YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

COMP4910 Senior Design Project 1, Fall 2019
Supervisor: Dr. Gizem Kayar

POF: Performance Optimized Fluid System

Final Report

By:

Baran Budak -15070001012

Cihanser Çalışkan -16070001020

İsmail Mekan -15070001048

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Last name	Student Number	Signature	Date
Baran Budak	15070001012		
Cihanser Çalışkan	16070001020		
İsmail Mekan	15070001048		

Project Supervisors:

Name, Last name	Department	Signature	Date
Gizem Kayar	Computer Engineering		

ACKNOWLEDGEMENTS

We would like to thank Gizem Kayar for discussions and suggestions in the development of the project. This project is supported by Yasar University Computer Engineering Department.

KEYWORDS

Term	Description
Cell	Axis aligned bounding box is divided into small identical cubes.
Color field quantity	It is a function that calculates how each particle is affected by all the other particles.
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 created by NVIDIA company.
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.
Spatial Hashing	Spatial hashing is a technique in which objects in a 2D or 3D domain space are projected into a 1D hash table allowing for very fast queries on objects in the domain space.

Table 1: Keywords

ABSTRACT

The POF system aims to provide surface identification and visualization in particle-based liquid simulations more efficiently and quickly. As a result of the first researches, the necessary algorithms for the system were determined and these algorithms were brought together and understood. Secondly, the system is divided into a structure with various algorithms and a main control panel that manages these algorithms. These infrastructures provide memory efficiency while gaining faster access to data. These structures are the spatial hashing algorithm, respectively. Defining the surface and determining the areas to be visualized and drawing on the screen. As the project is research-based, it is possible that the structures we have described will change and different structures will be replaced. Therefore, these algorithms and structures may change in the future.

ÖZET

POF sistemi parçacık temelli sıvı simülasyonlarında yüzey tanımlamasını ve görselleştirmesini daha optimize ve hızlı bir şekilde sunmayı amaçlar. İlk olarak yapılan araştırmalar sonucunda sistem için gerekli algoritmalar belirlenmiştir ve bu algoritmaların bir araya getirilerek anlaşılması sağlanmıştır. İkinci olarak sistem çeşitli algoritmalara ve bu algoritmaları yöneten ana bir kontrol paneline (controller) sahip bir yapıya bölünmüştür. Bu alt yapılar verilere daha hızlı erişim kazanırken hafıza verimliliği sağlar. Bu altyapılar sırasıyla konumsal karma algoritması, (Spatial hashing algorithm). Yüzeyin tanımlanması (Surface recognition) ve görselleştirilecek alanların belirlenip ve ekrana çizilmesidir (Marching cubes). Projenin araştırma temelli olmasından dolayı anlattığımız yapıların değişmesi ve yerlerine daha farklı yapıların gelmesi muhtemeldir. Bu yüzden ilerleyen zamanlarda bu algoritmalar ve yapılar değişebilir.

TABLE OF CONTENTS

Table 2: Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AABB	Axis Aligned Bounding Box. Bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set.
API	Application Programming Interface.
CPU	Central Processing Unit.
GPU	Graphic Processing Unit.
OPENGL	Open Graphics Library is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics.
POF	Performance Optimized Fluid
SSF	Screen Space Fluids Pro
MVC	Stands for Model View Controller. MVC is an application design model comprised of three interconnected parts (Model, View, Controller).

Table 3: List of acronyms/abbreviations

1. INTRODUCTION

This section gives main points about problem description, project goal and project output.

1.1. Description of the Problem

The main problem of the particle-based fluid simulation system is excessive numbers of the particles. There are millions of particles in a small number of liquids such as water. A particle is a rigid body sphere. Simulation applies physics to particles and these particles act as a liquid. Simulation having difficulties in calculations predicated on a surplus of particles. Indirectly, time and memory complexity increasing.

1.2. Project Goal

The main goal of the project researches whether there is a way to enhance fluid simulation. Increasing the efficiency and performance of an existing particle-based fluid simulation is a major goal. We aim to achieve these goals by implementing a variety of methods to the POF system such as using special structures to find store particles and visualize it by using various methods like the Marching cubes. In our project, there is no certain way because it is a research and development project and new more effective ways can be found during the project. Various methods and techniques will be researched and implemented while the project is in the development process.

1.3. Project Output

- Better performance.
- Better memory efficiency.
- Fluid-like appearance and behaviour.
- Testing of different algorithms for performance and efficiency.
- Higher frame rates per second.

2. DESIGN

This section describes about design of the POF system. High level and detailed designs are explained. Restrictions and conditions mentioned in this section.

2.1. High Level Design

2.1.1. Package Diagram

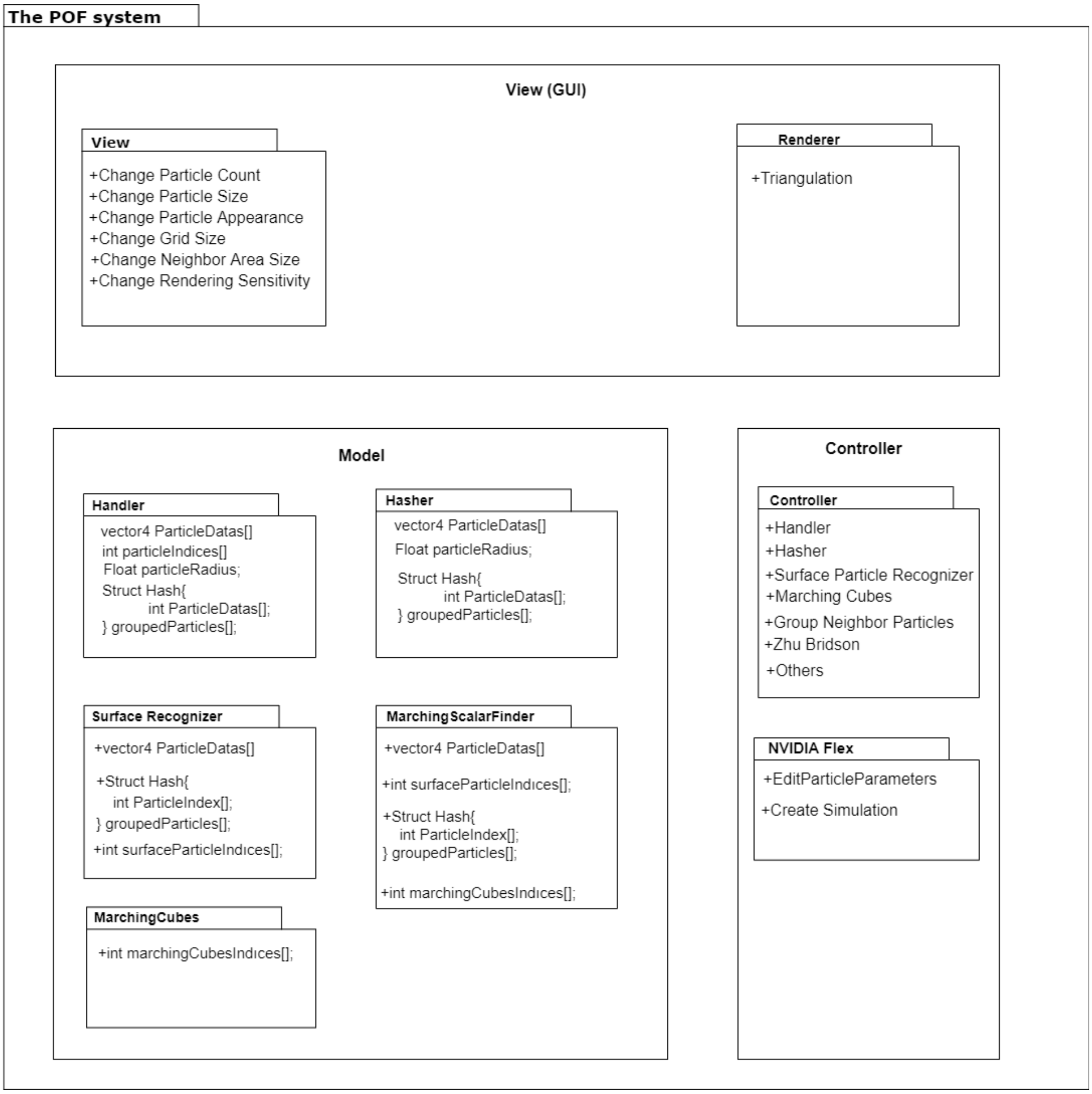


Fig 1: Package Diagram

Package diagram explains classes and their attributes along with relations. NIVIDA flex and Handler classes are the main parts of the system. Handler communicates to transmit data to relevant classes.

This section is extensively explained in design specification document [3] and it will be elaborated in future.

2.2. Detailed Design

Detailed design part is mentioned in design specification document. In DSD, [3] detailed design is explained with activity diagrams of the POF system. This section will be elaborated in future works.

2.3. Realistic Restrictions and Conditions in the Design

We had to neglect some aspects of the project to implement the project in a year. The security issue is ignored because the project aims to help everybody who has interested fluid simulations and contribute to science. We assumed that users of the POF system have the necessary equipment and software and know to how to use them.

3. IMPLEMENTATION and TESTS

This section describes implementation stages of the POF system. Code parts are given

3.1. Implementation of the System

3.1.1. Research Papers and System Structure

This project based on these two research papers, surface reconstruction algorithm that implemented by Zhu et al **[ZB05]** and Marching cubes algorithm **[WH87]**. However, some problems occur when system structure creation stage is started on visualization. Our main problem was performance and memory efficiency on that point, we added two algorithms to our project for passing over on performance and memory issues.

Main Problem	Solution of Problem
Searching particle data linearly due to 3D space positions and vector3 to integer translation.	Spatial hashing algorithm provides reaching particles by put them into cell data.
Too many particles appear in simulations and handling all of them occurs performance problems.	Do not put into calculations inactive and unnecessary particle on visualization (surface particle finding algorithm).

Table 4: Problem & Solution

After these solutions, we have started implement our system structure by creating classes, but we realize complexity getting higher due to interconnection between classes, so we create a handler class for control every classes in the one class.

3.1.2 Implementation of Hash algorithm

We use spatial hash algorithm to access particle position easily. Hash algorithm simplifies the three dimensions of float particle positions to integer id numbers in specific order.

Cube numbering is starts from top left and from left to right and then top to bottom. Then it implies the same operation for the third dimension. On x dimension, we found cell id by subtract minimum boundary area x from particle x, so it means that numbering increases from left to right. Same logic implies on y dimension, cell id number increase from top to bottom and similarly cell id number increases from backward to forward in z dimension.

```
int xId = (int)Math.Ceiling((particle.x - _bounds.min.x) / _length);
int yId = (int)Math.Ceiling((_bounds.max.y - particle.y) / _length);
int zId = (int)Math.Ceiling((particle.z - _bounds.min.z) / _length);
```

Fig 2: cell id numbering

We can reach cell id by position without storing it.

```
public struct vertexIndex
    {
        public int[] pointIndice;
        public vertexIndex(int[] pointIndice)
        {
           pointIndice = new int[1] { -1 };
           this.pointIndice = pointIndice;
        }
    }
```

Fig 3: Group struct

```
this._intervalx = (int)Math.Ceiling((_bounds.max.x - _bounds.min.x) / (_radius * 4));
this._intervaly = (int)Math.Ceiling((_bounds.max.y - _bounds.min.y) / (_radius * 4));
this._intervalz = (int)Math.Ceiling((_bounds.max.z - _bounds.min.z) / (_radius * 4));
groups = new vertexIndex[this._intervalx * this._intervaly * this._intervalz];
```

Fig 4: Hash size

Vertex index struct represents cells by creating an array on this struct. We created cell ids as indices. By finding how many grids in each dimension we find our hash table size represented as fig.3. We find particles by looking cells. Each cell represented by integer id number as seen in fig.2.

We have realized during the implementation when particle centre on intersection point of cell boundaries. We solve this problem by calculating subtract particle x/y/z from boundary max x/max y/max z and divide our grid size. If the remainder is equals to zero is on the boundary.

```
if ((_bounds.max.y - particle.y) % _radius == 0) {
    cubeID = (xId) + (this._intervalx * (yId--)) + (this._intervalx * this._intervaly * zId);

    checkS(cubeID-1, _indice);
}
```

Fig 5: Intersection boundaries check.

For example, in fig. 4. we found that particle on intersection boundaries and we found its second cell on y dimension
So that way we can find the particles by looking in cells instead of linear search of the particles. Ceiling is implemented to derive integer numbers because particles are float and they are derived according to the AABB size.

3.1.3 Particle neighbour algorithm

We must find the effect of the particles in a range on a specific particle and calculate it for each particle. In order to think mathematically consistent and coherent with the particles, the shape of the volume that we are checking should be spherical. However, required time and finishing the project according to given time interval has compelled project team to search cubical volumes. We must look the hashed cell ids in volume that we are searching to find neighbour particles.
We need three corner cells to solve the problem of finding neighbour particles. These three cells are called as tx,ty and tz. We find grid intervals that cells are going towards in every dimension. So that way we can find all cells in this volume just by looking at one cell that we have grouped neighbour particles in that cell.

// kod eklenecek.

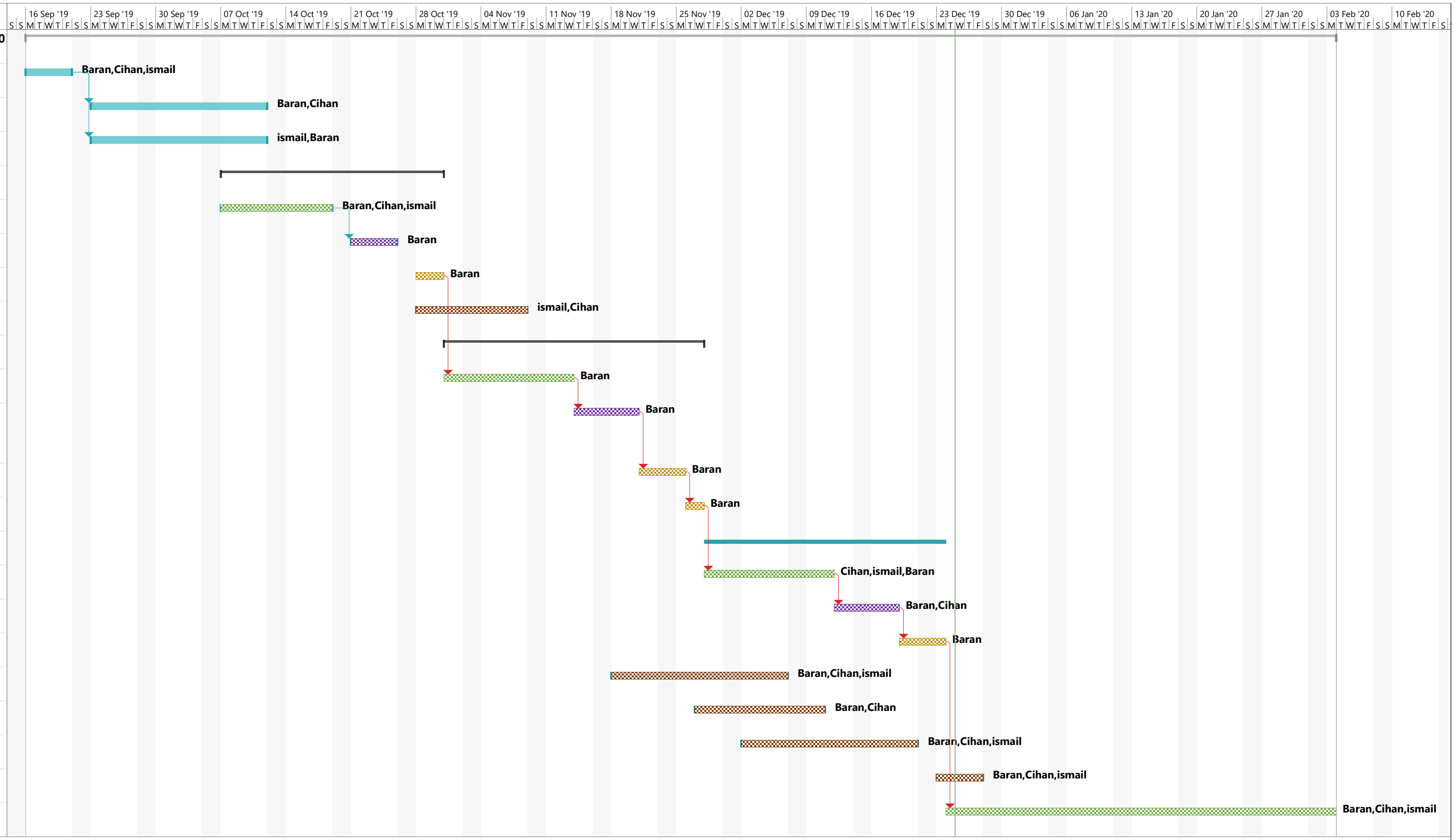
This project is not finished. Therefore, this section will be completed in future works.

3.2. Tests and Results of Tests

3.2.1 Availability of the Necessary Environment

Before finding NVIDIA Flex, we have tested three particle-based fluid simulation and we disqualify for these reasons.

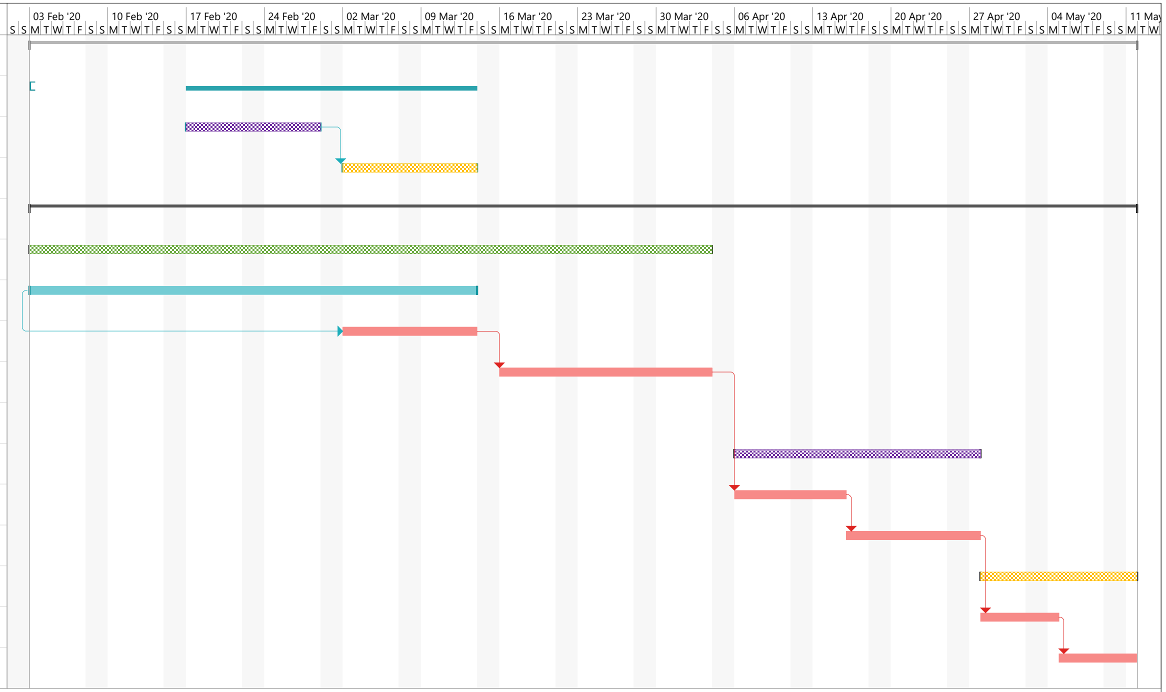
ID	WBS	Task Mode	Task Name	Duration	Start	Finish
0	0		**POF_Project_First_Semester**	**101 days**	**Mon 16/09/1!**	**Mon 03/02/20**
1	1		Select project Topic	5 days	Mon 16/09/19	Fri 20/09/19
2	2		Necessary platform research	15 days	Mon 23/09/19	Fri 11/10/19
3	3		Unity platform studying	15 days	Mon 23/09/19	Fri 11/10/19
4	4		**Divide Cells Algorithm**	**18 days**	**Mon 07/10/19**	**Wed 30/10/19**
5	4.1		Divide cells algorithm research	10 days	Mon 07/10/19	Fri 18/10/19
6	4.2		Divide cells algorithm implementation	5 days	Mon 21/10/19	Fri 25/10/19
7	4.3		Testing on experimental platform	3 days	Mon 28/10/19	Wed 30/10/19
8	5		Preparing the RSD document	10 days	Mon 28/10/19	Fri 08/11/19
9	6		**Surface Particle Algorithm**	**20 days**	**Thu 31/10/19**	**Wed 27/11/19**
10	6.1		Surface particle algorithm research	10 days	Thu 31/10/19	Wed 13/11/19
11	6.2		Surface particle algorithm implementation	5 days	Thu 14/11/19	Wed 20/11/19
12	6.3		Testing on experimental platform	3 days	Thu 21/11/19	Mon 25/11/19
13	6.4		Find surface particle constant	2 days	Tue 26/11/19	Wed 27/11/19
14	7		**Zhu Bridson algorithm**			
15	7.1		Zhu Bridson algorithm research	10 days	Thu 28/11/19	Wed 11/12/19
16	7.2		Zhu Bridson algorithm Implementation	5 days	Thu 12/12/19	Wed 18/12/19
17	7.3		Testing on experimental platform	3 days	Thu 19/12/19	Mon 23/12/19
18	8		Preparing the RSD2 document	15 days	Mon 18/11/19	Fri 06/12/19
19	9		Preparing the DSD document	10 days	Wed 27/11/19	Tue 10/12/19
20	10		Preparing the Final report document	15 days	Mon 02/12/19	Fri 20/12/19
21	11		Web Poster Design	5 days	Mon 23/12/19	Fri 27/12/19
22	12		Marching Cubes algorithm research	30 days	Tue 24/12/19	Mon 03/02/20



Project: POF_Project_First_Seme
Date: Wed 25/12/19

Summary · Project Summary · Manual Task · Duration-only · Manual Summary Rollup · Manual Summary · Start-only · Finish-only · External Tasks · Deadline · Critical · Critical Split · Progress · Manual Progress · Researching · Implementing · Testing · Documentation

ID	Task Mode	WBS	Task Name	Duration	Start	Finish
0		**0**	**POF Project second semester**	**71 days**	**Mon 03/02/20**	**Mon 11/05/20**
1		**1**	**Marching Cubes**		**Mon 03/02/20**	
2		1.1	Implementing Marching Cubes	10 days	Mon 17/02/20	Fri 28/02/20
3		1.2	Testing Marching Cubes	10 days	Mon 02/03/20	Fri 13/03/20
4		**2**	**Algorithms**	**71 days**	**Mon 03/02/20**	**Mon 11/05/20**
5		**2.1**	**Researching Algorithms**	**45 days**	**Mon 03/02/20**	**Fri 03/04/20**
6		2.1.1	Algorithm research for efficiency and performance	30 days	Mon 03/02/20	Fri 13/03/20
7		2.1.2	Performance testing of system	10 days	Mon 02/03/20	Fri 13/03/20
8		2.1.3	Development of algorithms	15 days	Mon 16/03/20	Fri 03/04/20
9		2.1.4	Determine algorithms	0 days		
10		**2.2**	**Implementing Algorithms**	**16 days**	**Mon 06/04/20**	**Mon 27/04/20**
11		2.2.1	Choosen algorithm implementation	8 days	Mon 06/04/20	Wed 15/04/20
12		2.2.2	Visualization changes	8 days	Thu 16/04/20	Mon 27/04/20
13		**2.3**	**Testing**	**10 days**	**Tue 28/04/20**	**Mon 11/05/20**
14		2.3.1	Testing of choosen algorithm	5 days	Tue 28/04/20	Mon 04/05/20
15		2.3.2	Testing of visualization	5 days	Tue 05/05/20	Mon 11/05/20



Project: POF Project second sem
Date: Wed 25/12/19

Task	Project Summary	Duration-only	Manual Summary	Finish-only	Critical Split	Manual Progress	implementing
Summary	Manual Task	Manual Summary Rollup	Start-only	Critical	Progress	Researching	Testing

uFlex	Had small bugs and errors in the code, even though we have fixed minor bugs, the particles were not recognizing the collider of the objects. Collider of the simple primitive objects was not recognized by the Uflex and particles were penetrating the objects. The only plane object was being recognized by the uFlex. The problem could not be solved, and we have changed the fluid simulation.
Obifluid	Obifluid is eliminated because of performance problems. The expected result was not satisfied by the Obifluid compared to other fluid simulations our expectation was reaching 30fps with a hundred thousand particles but in three thousand particles we have 3fps.
Screen Space Fluids Pro	Like uFlex we recognize small bugs and errors in the code, and we fixed it, but performance was very low on higher particle count.

4. CONCLUSIONS

In this section, the cost table of workers is given and explained. The cost of software and hardware is given with details and benefits of the projects are explained. This part of the final report summarizes our project and gives a cost analysis for the project. Future works mentioned.

4.1. Summary

The massive amounts of particles can be a computational hardship for the computer.
We implement various methods to get better results by making a research. Our project focuses on catalysing computational difficulties by increasing the performance and efficiency. Project should make easier to simulate with higher quantities of particles or getting better results with the same number of particles.

4.2. Cost Analysis

4.2.1 Cost of workers

Members	Day/Hour	Week/Hour	Semester/Hour	Salary/Hour	Salary/Monthly	TOTAL
Member	8	40	560	30 TL	4800 TL	16800 TL

Table 4: Cost Analysis of Workers

As shown in the cost analysis table, three people works in the project. Every people work equally as workload. Therefore, only one member is represented on the cost table.
Every member works 8 hours a day and 5 days a week. A semester consists of 14 weeks and salary is 30 Turkish lira per hour. Each member costs 4800 TL per month and costs 16800 in a

semester. The salary costs of all three members are 50400 TL per semester. The equivalent of 16800 TL is $2894, 67. Currency translation has made from Dollar / Turkish Lira = 1 / 5.80 in 10 December 2019.

4.2.2 Cost of Software

Title of Software	**Cost**
uFlex	$30
Obi Fluid	$30
SSF	$7
Total Cost	**$67**

Table 5: Cost of software

4.2.3 Cost of Hardware

4.2.3.1 PC components that used in Project

Total cost = Total employee cost + Total software cost + Total Hardware cost (PC1)

PC 1 components that used in Project	**Description**
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
Cost of PC 1 per user	$1693, 21
Total cost (for 1 worker)	**$4684, 88**
Total cost (for 3 workers)	**$14054, 64**

Table 6: PC 1 cost of components

4.2.3.2 Optimal Simulation Computer (PC 2)

Total cost = Total employee cost + Total software cost + Total Hardware cost (PC2)

Optimal Simulation Computer (PC 2)	Description
Operating System	Windows 10(64-bit Pro)
Processor	8-core Intel i7 5.1 GHz
Memory	32 GB RAM- DDR4- 2666MHz
GPU	NVIDIA Quadro P2200 5GB
Cost of PC 2 per user:	$5017
Total cost (for 1 worker)	**$8008,67**
Total cost (for 3 workers)	**$24026,01**

Table 7: PC 2 cost of components

4.3. Benefits of the Project

Our project can benefit in all areas where liquid simulation is available.

4.3.1 Animations and Movies: The POF system can be used in any movies, animations that used fluids.

4.3.2 Scientific work: Our project benefit scientific areas the most because the project is heavily research and development based of the research papers about the particle-based fluid simulations. Scientist and researchers can use the POF system for their scientific researches

4.3.3 Games: Some games need a fluid simulation system to make more realistic games. The POF system can be a good factor for the makes realistic games. For instance, in sailing simulator game is a perfect match for our system.

4.3.4 Construction: The construction and Architecture sector can benefit from our system because the simulation is physics-based which means the POF system is almost realistic. The POF system neglects some imperceptible elastic deformations. For instance, a civil engineer can build a barrage and want to test endurance, on the computer simulation. Therefore, our system can be used for construction and architecture testing.

4.4. Future Work

We will develop our project in order to achieve performance and efficiency goals. The functionality of the project will remain the same. However, small changes in the calculations will be changed to get better results.

References

1. Requirement Specification Document revision 1.0 (RSD 1.0)
2. Requirement Specification Document revision 2.0 (RSD 2.0)
3. Design Specification Document revision 1.0 (DSD 1.0)
4. **[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.
5. **[ZB05]** Zhu, Y., & Bridson, R. (2005). Animating sand as a fluid. (New York, NY, USA, 2005) *ACM Trans. Graph., 24*, 965-972.

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

// BURAYA RSD EKLENECEK

APPENDIX B: DESIGN SPECIFICATION DOCUMENT

// BURAYA DSD EKLENECEK