POF: Performance Optimized Fluid System Requirements Specification

2.11.2019

Revision 1.0

By:

Baran Budak-15070001012

Cihanser Çalışkan-16070001020

İsmail Mekan-15070001048

Contents

Revision History

Revision	Date	Explanation
1.0	03.11.2019	Initial requirements

1.0 Introduction

This section introduces the requirement specification document for the Performance Optimized Fluid System. It provides the purpose and scope of the system.

1.1. Purpose

Purpose of the requirement specification document is describing the functions and specified requirements for performance optimized fluid systems. The system should help increasing the efficiency of running simulation by means of running it faster while occupying less memory of the computer.

The aim of the performance optimized fluid system is to enhance the performance of the existing fluid simulation system by reconstructing the surface and doing it by means of benefiting from research papers (other various research papers applied along the project). It was decided to use unity game engine. However, it can be used another tool program such as unreal engine in order to simulate fluid system.

1.2. Scope

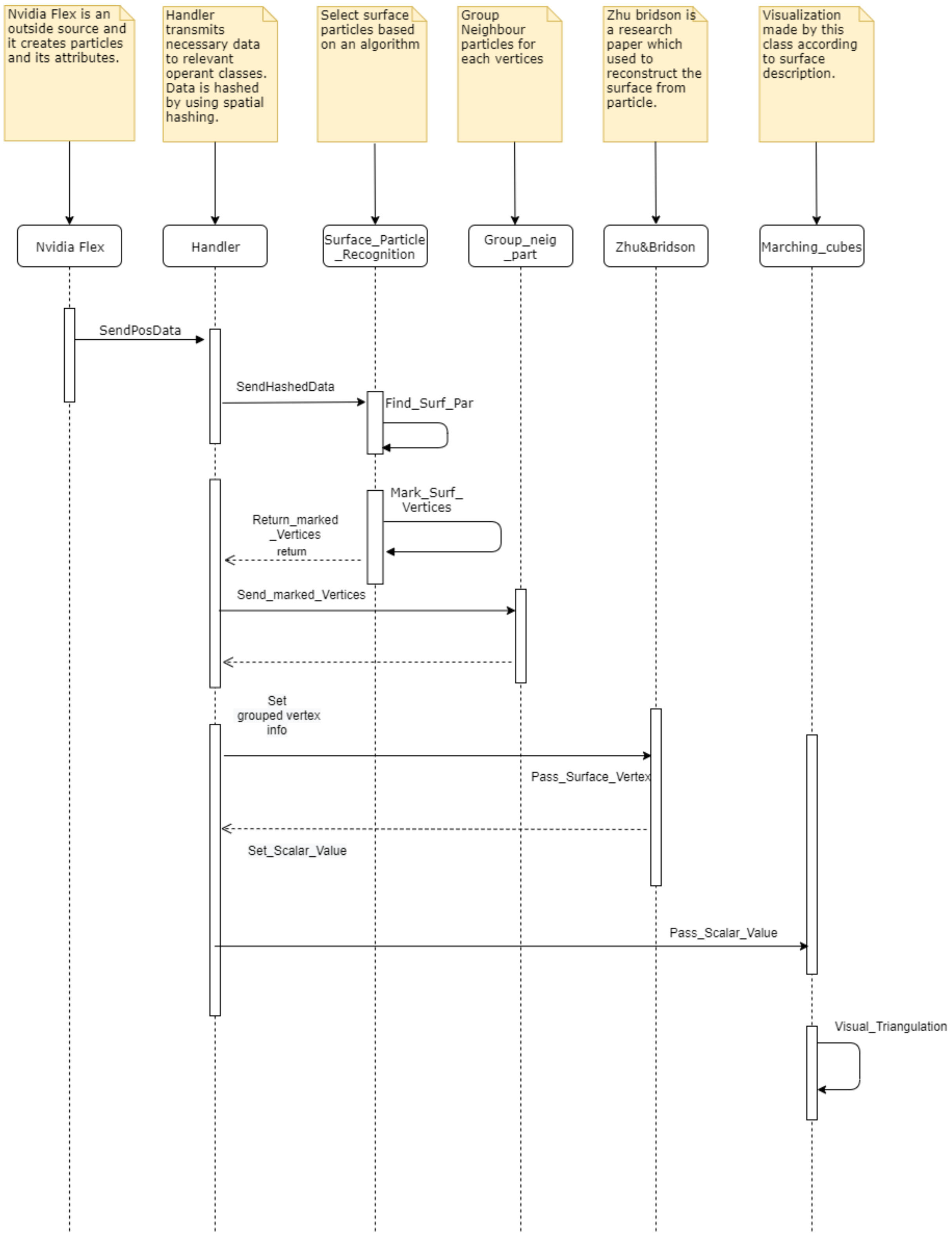
The POF system shall help to increase performance for simulating fluids. System reduces the necessary computation operation for particles during the simulation. System approaches as a whole rather than each one of particle. It takes the particles' position and necessary simulation data and POF and simplifies it the way that requires less computation.

1.3. Overview

This document provides a high-level description of the performance optimized particle-based fluid system. It identifies the involved users and helps to define their roles in system. The majority of this document focuses on the specified requirements. System functions are defined by expressing functional and non-functional requirements. NVIDIA flex is simply explained, and its usage is simply described. Use case and sequence diagrams are included to make it more understandable. User characteristics and constraints specifies that how system can work according to specific circumstances.

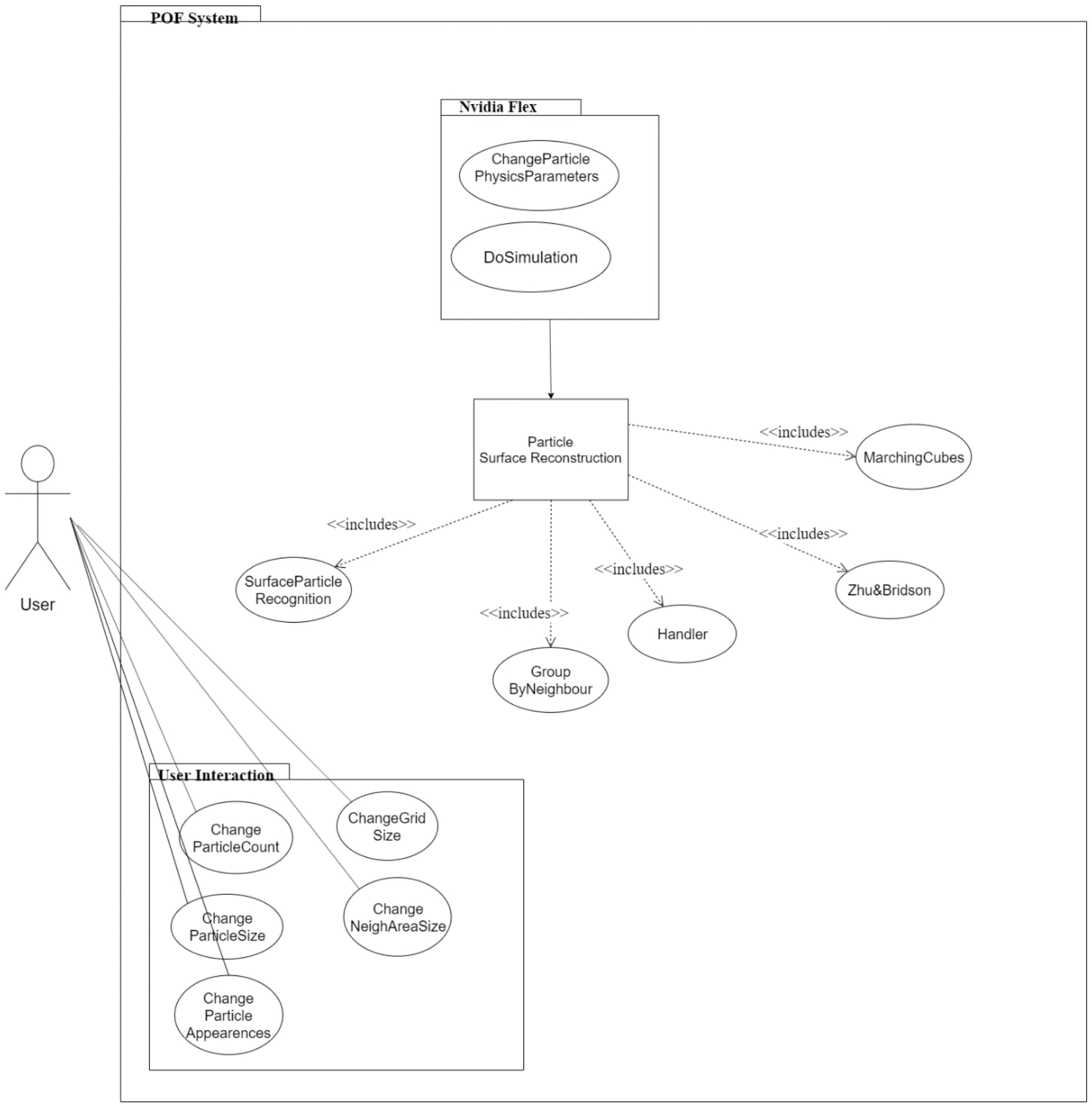
2.0 Diagrams

2.1. Sequence diagram



Nvidia Flex is an outside source and it creates particles and its attributes.
Handler transmits necessary data to relevant operant classes. Data is hashed by using spatial hashing.
Select surface particles based on an algorithm
Group Neighbour particles for each vertices
Zhu bridson is a research paper which used to reconstruct the surface from particle.
Visualization made by this class according to surface description.
Nvidia Flex
Handler
Surface_Particle _Recognition
Group_neig _part
Zhu&Bridson
Marching_cubes
SendPosData
SendHashedData
Find_Surf_Par
Mark_Surf_ Vertices
Return_marked _Vertices
return
Send_marked_Vertices
Set grouped vertex info
Pass_Surface_Vertex
Set_Scalar_Value
Pass_Scalar_Value
Visual_Triangulation

2.2. Use case diagram



3.0 General Description

This system is used to provide a high-level description of the system, as well as identify the users involved and help explain their roles.

3.1. System functions

The POF system shall retrieve the position data of the particles which is created by NVIDIA flex particle-based fluid simulation system. System detects the required boundaries that particles occupied in three dimensional space. Grids are created after the bounding box. System will detect the surface particles for the specific algorithm and use the hash function for accessing surface particles more effective.

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 construction and it is not aimed to create a fluid simulation system from scratch. In simulation, a stack of water consists from millions of water particles.

4.0 Functional Requirements

4.0.1 ***Take the particle data from NVIDIA flex:*** NVIDIA Flex creates the particles and we retrieve the data to another function which will use these data to apply our algorithm.

4.0.2 ***Find the boundaries*****:** Particles occupies a space in three-dimensional coordinate system. Particles' minimum and maximum boundaries should be found for the specifying the volume that particles occupy. Simply, boundaries imply an Axis Aligned Bounding Box which is memory efficient way of representing a volume. It is necessary preliminary step for the dividing into small cubes.

4.0.3 ***Divide into grids:*** Axis aligned bounding box should have divided into small cubes to analyse the particles and apply the algorithm. According to our researches, it was decided to use ratio of one-eighth of the particle radius, but it will change in during project time for testing performance and efficiency.

4.0.4 ***Surface recognition:*** The algorithm detects surface particles and their cells so we can discard inactive cells (for marching cubes of vertices). With this method we have more efficient and better performance by discarding unnecessary cells.

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

4.0.6 ***Marching cubes:*** The algorithm is used for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field. In this project, marching cubes algorithm is used with Zhu-Bridson algorithm. Zhu-Bridson algorithm is used in marching cubes algorithm in order to get better visual outputs.

4.0.7 ***Performance:*** This requirement can be accepted as both kind of requirement type. 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

***Performance*:** The system's performance should be increased after the application POF to the system. Due to POF system, particle simulation has 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 another various platform. However, our project will be deployed into Unity game engine which is supported on windows and macOS.

***Efficiency*:** The aim of the POF system is efficient memory usage.

5.0 User Characteristics

There are two kinds of people who will use our system. The performance optimized fluid system can be used from scientists which they can reverse engineer our system and apply it to another related research and development system however they want it. Besides, it can be used from students who have interest about surface reconstruction topics in college who researches about these issues.

6.0 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

7.0 References

1) Zhu and Bridson

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

2) NVIDIA Flex Documentation

https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/flex/index.html

3) Marching cubes

http://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

4) Surface recognition

https://cg.informatik.uni-freiburg.de/publications/2012_CGF_surfaceReconstructionSPH.pdf