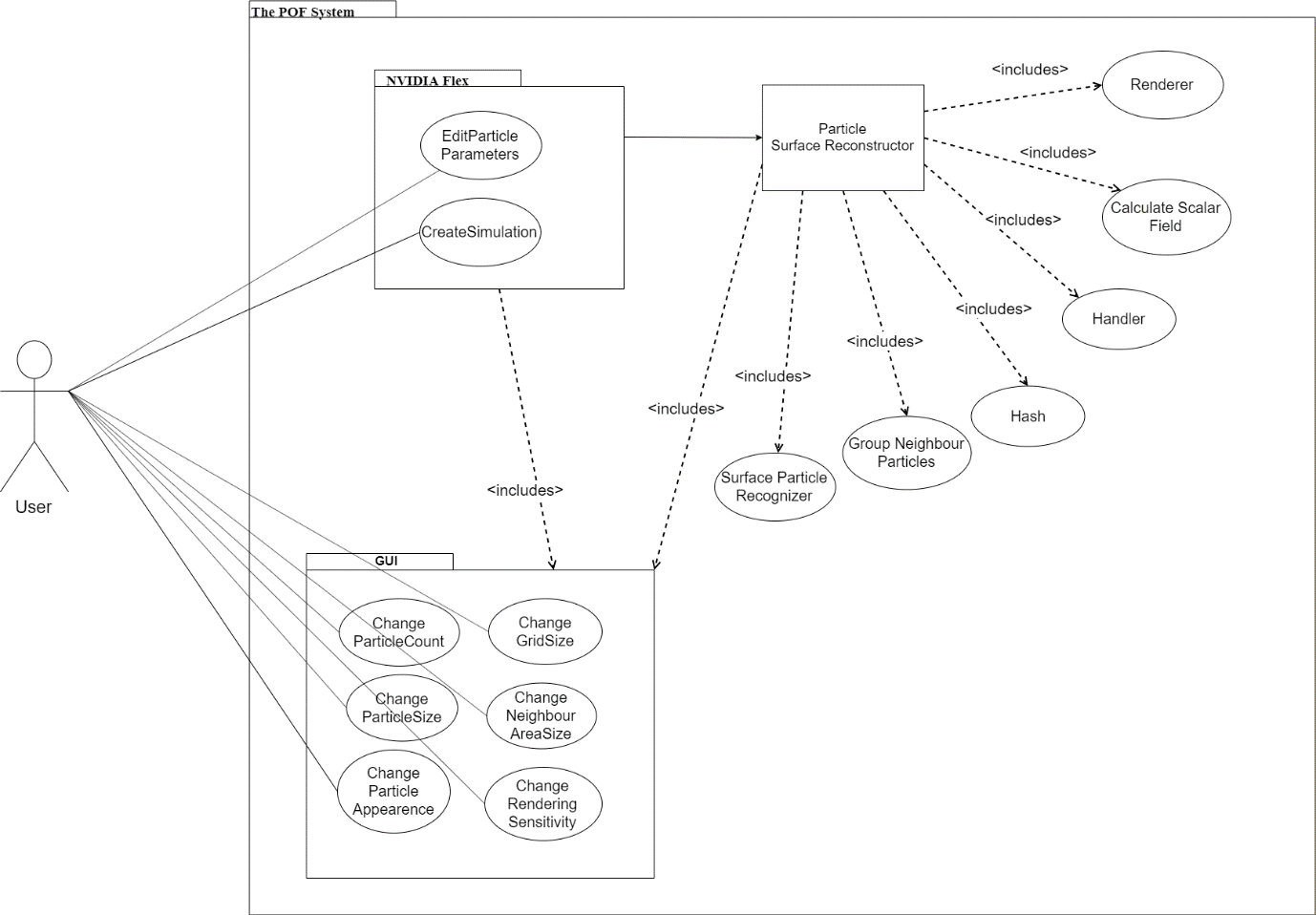
This document describes the POF system. Requirements specification document involves diagrams that define user roles in the system and more importantly, explaining how the system operates in the background. The document mostly focuses on the specified requirements. System functions are defined by expressing 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**

**2.1. *Sequence diagram***



**2.3. *Use case diagram***



**3.0 General Description**

The requirement specification document is used to provide a high-level description of the POF system. The documentation describes the mechanism and explain their 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 color 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 [ZB05]. The system visualizes the particles by using the Marching cubes algorithm [WH87].

***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 simulation enhancement. We will use NVIDIA flex for creating particles and using particle data to process it for our algorithm. Besides, it is unnecessary to strive with particle physics for our project because it is aimed that enhancing the performance of the already existed particle-based fluid system by surface reconstruction and it is not aimed to create a fluid simulation system from scratch.

1. **Functional Requirements** 
   * 1. ***Retrieve the 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.
     2. ***Divide into cells:*** The reason of using AABB is to make search algorithm is more efficient. Axis aligned bounding box is divided into cubic cells to analyze the particles situation. Cells are divided by the ratio of one-eight times of radius for Marching cubes algorithm [WH87]initialization. Cubes are an easy way to reach vertex information. Instead of holding eight vertex data, system holds a cube position and it is a memory efficient way. The POF system use these cells to calculate the scalar values of the particles inside the cells by using [ZB05].
2. **Zhu and Bridson & Surface Recognition**

Zhu and Bridson [ZB05] calculate a scalar value of vertices that outside of the fluid to send marching cubes to visualize.

1. **Mathematical Equations**

k is a kernel function; k(s) = max ()

1. **What is kernel function?**

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

1. **What is weight?**

Weight function calculates a single particle how much affected by the summation of every other particle.

1. **Importance of particle classification for memory efficiency**

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.

* + 1. ***Surface recognition:*** The algorithm detects surface particles and their cells so we can discard inactive cells and focus on the surface particles. This method makes system more efficient and results with better performance by discarding unnecessary cells. The POF system finds each particle

1. **Color field quantity**

Color 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.

//Formul koy

1. **Kernel function**

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

//Formul ekle

1. **Weight function**

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

//Formül ekle

**D) Marking cells and vertices**

When the cells are found, afterward we find the particles in those cells and calculate how every particle affects the other particles as a scalar value of the vertex in which the method is defined.

* + 1. ***Animating Sand as a Fluid:*** Animating Sand as a Fluid is a research paper that explains animating sand as a fluid. However, the research paper also mentions about it can be used in fluids which is the significant part that it will use in the project. The paper mentions surface reconstruction from particle and gives the functions and formulas in order to apply the method.

1. **Weight function**
   * 1. ***Marching cubes:*** The algorithm is used for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field. In this project, the marching cubes algorithm is used with the [ZB05]algorithm. [ZB05]algorithm is used in the marching cubes algorithm to get better visual outputs.
     2. ***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.

**4.1 Non-functional Requirements**

|  |  |
| --- | --- |
| **Non-Functional Requirements** | **Description** |
| Efficiency | The aim of the POF system is efficient memory usage. |
| Performance | The system’s performance should be increased after the application POF to the 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 in OpenGL or other various platforms. However, our project will be deployed into the Unity game engine which is supported on Windows and macOS. |

1. **Glossary**

|  |  |
| --- | --- |
| **Term** | **Description** |
| API | Acronym for Application Programming Interface. |
| Axis Aligned Bounding Box (AABB) | 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. |
| Color field quantity | It is a functionthat 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. |
| 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 | An Acronym stands for the performance-optimized fluid system. |
| Polygonal Mesh | A polygon mesh is the collection of vertices, edges, and faces that make up a 3D object. |
| 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. |

1. **User Characteristics**

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

**7.0 General Constrains**

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.

In order 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

**8.0 References**

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

[AIA12] https://cg.informatik.uni-freiburg.de/publications/2012\_CGF\_surfaceReconstructionSPH.pdf

**[BP94]Paul Bourke 1994, Marching Cubes, viewed 1 December 2019, <**[**http://paulbourke.net/geometry/polygonise/**](http://paulbourke.net/geometry/polygonise/)**>**

**[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.**

[PBS03] http://www.sci.utah.edu/~tolga/pubs/ParticleFluidsHiRes.pdf

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

[OSH03]http://matthias-mueller-fischer.ch/publications/tetraederCollision.pdf

**[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.**

[WH87] https://www.researchgate.net/publication/202232897\_Marching\_Cubes\_A\_High\_Resolution\_3D\_Surface\_Construction\_Algorithm

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

[ZB05] https://www.cs.ubc.ca/~rbridson/docs/zhu-siggraph05-sandfluid.pdf