ARTISTIC CONTROL WITH SAMPLING FOR LIQUID SIMULATIONS

An Undergraduate Research Scholars Thesis

by

SAMANTHA BLAIR HALLAM

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. John Keyser

May 2023

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Samantha Blair Hallam, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

Page

ABSTRACT

Artistic Control with Sampling for Liquid Simulations

Samantha Blair Hallam
Department of Computer Science and Engineering
Texas A&M University

Faculty Research Advisor: Dr. John Keyser
Department of Computer Science and Engineering
Texas A&M University

Fluids such as gases and liquids are often animated through physically based techniques rather than manually. While this method can generate highly detailed fluids without tedious work from an artist, it also diminishes the amount of control an artist has over the end simulation. One proposed solution to this problem is to provide a low resolution preview to allow an artist to visualize a potential simulation before committing to a final, high resolution animation. However, increasing the resolution will also result in non-negligible differences between the preview and final versions of a fluid animation.

A potential technique to combat this problem is match point sampling. In this scheme, match points are placed in regions of interest and are used as a guideline to force the fluid in that region to conform to how it behaved during its preview stage. This method has shown promising results, but its development has been limited to grid-based Eulerian simulations, which are primarily utilized for gas simulations.

This work focuses on adapting the original sampling technique for use in particle-based Lagrangian simulations, which are primarily used for liquid simulations. Specifically, we investigate different methods of distributing and moving match points through three-dimensional space in the absence of a fixed grid and evaluate their efficacy by examining any differences between the preview and final versions of different liquid simulations.

DEDICATION

To friends, family, mentors, and anyone else who supported me along the way.

ACKNOWLEDGMENTS

Contributors

First and foremost, I would like to thank my faculty advisor, Dr. John Keyser, for his guidance and advice throughout the course of this research. Without his expertise, patience, and encouragement, this research would have not been possible. His commitment to meeting every week to offer support ensured that this work could be completed.

Thanks also go to my friends, colleagues, and professors who supported me throughout my time at university and through this project. I would also like to express my gratitude towards the Texas A&M Department of Computer Science faculty and staff for the opportunities they provided and their role in continuing to nurture my passion for computer science.

While all the presented results are the products of my own fluid simulator, they would have not been possible without the work of open-source libraries, such as Dear ImGui, LEAVEN, and cyCodeBase. The existence and maintenance of these code bases made it possible for me to focus on researching new methods of fluid control without worrying about other details such as particle position initialization and GUI creation.

I would also like to thank the Undergraduate Research Scholar program for their assistance in writing a thesis. Their templates helped increase the professionalism of the final document, and their deadlines helped ensure the thesis was developing appropriately throughout the two semester journey.

Funding Sources

No funding was provided for this research.

NOMENCLATURE

SPH	Smoothed particle hydrodynamics
PBD	Position-based dynamics
FPS	Frames per second
GPU	Graphics processing unit
LR	Low resolution
HR	High resolution

1. INTRODUCTION

1.1 Motivation

In graphics, fluid simulation is a form of physics-based animation that assists artists in the creation of convincing gases and liquids. It is often divided into two forms of simulation. The first type is Eulerian, which is most popular for gas simulations. The second type is Lagrangian simulation, which is most popular for liquid simulations and is the focus of this paper.

Lagrangian frameworks simulate fluids by dividing the actual fluid's volume into discrete particles. The forces acting on each individual particle are integrated over time to yield acceleration. With this acceleration, the velocities and positions of each particle can be estimated for a given time. By completing this integration scheme for every particle, the overall movement of the simulated liquid can predicted and animated.

In 2003, Müller, Charypar, and Gross pioneered the application of smooth particle hydrodynamics, hereafter referred to as SPH, for liquid simulation in graphics [8]. This method takes advantage of the Navier-Stokes equations that govern fluid dynamics to determine the viscosity and pressure fields acting on each particle. Additionally, SPH liquid simulation introduced the concept of smoothing kernels. Since then, SPH simulations have been refined through improvements such as the development of predictive-corrective incompressible SPH, which solved some physical inaccuracies left in the original model [10].

One of the extant challenges in fluid simulation is how to provide means of artistic control. While SPH models can yield impressive results, the computation required to simulate and render the fluid takes a significant amount of time. As a result, artists may wait hours for a simulation to finish yet be left with an unsatisfactory result. The research presented in this thesis aims to evaluate the efficacy of the sampling method to provide control in a Lagrangian framework.

1.2 Overview of SPH

Smoothed particle hydrodynamics or SPH is a method of animating particle systems. SPH's origins lie within astrodynamics to assist with simulations within that field of study. However, since Müller's work in adapting it for graphics in 2003, it has consistently been applied to fluid animations. The main principle driving SPH is smoothing kernels: functions that are capable of determining a scalar quantity of a particle as a function of other particles within a finite neighborhood. Ideally, these smoothing kernels should be symmetric, and the integral of the kernel with respect to the distance between particles should be one:

$$W(\mathbf{d}, h) = W(-\mathbf{d}, h) \tag{1}$$

$$\int W(\mathbf{d}) d\mathbf{d} = 1 \tag{2}$$

where $\mathbf{d}$ is the difference between a particle's position and the point in which a sample is being taken, and h is the maximum distance supported by the kernel function. If these conditions are met, any given scalar quality *A* at some location can be computed as a function of its neighboring particles:

$$A = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r_j}, h) \tag{3}$$

where the summation iterates through each neighboring particle j, m_j is the mass at particle j, A_j is the scalar quantity A at particle j, ρ_j is density at particle j, W is the kernel function, $\mathbf{r}$ is the location of the particle for which the scalar quantity is calculated, $\mathbf{r}_j$ is the location of particle j, and h is the maximum distance of the kernel function. This is incredibly useful for calculating fluid properties such as density. In the common case where the gradient or Laplacian of a scalar is required for a calculation, one only needs to take the gradient or the Laplacian of the weighted kernel:

$$\nabla A = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r_j}, h) \tag{4}$$

$$\nabla^2 A = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r_j}, h) \tag{5}$$

Equations 3, 4, and 5 can be combined with the Navier-Stokes equations and used to calculate useful fluid quantities such as density, pressure force, and viscosity force:

$$\rho = \sum_j m_j W(\mathbf{r} - \mathbf{r_j}, h) \tag{6}$$

$$\mathbf{f_{pressure}} = -\sum_j m_j \frac{p + p_j}{2\rho_j} \nabla W(\mathbf{r} - \mathbf{r_j}, h) \tag{7}$$

$$\mathbf{f_{viscosity}} = \mu \sum_j m_j \frac{v + v_j}{2\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r_j}, h) \tag{8}$$

where ρ is the density, ρ_j is the density at neighboring particle j, p is the pressure, p_j is the pressure at neighboring particle j, μ is the viscosity constant, v is the velocity, and v_j is the velocity at neighboring particle j. These resulting values can then be used to determine the acceleration (and thus, the velocities and positions) through time:

$$a = \frac{\sum \mathbf{f}}{\rho} \tag{9}$$

where a is acceleration, $\mathbf{f}$ is any force acting upon the particle, and ρ is the fluid density at the particle. The forces acting on the particle may extend beyond than those presented in this paper. In addition to pressure and viscosity, other common forces on acting on the particles include external forces such as gravity and surface tension.

1.3 Related Work

Several methods of providing animators with artistic control over fluids have been explored through the years. Huang et al. proposed match point sampling, the technique focused and expanded on by this research [5] . Under this control scheme, animators are provided with a low resolution preview before committing to a high resolution simulation. Without further modification, the high resolution simulation will contain significant visible differences from the preview as

a result of changes in physics from having a higher resolution. However, Huang's method samples the low resolution preview for key values at defined match points (either automatically selected or selected by an artist) and uses the differences between those values and the values calculated by the high resolution model to force the final simulation to conform to the properties seen in the preview. This method proved successful, and Huang et al. continued to improve it [4].

Another example of relatively successful fluid simulation control is target-driven animation [3]. The driving concept of this method is to provide "targets" for the fluid animation, such as a ship sailing through a ring, and use a gathering mechanism to morph the fluid to those targets in a plausibly realistic way. One drawback to this method is that there is no way for an artist to control how much a target should be approximated at a given time.

Another method of interest is the adjoint control method. McNamara et al. introduced this method for providing fine control of large simulations [7]. By using the adjoint method, derivatives of millions of control parameters could be calculated with enough efficiency for practical use. These derivatives could then be used to drive the fluid into a certain shape at a certain time based on keyframes provided by an artist. However, this model does have limitations. While keyframing provides artists with a large amount of control, this method at present cannot account for initializing certain fluid phenomena, such as a wave breaking, without providing a keyframe or encoding additional controls.

Position-based dynamics (PBD) is another alternative to SPH fluids. In PBD animation, a particle is moved along some velocity vector (usually a function of external forces on a system) and then has its position further transformed by applying physically-based constraints. This possibility was investigated by Macklin and Müller and was shown to outperform PCISPH in speed and stability [6]. However, one flaw with this method was that it was challenging to modify a single parameter without having to modify others as well.

Sampling has also shown to be useful for guiding simulations generated through position-based dynamics, including simulations outside of the realm of fluids. Sommer et al. pioneered applying upsampling to granular materials to control high resolution simulations [12]. With this

strategy, a PBD low resolution simulation would first be run and used as a guide during the high resolution simulation. The materials simulated in the paper were more analogous to sand or gravel than to a traditional fluid like gas or liquid, but it does not require a stretch of the imagination to see how this work could be applied to fluids.

Position-based dynamics can also be used to enhance SPH instead of being used as an alternative. For example, one problem with traditional SPH fluid simulations is unstable handling of solid boundaries. This means that in cases where there is a large difference between the velocity of the fluid and the solid it is colliding with penetration of the solid object can occur even when it is not physically possible. Traditionally, small time steps were required to prevent this phenomenon from occurring. However, Shao et al. discovered a possible solution to this challenge through position-based dynamics [9]. In their research, PBD was used to assist position constraints between the fluid and solid, as well as determine how the collision of the solid affected the fluid.

The focus of this paper will be furthering the development of sampling for artistic control. Of the methods described in this section, sampling will be the method investigated. Specifically, this research explores how the match point sampling system utilized for Eulerian simulations can be adapted and implemented for use in Lagrangian simulations.

2. METHODS

2.1 Initializing Match Points

In order to initialize a match point, three key values are needed. First, since a match point is a point in space, a 3D vector is required to represent the match point's position. Additionally, a scalar value is used to represent the radius of the support of the match point. This determines the boundaries of the neighborhood for the match point and is also used to define the radius used in the Gaussian weight function for sampling. Finally, match points have some scalar weight. The weight value is used to determine the magnitude of the error correction during the matching phase.

A few different strategies for match point initialization were used during this project. The first is automatically initializing a grid of evenly spaced match points of equal radii and weight over half of a generated scene. This is not an ideal setup in practice, as it will attempt to match half of the scene as closely as possible. Additionally, the number of match points required to adequately cover half of the scene might also be enough match points to significantly impact the interactivity of the simulation during the LR (low resolution) phase. However, this method was effective for evaluating whether or not the sampling and matching system was useful at all.

Another strategy was uniform random placement based on initial particle positions. These particles would be assigned an equal weight and a random radius between some arbitrary minimum and maximum radius. The primary disadvantage of initializing match points in this manner is that if a key feature of a simulation occurs at a location that is not near the original particle positions, there is no way to capture it using this method.

The next strategy used was manual match point placement. This allows a user of the program to input where they would like to add a match point directly. This is the method that would allow an artist the most control over what features their HR (high resolution) animation should capture from its LR preview simulation. While this process can be tedious, it can also easily be combined with other initialization strategies to help minimize the amount of match points the artist

has to specify.

Manual match point placement also supports a grid placement. This is done by allowing the user to input a minimum and maximum coordinate that represent corners of a box and a match point radius. With this information, match points with the desired radius are placed from the minimum coordinate to the maximum coordinate in a grid pattern such that no match points are overlapping. This can help alleviate some of the tedium of placing match points individually in areas of the fluid where capturing a larger feature is desired.

Figure 1 shows different possible arrangements of match points. However, the strategies presented are not an exhaustive list. It is entirely possible that there is a better way to place match points in the LR preview to improve results, efficiency, or ease of use. However, due to time constraints, investigations into alternative methods are out of the scope of this research.

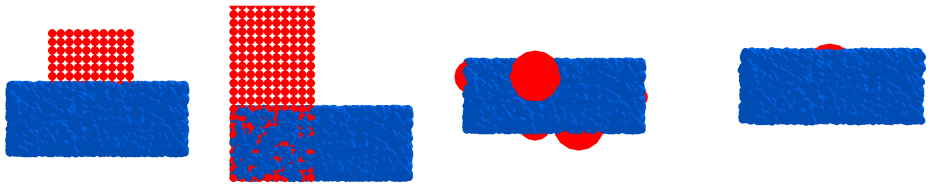

Figure 1: Different possibilities for initializing match points in which the match points are the red spheres. From left to right: manual grid input, automated half grid, uniform random, and single manual input.

2.2 Sampling Match Points

The sampling method used for the SPH simulator is the sampling method found in the past research on Eulerian fluid simulations [5]. Sampling a match point for a vector value (such as velocity) and its curl is performed according to the following equations:

$$\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{1}} = \frac{1}{\sum_x G_r} \sum_x G_r(\mathbf{x_i}, \mathbf{x}) \mathbf{D}(\mathbf{x}) \tag{10}$$

$$\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{2}} = \frac{1}{\sum_x G_r} \sum_x G_r(\mathbf{x_i}, \mathbf{x}) \mathbf{D}(\mathbf{x}) \times (\mathbf{x} - \mathbf{x_i}) \tag{11}$$

where Equation 10 results in the sampled vector value and Equation 11 results in the curl of a vector value. The function G is a Gaussian weight function. This function determines the weight a particle's vector or vector curl has on the overall sample as a function of its distance from the sample, $|\mathbf{x_i} - \mathbf{x}|$ and the radius of support r. The Gaussian weight function used is:

$$G_r(\mathbf{x_i}, \mathbf{x}) = e^{-r|\mathbf{x_i} - \mathbf{x}|^2} \tag{12}$$

After every time step in the simulation during the low resolution preview, a keyframe is created. This keyframe stores a collection of all match points at that time, their sampled values, and the time itself. These keyframes are then utilized as a frame of reference for the final model during the high resolution animation.

One key difference between sampling for Lagrangian simulations versus Eulerian simulations is that in Lagrangian simulations the number of neighbors a match point has can change over time. Because Eulerian simulations rely on a fixed arbitrary mesh or grid, the number of points in the grid encompassed by a single match point will be constant throughout a simulation for a given resolution. In contrast, the particles in Lagrangian simulation are in constant motion, creating a situation where it is probable that the number of neighbor particles a match point has will change over time.

As a result, it is entirely possible that the sampled value at a given match point will be incapable of providing reasonable data for use during the matching stage. In order to prevent attempting to match to an inaccurate value, match points that have a sparse neighborhood during the LR preview should be flagged to be ignored during the matching phase.

Determining whether or not a neighborhood should be considered sparse is one of the more difficult tasks required by this research. There is no universal mathematical or physical definition of what constitutes a sparse neighborhood. One early idea was to define an arbitrary number of

neighbors a match point had to have to be considered a good sample, but this proved to be inadequate. The number of particles needed might differ between low resolution and high resolution simulations. Additionally, a match point could have enough neighbors under that definition but still be an insufficient sample if all the neighbors are on the edge of the radius of support.

An idea inspired the upsampling of granular materials research was to define sparsity not as a measure of the number of particles in a given neighborhood, but instead as a function of the maximum weight found [12]. With this method, a match point is considered to have too sparse a neighborhood to be considered a valid sample if the maximum weight of a particle within the support radius is less than some constant or if the ratio of the maximum weight to the total weight is greater than some constant:

$$S(x_i) = \begin{cases} 1 & \text{if } G_{r_{max}}(\mathbf{x_i}, \mathbf{x}) < c_1 \text{or } \frac{G_{r_{max}}(\mathbf{x_i}, \mathbf{x})}{\sum_x G_r(\mathbf{x_i}, \mathbf{x})} > c_2 \\ 0 & \text{else} \end{cases} \tag{13}$$

where 1 represents match point x_i being too sparse and 0 represents match point x_i having a dense enough neighborhood to serve as a reasonable sample. Both c_1 and c_2 are constants that can be tuned by the user where c_1 can be used to filter out match points in which no particle within a neighborhood has significant influence and c_2 can be used to filter out match points in which a single particle dominates the sample. For the purposes of this research, $c_1 = .00001$ and $c_2 = .4$.

2.3 Keyframe Matching

Once the high resolution simulation starts, keyframe matching is used to help determine the velocities of the particles within the radius of support of the match points. While simulating in HR, the current simulation time is checked against the recorded simulation time of the next ordered keyframe. If these times match, control is applied after the standard SPH simulation for that time step.

The control method used at each keyframe is nearly identical to the method presented in the controlled Eulerian simulation [5]. The match points used during the LR preview are sampled

again during the HR simulation using the same Gaussian weight functions. These values will be referred to as $\overline{\mathbf{D^r_{i,1}}}$ for a HR sampled vector's direction and $\overline{\mathbf{D^r_{i,2}}}$ for a HR sampled vector's curl. Then, the weighted error between the LR and HR match points is calculated as the following for vector direction and curl respectively:

$$\Delta^r_{i,1} = m^r_i(\vec{\mathbf{D}}^{\mathbf{r}}_{\mathbf{i,1}} - \overline{\vec{\mathbf{D}}^{\mathbf{r}}_{\mathbf{i,1}}}) \tag{14}$$

$$\Delta^r_{i,2} = m^r_i(\mathbf{D}^{\mathbf{r}}_{\mathbf{i,2}} - \overline{\mathbf{D}^{\mathbf{r}}_{\mathbf{i,2}}}) \tag{15}$$

where m^r_i is the weight of a the match point being sampled. Once these weighted errors are calculated, control can be spread using a Gaussian spatial blending function. This serves the purpose of spreading the control by making adjustments to all particles within a given radius. This spatial blending function for a particle $\mathbf{x}$ around match point $\mathbf{x}_i$ is the following:

$$G'_r(\mathbf{x}_i, \mathbf{x}) = \frac{G_r(\mathbf{x}_i, \mathbf{x})}{\sum_{\mathbf{x}}(G_r(\mathbf{x}_i, \mathbf{x}))^2} \tag{16}$$

The Gaussian function used for spreading control was equivalent to that used for the sampling process. This equation can be combined with the weighted error to iteratively correct differences between values in the LR and HR simulations. The equations to provide correction for velocity direction and curl are as follows:

$$\Delta^r_{i,1} G'_r(\mathbf{x}_i, \mathbf{x}) \tag{17}$$

$$G'_r(\mathbf{x}_i, \mathbf{x})(\Delta^r_{i,1} \times (\mathbf{x}_i - \mathbf{x})) \tag{18}$$

These values can be directly added to the vector values of interest to apply control to direction and curl, respectively. Once this step is completed, $\Delta^r_{i,1}$ and $\Delta^r_{i,2}$ are recalculated and the process of applying control to particles in the HR simulation is repeated until ideally both $\Delta^r_{i,1}$ and $\Delta^r_{i,2}$ are under some defined error threshold. Alternatively, it is possible that the error never

converges under the defined threshold. For this reason, it is required to additionally define a max number of iterations of control for the simulation to attempt before giving up on resolving the error values in order to prevent infinite loops.

2.4 Blending

A novel idea proposed by this research is to blend between the recorded keyframes by projecting match points backwards through time and creating a more gradual shift in sampled values. The goal of this method is to create a more natural, gradual looking animation between keyframes provided in the LR preview. This is especially useful if the time step used for the physics simulation in the HR simulation is much smaller than the time step used during the LR preview. If the current simulation time in the HR simulation does not match the simulation time recorded in the next keyframe, a blending process begins. First, α is calculated as the following:

$$\alpha = \frac{t_k - t_s}{t_k - t_{k-1}} \tag{19}$$

where t_s is the current simulation time, t_{k-1} is the recorded simulation time of the last known keyframe, and t_k is the recorded simulation time of the next keyframe. Next, instead of sampling the HR values in the same positions of the match points used during the LR simulation, the positions of the HR match points are calculated using the following equation:

$$\mathbf{x_i}^{HR} = \mathbf{x_i}^{LR} - \alpha \Delta t v_i^{LR} \tag{20}$$

where $\mathbf{x_i}^{LR}$ is the position of a match point in low resolution, Δt is the time step being used for the physics simulation, and v_i^{LR} is the sampled velocity direction in the low resolution simulation. This new match point position is sampled using the same methods described previously in this chapter.

Rather than using the sampled velocities of the match points at the next keyframe as a goal value when calculating error, new goal quantities are calculated as a function of α:

$$\mathbf{D}^{\mathbf{r}}_{\alpha \mathbf{i},\mathbf{1}} = (1-\alpha)\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{1}} + \alpha\overline{\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{1}}} \tag{21}$$

$$\mathbf{D}^{\mathbf{r}}_{\alpha \mathbf{i},\mathbf{2}} = (1-\alpha)\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{2}} + \alpha\overline{\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{2}}} \tag{22}$$

$\mathbf{D}^{\mathbf{r}}_{\alpha \mathbf{i},\mathbf{1}}$ represents a vector direction and replaces $\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{1}}$ in Equation 14 when calculating weighted error. Likewise, $\mathbf{D}^{\mathbf{r}}_{\alpha \mathbf{i},\mathbf{2}}$ represents vector curl and replaces $\mathbf{D}^{\mathbf{r}}_{\mathbf{i},\mathbf{2}}$ in Equation 15. With these replacements, the control procedure iterates as usual until both weighted errors are under some defined threshold of error or a maximum number of iterations has been reached.

3. IMPLEMENTATION

3.1 Source Control

GitHub was used for source control for this project. Using Git permitted for bugs to be fixed and new features implemented while preserving a working version of the simulator at all times. This ensured that research could progress forward at a steady pace without concerns about whether new changes would cause the simulator to cease to function.

3.2 Rendering

OpenGL was the library used for rendering the simulations as 3D animations. OpenGL is a C++ library that uses the rasterization pipeline for graphics to help create the interactive graphics needed for the low resolution preview stage of the fluid simulation. The GLSL shaders used for this research were utilized for aesthetic purposes only and were not used for any other physics calculations. The fluid was rendered as a collection of spheres rather than a fluid surface which was out of the scope of this project. The sphere model used for the particles was a low-poly sphere created with Blender and exported as an OBJ to be used in the application. Within OpenGL, the GLFW library was used for creating and polling windows.

3.3 Vector and Matrix Math

The GLM library was used for the complex mathematics needed for the graphics calculations in this project. GLM is a mathematics library designed for use with OpenGL (but also useful with other graphics libraries) that contains essential functions for graphics such as vector normalization, matrix multiplication, and calculating a matrix's inverse. The classes and capabilities imported from this library were used for tasks anywhere from storing a particle's position as a vector to normalizing distances between particles in a scene.

3.4 GUI

In order to quickly adjust fluid parameters to obtain a satisfactory animation, a GUI was needed. The alternatives, providing command line arguments or hard coding the parameters in the source code itself and recompiling in between animation attempts, proved to be much too arduous for practical use. An open source library known as Dear ImGui was used to provide an interface that allowed the user to adjust fluid parameters without having to restart the program [2]. Additionally, the GUI created with this library provided controls that allowed the user to pause the simulation and initialize recording in low, medium, or high resolution and add match points to the simulation. Figure 2 shows what this GUI looks like.

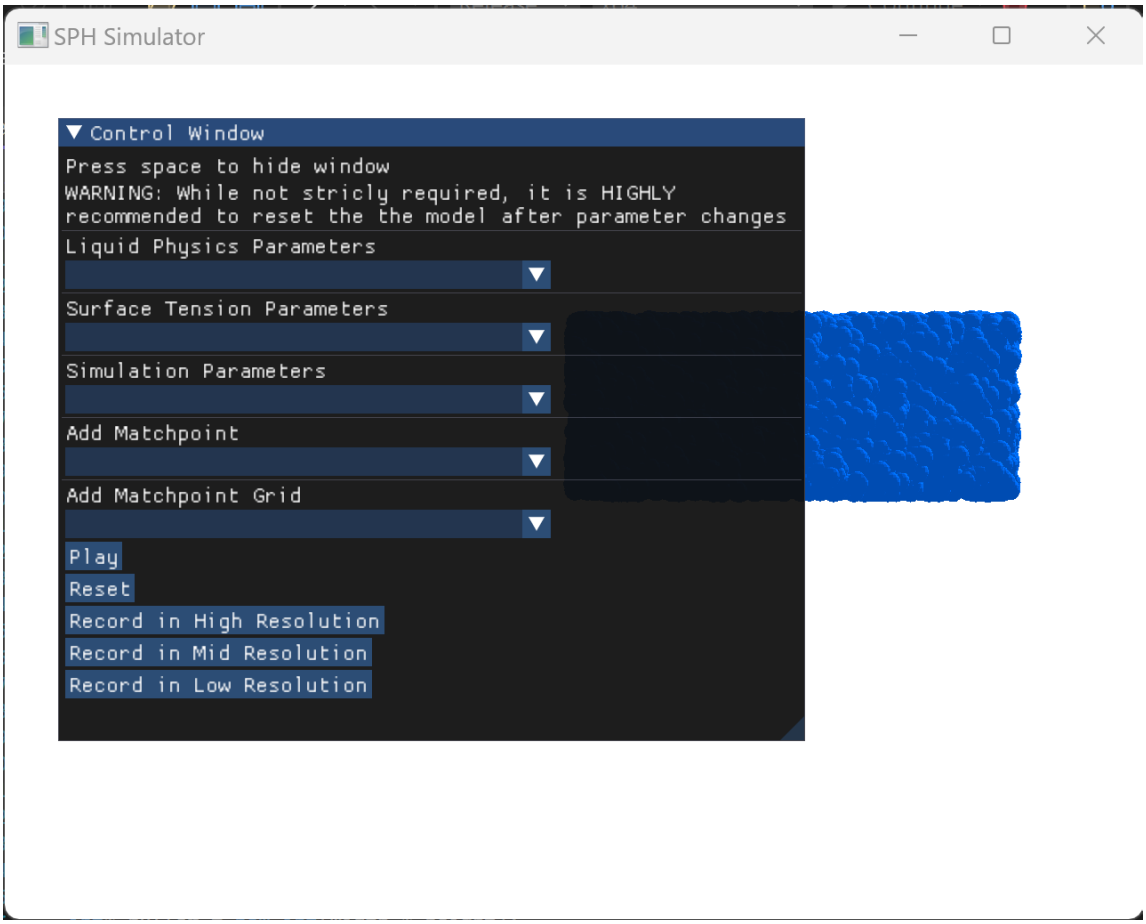

Figure 2: The window in which the simulation is played. The "Control Window" is a GUI created using the Dear ImGui library.

3.5 Particle Initialization

At the beginning of each simulation, particles are initialized in a scene that is determined at compile time. The scene options include a drop of water falling into still water to create a splash and a dam breaking and the fluid moving to fill its new container. Certain particle characteristics, such as radius and velocity, are trivial to determine at the beginning of the simulations.

In contrast, finding the ideal position for a particle to be at during initialization proved to be somewhat difficult. Specifically, finding an arrangement of particles such that they were either at rest or close to being at rest was challenging. One method that showed acceptable results was to distribute the particles in a uniform random distribution through their ideal starting shape, often a box of some form. After this, the average mass that each particle would have to be to have a density close to the rest density of the particles is calculated and used as each particle's mass. The particles in the box formation would retain their positions.

For arbitrary shapes, the same mass calculated in the above procedure is used as the mass of each particle. However, an external library was used to determine particle placements in these cases. This library, known as LEAVEN, was developed as the subject of another computer graphics paper [11]. The results produced by incorporating it were significantly better than any original attempts by this author, and it was used to create the water droplet in the splash scene.

3.6 SPH Fluid Simulation

All SPH fluid simulation was programmed from scratch. The kernel functions used were the same as those presented in the original Müller paper [8]. While there have been other methods of SPH to maintain the incompressibility of the fluid, such as PCISPH, they are outside of the scope of this research.

The first implementation of SPH in this research proved to be much too slow. This is because the physics required for each particle is computationally intensive, and each particle was processed in a sequence rather than in parallel. As a result, even the low resolution preview stage of the simulation was too slow to truly be interactive.

The next improvement was to utilize multi-threading to be able to process particles in parallel. This was done by using the threading libraries available in C++. For each needed physics operation, such as calculating density or integrating acceleration to determine velocity and position, the particles were processed concurrently over ten threads. While there was a noticeable increase in frame rate using this method, there was still considerable lag. This was because the parallelization was severely limited by the number of cores available in the CPU.

The final implementation of the SPH simulator uses CUDA to provide more support for parallelization. CUDA is a toolkit released and managed by Nvidia which permits users to run code on the GPU. This is advantageous because GPUs have more cores than CPUs, allowing for a higher level of concurrency. By shifting the computation from the CPU to the GPU, multiple frames were able to be computed in under a second. The one exception is the frame in which neighbors are updated, which remains as a bottleneck in the project.

3.7 Finding Neighbors

In order to simulate fluids using SPH, the neighbors of each particle in the system must be found. The process of finding these neighbors is the most computationally expensive step of the simulation. Using a brute force check and checking every particle against every other possible particle in the system takes $O(n^2)$ time. However, techniques exist in order to help decrease the amount of time required to find neighborhoods. One of the most popular algorithms for this is a kd-tree.

Kd-trees are a form of binary tree used to partition the space in which an object or objects of interest reside [1]. Kd-trees can be used for objects of an arbitrary dimension and work by continuously splitting the given space in half by using the median value of the given data points (in this case, the positions of the particles occupying the system) in one of their dimensions. By searching the tree created by this method, the average time to find all neighbors of a node within some radius was found to be within $O(n \log n)$ time, a dramatic improvement over other nearest-neighbor algorithms.

The kd-tree approach was used when finding neighbors for this research. The library for the

creation and usage of kd-trees was made by Cem Yuskel [14]. However, even with the use of the kd-tree, there was an obvious slowdown as a result of updating the tree and particle neighborhoods. As a result, the kd-tree and neighborhoods would only be updated after a certain amount of simulation steps instead of after every step in the simulation. Three steps were used as a default, but this value could be tuned by the user.

4. RESULTS

4.1 Standard Matching

Standard matching refers to the use of the matching algorithm described in the methods section sans the blending portion. This matching method was tested with two different scenes: the water splash and the dam break. Match point placement strategies tested included the half grid method, uniform random placement, manual placement by single points, and manual placement by grids.

Both of the automated methods were tested with the splash scene first. In Figure 3, frames of interest from the LR preview are in the leftmost column. The next column represents the HR simulation without any control methods. The third column shows the results of using uniform random positioning to place the match points and control their radii. At present, this method does not seem very effective, as there is little noticeable difference between these frames and the frames produced by the HR simulation with no control whatsoever. However, the final column used match points placed throughout the left half of the simulation in a grid pattern. Although this simulation still looks considerably different from the LR preview, most of the explosive splash seen in the other two HR simulations in the figure is contained within the right half of the simulation. This illustrates that sampling and controlling via a match point system is capable of making a demonstrable difference.

After the automated strategies proved that sampling can be used as a means of control, the manual strategies were tested. Like in Figure 3, the first and second columns of Figure 4 represent the LR preview and the HR simulation with no control, respectively. The next column represents inputting individual control points manually through the GUI system. In this case, a single large control point was placed in the middle of the body of liquid where the droplet lands. This method of match point placement did not create a significant change. In contrast, the final column, which represents allowing the user to input a minimum and maximum coordinate to create a grid of smaller match points, showed significant promise. This grid was also located in the center of the

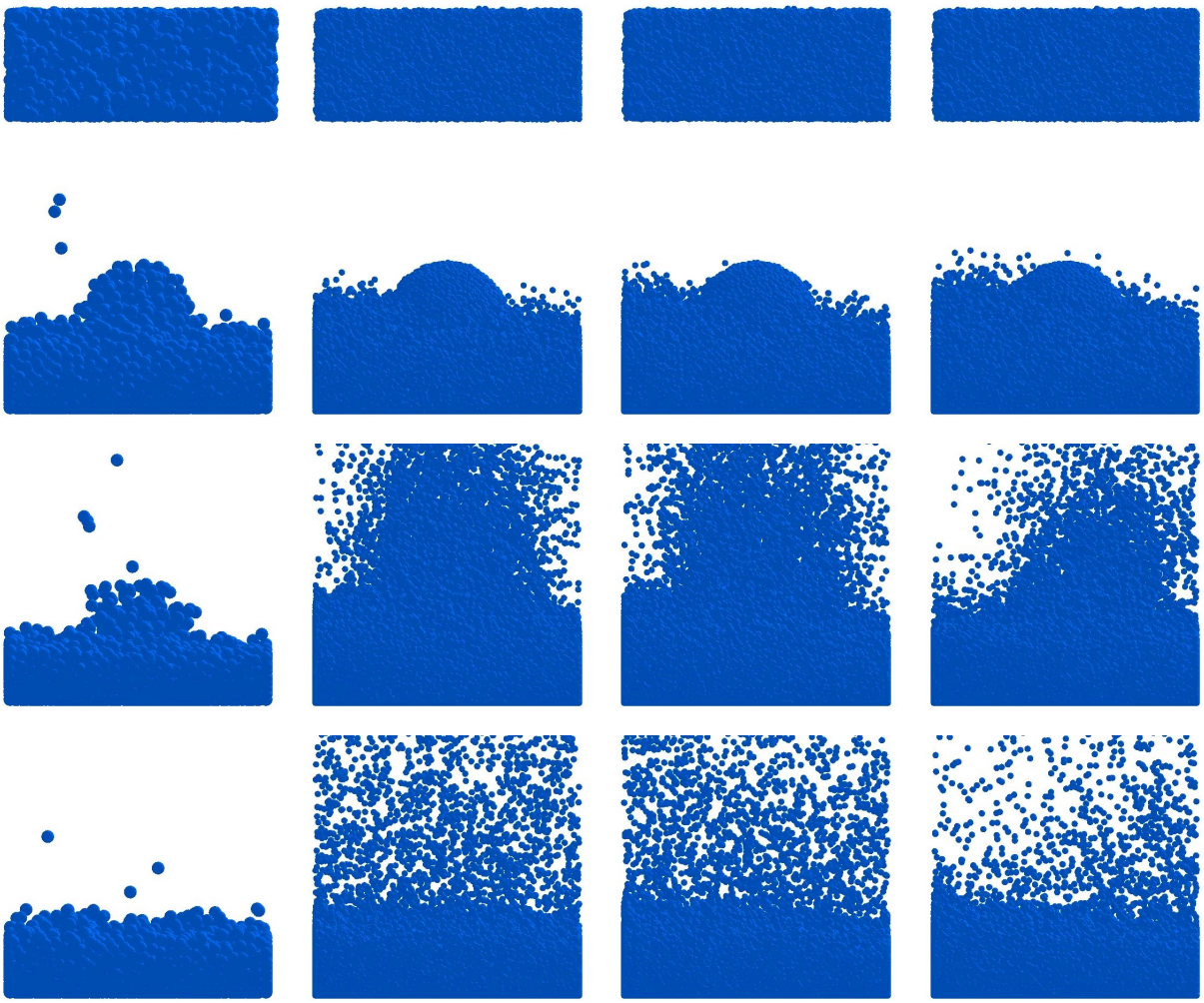

Figure 3: Varying frames from simulations of the splash scene with different resolution and control settings. From left to right column: LR, no control HR, uniform random control HR, automated left half grid HR.

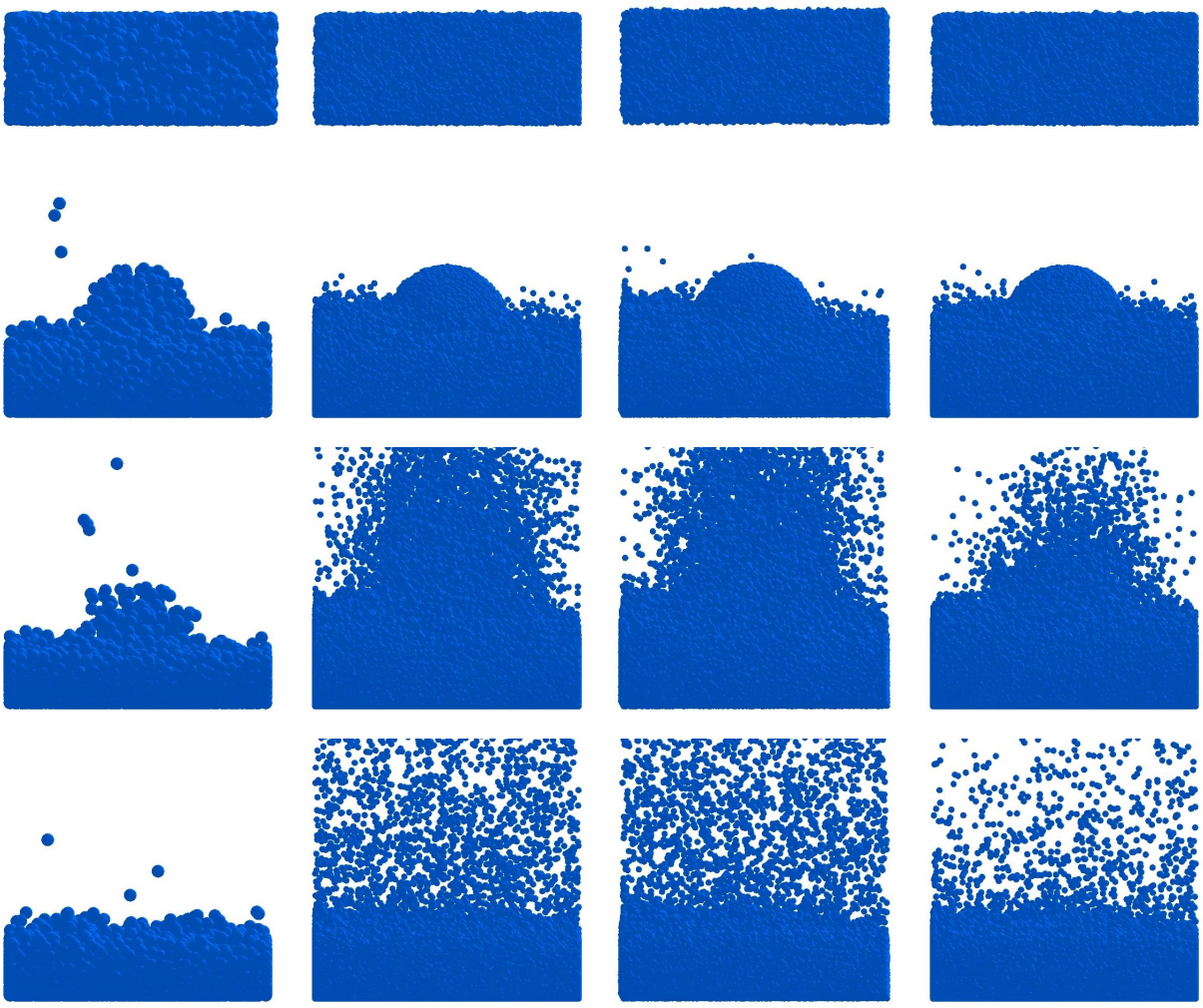

Figure 4: Varying frames from simulations of the splash scene with different resolution and control settings. From left to right column: LR, no control HR, manual point HR, manual grid HR.

liquid. While there still is noticeable difference from the LR preview, in this HR simulation the splash after contact that is captured in the third shown frame did not explode nearly as much as the other two HR simulations and is closer to the LR preview.

Table 1: Complete splash simulation times in seconds

LR - no control	HR - no control	LR - grid control	HR - grid control
38.3193	962.382	47.1409	1153.41

Timing data for the simulation splash scene in LR and HR with no control and with manual grid control is displayed in Table 1. While the controlled versions of the simulation required more time in both LR and HR to capture key data and apply control, the time increase was not incredibly large. The increase in LR time was approximately 23.0%, and the increase in HR time was approximately 19.8%. Thus, sampling seems to show promise in providing a method of artistic control without greatly incurring time costs.

For the dam scene in Figure 5, only methods of manual control were tested because the splash scene demonstrated that the automated method of uniform random placement was ineffective, and the automated half grid method was only meant to illustrate that sampling was capable of causing significant differences in the simulation. In comparison to the splash scene, the results from the dam scene are more inconclusive. However, it is difficult to look for meaningful differences between the different HR simulations as the original LR model in the leftmost column and the HR simulation without control in the next column are fairly similar without extra control methods. Similar to the setup for the other presented figures, the third column represents control through individually inputted match points, and the fourth column represents control through a match point grid system. In both of these control schemes, match points are placed where the dam water collides with a secondary wall after the initial dam break. While these results are much less convincing than the splash scene, it is also useful to know in a case where the HR simulation is already similar to the LR simulation that adding unnecessary control will not visually degrade the

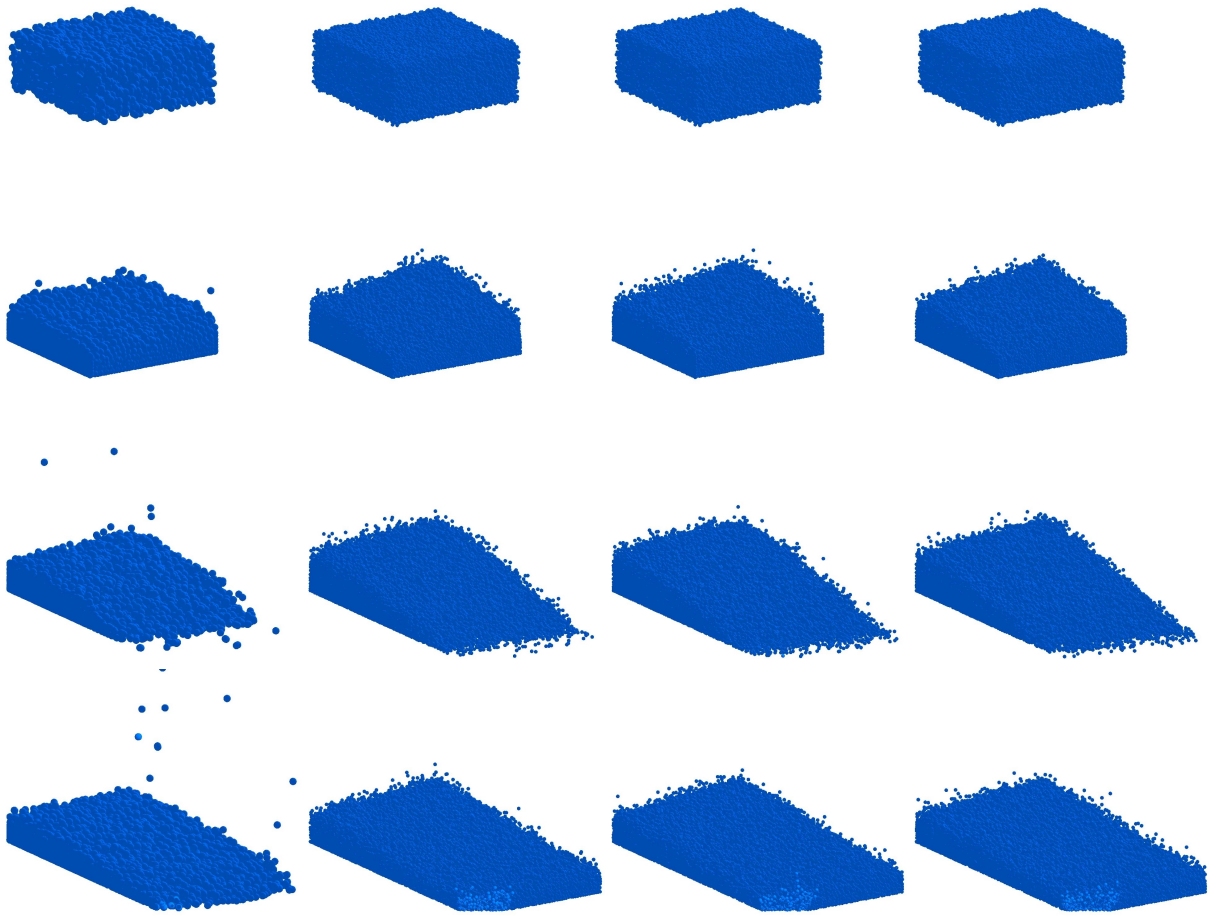

Figure 5: Varying frames from simulations of the dam scene with different resolution and control settings. From left to right column: LR, no control HR, manual point HR, manual grid HR.

animation.

Table 2: Complete dam simulation times in seconds

LR - no control	HR - no control	LR - grid control	HR - grid control
74.3804	1861.28	89.2635	2262.54

The timing results from the dam simulation seem similar to those from the splash simulation and are shown in Table 2. In both the LR and HR simulations, the simulation time increased but not by an excessive amount. The LR simulation time increased by approximately 20.0% and the HR simulation time increased by approximately 21.6%.

Overall, from the scenes and match point placement strategies tested, the manual grid strategy seemed to work the best, and the automatic half grid strategy was the most effective in demonstrating a clear difference between controlling via match points and not having any control whatsoever. In contrast, both the uniform random placement strategy and single point manual placement seem ineffective at controlling the simulation at present. Additionally, time costs incurred by using sampling were not overly costly regardless of scene.

4.2 Blended Matching

Blended matching refers to the use of the matching algorithm described in the methods sections including the blending portion. The goal of blending is to help alleviate some of the finer visible discrepancies between a LR preview and a controlled HR simulation. The scenes and methods utilized for testing this algorithm were the similar to those discussed in the standard matching section.

Since in the standard matching section the automated half grid was the most effective method of the automatic match point initialization methods at showing a difference between no control and control via sampling, it was the initial method used to test the efficacy of blended matching. The leftmost column in Figure 6 shows the LR preview, and the next column shows the HR simulation with no control. The third column shows the automated left half grid with standard

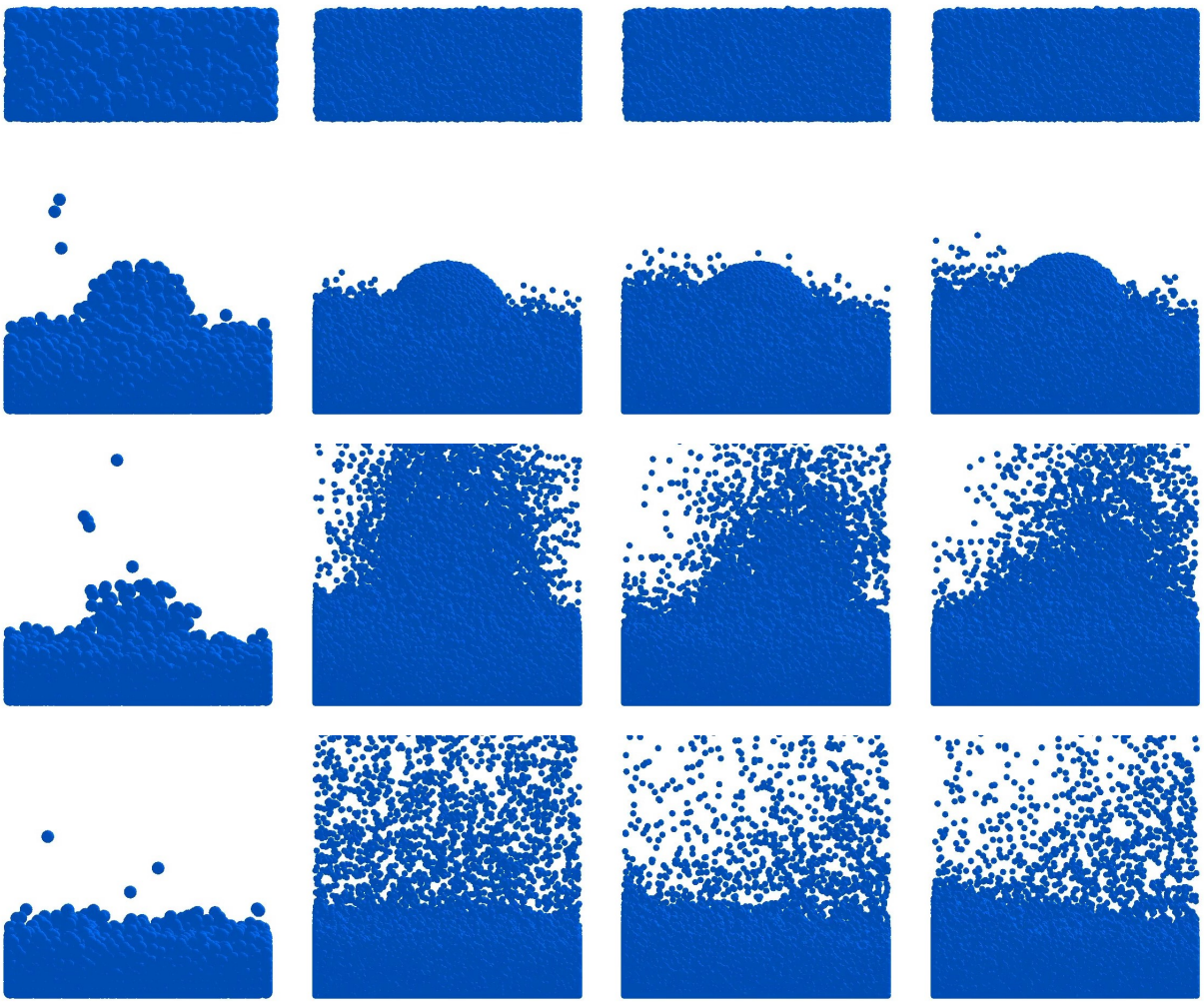

Figure 6: Varying frames from simulations of the splash scene with different resolution and control settings. From left to right column: LR, no control HR, automated half grid standard HR, automated half grid blended HR.

matching, and the last column shows the automated left half grid with blended matching. In these frames, there does not appear to be a significant difference between using standard versus blended matching.

In comparison to the automated grid, the difference between standard and blended control is more apparent in the manual grid for the splash. The columns in Figure 7 are organized the same as before, with the leftmost column showing the LR preview, the next column showing the HR simulation with no control, the third column showing the HR simulation with standard matching, and the final column showing the HR simulation with blended matching. In Figure 7, there is a clear difference in the third shown frame between the last two columns. With standard matching, the particles that are part of the splash are much more spread out than those same particles in the HR simulation that used blended matching. As a result, the blended matching version has a more spherical shape, closer to the splash seen in the original LR preview. Thus, in some cases, it appears that using the blended matching method can yield a smoother animation that is more reminiscent of the LR simulation.

Table 3: Complete splash simulation times in seconds

LR - no control	**HR - no control**	**LR - grid control**	**HR - grid control**
38.3193	962.382	43.6334	992.153

The simulation times captured for blended matching actually outperformed those seen in standard matching for the splash simulation and are presented in Table 3. For the blended matching system, the LR preview with control only took approximately 13.9% longer than the LR preview without control, and the HR simulation with control took approximately 3.1% more time than the HR simulation without control. While the difference in time in the LR preview between this and the standard matching model is minor, the difference for the HR simulation is significant. One possible explanation for this is that running the match point control algorithm at each time in the simulation rather than only at times that correlate directly to a keyframe helps prevent large errors

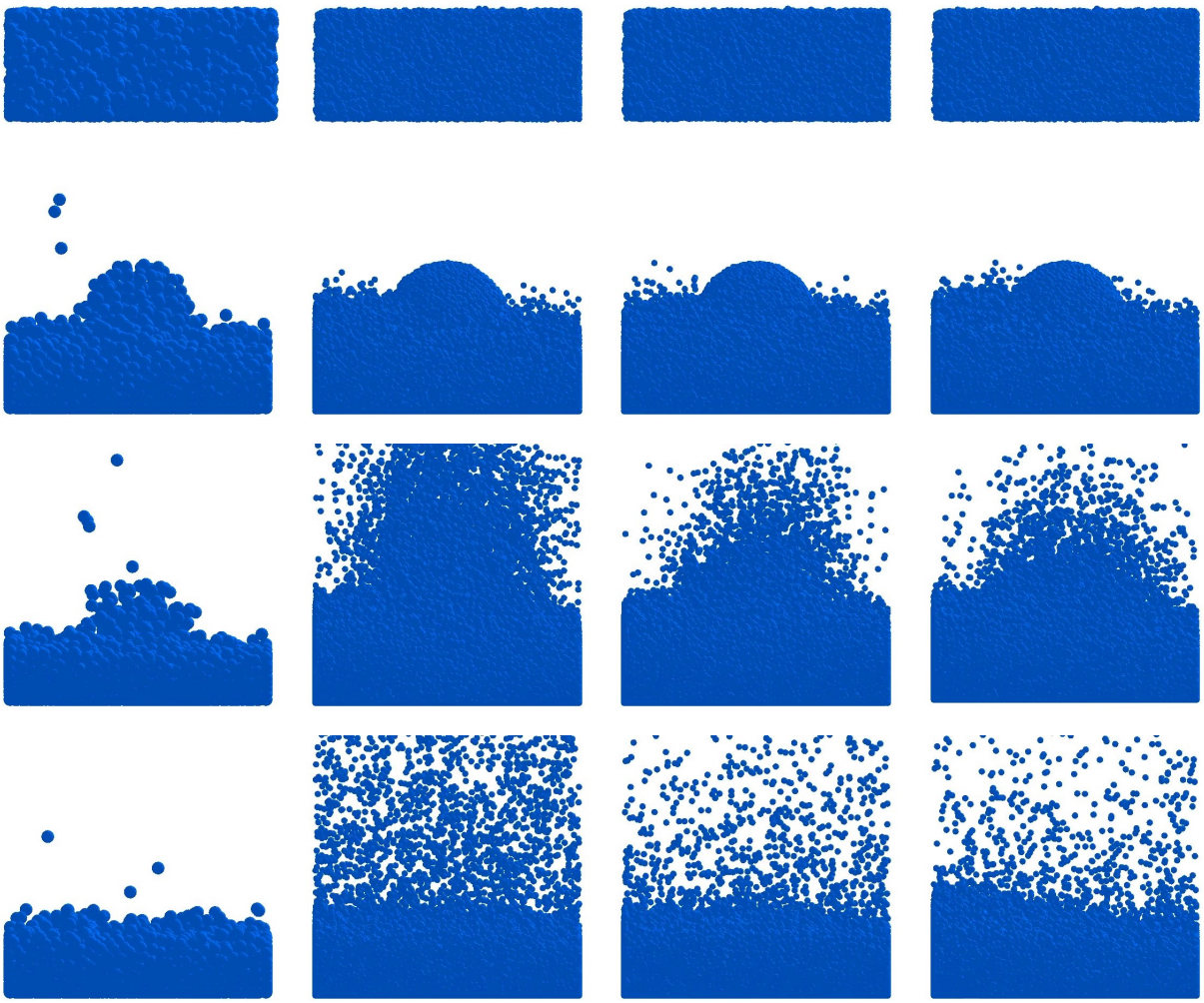

Figure 7: Varying frames from simulations of the splash scene with different resolution and control settings. From left to right column: LR, no control HR, manual standard grid HR, manual blended grid HR.

that would require several correction iterations from occurring.

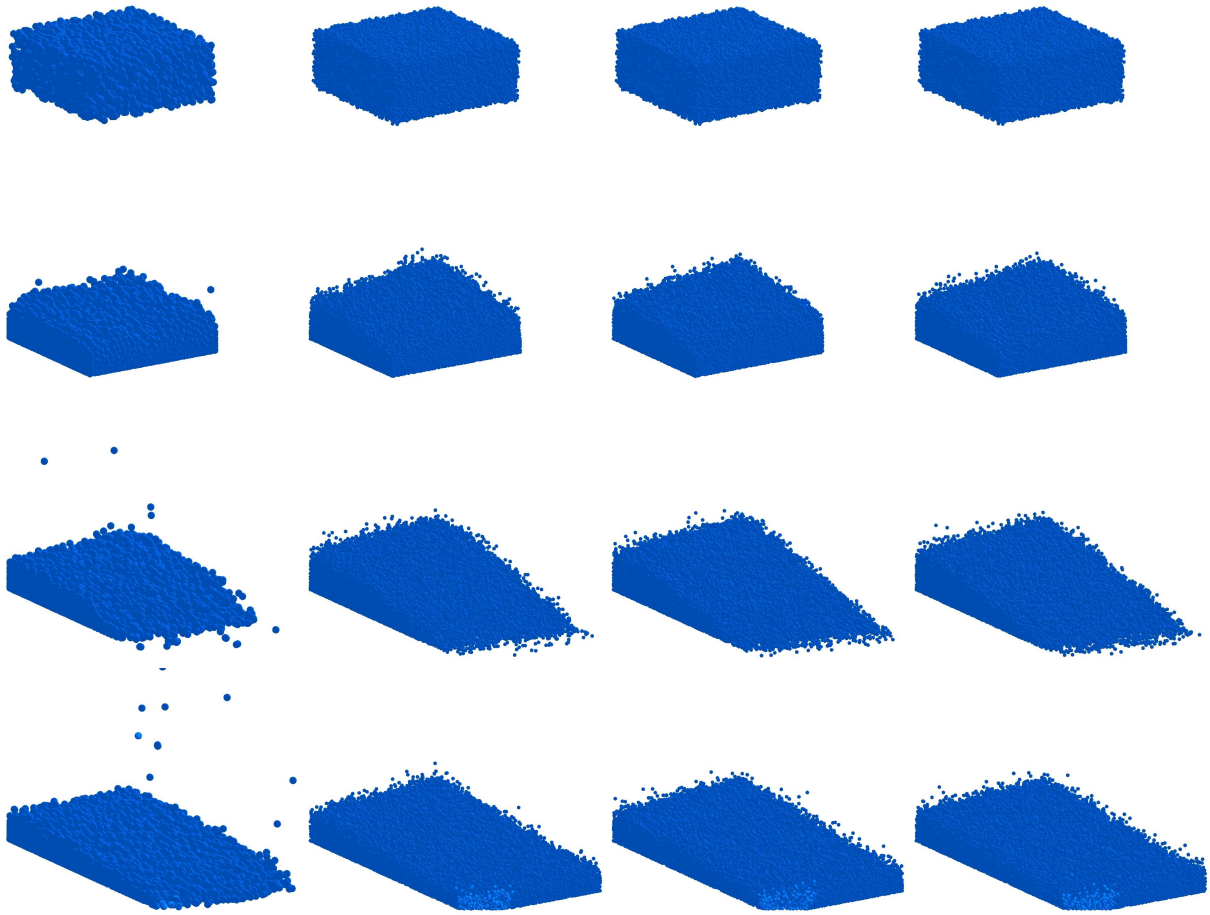

Figure 8: Varying frames from simulations of the dam scene with different resolution and control settings. From left to right column: LR, no control HR, manual standard grid HR, manual blended grid HR.

Column order for the dam simulations in Figure 8 is the same as those discussed for the splash manual grid simulations. In these frames, there is no significant visual difference between the standard and blended matching methods. However, since there is already little difference between the HR without control and the HR with control (standard or blended matching), the dam scene might be insufficient to further test blended matching.

Table 4: Complete dam simulation times in seconds

LR - no control	HR - no control	LR - grid control	HR - grid control
74.3804	1861.28	84.8077	1971.21

Despite offering no visual advantages, the collected timing data for the dam simulation indicated that blended matching provides quicker results than standard matching. The LR preview took approximately 14.0% more time than its uncontrolled counterpart, and the HR simulation took approximately 5.9% more time than the HR simulation with no control. These results seem consistent with those seen in the splash simulation, but the time advantage provided by blended matching should be explored in more depth before blended matching is used for the sole purpose of reducing time cost.

Blending seemed to help alleviate some of the discrepancies between LR and HR simulations seen in the standard matching results. In at least one case, blended matching resulted in a smoother animation. Additionally, blended matching appeared to have a lesser time cost than standard matching, especially during the HR simulation. Blended matching seems to show some promise at creating more natural looking simulations and improving the overall time of incorporating sampling into a fluid simulation but should be investigated further.

5. CONCLUSION

5.1 Conclusion

The match point and sampling method presented originally for Eulerian simulations could successfully be adapted to Lagrangian simulations. While the majority of the algorithm is consistent with what was proposed in earlier research, two novel ideas presented in this paper include applying a sparsity check to achieve better sample quality and blending between keyframes using the match points stored in each one. A significant difference could be seen between controlled versions of a HR animation versus their uncontrolled counterparts, and the time cost incurred by sampling was not excessive. As a result, sampling shows promise as a control scheme for Lagrangian simulations.

5.2 Limitations and Future Work

In the previous sampling work, an algorithm was developed to detect features of interest within the simulation [4]. The goal of this method was to be able to spare artists tedium by providing a means of identifying key areas in the simulation by examining quantities such as fluid density. While this measure worked well for gaseous Eulerian simulations, modifying density through match point sampling does not work as effectively for Lagrangian simulations due to differences between grid-based and particle-based simulations. However, this does not mean there is not a method of identifying key fluid features for Lagrangian simulations. This possibility was not explored by this research but should be seriously considered as a valid avenue of future research.

Additionally, a measure of sparsity proposed by another paper was used to determine whether a match point sample was good enough to be used to apply control [12]. While this proved to be sufficient to create a working implementation of sampling, it resulted in finer details that may only consist of a few particles during the LR preview to be omitted entirely from any final HR simulation. Investigating finer methods of determining the sparsity of a neighborhood surrounding a match point might yield a system that is capable of higher levels of precision.

The amount of match points required to accurately capture a feature can be immense. In the simulations in which half of the scene was covered by match points, a noticeable slowdown occurred even during the LR preview stage. This is detrimental to the efficacy of this method because an artist should be able to quickly iterate and review the animation during this stage. One potential solution to this problem could be shifting the sampling process from the CPU where it currently resides to the GPU using CUDA.

Another noticeable bottleneck in the system is finding neighboring particles. This is another challenge that could potentially be resolved by shifting more work to the GPU. The implementation used in this paper was borrowed from the publicly available cyCodeBase library [14]. The kd-tree implemented is run on the CPU despite the rest of the physics simulation running on the GPU through CUDA. This makes frames where the simulation is conducting a neighbor search, which is already one of the most computationally expensive parts of the simulation, result in an obvious slowdown. While the slowdown as a whole is unavoidable, a GPU implementation might be able to help alleviate it.

One possibility overlooked by this research is that some case may exist where the error at a match point may never converge to an acceptable value. In contrast to the Eulerian sampling model, there is currently no proof for convergence, and it cannot be guaranteed. At present, the model is given some heuristically determined number of corrective iterations to attempt before giving up on matching a quantity between LR and HR simulations. However, it would be useful in the future to either prove that this process will either converge or prove that it will not converge definitively.

One major limitation of this model is that it was created using generic SPH rather than PCISPH. As a result, the fluid in the simulation might not be truly incompressible. For certain animations, incompressibility may be a necessity. Moving the sampling and keyframe matching work to a simulator that uses PCISPH is of great interest to furthering the development of this control method.

The final results of this research could be improved aesthetically as well. As seen in the

images in Chapter 4, all fluids in this project were rendered as a collection of particles. While this technique was sufficient for the purposes of testing the low resolution and high resolution simulations for this paper, it does not create a convincing liquid. Using the anisotropic kernel algorithm with marching cubes could be used to create a liquid surface for a more realistic fluid [13].

REFERENCES

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[2] O. Cornut. Dear imgui. https://github.com/ocornut/imgui, October 2022.

[3] R. Fattal and D. Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, August 2004.

[4] R. Huang and J. Keyser. Automated sampling and control of gaseous simulations. *Vis. Comput.*, 29(6–8):751–760, June 2013.

[5] R. Huang, Z. Melek, and J. Keyser. Preview-based sampling for controlling gaseous simulations. pages 177–186, 08 2011.

[6] M. Macklin and M. Müller. Position based fluids. *ACM Trans. Graph.*, 32(4), July 2013.

[7] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3):449–456, August 2004.

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

[9] X. Shao, E. Liao, and F. Zhang. Improving sph fluid simulation using position based dynamics. *IEEE Access*, 5:13901–13908, 2017.

[10] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28(3), July 2009.

[11] A. Sommer and U. Schwanecke. Leaven - lightweight surface and volume mesh sampling application for particle-based simulations. *CSRN*, 2021.

[12] A. Sommer, U. Schwanecke, and E. Schoemer. Interactive high-resolution simulation of granular material. *Journal of WSCG*, 30:9–15, 01 2022.

[13] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, 32(1), February 2013.

[14] C. Yuskel. cycodebase. https://github.com/cemyuksel/cyCodeBase, December 2022.