**A picture containing clipart

Description automatically generated**

**COMP4910 Senior Design Project 1, Fall 2019**

**Advisor: Gizem Kayar**

**POF: Performance Optimized Fluids**

**High Level Design**

**Design Specifications Document**

**Revision 1.0**

**8.12.2019**

**By:**

**Baran Budak -15070001012**

**Cihanser Çalışkan -16070001020**

**İsmail Mekan -15070001048**

# Revision History

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Explanation** |
| 1.0 | 8.12.2019 | Initial high level design |

# Table of Contents

[Revision History 2](#_Toc501993332)

[Table of Contents 3](#_Toc501993333)

[1. Introduction 4](#_Toc501993334)

[2. POF System Design 4](#_Toc501993335)

[2.1. POF System Architecture 4](#_Toc501993336)

[2.2. POF System Structure 4](#_Toc501993337)

[2.3. POF Environment 4](#_Toc501993338)

[3. POF System Detailed Design: 4](#_Toc501993339)

[4. Testing Design 4](#_Toc501993340)

[References 5](#_Toc501993341)

# 1. Introduction

The purpose of the POF system is to research and apply existed methods to simulate fluids and looking for a better way to simulate it. Various methods will be implemented and tested during the research and development of this project. The main goal is making research and sharing our observations of the project results. One of the major project objectives is to reach a more efficient and better performance fluid simulation system but it is not promised because there is no certain way to achieve it and as mentioned, the project is mainly research-based.

The design is based on The POF system Requirements Specification Document, Revision 2.0 [1]

This design process conforms to the Requirements Specification Document and its diagrams. The project conforms to UML diagrams. Diagrams are describing the project to understand mainly operations of the POF system. Imperceptible parts of the POF system can be changed but the main functioning of the system will remain the same as before. If any change occurs during the development of the POF system, this document and diagrams will be changed.

The system architecture and overall high-level structure of the POF system is given in second section. Detailed design of all system functions and the user interface in terms of are methods of all classes will be given later in third section of this document.

# 2. POF Computer System High Level Design

# 2.1. POF System Architecture

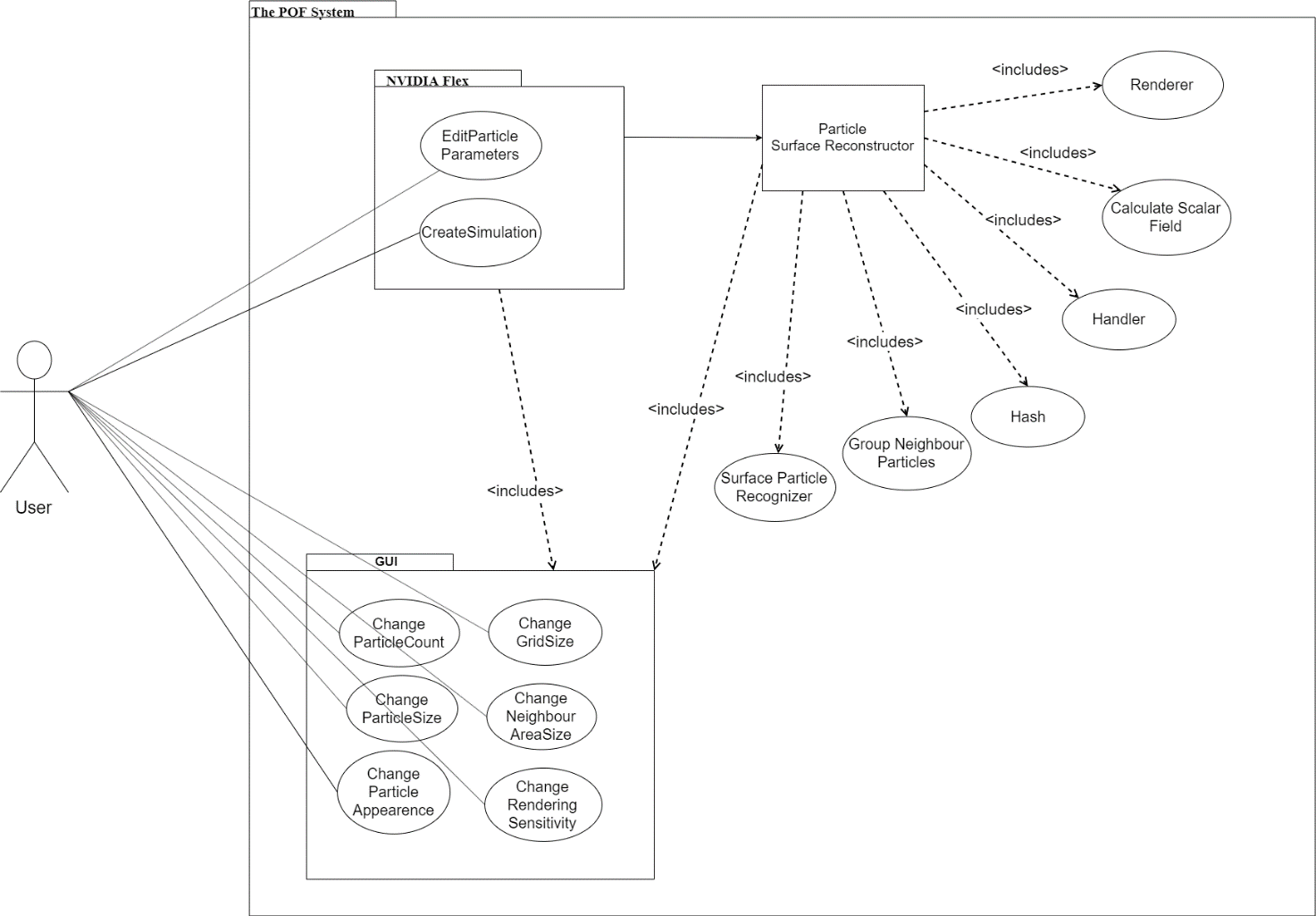
The POF system architecture works with NVIDIA Flex as an outsource asset. NVIDIA flex is mandatory because of the particle positions and axis aligned bounding box (AABB) in three dimensional space. System has a handler between the NVIDIA flex and the POF system. Initially, Flex starts the simulation and creates the particles and AABB. Handler retrieve this data to relevant classes.

/////////////////////////////////////////////////////

* Overall computer system architecture
* Architecture of large system components, if such components with different architecture/structure exist. For example, a client-server system may have complex client and server side components with different architectures. Similarly for peer-to-peer and/or large distributed systems/
* Discussion and justification of design decisions to choose specific architectures.

# 2.2. POF System Structure

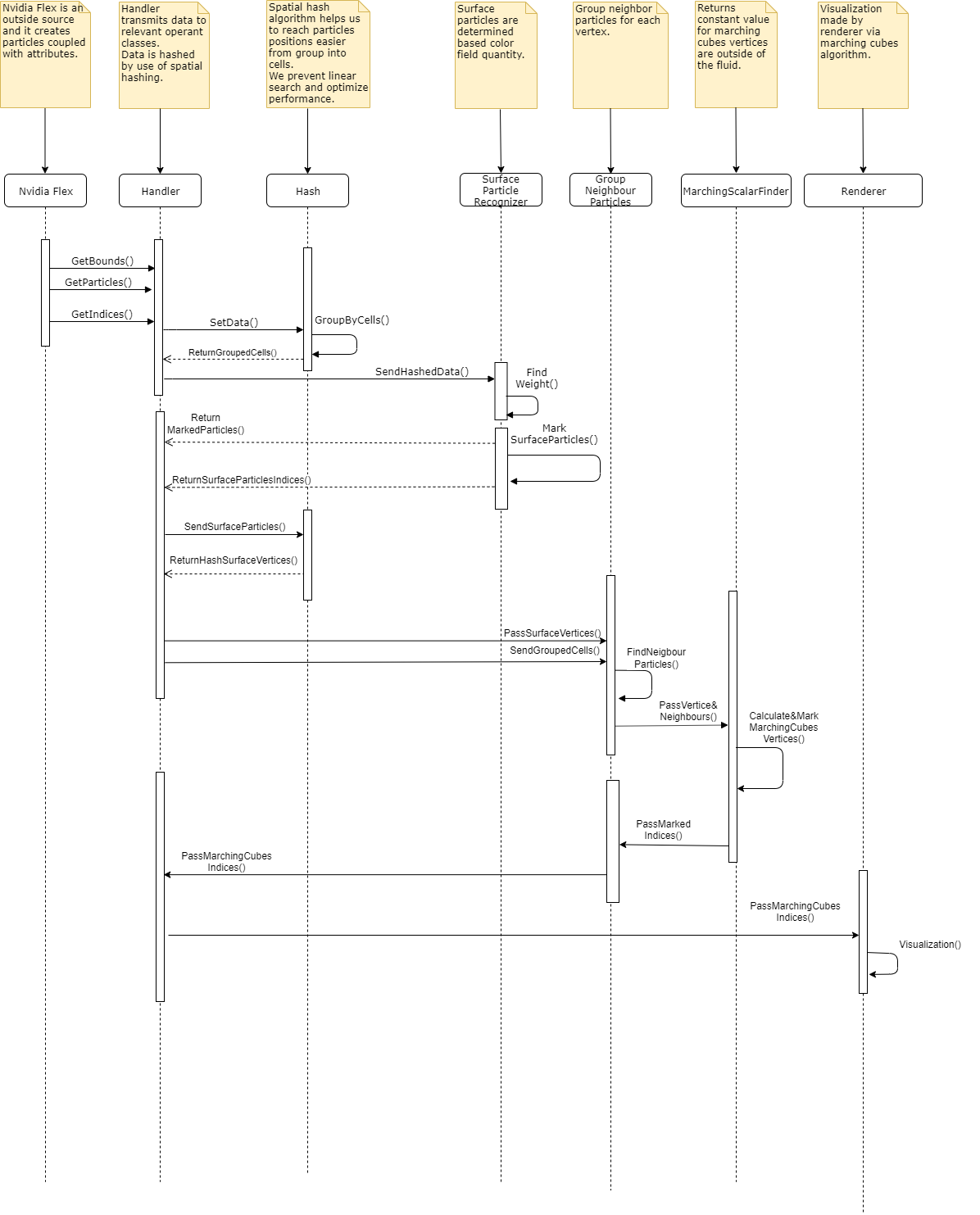
**2.2.1 Use Case Diagram**



|  |  |
| --- | --- |
| **Title** | **Description** |
| **Calculate Scalar Field** | Calculates a constant value of a particle in given range. |
| **Change grid size** | Interval of grid size of axis aligned bounding box can be changed with this function. |
| **Change neighbor area size** | Neighbor particle range of volume can be changed which affects the visualization of particles and changes the shape. |
| **Change particle appearance** | Particle color, texture and light settings can be changed with this function. |
| **Change particle count** | Particle number in the scene canbe edited. |
| **Change particle size** | Radiusof the particle can be changed with this function. Increase in the particle size will result slow performance compared to less particle size. |
| **Change rendering sensitivity** | Rendering sensitivitycan be changed with this function. If sensitivity increase, fluid visualization will be more precise however processing time will increase. |
| **Create Simulation** | NVIDIA Flex simulation initialize when this function called. |
| **Edit particle parameters** | Particle attributes can be edited by user from GUI. Parameters can be maximum particle number, particle size, friction, adhesion etc. |
| **Group Neighbour Particles** | Neighbour particles are grouped by looking a specific range. |
| **Handler** | Handler transmits data between layers and relevant classes. Handler manage data transmission. |
| **Hash** | Hashes the particle and cell position |
| **Renderer** | Visualize fluid by drawing the given polygons. |
| **Surface Particle Recognizer** | Surface particles marked andprocessed for the necessary calculations in this function. |
| **User** | Users can be anyone who has access to the program. |

**Table 1:** Description of the use case diagram

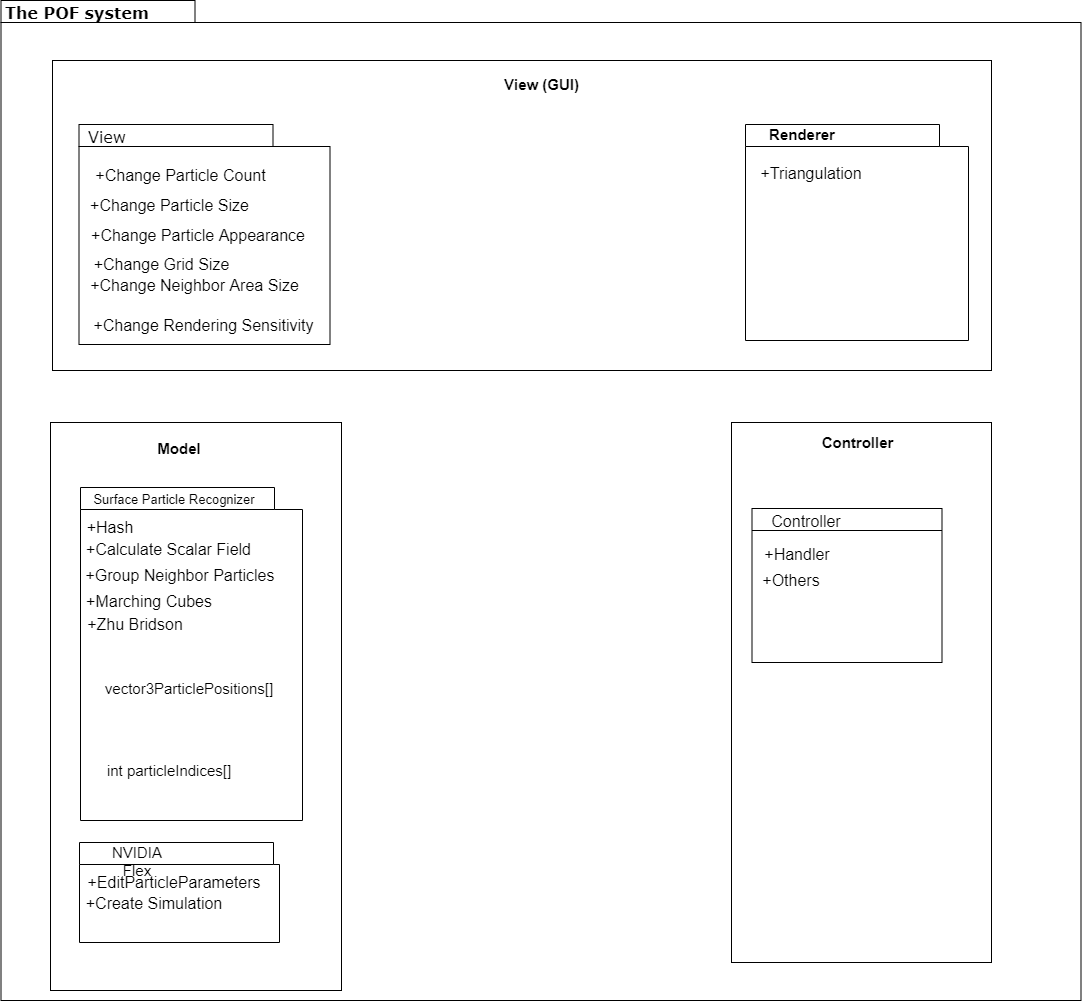
**2.2.2 Sequence Diagram**

****

**Description:**

Handler manages data between other sections. If any data have to transmit to another class, Handler execute this operation. NVIDIA flex is an already existed particle-based fluid system which is outsourced and initializes the fluid simulation. Hash applies a special algorithm and makes it easier and faster store and reach it to particle and cell data. Surface particle recognizer finds the surface particles by looking color field quantity. Group neighbor particles group neighbor particles for each vertex. Marching scalar values computes a constant value for the particle vertices that outside of the fluid. Renderer make the triangulation of the specified vertices and draws the fluid for each frame.

**2.2.3 Package Diagram**

****

**Description:**

Package diagram is based on MVC (model view controller) system. Model section consists of NVIDIA flex and surface particle recognizer. NVIDIA flex create simulation and can change hydrodynamic attributes of the particles. Surface particle recognizer is another package in the model section. Hash calculates cell id based on the boundaries of cells. Calculate scalar field returns a constant value of the particle. Group neighbor particles computes a particles weight by checking nearby particles in a specific range. Marching cubes algorithm determines which vertices will be triangulated. Zhu and Bridson is needed for the reconstructing the surface. Controller part has a handler which controls the data transmission and communication between sections. View section can change particle attributes such as particle count, size and appearance. Change grid size affects cells sizes. Change neighbour area size can affect the particle rendering. Changing rendering sensitivity of the rendering affects the appearance of the POF fluid system directly.

# 2.3. POF System Environment

The POF system environment constraints:

* D3D11 capable graphics card
* NVIDIA: GeForce Game Ready Driver 372.90 or above.
* AMD: Radeon Software Version 16.9.1 or above.
* Microsoft Visual Studio 2013 or above.
* G++ 4.6.3 or higher
* CUDA 8.0.44 or higher
* DirectX 11/12 SDK
* Windows 7 (64-bit) or higher.
* Unity 3D 2017.3 version or higher

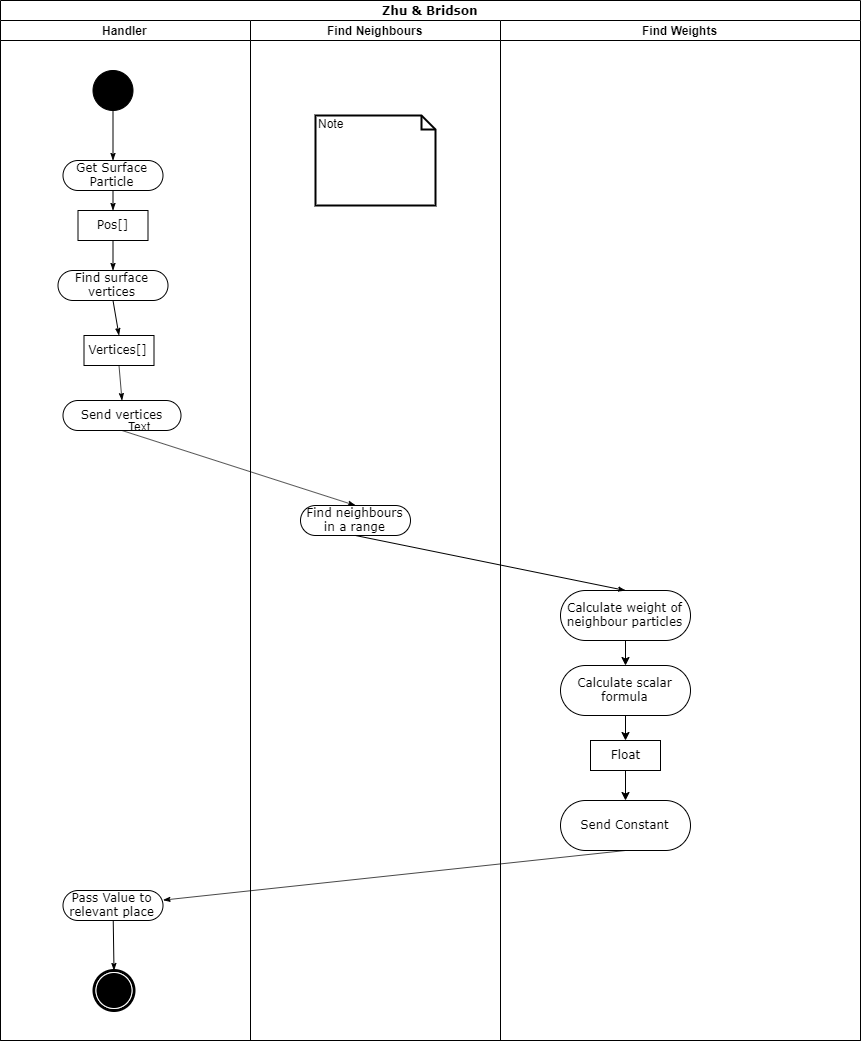
The main project made on the system:

* Operating System: Windows 10 (64-bit)
* Processor: Intel Core i7-4700 HQ CPU
* Memory: 16 GB RAM – DDR3L-1600 Mhz
* GPU: NVIDIA GeForce GTX850M 4GB DDR3

This system has low performance on this project because it can handle very small amount of particles. The optimal system should be workstation defined in final report [3].

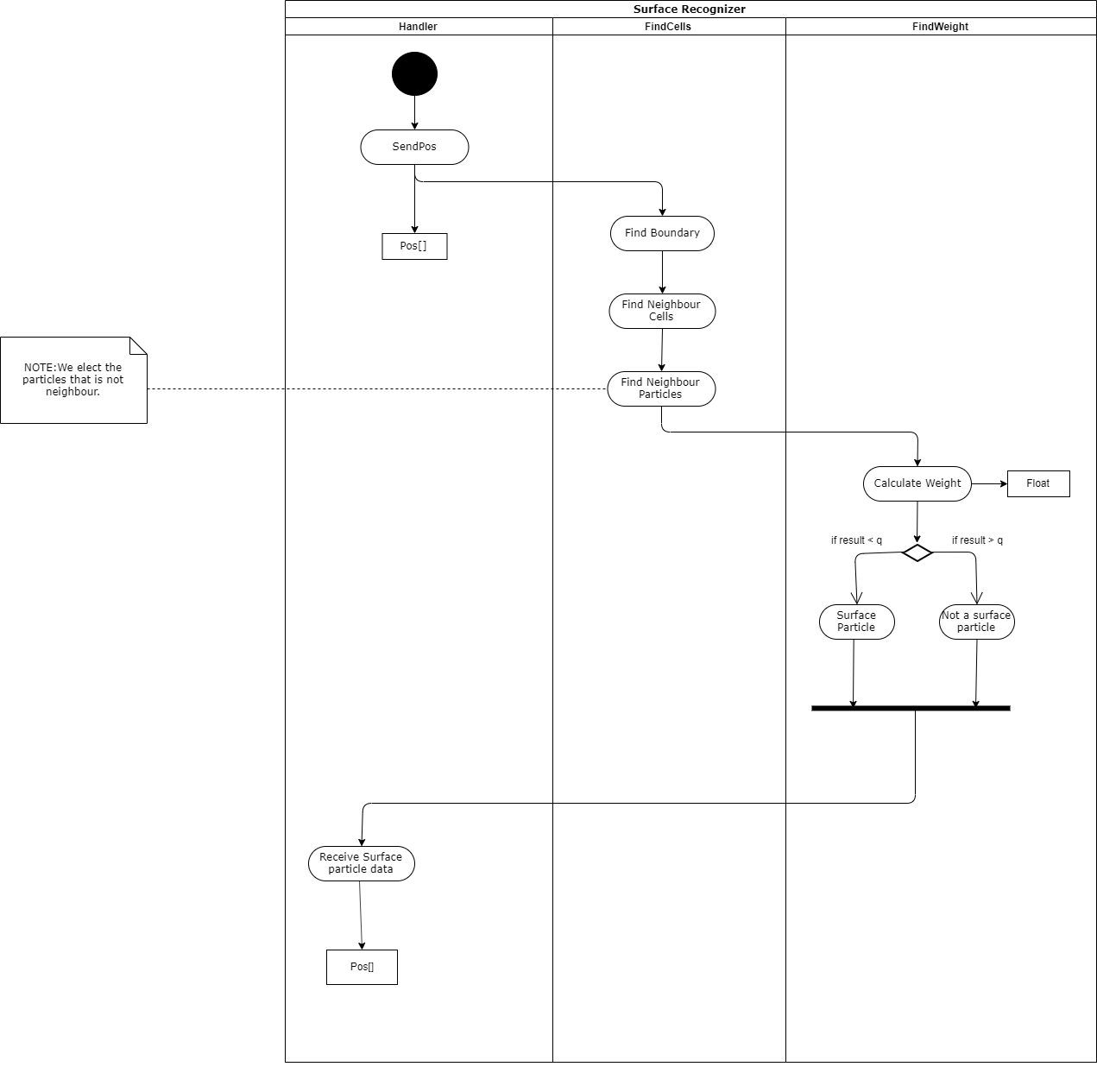
# 3. POF System Detailed Design:

**3.1 Activity Diagram of Zhu & Bridson**

****

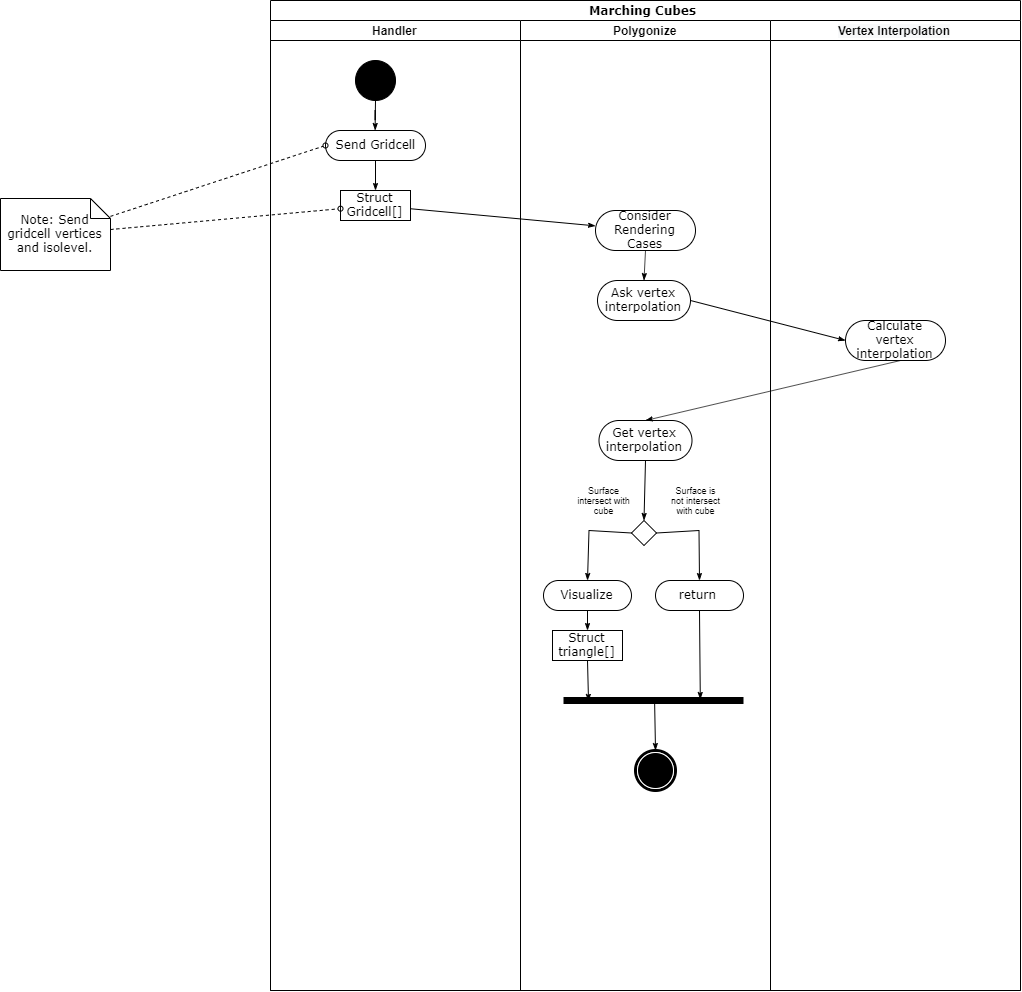
**Description:** Handler gets the surface particle from relevant sections. Handler receives the surface vertices and send it to find neighbors section which is responsible to find a particle neighbors in a specific range. Find neighbors passes neighbor particle data to find weight section of the

**3.2 Activity Diagram of Surface Recognizer**

****

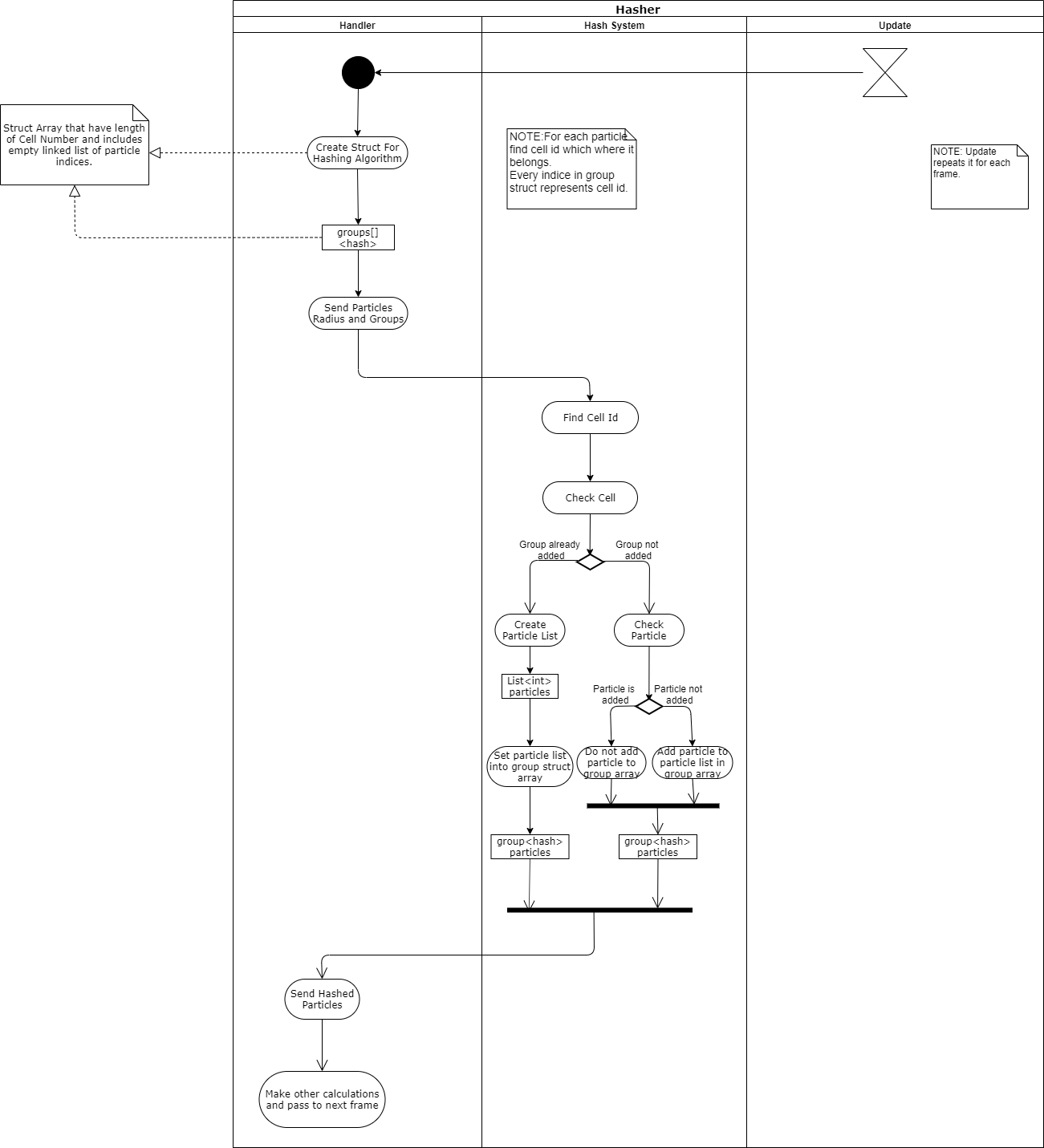
**Description:**

**3.3 Activity Diagram of Marching Cubes**

****

**Description:**

**3.4 Activity Diagram of Hasher**

****

**Description:**

# 4. Testing Design

Since our project is heavily research based, testing design is not handled in the earlier stages of our project. We will implement various methods that mentioned in research papers to our project. The Unity 3D game engine will be used to test results. The best methods will be determined according to the test results. The main standards are performance and efficiency. Memory and CPU usage is important. NVIDIA flex communicates with the GPU for efficiency of simulation. Detailed comparison tables will be sketched and elaborated in the second future.

# References

1. Requirement Specification Document revision 2.0 (RSD 2.0)
2. Use case, sequence, package diagrams in RSD 2.0
3. Final Report revision 1.0
4. Other references to additional documents, like other internal organizational documents, software project management documents, software design tool documents, etc.
5. References to additional bibliographic sources, like professional books, textbooks, handbooks, patents, standards, technical reports, journal/conference papers, etc.