UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JOSE ABEL TICONA LARICO

Towards Interactive Simulation of Soft, Rigid and Viscous Objects in Immersive Virtual Reality

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof^{a}. Dra. Luciana Nedel
Coadvisor: Prof. Dr. Rafael Torchelsen

Porto Alegre
September 2020

CIP — CATALOGING-IN-PUBLICATION

Ticona Larico, Jose Abel

Towards Interactive Simulation of Soft, Rigid and Viscous Objects in
Immersive Virtual Reality / Jose Abel Ticona Larico. – Porto Alegre: PPGC da UFRGS, 2020.

96 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Luciana Nedel; Coadvisor: Rafael Torchelsen.

1. Physically-based Animation. 2. Extension Position-Based Dynamic. 3. Smooth Particle Hydrodynamic. 4. Virtual Reality. 5. Shifting Phase. 6. 3D Sketching. I. Nedel, Luciana. II. Torchelsen, Rafael. III. Título.

“It doesn’t matter how slowly you go as long as you do not stop.”

— CONFUCIUS

ACKNOWLEDGEMENTS

I want to acknowledge in Spanish, my mother language, because I feel my words will come out more easily.

En primer lugar dar las gracias a Dios, por cuidar de mi familia durante mi ausencia, y por darme salud y fortaleza para seguir adelante y cumplir con mi objetivo trazado. Esta Tesis está dedicada en memoria de mi abuelo Avelino Ticona Flores, que me dejó durante mi estadía en Brasil, se que hubiera estado muy orgulloso de mi. Me gustaría comenzar agradeciendo a mi orientadora la profesora Luciana Nedel por darme la oportunidad de hacer la maestría aquí en la UFRGS, y de estar presente en los momentos indicados, a mi co-orientador el profesor Rafael Torchelsen, por ayudarme con su experiencia en gran parte de mi maestría, al profesor Anderson Maciel por su colaboración y soporte técnico en los diferentes trabajos realizados. Agradecer a mis amigos Jorge y Fabian por su amistad incondicional y por que me enseñaron muchas cosas que me permitieron continuar adelante. A todos mis compañeros de laboratorio que de alguna manera me ayudaron durante este periodo además de compartir gratos momentos. Agradecer también a todos mis parceros peruanos y colombianos que siempre estuvieron presentes para motivarme y aconsejarme en los momentos indicados. Y por último expresar mis sinceras agradecimientos a mi abuela Dora, mis padres Abel y Rosa, mis hermanos Stive, Michael y Nayely, y a toda mi familia (*grande y numerosa* como decía mi abuelo) que a pesar de estar lejos supieron entenderme y aconsejarme siempre.

ABSTRACT

Objects composed of fluids, rigid, and soft bodies are present in our planet, and people interact with them in their daily life. Also, some materials change their shape and behavior due to changes on physical properties depending on environmental conditions such as air pressure, and heat. This is known as phase-change phenomena. They are present in the melting of snow when winter ends, the solidification of rocks into a volcanic eruption, the evaporation of a fluid when it reaches its boiling temperature, or the evaporation of the oceans due to the temperature increase.

The current development of physics-based animation attempts to simulate materials and their phase-change transitions. However, there is still not a method to unify all transitions between the states of the material, nor an intuitive tool to allow users to interact with materials in different states in real time. In this work, we propose to integrate interactive simulations based on a particle-based framework in virtual environments. Extended Position-Based Dynamics (XPBD) and Smoothed Particle Hydrodynamic (SPH) methods were modified and adapted to create a general method that unifies materials and allows the interaction with and among them. We simulate objects composed of viscous, rigid, and soft materials and allow the transition between their physically properties modeling the phase-change phenomena. We also propose a virtual tool that allows interacting in the virtual environment and sketching new objects using these objects within a virtual reality experience.

We demonstrate that the interactive methods proposed allow reaching plausible visual simulations. Our results show that the reproduction of melting, solidification, evaporation, and condensation phenomena is closer to reality, furthermore including the visualization of the dilation and convection effects. Another result reached is the high realism of the 3D models in Virtual Reality because of the modeling inspired in Physics to enrich 3D drawings with dynamic behaviors, and the ability for users to interact with the model dynamically, modeling it as it moves, deforms, and falls. Our proposal is based on interactive methods and has a low computation cost.

Keywords: Physically-based Animation. Extension Position-Based Dynamic. Smooth Particle Hydrodynamic. Virtual Reality. Shifting Phase. 3D Sketching.

Rumo à simulação interativa de Objetos deformáveis, rígidos e viscosos em realidade virtual imersiva

RESUMO

Objetos compostos por fluidos, corpos rígidos e flexíveis estão presentes em nosso planeta, sendo que as pessoas interagem com eles em sua vida cotidiana. Alguns materiais mudam sua forma e comportamento devido a alterações nas propriedades físicas, dependendo das condições ambientais, como pressão do ar e calor, também conhecidos como fenômenos de mudança de estado. Exemplos de mudanças de estado são o derretimento da neve quando o inverno termina, a solidificação das rochas em uma erupção vulcânica, a evaporação de um líquido quando atinge a temperatura de ebulição, ou a evaporação dos oceanos devido ao aumento da temperatura.

A animação baseada em Física tenta simular materiais e suas transições de mudança de estado. No entanto, ainda não existe um método para unificar todas as transições entre os diferentes estados do material, nem uma ferramenta intuitiva para permitir que os usuários interajam com materiais em diferentes estados em tempo real. Neste trabalho, propomos a integração de simulações interativas em ambientes virtuais, sobre um framework baseada em partículas. Conseguimos simular objetos compostos de materiais viscosos, rígidos e macios e transformar suas propriedades físicas modelando os fenômenos de mudança de fase, além de uma ferramenta virtual que permite interagir e esboçar novos objetos usando esses materiais na realidade virtual. Os métodos de Dinâmica Baseada em Posição Estendida (XPBD) e Hidrodinâmica de Partículas Suavizadas (SPH) foram modificados e adaptados para criar um método geral que unifica os materiais e permite a interação com e entre eles.

Demonstrou-se que os métodos interativos propostos permitem alcançar simulações visuais plausíveis. Os resultados mostram que a reprodução dos fenômenos de fusão, solidificação, evaporação e condensação está próxima da realidade, incluindo ainda a visualização dos efeitos de dilatação e convecção. Outro resultado alcançado é o alto realismo dos modelos 3D em Realidade Virtual, devido à modelagem inspirada na Física para enriquecer os desenhos 3D com comportamentos dinâmicos e a capacidade dos usuários de interagirem dinamicamente com o modelo, criando novas formas à medida que os objetos se movem, deformam e caem. Além disso, esta proposta é baseada em métodos interativos e possui baixo custo computacional.

Palavras-chave: Animação baseada em Física. Dinâmica Baseada em Posição extendida. Hidrodinâmica de Partículas Suavizadas. Mudança de fase. Realidade Virtual. Esboço 3d.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

FEM	Finite Element Methods
PBD	Position-Based Dynamic
XPBD	Extended Position-Based Dynamics
SPH	Smooth Particle Hydrodynamic
MPM	Material Point Method
FLIP	Fluid Implicit Particle
WCSPH	Weakly compressible Smooth Particle Hydrodynamic
PCISPH	Predictive-Corrective Incompressible Smooth Particle Hydrodynamic
IISPH	Implicit Incompressible Smooth Particle Hydrodynamic
PF	Projective Fluid
PBF	Position-Based Fluid
DFSPH	Divergence-free Smooth Particle Hydrodynamic
SDF	Signed Distance Fields
FEM	Finite Elements Method
VR	Virtual Reality
HMD	Head-Mounted Display

LIST OF SYMBOLS

b	scalar b
$\mathbf{b}$	vector b
$\dot{\mathbf{b}}$	first derivate of the vector **b**
$\ddot{\mathbf{b}}$	second derivate of the vector **b**
t	time
Δt	time step
$\#_{iter}$	iteration count
m	mass
w	inverse mass
$\mathbf{v}$	velocity
$\mathbf{a}$	acceleration
F_{res}	result Force
F_{int}	internal Forces
F_{ext}	external Forces
$\mathbf{x}_i$	position vector of the particle i
$\mathbf{x}_i^*$	position vector correctly of the particle i
$\Delta\mathbf{x}_i$	correction vector of the particle i for the next time step
$\mathbf{v}_i$	velocity vector of the particle i
$\mathbf{a}_i$	acceleration vector of the particle i
$C_j(\mathbf{x})$	constraint j that represent a scalar function
$\nabla C_j(\mathbf{x})$	gradient vector of the constraint j
$\mathbf{C}$	vector of constraint functions
$U(\mathbf{x})$	energy potential
λ	Lagrange multiplier

$\Delta\lambda$	correction Lagrange multiplier
k	stiffness of the constraint
α	compliance of the constraint
$\boldsymbol{\alpha}$	compliance diagonal matrix
$\tilde{\boldsymbol{\alpha}}$	compliance matrix
v	viscosity
ρ_i	density of the particle i
V	volume total
V_i	volume of the particle i
p_i	pressure of the particle i
T_i	temperature of the particle i
C	specific heat capacity
K	thermal conductivity
E	stiffness (Young's modulus)
e	linear coefficient of expansion
ΔT	variation of the Temperature
d_0	initial distance between particles

CONTENTS

1 INTRODUCTION

Physically-based animation is a computer graphics topic that, during the last decades, has contributed to the development of video games, medical simulations, special effects in movies, virtual reality, and simulation of natural phenomena. These applications require high-performance, optimization of the resources, high-quality rendering, robust stability, and real-time processing, and are usually used in 2D or 3D simulated environments. Orienting these types of applications for virtual reality environments brings significant realism and allows people to interact directly with each simulation, respecting the physical principles.

The traditional methods based on impulse dynamic (internal and external forces) using implicit integration for solving large systems of equations, have high computational cost and are usually used in off-line workflows (BARAFF; WITKIN, 1998). An explicit/implicit mixed time integration scheme was attempted to speed up integration time (BRIDSON; MARINO; FEDKIW, 2005); both works present results with plausible behavior. The Finite Elements Method (FEM) has higher accuracy and is used in computer graphics; however, it is very slow. An optimization was formulated by Euler's implicit method with the corotated FEM (KUGELSTADT; KOSCHIER; BENDER, 2018). Another group of the particle-based methods is called Lagrangian that in the last few years has been very popular because its methods can be easily parallelized and allow us to use GPUs to accelerate the processes (MACKLIN et al., 2014; IHMSEN et al., 2011).

In this work we present the exploration and adaptation of the Extended Position-Based Dynamics (XPBD) and Smoothed Particle Hydrodynamic (SPH) methods to simulate objects in their three phases of matter (solid, liquid, and gas). Afterwards, we simulate the transitions between the matter phases through the modeling of all phase-changes phenomena. Then we propose and implement a virtual reality tool that allows us to sketch 3d objects using the materials modeled. The outcome of this work is the integration between simulation and interaction in virtual reality, where users are able to create viscous, rigid and deformable objects and, at the same time, to test their physical behavior.

1.1 Motivation

A challenge for the physics-based animation research is to be able to replicate and visualize the behavior of the matter changing phenomena in a controllable manner, that will allows comparing, analyzing, making predictions, and validating theories.

The water is the only material that we can find naturally in its three states: solid, liquid, and gaseous (ice, water, and steam, respectively). The shifting phase is a phenomenon that modifies the properties of the material, involves processes such as solidifying, melting, evaporating, and condensing, and is directly dependent on pressure and temperature.

Some works approach the problem by separately reproducing each process. Gao et al. show only melting simulation based on the heat-based fluid (GAO et al., 2017b). Stomakhin et al. introduce a novel material point method for melting and solidifying simulation (STOMAKHIN et al., 2014); another similar work is that of Farrokhpanah et al. that use SPH method and introduces the equations of the latent heat (FARROKHPANAH; BUSSMANN; MOSTAGHIMI, 2017). To simulate the evaporation and condensation transition, Hochstetter et al. use a coarse grid for the air phase and mass preservation, along with a rigid and liquid body simulation based on SPH (HOCHSTETTER; KOLB, 2017). Since there is no unified method that allows integrating the four transitions, it is essential to propose a bidirectional transition model to simulate the phase-change.

In the area of Virtual Reality, much progress is being done on the creation of new devices and applications that increase our experience in immersive environments. Tasks like virtual shopping, walking in the streets, manipulating laboratory objects, 3D modeling, sculpting, were proposed but still do not reach a realistic level; an essential factor is the effect of animation given to 3D materials to improve the immersive experience.

The 3D modeling involves defining the shape, choosing the material, and the right tool to support the creation of a 3D object to allow users to customize their prototypes. Additionally, it is important that users can interact with the simulation in real-time, just like the Fluid Sketching-Immersive tool does (EROGLU et al., 2018), enabling artists to draw fluid-like sketches using a 3D brush and offering natural interaction methods for shaping the drawn sketches. Other works on virtual reality that also allow interaction with 3D objects are commercial products with a high degree of immersion, like the TiltBrush (GOOGLE, 2016) (See Figure 2.6) and the Quill (FACEBOOK, 2016), from Google and Facebook, respectively. Their most noticeable weakness is that the 3D prototypes are static, it only remains in rendering, and do not interact with the user.

1.2 Contribution

In this work we propose a continuously full Lagrangian Unified method to simulate solid, liquid, and gaseous materials with their transitions based on the modeling of phase-change phenomenon. We used the Extended Position-Based Dynamic (XPBD) (MACKLIN; MÜLLER; CHENTANEZ, 2016) to simulate rigid and soft bodies and the Smooth Particle Hydrodynamic (SPH) (AKINCI et al., 2012) to simulate fluids (liquid and gas). Both methods are particle-based, and allow easy integration and adaptation for heat-transfer support by particle contact.

The stiffness, distance, viscosity, density, and gravity parameters usually have constant values during simulation; however, we vary their values depending on the change in temperature and latent heat. We propose a solid manager to create and destroy distance constraints during the transition of solid-liquid (melting and solidification). To change the values of the other parameters continuously, we define smoothed functions derived on the sigmoid and logarithm functions, and this makes the stable transitions.

We also integrate the simulation methods within virtual reality to offer a modeling tool using a grid structure that allows controlling the insertion and erasing particles, avoiding overlapping during the simulation. For this, we propose a particle-manager for defining the behavior depending on the type of material in use.

We demonstrate the versatility and stability of the interactive methods used, achieving plausible visual results. In the proposed tool, users can interact directly with simulations in real-time. It enables us to create powerful tools like replicating natural phenomena and 3D modeling in VR using different types of materials, with a high degree of immersion, while safeguarding physical behavior.

1.3 Collaboration in this Work

The motivation for this work came from a first year Master course. During the course on Animation and Physics-based Simulation, students are invited to develop an implementation work in pairs. In this context, we proposed to simulate phase-change transitions using thermodynamics and solving the heat equation, a very ambitious project.

This work was developed in a tandem with the Master student Steeven Villa Salazar. Because of the promising results achieved, even with the end of the course, we continued working together, pursuing other still open aspects of this project (the first results were published in the Computer & Graphics journal). As a consequence, both students had very similar Master Thesis topic and results.

The main contributions of Steeven Villa Salazar were the modeling of heat transfer using SPH, the incorporation of latent heat, the research on methods of phase-change, the support with user tests formulation, technical advice on human-computer interaction, and help with the design of figures and writing the papers.

I contributed with the modeling of solid and deformable objects using XPBD, and the fluid–solid integration. I also proposed the model that allows the transition from solid–liquid and liquid–gas, and modeled and implemented the virtual tool to interact with the particles.

Together, we have a pending project that consists on the integration of the particle model with force rendering using haptic devices, what would make our system not only looks real, but also feels real, producing a greater immersive experience.

1.4 Document Organization

This document is organized as follows. In Chapter 2, we review previous works and existing applications. In Chapter 3, we present the core concepts, main algorithms, simulation methods, physics, and mathematic equations used as the base of this work. Chapter 4 shows the implementation and adapted method for supporting the phase-change simulation, including the results achieved. Chapter 5 presents the construction of the virtual tool for 3D sketching. Finally, in Chapter 6 we present our conclusions, limitations of this work and future work.

2 RELATED WORK

We reviewed the most relevant works for this project. First, we present the physics-based methods to simulate materials such as solid and soft bodies, liquid, and gas. Then, we introduce methods to simulate the phase-change phenomena heat-based of the materials. Finally, we cover works and applications to model and design 3D materials in virtual reality environments.

2.1 Material Simulation Methods

Jakobsen introduced the position-based approach, and presented a general approach that handles general constraints; his central idea was to use a Verlet integrator and to manipulate positions directly (JAKOBSEN, 2001). Because current and previous positions implicitly calculate the velocities, it allows the simulation of cloth, soft, and rigid bodies.

In order to make an overview of simulation methods of deformable objects, Nealen et al. explained different ways to model it. The finite elements, mass-spring systems, mesh-free methods are the more significant ones (NEALEN et al., 2006). They also give us a vision of some topics, such as elastoplastic deformation and fracture, virtual surgery simulation, interactive entertainment, and fluid/gas animation, and suggest areas for future research.

Position-based dynamics (PBD) was formulated by Müller et al. They provide a mathematical framework of constraints to generate various behaviors; this work has been popular in the last years because it is stable and runs in real-time (MüLLER et al., 2007). Years later, Bender et al. present a tutorial explaining each position-based constraint (BENDER; MÜLLER; MACKLIN, 2015). However, the issues are that it does not converge to a real solution and dependent on the stiffness constraints.

Nucleus is a Unified Dynamics Solver developed by Autodesk. This model is a generalization of a triangle mesh that includes points, curves, and solids; it allows interactions such as collisions between various elements of different dimensionalities (STAM, 2009). Internal deformations such as stretch and bend are handled through constraints instead of springs making the simulation more stable.

Müller et al. propose a new fast and robust method to simulate various types of solid, including rigid, plastic, and soft bodies as well as one, two, and three-dimensional structures such as ropes, cloths, and volumetric objects (MÜLLER; CHENTANEZ, 2011). The underlying idea is to use oriented particles storing rotation and spin, along with the usual linear attributes, such as position and velocity.

In the last few years, methods have appeared that improve accuracy and convergence. Projective Dynamic (PD) proposed by Bouaziz et al. is a method for implicit time integration of physical systems, inspired by FEM and PBD. Based on continuum mechanics, they derive a set of continuum-based potentials that are efficiently incorporated (BOUAZIZ et al., 2014).

Extended Position-Based Dynamics (XPBD) is an extension of PBD that improves the precision and efficiency of the simulation implicitly, derived from an energy potential $U(\mathbf{x})$ (see Equation 2.1) (MACKLIN; MÜLLER; CHENTANEZ, 2016). PD and XPBD methods are only an approximation of an implicit Euler integrator. In this work, we will implement the XPBD method.

$$\mathbf{M}\ddot{\mathbf{x}} = -\nabla U^T(\mathbf{x}) \tag{2.1}$$

Bender et al. present a general survey on position-based dynamic methods (BENDER; MÜLLER; MACKLIN, 2017). The authors introduce the basic concept of position-based dynamics, and its improvement to XPBD also presents different solvers to simulate elastic rods, cloth, volumetric deformable bodies, rigid body systems, and fluids. Approaches are fast, stable, and controllable for use in interactive environments.

An important topic in literature is the simulation of fluids. There are various methods that simulate fluids. The most important methods, are the Fluid Implicit Particle (FLIP) proposed by Brackbill et al. (BRACKBILL; KOTHE; RUPPEL, 1988) and the Smooth Particle Hydrodynamic (SHP) introduced by Monaghan (MONAGHAN, 1994). FLIP is an adaptation to fluids of the implicit moment method for simulating plasmas, using the particle data. Lagrangian moment equations are solved on a grid. The solutions are then used to advance the particle variables from time step to time step. SPH is one of the most popular methods extended to deal with incompressible fluids, and introduced the concept of a density estimator based in the search neighborhood. Both methods have served as the basis for the development of new methods to this day.

Solvers such as Weakly Compressible Smooth Particle Hydrodynamic (WCSPH), developed by Becker et al. preserves the initial volume of the fluids using forces of the superficial tension (BECKER; TESCHNER, 2007). Predictive-Corrective Incompressible Smooth Particle Hydrodynamic (PCISPH) was proposed by Solenthaler et al. and achieves the incompressibility, proposing a correction scheme to determine pressures between particles (SOLENTHALER; PAJAROLA, 2009). Position-Based Fluid (PBF) introduced by Macklin et al. defined new density constraints to integrated into PBD solver, it is stable because computes a Langrarian operator faster (MACKLIN; MÜLLER, 2013).

Another stable method is the Projective Fluid (PF) proposed by Weiler et al. based in the Projective Dynamic (PD) and the Smooth Particle Hydrodynamic (SPH), solving pressure constraints (WEILER; KOSCHIER; BENDER, 2016). Recently, Jakob et al. presented a tutorial at Eurographics 2019 conference, covering various aspects of SPH simulations (KOSCHIER et al., 2019). They discussed aspects as governing equations for mechanical phenomena and their SPH discretizations. They also introduced concepts and implementations of core components such as neighborhood search algorithms, pressure solvers, and boundary handling techniques.

The soft materials are essential for our work, and we reviewed some works that helped us to model it. Gray et al. introduced a method that generated artificial stress to simulate the fracture of the solid using SPH (GRAY; MONAGHAN; SWIFT, 2001). Another work from Becker et al. presented a corotational formulation for elastic solids based on SPH. The rotations in the deformation field are computed using an SPH variant of the shape matching method (BECKER; IHMSEN; TESCHNER, 2009). Bender et al. introduced a novel fast and robust simulation method for deformable solids supporting complex physical effects such as a lateral contraction, anisotropy, or elastoplasticity (BENDER et al., 2014).

Interactive rigid body simulations have become an essential part in different application areas. Its characteristic is to keep its original shape during the simulation. In order to get this, it is required efficient and accurate methods for contact constraints as well as fast collision detection. Bender et al. explain that the solid simulations are crucial in part of many modern tools, not only for games but also for designers and engineers (BENDER; ERLEBEN; TRINKLE, 2014).

Deul et al. incorporated new rigid constraints that can be integrated into Position Based Dynamic (PBD) (DEUL; CHARRIER; BENDER, 2016) to enables two-way coupling rigid and deformable bodies. Another behavior of the rigid body is bending and torsion effects, which are known as rods. Kugelstadt et al. introduced a new constraint to couple orientations, which are represented by unit quaternions and integrated into existing PBD (KUGELSTADT; SCHÖMER, 2016). A recent improvement made by the same author introduced the Langrariam operator based on the XPBD (DEUL et al., 2018).

Another material to simulate is the snow, which has a typical granularity behavior (solid and fluid-like). Here we found a work done by Stomakhin et al. using Material Point Method (MPM), which is an hybrid method between particle-grid (Langrangian/Eulerian) (STOMAKHIN et al., 2013).

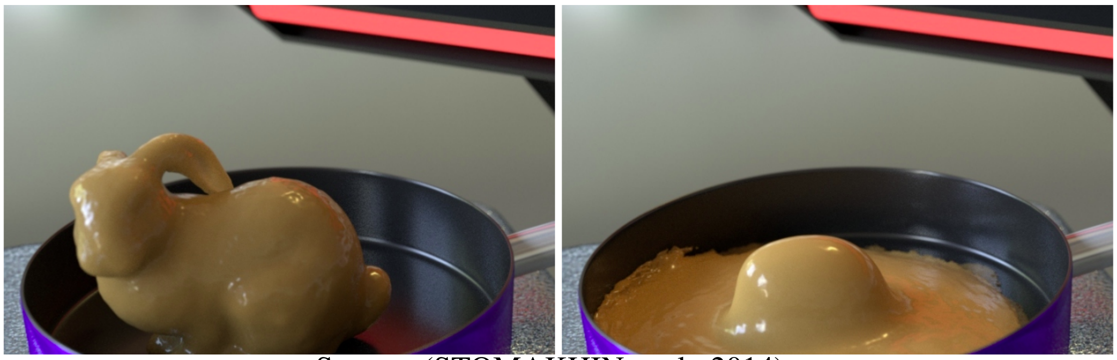
In minor quantity, there are works for gas simulations. Macklin et al. proposed a method based in PBF which begins filling a container with fluid particles, and reducing gravity for gas particles. Ren et al. present a fast SPH simulation for gaseous fluids. Instead of using particles to simulate air, it replaces them with external forces to compensate the density (REN et al., 2016).

2.2 Phase-change Methods

The revision of previous works serve as a basis for seeing more complex simulations involving interaction and coupling between different types of materials. Some of these methods to simulate melting and solidifying phenomena, such as the Material Point Method (MPM) (STOMAKHIN et al., 2014), allows the variation of the material parameters by adapting the heat equation to the MPM solver, and present beautiful visual results but running offline, as can be seen in Figure 2.1.

A similar work was presented by Weiler et al. and used an implicit solver for the simulation of highly viscous fluids using SPH (WEILER et al., 2018). Their strategy is to make very viscous materials to achieve rigid behaviors by making them more natural solid-fluid coupling. Additionally, they discretize the heat equation to get melting simulation.

Figure 2.1: Simulation of a bunny that is melted by hot air contact producing a melting effect ($1.2x10^6$ particles and 11.4 min per frame were used). MPM implements the heat transfer in its solver through the boundaries of its cells.



Source: (STOMAKHIN et al., 2014)

Gao et al. presented a coupling fluid-solid phase using FLIP and PBD-based shape matching constraint to simulate only the melting and strong coupling between fluid and deformable solids (GAO et al., 2017a). In the same year, the authors included the influence of the air in the simulation of solid-fluid coupling and fluid-gas interaction using two methods: FLIP and SPH (GAO et al., 2017b). The visual characteristic is the generation of bubbles (see Figure 2.2).

Domaradzki et al. present a method for simulating melting of ice implementing the heat transfer between ice objects and fluids (water and air). They proposed a new particle-based air model to consider the influence of the natural air (DOMARADZKI; MARTYN, 2014). Another work that also simulate ice was developed by Miao et al. and present a freezing model, coupling phase-based viscosity, and rigid-shape matching constraints to particles (MIAO; XIAO, 2015).

Müller et al. take into account the interaction between SPH-based fluid-fluid, where the air is modeled also as a fluid, generating bubble effects (MÜLLER et al., 2005b). Their technique makes possible the simulation of phenomena such as boiling water, trapped air, and the dynamics of a lava lamp, but is not physically-based.

The transition from a liquid state to a gas state is known as evaporation and condensation phenomena. Hochstetter et al. tried to simulate this phenomenon using a grid for the air phase, and a mass-preserving couple for a SPH-based liquid and rigid body simulation (HOCHSTETTER; KOLB, 2017).

In this work, we propose a method based in the Smooth Particle Hydrodynamic (SPH) and the Position-Based Dynamic (PBD) that implemented the three-phase transitions solid–fluid–gas, achieving solidification, melting, evaporation, and condensation effects. Our method uses distance constraints to simulate the solid–fluid phase, and a

Figure 2.2: An ice block is dropped into the boiling water. As the ice melts, the water temperature is lowered, and then the water stops boiling. Re-boiling happens when the water is heated back to the boiling point again. This scenario shows the generation of bubbles during the fluid-gas transition.

Source: (GAO et al., 2017b)

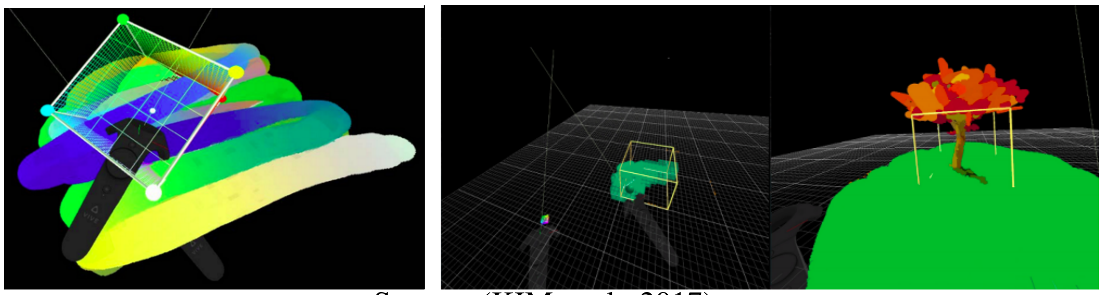
smooth function to vary the density and gravity for the fluid–gas phase (SALAZAR et al., 2018). A similar work was proposed by Bian et al. They designed an unified computational process to model the three phases of water (BIAN; XIAO; LI, 2018). This work is very similar to ours but there are two differences. First is the use of the solver. They use the PBD that has convergence and stability problems, while our work uses the XPBD, which improves the deficiencies of the PBD (MACKLIN; MÜLLER; CHENTANEZ, 2016). Another difference is that they consider heat transfer mechanism and mass transfer mechanism, while in our work we only consider the heat transfer mechanism.

Other types of simulations do not change the properties of the materials but do change their initial shape and structure. The cuts are one type of them, which are widely used in surgery simulations. Pan et al. allow for manipulating and cutting soft tissue models using PBD in real-time (PAN et al., 2015). Similar work was presented by Berndt et al. that introduced a traditional surgery simulation using a heat equation to break connections between particles, keeping the mesh update while supporting interactive tissue cutting with haptic feedback (BERNDT; TORCHELSEN; MACIEL, 2017).

2.3 Modeling and Designing in VR

CanvoX ((KIM et al., 2017)) proposed a new tool for VR painting in 3D space that supports conventional painting tools for artists such as color mixing, eraser, recoloring, and color-blending. As a choice for painting user interfaces, it uses off-the-shelf VR controllers such as the HTC Vive controllers (see Figure 2.3). CanvoX also enables artists to draw huge scenes, and to navigate in a large canvas space with full control of the spatial color fields.

Figure 2.3: Canvox: VR painting system. Left Image: a user picks and changes colors while applying strokes at the same time. Right Image: a typical painting procedure. First, the grow is painted in the scene plane, and on it, a tree is painted.

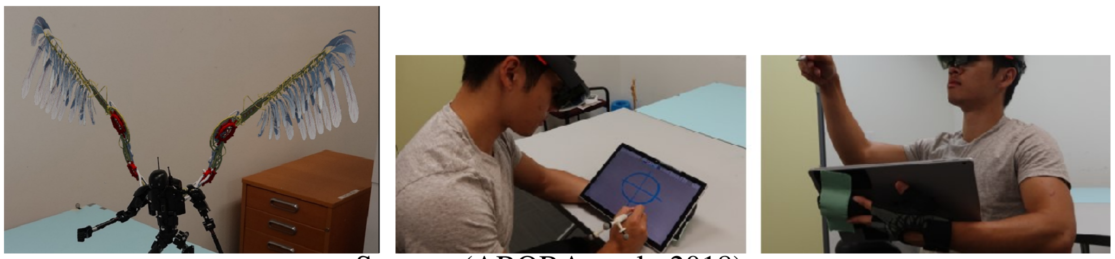


Source: (KIM et al., 2017)

Drawing geometric figures with pressure in 3d dimension is very challenging for users, so Maira D. et al. in his work Multiplanes shows that it is possible to combine the flexibility of freehand drawing and the ability to draw accurate shapes in 3D by affording both planar and beautified drawing. Multiplanes is a VR drawing system generates snapping planes and beautification trigger points based on previous and current strokes and the current controller pose. Based on geometrical relationships to previous strokes, beautification trigger points serve to guide the user to reach specific positions in space. (MACHUCA et al., 2018).

Another work that combine 2D and 3D drawings is SymviosisSketch ((ARORA et al., 2018)), that offers immersive and life-size interaction in VR (3D with HMD), as well as precision and ergonomics for AR (2D with a tablet). SymbiosisSketch directly allows users to interact and draw relative to 3D objects in the physical world (see Figure 2.4). Drawing directly over many physical objects, however, can be hard due to their form, material, or size. HoloLens's capabilities help to map the physical world as a 3D triangle mesh.

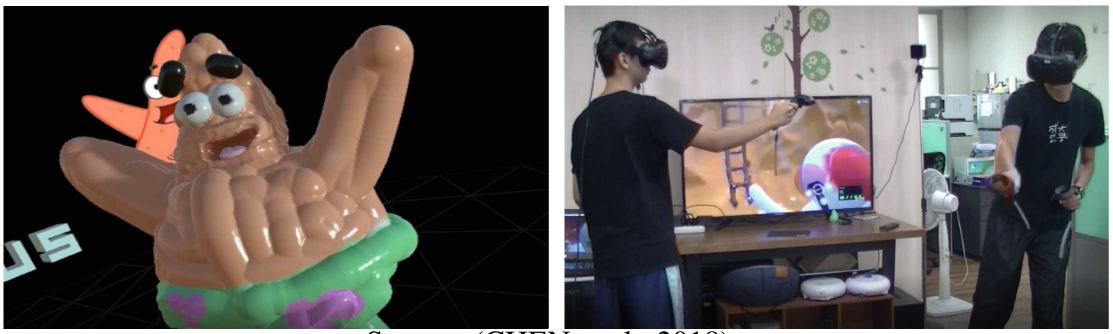
Figure 2.4: SymbiosisSketch: designs can seamlessly blend into the real world, as seen in left, where the wings of a toy robot are drawn with an artistic detail. Setup: the user wears the HoloLens and draws with a motion tracked stylus, on a tablet (middle), or mid-air (right) using a mouse attached to the back of the tablet.



Source: (ARORA et al., 2018)

Other works allow users to paint, sculpt, color, transform, and collaborate with other users in a virtual environment using visual/temporal effects such as the Aura Garden (SEO; BRUNER; AYRES, 2018) and Ontlus (CHEN et al., 2018) (see Figure 2.5). Both allow two participants to simultaneously work together in the same virtual space. Participant have their own controllers and can synchronize and coordinate their movements to create otherwise impossible sculptures.

Figure 2.5: The left image shows a 3D model created by using the Ontlus system. The right one shows two users cooperating in the multi-user mode for simultaneously editing.



Source: (CHEN et al., 2018)

Google proposes the TiltBrush (GOOGLE, 2016), a system developed for professional design and sculpting in VR, mainly focused on artists in general. Figure 2.6 shows how the tool works. Quill was developed by Facebook and is a VR illustration and animation tool for artists and creators (FACEBOOK, 2016). Both systems are completely immersive, but the problem is that they are not physically-based. Additionally, the static behavior and the visual effects presented make the visual perception poor and the sense of presence low.

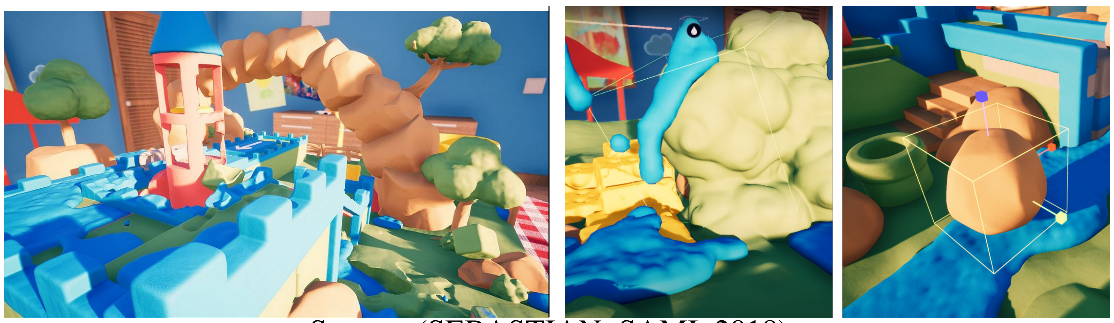
Figure 2.6: Tilt Brush tool allows users to model any object using different types of material. This system presents a simple and intuitive user interface, with a list of all the materials and textures supported. In the example, the user is creating a volcano with lava inside.

Source: by Google

In the literature, some works intended to integrate Physics-based behaviors into VR environments, such as the Fluid Sketching, for instance (EROGLU et al., 2018). It is a novel tool for creating 3D shapes in immersive virtual environments. Based on fluids simulation, it allows the user to modify the properties of the fluids, such as density, viscosity, and adhesion. The drawback here is that it does not support soft and rigid bodies, or cloth simulating.

The work most similar to our work is Claybook, developed by Second Order (SEBASTIAN; SAMI, 2018) and released for video games with support for multiple platforms (Xbox, PS4, and PC). This application allows the generation of dynamic contents with Physically-based behavior. It offers support for different types of materials such as fluids, deformable bodies, clay, and solids based in SPH and SDF method (see Figure 2.7). However, it is not immersive, and does not adequately simulate rigid solids.

Figure 2.7: Claybook allows to render and simulate rich volumetric clay environments. This game makes possible new kinds of user-generated content, as, for example, a clay city (left). Some modeling tools allow the fluid insertion to the scene easily (middle). Another tool allows selecting and moving objects (right).



Source: (SEBASTIAN; SAMI, 2018)

Our work, so called Phys-Sketch, integrates a position-based dynamics simulation solver within virtual reality so that users can design and interact with 3D models in a realistic way in real-time (TICONA et al., 2019).

3 INTERACTIVE SIMULATION METHODS

In the last years, the Lagrangian simulation methods such as the Extended Position-based Dynamics (XPBD) and the Smooth particle hydrodynamic (SPH) attained high popularity in the graphics community. These methods, according to the literature, have shown excellent results to allow the modeling of a wide variety of materials, look stable during simulation, generate behaviors very close to reality, and are easy to parallelize. This chapter introduces and explains the main algorithms, data structures, mathematical and physical models used in this project and is based on the Survey on Position-Based Dynamics (BENDER; MÜLLER; MACKLIN, 2017) and Smooth Particle Hydrodynamic (IHMSEN et al., 2014).

3.1 Equations of Motion

We first introduce the basic theory of physically-based simulation. Internal and external forces change the motion of bodies with mass m. The resulting force of the sum of these forces $F_{res} = F_{int} + F_{ext}$ determines the direction and allows us to calculate acceleration using Newton's second law of motion. Moreover, an integration method is used to compute the velocity and the new position of the object on time t_n.

$$a = \frac{F_{res}}{m} \tag{3.1}$$

Our method is particle-based, so we must calculate the motion equations for a particle, in order to know the first and second derivative of position – the velocity and the acceleration – respectively. Then, from the Equation 3.1 above we have:

$$\ddot{\mathbf{x}}_i = \frac{1}{m_i}\mathbf{f}_i \tag{3.2}$$

$$\dot{\mathbf{x}}_i = \mathbf{v}_i \tag{3.3}$$

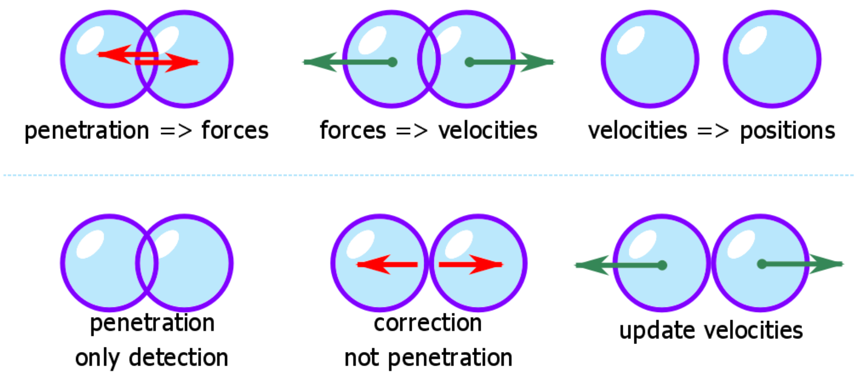
where $\mathbf{f}_i$ is the sum of all forces acting on particle i, $\ddot{\mathbf{x}}_i$ and $\dot{\mathbf{x}}_i$ are the second and first derivative of the position $\mathbf{x}_i$. Equations 3.2 and 3.3 show that the acceleration can be represented as $\mathbf{a}_i = \dot{\mathbf{v}}_i = \ddot{\mathbf{x}}_i$.

3.2 Position-Based Dynamics

In contrast, the position-based dynamics method omits the velocity and acceleration layer and immediately works on the positions. The main advantage of a position-based approach is the controllability over the positions, allowing them to be corrected to ensure the stability of the simulation. A comparison between both methods is represented in Figure 3.1. In general terms, position-based dynamics is a method for calculating the movement of a set of particles or vertex over time. The features and advantages described by Müller are (MüLLER et al., 2007):

- It gives control over explicit integration and removes the typical instability problems
- Positions of vertices and parts of objects can directly be manipulated during the simulation
- It allows handling general constraints in the position-based setting
- The solver is easy to understand and implement.

Figure 3.1: Comparative illustration between the Force-Based Update (upper line) and the Position-Based Update (lower line).



Source: the author.

3.2.1 Representation

A dynamic object is represented by a set of N particles and M constraints. We choose the particle as the fundamental unit for the construction of all types of objects and the constraint to define the relationships and rules between particles, as well as the effects that we want to simulate:

- A particle p_i, has a mass m_i, a position x_i and a velocity v_i, where $i \in [1, ..., N]$, and N is the number of particles.

- A constraint $C_j(p_{i_1}, p_{i_2}, ..., p_{n_j})$, is a function $C_j : R^{3n_j} \to R$, where $\{i_1, i_2, ..., i_{n_j}\}$ is a set of particle indices. C_j consists of a cardinality n_j (number of particles needed by the constraint C_j), $j \in [j, ..., M]$, and M is the number of constraints. This function can be type as equality or inequality.

3.2.2 Algorithm Overview

The Algorithm 1 describes the steps to get a position-based simulation. The lines (1) - (3) initialize the features of the particle i, while the main loop for the simulation for each time step Δt is defined in the lines (4) - (23). The first step is to compute the explicit Euler integration method on the velocities and positions in the lines (5) - (7) and (9) - (11), respectively, for all particles. Line (6) allows adding external forces that cannot be converted to positional constraints. If we need to use a damped function to achieve some effect on the simulation, this should be done on the line (8). On line (10), the new position $\mathbf{x}_i^*$ of the particle i is computed to be used as predictions. The collision constraints for each time step are estimated in the line (13).

Lines (15) - (17) are the core of the method, where the correction of the position $\mathbf{x}_i^*$ is estimated. We repeat these calculations until we reach an acceptable solution. Here, the term of the iteration count $\#_{iter}$ is introduced. Another essential part of the method is in the lines (18) - (21), where the optimized movement is calculated. In line (19), the velocity $\mathbf{v}_i$ is recalculated from the estimated position $\mathbf{x}_i^*$, and them in line (20), the position $\mathbf{x}_i$ is updated. Finally, in line (2), the velocities of colliding vertices are modified according to friction and restitution coefficient.

Algorithm 1 Position-based dynamics

```
1: for all particles $i$ do
2:     initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
3: end for
4: loop
5:     for all particles $i$ do
6:         $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
7:     end for
8:     dampVelocities($\mathbf{v}_1, ..., \mathbf{v}_N$)
9:     for all particles $i$ do
10:        $\mathbf{x}_i^* \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
11:    end for
12:    for all particles $i$ do
13:        generateCollisionConstraints( $\mathbf{x}_i \rightarrow \mathbf{x}_i^*$)
14:    end for
15:    loop solverIteration  times
16:        projectConstraints( $C_1, ..., C_{M+M_{coll}}, \mathbf{x}_1^*, ..., \mathbf{x}_N^*$)
17:    end loop
18:    for all particles $i$ do
19:        $\mathbf{v}_i \leftarrow (\mathbf{x}_i^* - \mathbf{x}_i)/\Delta t$
20:        $\mathbf{x}_i \leftarrow \mathbf{x}_i^*$
21:    end for
22:    velocityUpdate($\mathbf{v}_1, ..., \mathbf{v}_N$)
23: end loop
```

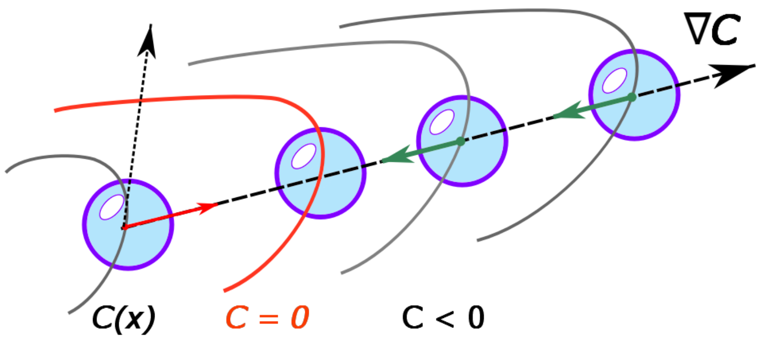
3.2.3 Constraint Projection

In the Section 3.2.1, we mention the information that the constraints represent. Now we will deal with some mathematical aspects that must be fulfilled. First, we assume that a constraint is a kinematic restriction that can be an equation or an inequality that constrain the motion. In PBD, constraints only consider the positions for its formulation. The goal is to find a configuration for a constraint j along its gradient vector ∇C_j, that satisfy the equation $C_j(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_{n_j}}) = 0$ or the inequality $C_j(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_{n_j}}) \geq 0$ (see Figure 3.2). For the sake of simplification, the symbol $\succ$ denotes either $=$ or $\geq$.

The Constraint Projection consists in computing the next position that satisfies the relations mentioned above. Given $\mathbf{x}$, we want to find a correction $\Delta\mathbf{x}$ such that $C(\mathbf{x} + \Delta\mathbf{x}) \succ 0$. To solve and find a solution, PBD linearizes the constraint function with the following approximation:

$$C(\mathbf{x} + \Delta\mathbf{x}) \approx C(\mathbf{x}) + \nabla C(\mathbf{x}).\Delta\mathbf{x} \succ 0 \tag{3.4}$$

Figure 3.2: Constraint function in different steps, and how the PBD algorithm looks for a solution in the direction of the gradient.



Source: the author.

To notice that the correction $\Delta\mathbf{x}$ is in the direction of ∇C, means choosing a scalar λ (a Lagrange multiplier) such that

$$\Delta\mathbf{x} = \lambda\mathbf{M}^{-1}\nabla C(\mathbf{x})^T \tag{3.5}$$

where $\mathbf{M} = diag(m_1, m_2, ..., m_N)$. Substituting Equation 3.5 into Equation 3.4, and solving the λ we have that

$$\lambda = -\frac{C(\mathbf{x})}{\nabla C(\mathbf{x})\mathbf{M}^{-1}\nabla C(\mathbf{x})^T} \tag{3.6}$$

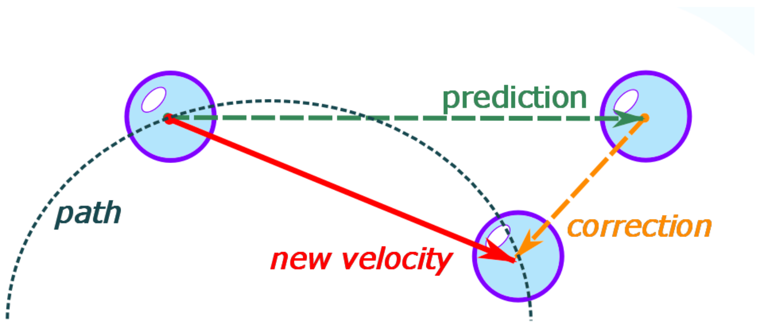
For computing the correction vector of one particle i, we need to first solve the Equation (3.6) to obtain the value of λ and finally, this yields the correction vector

$$\Delta\mathbf{x}_i = \lambda w_i \nabla_{x_i} C(\mathbf{x})^T \tag{3.7}$$

$$\lambda = -\frac{C(\mathbf{x})}{\sum_l w_l |\nabla_{x_l} C(\mathbf{x})|^2} \tag{3.8}$$

where l represents the number of particles that are affected by the constraint. Sometimes we consider a stiffness k for the constraint, which multiplies the Equation 3.7 such as $k\Delta\mathbf{x}_i$ where $k \in [0...1]$. Once we have calculated the correction $\Delta\mathbf{x}_i$ for the position $\mathbf{x}_i$, we can say that the next position is obtained by $\mathbf{x}_i^* = \mathbf{x}_i + \Delta\mathbf{x}_i$, to a particle i (see Figure 3.3).

Figure 3.3: Correction of the position **x**. We see the path that a particle must travel, first we predict its next position, then the correction that satisfies the restriction of the particle is computed, and finally the new velocity is calculated.



Source: the author.

Note: Keep in mind that Equations 3.7 and 3.8 are evaluated in the *projectConstraints* function presented in the line 10 of the Algorithm 1, and that we can define the relationship $\forall t_n \rightarrow$ *projectConstraints* to run $\#_{iter}$ times, with a $\Delta t = t_{n+1} - t_n$, where $n \in [0, ...]$.

3.3 XPBD

The Position-Based Dynamics (PBD) is of easy implementation, fast execution, plausible visualization, and allows to simulate different behaviors, as seen in the previous section. However, the main problem is that it is not based on physics, resulting in a lack of accuracy and no convergence to the best solution. Furthermore, the stiffness of the constraint depends directly on the time step and the iteration counting.

In the last few years, some new methods were proposed looking for improving accuracy and convergence. Projective Dynamics (PD) (BOUAZIZ et al., 2014) is a method that uses implicit time integration of physics-based systems, inspired by Finite Element Methods (FEM). The Extended Position-Based Dynamics (XPBD) (MACKLIN; MÜLLER; CHENTANEZ, 2016), is an extension of the Position-Based Dynamics (PBD) derived from an energy potential and improves the precision and efficiency of the simulation. Both methods are only an approximation of an implicit Euler integrator. In this work, we implemented the XPBD method.

Extended Position-Based Dynamics (XPBD) has a direct correspondence with elastic and dissipation energy potentials. It starts with the Newton's Equation of motion, subjects to forces derived from an energy potential $U(\mathbf{x})$

$$\mathbf{M}\ddot{\mathbf{x}} = -\nabla U^T(\mathbf{x}) \tag{3.9}$$

Macklin et al. derive the Equation 3.9 and demonstrate that it is possible, solving constraints in a time step and iteration count in an independent manner (MACKLIN; MÜLLER; CHENTANEZ, 2016). The new algorithm works by calculating a variable $\Delta\lambda$ for each constraint during an iteration of the solver, as an incremental change to the total Lagrange multiplier λ. Additionally, it introduces a compliance factor α, which is associated with the inverse stiffness for each constraint ($\alpha = k^{-1}$), thus modifying Equations 3.5 and 3.6 as follows:

$$\Delta\mathbf{x} = \Delta\lambda\mathbf{M}^{-1}\nabla C(\mathbf{x})^T \tag{3.10}$$

$$\Delta\lambda = -\frac{C(\mathbf{x}) - \tilde{\boldsymbol{\alpha}}\lambda}{\nabla C(\mathbf{x})\mathbf{M}^{-1}\nabla C(\mathbf{x})^T + \tilde{\boldsymbol{\alpha}}} \tag{3.11}$$

where $\boldsymbol{\alpha} = diag(\lambda_1^{-1}, \lambda_2^{-1}, ..., \lambda_m^{-1})$, and $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}/\Delta t^2$ represent the compliance matrix. From Equations 3.10 and 3.11 for the calculus of each iteration, the system must be solved in the following order:

$$\lambda^{n+1} = \lambda^n + \Delta\lambda \tag{3.12}$$

$$x^{n+1} = x^n + \Delta x \tag{3.13}$$

where λ is initialized with zero for a $n = 0$ of each frame. The solution of λ^{n+1} serves as input for Δx calculation.

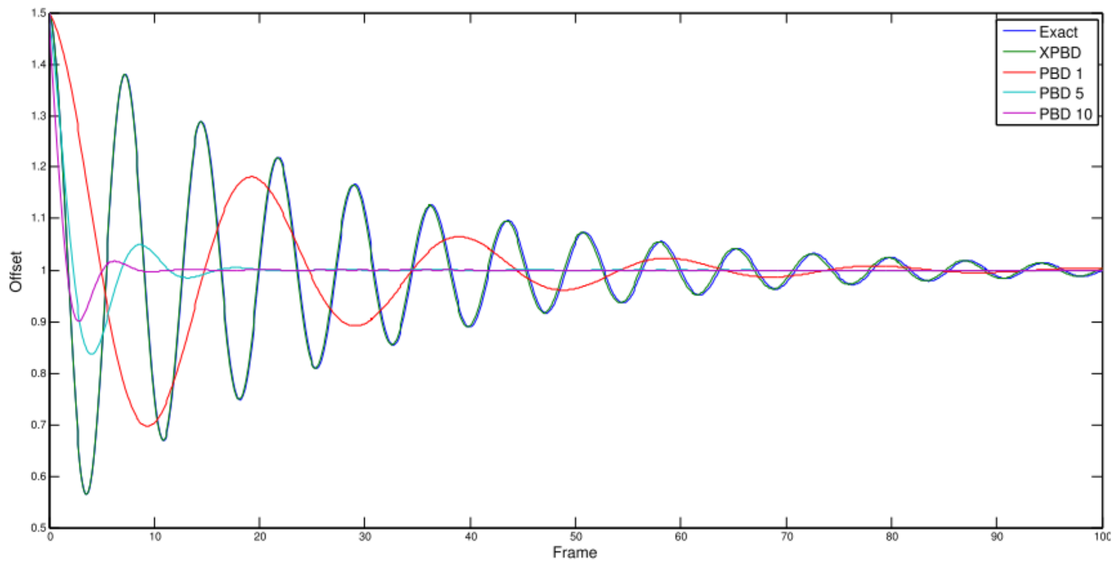
Algorithm 2 is an iterative method that converges to a solution reducing the error. So in line 16 of Algorithm 1 is replaced by Algorithm 2, this substitution improves the stability of the simulation. Line 2 computes the correction $\Delta\lambda$ to converge to Lagrange multiplier λ, line 3 computes the correction $\Delta\mathbf{x}$, to converge in real position, and finally in lines 4 and 5 update λ and $\mathbf{x}$ respectively for the next iterations.

Algorithm 2 Projection Constraints with XPBD

```
1: procedure PROJECTCONSTRAINTS($C_1, ..., C_{M+M_{coll}}, \mathbf{x}_1^*, ..., \mathbf{x}_N^*$)
2:     compute $\Delta\lambda$. using Eq. 3.11
3:     compute $\Delta\mathbf{x}$. using Eq. 3.10
4:     update $\lambda$. using Eq. 3.12
5:     update $\mathbf{x}$. using Eq. 3.13
6: end procedure
```

As we can see from Figure 3.4, Position-Based Dynamics (PBD) cannot reproduce the correct period of oscillation. The graphic shows that Extended Position-Based Dynamics (XPBD) closely reproduces the analytic result regardless the time step and iteration count.

Figure 3.4: Extended Position-Based Dynamics (XPBD) matches the analytic solution closely, while the Position-Based Dynamics (PBD) solutions are slowly dependent on the number of iterations.



Source: Macklin et al. (2016)

3.4 SPH

Smoothed particle hydrodynamics (SPH) is a Lagrangian method for obtaining approximate numerical solutions of the equations of fluid dynamics by replacing the fluid with a set of particles. The particles are interpolation points from which the properties of the fluid can be calculated. The interpolate can be constructed using analytical functions, and spatial derivatives of the interpolated quantities can then be found using ordinary calculus. There is no need to use a grid, and the description of free surfaces (MONAGHAN, 2005).

The range of applications include the disruption of free surfaces when a wave hits a rocky beach, multi-fluid problems that may involve the motion of rigid and elastic bodies, non-Newtonian fluids, heat transfer, virtual surgery, and chemical precipitation from fluids moving through fractured media (MONAGHAN, 2012). Moreover, this method has been established during the last years as one of the most important for fluid animations.

3.4.1 Representation

The fluid body is subdivided into moving particles with fluid properties. A fluid particle i is represented by a sample position $\mathbf{x}_i$ with their velocity $\mathbf{v}_i$, have a fixed mass m_i, volume V_i, density ρ_i, and pressure p_i. The shape, initialized as a cube, is variable and not defined, but typically visualized as a sphere.

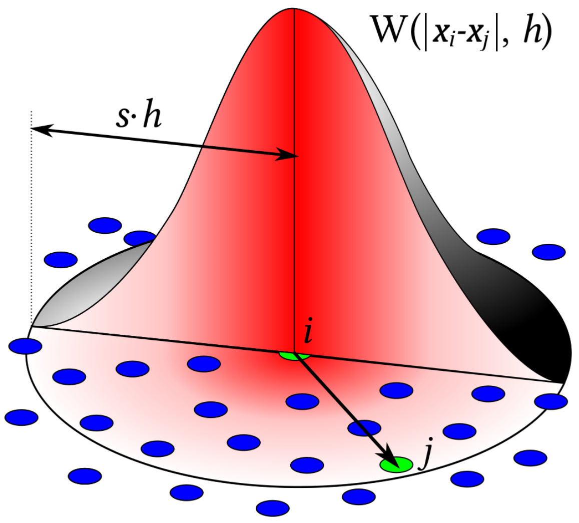
3.4.2 Navier-Stokes Equation

The particles move with the fluid flow, creating a velocity vector field that changes over time due to pressure gradient force, internal viscosity force, and external body forces. The dynamics of incompressible fluids is governed by the Lagrange form of the Navier-Stokes Equations, a set of differential partial equations that are supposed to hold throughout the fluid (see Equation 3.14).

$$\ddot{\mathbf{x}}_i = -\frac{1}{\rho_i}\nabla p_i + \upsilon\nabla^2\dot{\mathbf{x}}_i + \frac{1}{m_i}\mathbf{f}_i \tag{3.14}$$

where, $-\frac{1}{\rho_i}\nabla p_i$ is the pressure term and defines internal forces generated due to the pressure differences, $\upsilon\nabla^2\dot{\mathbf{x}}_i$ is the viscosity term that defines the internal friction forces due to the material friction, and $\frac{1}{m_i}\mathbf{f}_i$ represents the external forces.

Figure 3.5: Kernel function that interpolates the influence of a neighborhood j of a particle i with a radius h.



Source: By Jlcercos - Own work, CC BY-SA 4.0

3.4.3 Interpolation

A quantity A_i at an arbitrary position $\mathbf{x}_i$ is approximately computed with a set of known quantities A_j at neighboring particle positions $\mathbf{x}_j$

$$A_i = \sum_j V_j A_j W_{ij} = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \tag{3.15}$$

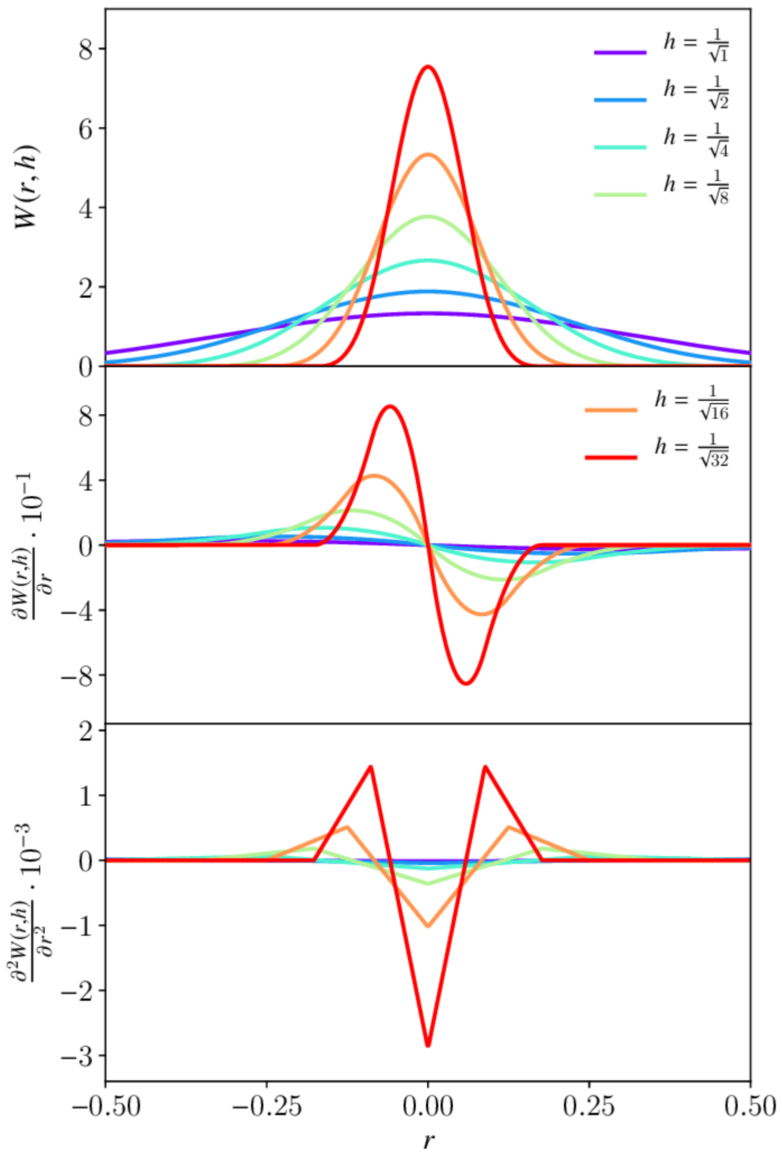
with W_{ij} representing a kernel function that weights the contributions of sample positions $\mathbf{x}_j$ according to their distance to $\mathbf{x}_i$ as

$$W_{ij} = W(\frac{||\mathbf{x}_i - \mathbf{x}_j||}{h}) = W(q) \tag{3.16}$$

where h is the called kernel's smoothing length. The smoothing length controls the amount of smoothing and, consequently, how strongly the value of $\mathbf{A}$ at position $\mathbf{x}$ is influenced by the values of the neighbor, as shown in Figure 3.5. Kernel functions should be close to a Gaussian, with support typically between h and $3h$. A kernel's function is the cubic spline kernel as

$$W(q) = \frac{3}{2\pi} \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \leq q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \tag{3.17}$$

Figure 3.6: Cubic spline kernel: Upper Figure shows different six functions for different h $(1/\sqrt{1}, 1/\sqrt{2}, 1/\sqrt{4}, 1/\sqrt{8}, 1/\sqrt{16}, 1/\sqrt{32},)$. The Middle and Bottom Figures represent the first and second derivatives respectively for each function of the nucleus.



Source: **??**

Spatial derivatives can be approximated from the Equation 3.15 as the differential operator $\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$ and Laplace operator $\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$. The advantage of using a kernel function is that it allows us to easily compute its first and second derivatives, as shown in Figure 3.6.

To compute the density, we use its explicit form based on the the interpolation Equation 3.15, so the density of particle i is

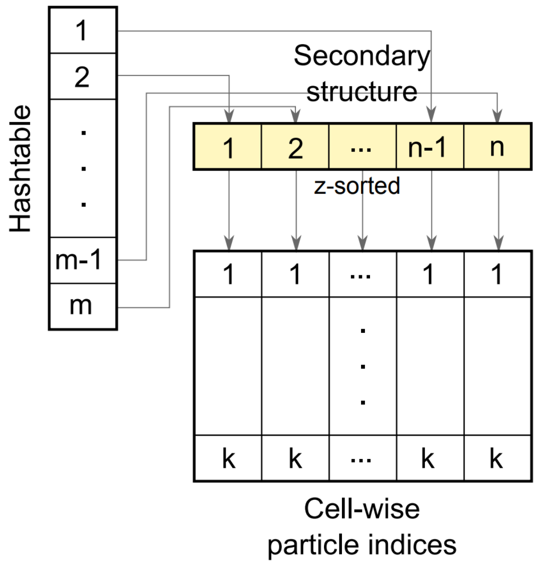
$$\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \sum_j m_j W_{ij} \tag{3.18}$$

3.4.4 Neighborhood Search

For the SPH method to work correctly, it is essential to have a technique that allows knowing all the neighbors of a particle i. If we try to calculate all the distances between the particles, the algorithm will be of order $O(n^2)$. To improve that cost, we present a data structure to optimize the complexity, with the following characteristics:

- to maintain a temporal coherence
- to control the changings of the set of particles
- to guarantee the fast generation with an update for each timestep
- to fast query for the neighbors.

Figure 3.7: A handle array of size m is allocated for the hash table, that points to a compact list storing the small number of n used cells (yellow). For each used cell, memory is reserved for k entries.



Source: Ihmsen, M. et al. 2011

Compact hashing: Ihmsen et al. present an efficient system for computing SPH fluid simulations. They propose a compact hashing and z-index sort as optimizations for the commonly used spatial hashing and index sort schemes (IHMSEN et al., 2011) (see Figure 3.7). Hash cells store a compact list of used cells, while k entries are pre-allocated for each element in the list of used cells. Elements in the used-cell list are generated if a particle is placed in a new cell, and deleted if a cell gets empty. We query the list of used cells for the neighborhood search in order O(1).

3.5 Types of Constraints

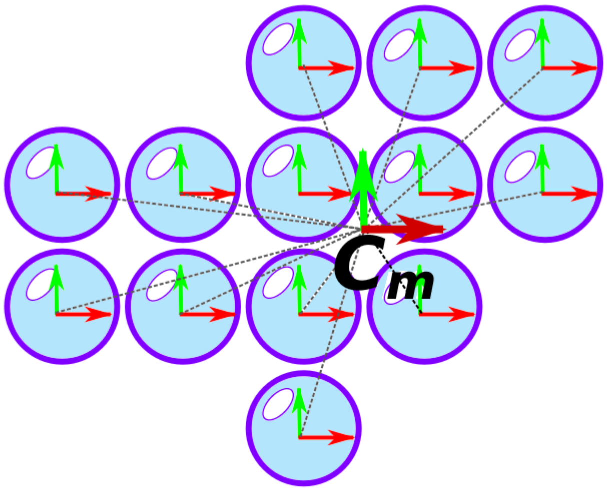
Everything is a set of particles connected by constraints. There are several constraints to describe a great variety of behaviors. PBD allows unifying and integrating different constraints, which allows solving with the same method (MACKLIN et al., 2014; BENDER; MÜLLER; MACKLIN, 2017). Section 3.2.1 presented the definition of constraint, while now we will show the types of constraints used in this work. We define $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ as a notation for better understanding.

3.5.1 Rigid

A rigid body is a body that conserves its shape during the simulation. We represent it as a set of particles that have a relation with its center of mass. However, collisions between particles or with surfaces can modify the positions of the particles during the simulation. We use the shape-matching method to restitute the original shape, as presented below (MÜLLER et al., 2005a).

Shape matching constraint: we can use the shape matching to formulate a constraint projection directly on the points. Given the set of initial positions $\mathbf{x}^0$ and the set of deformed positions $\mathbf{x}$ with its respective mass center $\mathbf{c}^0$ and $\mathbf{c}$, we want to find the rotation matrix and the corrections of the positions needed to maintain the original shape (see Figure 3.8).

Figure 3.8: Calculus of the solid center of mass $\mathbf{C}_m$ from a particle set. After compute the distance between the particles to its center of mass, the movement of each particle depends on the center of mass.



Source: the author.

According to Macklin et al. (MACKLIN et al., 2014), the position delta is given by the next equation

$$\Delta\mathbf{x}_i = \mathbf{Q}(\mathbf{x}_i^0 - \mathbf{c}^0) + \mathbf{c} - \mathbf{x}_i \tag{3.19}$$

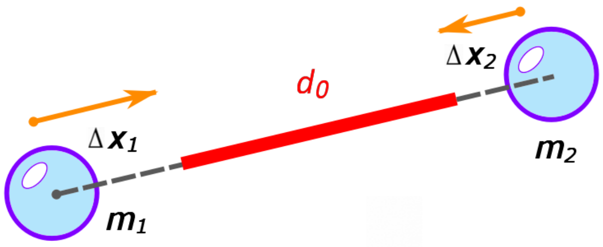
where $\mathbf{Q}$ is a rotation matrix, given by the polar-decomposition of the deformed matrix $\mathbf{A}$, calculated as

$$\mathbf{A} = \sum_i (\mathbf{x}_i - \mathbf{c}).(\mathbf{x}_i^0 - \mathbf{c}^0)^T \tag{3.20}$$

3.5.2 Soft

One way to achieve a deformable behavior is using multiple *shape-matching constraints*, which consists of dividing the object into overlapping clusters of particles, where each cluster has an associated constraint, as seen on Subsection 3.5.1. Another way to simulate deformable objects is to use the well-known distance constraint, as explained below.

Figure 3.9: Distance Constraint: the corrections Δx_i are weighted according to the inverse masses $w_i = 1/m_i$.



Source: the author.

Distance constraint: constraint mimics a linear spring force as in the classic mass-spring model. The objective is to maintain two particles $\mathbf{x}_1, \mathbf{x}_2$ at a distance d. See Figure 3.9.

$$C(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2| - d \tag{3.21}$$

Let us consider the distance constraint function in the Equation 3.21. The position delta is given by

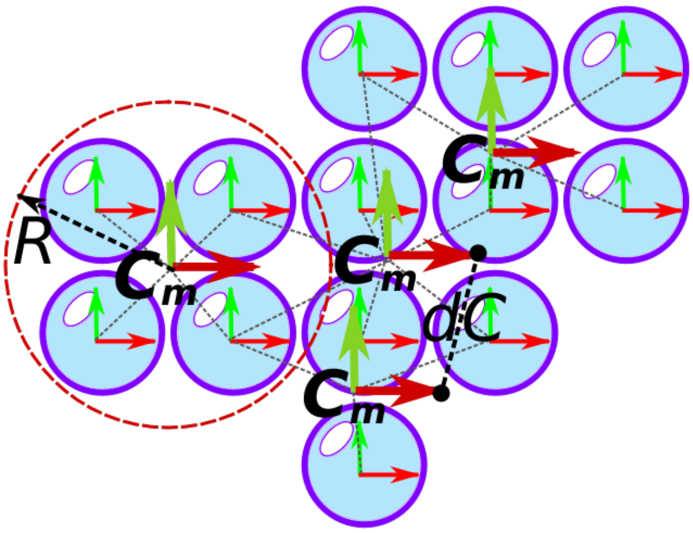
$$\begin{aligned} \Delta\mathbf{x}_1 &= -\frac{w_1}{w_1 + w_2}(|\mathbf{x}_{1,2}| - d)\mathbf{n} \\ \Delta\mathbf{x}_2 &= +\frac{w_2}{w_1 + w_2}(|\mathbf{x}_{1,2}| - d)\mathbf{n} \end{aligned} \tag{3.22}$$

Shape matching constraint: Another way to model soft is to use Shape matching constraint, with the main difference we will have several mass centers. Mass centers are calculated from cluster of particles, given a radius R and a minimum distance between clusters dC , this results on a structure of articulated bodies inside a total body, the Figure 3.10 shows how the coordinate system is created. A given particle can belong to more than one cluster. As in solid bodies, every particle belonging to a cluster has a relative position to the mass center of the cluster (See Subsection 3.5.1).

3.5.3 Fluid

A fluid behavior is simulated using the position-based fluids method proposed by Macklin et al. (MACKLIN; MÜLLER, 2013), where the fluid density is calculated by the neighborhood using SPH and PBD to the correction of the positions.

Figure 3.10: Soft bodies conformation: Several coordinate systems are created based on two properties: a radius R and a minimum distance between clusters dC. Each cluster has a central CM, and the particles inside have a relative position to the cluster. Soft bodies behaves as articulated solid bodies



Source: the author.

Density constraint: borrowing the concept of a density estimator from Smoothed Particle Hydrodynamics (SPH) (MONAGHAN, 1994), a density constraint is constructed for each particle i in the system as follows.

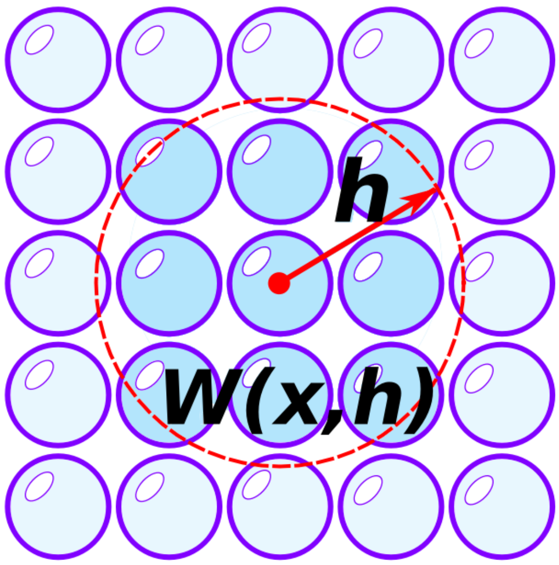
$$C_i(\mathbf{x}_1, ..., \mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1 \tag{3.23}$$

where ρ_0 is the fluid rest density and ρ_i is the density at particle, that is calculated in the Equation 3.18. The position delta for the fluid is

$$\Delta\mathbf{x}_i = \frac{1}{\rho_0}\sum_j(\lambda_i + \lambda_j)\nabla W(x, h) \tag{3.24}$$

where λ is the Lagrangian factor calculated in the Equation 3.8 and the function Kernel $W(x, h)$ is defined in the Equation 3.17. Figure 3.11 shows the kernel in action, tracing a radius of range on the neighboring particles, which will be used to compute the density of the current particle.

Figure 3.11: Calculus of the density from the neighborhood of particles for a particle i with position x given a radius h.



Source: the author.

3.6 Heat Transfer

Temperature is the core variable in this work since the phase change phenomena occur when a specific material reaches its temperature threshold (we will extend this assertion later in this section). In an homogeneous medium, the heat transfer is introduced by Hochstetter et al. (HOCHSTETTER; KOLB, 2017) in the Equation 3.25.

$$\frac{dT}{dt} = \frac{\nabla(k\nabla T)}{\rho c_p} \tag{3.25}$$

The literature offers several SPH-oriented models for heat transfer. Besides, a 1999 work proposed by Cleary and Monaghan (CLEARY; MONAGHAN, 1999) is until now one of the most popular models used to simulate temperature-based phenomena in SPH. They modeled heat transfer taking into account the specific heat capacity (C) and the thermal conductivity (K), allowing for different behaviors. Equation 3.26 shows the relation between all thermodynamic properties mentioned before.

$$\left(\frac{dT}{dt}\right)_{i,j} = \frac{V_i}{m_i C_i} \sum_j \frac{4K_i K_j}{K_i + K_j} V_j \left(T_i - T_j\right) \nabla W_{ij} \tag{3.26}$$

In our modified model, we handle the boundaries by introducing fixed temperatures when evaluating boundary particles to induce temperatures coming from these boundaries.

$$\left(\frac{dT}{dt}\right)_{i,b} = \frac{V_i}{m_i C_i} 2K_i \sum_b \frac{\rho_0}{\delta_b} \left(T_i - T_b\right) \nabla W_{ib} \tag{3.27}$$

3.7 Summary

This chapter is useful to familiarize the reader with the concepts and equations of the Smoothed Particle Hydrodynamics (SPH) and the Extended Position-Based Dynamics (XPBD). This is a base theory to understand the contribution of our work that will be seen in the following chapters.

4 THERMODYNAMIC MODELING

As discussed before, one objective of the physics-based simulation is to model or reproduce natural phenomena. In this chapter, we present a temperature-based method that drives phase-transitions phenomena based on the latent heat using extended position-based dynamics (XPBD) and smooth particle hydrodynamics (SPH). We simulate the phase-change phenomenon by varying parameters of the material such as the stiffness, distance, viscosity, density, and gravity to reach the different behaviors. This chapter uses some results and paragraphs from our paper entitled *Heat-based bidirectional phase shifting simulation using position-based dynamics* (SALAZAR et al., 2018).

4.1 Method Overview

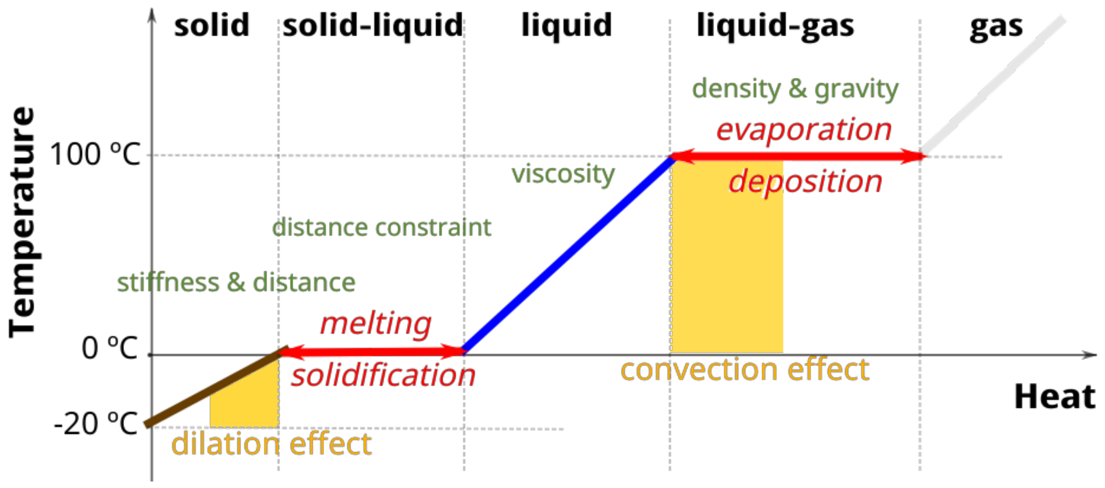
We introduce a novel combination of methods and techniques into an integrated solution to cover all state changes in the same framework. We then combine PBD and PBF particles with a SPH-based temperature model to track heat flow among them.

In our model, we consider solid, liquid, gas, and boundary particles. The first thing to do is to couple all these types of particles. Then we use a heat transfer model to vary the temperature and heat of the particles. Our model depends directly on the temperature and latent heat variation. In order to induce a change in temperature, we use boundary particles, that have the temperature as a property, but no movement.

The classic graph of the water phase-change serves to explain how our framework works during the simulation. In Figure 4.1, we can see the three states of matter and the transitions between them, represented by the red areas that are the critical points where the phase-change happens (melting, solidification, evaporation, and condensation). The yellow regions represent the additional effects that occur to achieve dilation and convection simulation, respectively.

Figure 4.1 also shows the variables that change in each phase, such as stiffness for solid, distance for solid – liquid, viscosity for liquid, and gravity and density for liquid – gas. For gas, all the parameters are fixed. Table 4.1 summarizes the variables and its dependencies. Our model is bidirectional so you can go from solid – liquid – gas to gas – liquid – solid.

Figure 4.1: General diagram showing all the phase transitions achieved in this work, passing from solid – liquid – gas.



Source: the author.

Latent heat is the driver of all phase changes. The execution of the phase-change is shown in the Algorithm 3. This algorithm is an adaptation of the Algorithms 1 and 2, seen in Section 3.2 and 3.3, respectively. We start initializing the variables in Lines 2-4. The values of these variables depend on the material properties, and interactions between the different materials are allowed. In Line 7, the next positions are calculated according to Euler integration. Then, in Line 10, the gravity values are assigned depending on the latent heat of vaporization. From Lines 14 to 24, all constraints are solved. Then, in Line 25 we call a procedure to manage distance constraints on solid particles. In Line 28, the heat transfer among the particles is calculated. Finally, in Lines 30, 32, and 34,

Table 4.1: The parameters list varying in our equations.

Symbols	Meaning	Solver	Depends on
E	stiffness	XPBD	temperature
d	distance	XPBD	latent heat
V	viscosity	XSPH	temperature
ρ	density	PBF	latent heat
g	gravity	Newton integration	laten heat

temperature, velocity, and neighborhood values are respectively updated.

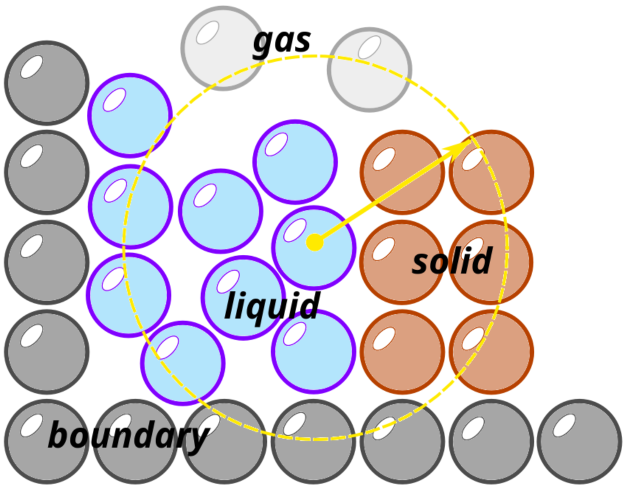
Algorithm 3 Main Loop

```
 1: for all Particles do
 2:     $x_i \leftarrow 0, v_i \leftarrow 0, w_i \leftarrow \frac{1}{m_i}$
 3:     $T_i \leftarrow T_0, L_m \leftarrow 0, L_e \leftarrow 0$
 4:     Set $T_m, T_e, L_{m,threshold}, L_{e,threshold}, c_i, k_i, \phi_i$
 5: end for
 6: loop
 7:     for all Particles do Calculate Next Positions
 8:     end for
 9:     for all Fluid Particles do
10:         Set Gravity Acceleration($L_e$) (Using Equation 4.9)
11:     end for
12:     for all Particles do Neighborhood Search
13:     end for
14:     loopIterations:
15:         for all Particles do
16:             Density Constraint($L_e$) (Using Equation 3.24 and 4.10)
17:         end for
18:         for all Fluid Particles do
19:             Viscosity Constraint (Using Equation 4.8)
20:         end for
21:         for all Solid Particles do
22:             Distance Constraint($T_i$) (Using Equation 3.22)
23:         end for
24:     end loop
25:     for all Particles do Solid Manager($L_m$) (Using Algorithm 5)
26:     end for
27:     for all Particles do
28:         Heat Transfer (Using Equation 4.2 and algorithm 4)
29:     end for
30:     for all Particles do Update Temperatures
31:     end for
32:     for all Particles do Update Velocities
33:     end for
34:     for all Particles do Update Neighborhood
35:     end for
36: end loop
```

4.1.1 Coupling between Solid, Fluid, and Gas

We notice that density is a property present in the three states of matter. Then, there is a way of coupling the three types of phases. We consider that all the solid, liquid, and gas particles will have the property of density, and that we can apply the density constraint using the Position-Based Fluid methods (PBF) (MACKLIN; MÜLLER, 2013). We extended the approach proposed by Akinci et al. (AKINCI et al., 2012) to couple between solid, liquid, and gas using the density property. The density of a particle i will be calculated from the neighboring particles (considering the three types) and will take different values for the three states (see Figure 4.2).

Figure 4.2: Representation of the influence of the density of a liquid particle, concerning all the others (boundary, liquid, solid, gas).



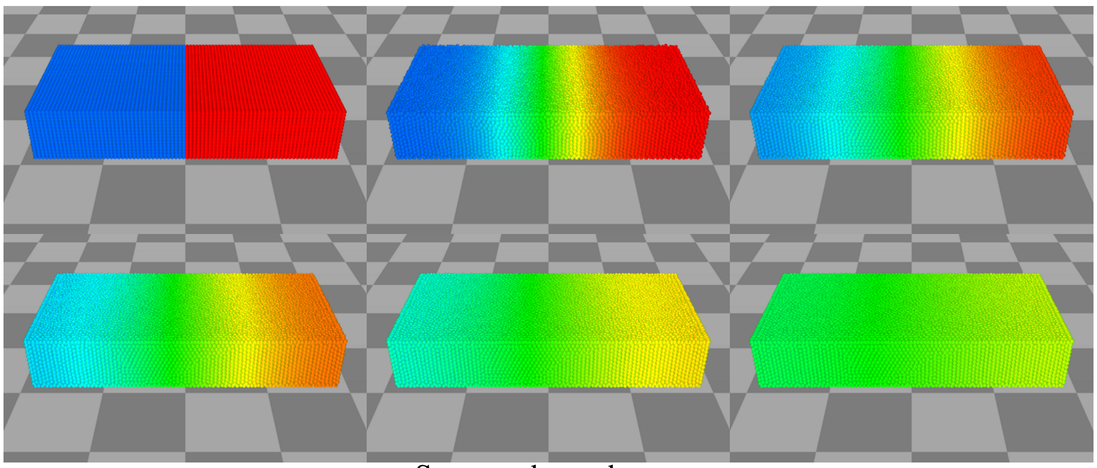
Source: the author.

To deal with boundary interactions, we calculate the density number $\delta_i = \sum_k W_{ik}$, as done by (AKINCI et al., 2012). Solid, liquid, gas, and boundary particle influence is taken into account while calculating the densities of the particle. It is important to highlight that we correct the solid particle position by applying the density constraint to them. In such a way, all phases interact in the same framework. Equation 4.1 shows how density is calculated for a particle i.

$$\rho_i = \sum_s m_s W_{is} + \sum_l m_l W_{il} + \sum_g m_g W_{ig} + \sum_b \frac{\rho_0}{\delta_b} W_{ib} \tag{4.1}$$

where s particles are solid, l particles are fluid, g particles are gas, and b particles are boundary particles that represent the walls. So, i can be a solid, liquid, or gas particle. Using this approach, we can obtain solid – fluid – gas interaction in a straightforward manner.

Figure 4.3: Heat transfer: The upper left image shows two liquids with different initial temperatures: $10°C$ (blue fluid), and $90°C$ (red fluid). The following images show how heat spreads until it reaches a state where the temperature is the same (lower right image).



Source: the author.

4.1.2 Solving the Heat Equation

At the beginning of the previous section, we stated that a phase change occurs when a specific material reach its critical temperature. The temperature is a variable and its changing does not produce a phase change; it only influences some parameters of the material such as, for example, the stiffness and the viscosity. As shown in Figure 4.3, two fluids at different temperatures only exchange heat, do not achieving the phase-change. The shift in temperature only serves to reach the critical temperature limit of a material; here is where the phase change happens. In this case, the temperature remains constant, and an energy in the form of latent heat begins to accumulate until exceeding the threshold to move to the next phase.

Based on Equations 3.26 and 3.27, the temperature for one particle i can be calculated as follows:

$$\left(\frac{dT}{dt}\right)_i = \left(\frac{dT}{dt}\right)_{i,j} + \left(\frac{dT}{dt}\right)_{i,b} \tag{4.2}$$

The changing rate in latent heat is calculated from the change in temperature and specific heat as:

$$dL = c_i \Delta T dt \tag{4.3}$$

where ΔT is difference of the temperature and obtained from the heat model.

Algorithm 4 Latent Heat Update

```
1: switch $State\ Particle\ i$ do
2:     case $Phase\ change_{m-f}$
3:         $L_{m-f} \leftarrow L_{m-f} + \Delta L$ (Using Equation 4.3)
4:         if $L_{m-f} < 0$ then
5:             $L_{m-f} \leftarrow 0$
6:         else if $L_{m-f} >= L_{m-f,threshold}$ then
7:             $L_{m-f} \leftarrow L_{m-f,threshold}$
8:         end if
9:     case $Phase\ change_{e-c}$
10:        $L_{e-c} \leftarrow L_{e-c} + \Delta L$ (Using Equation 4.3)
11:        if $L_{e-c} < 0$ then
12:            $L_{e-c} \leftarrow 0$
13:        else if $L_{e-c} >= L_{e-c,threshold}$ then
14:            $L_{e-c} \leftarrow L_{e-c,threshold}$
15:        end if
16:    case $Solid$
17:        if $T_i^{n+1} \geq T_{m-f}$ then
18:            $State_i \leftarrow Phase\ change_{m-f}$
19:        end if
20:    case $Fluid$
21:        if $T_i^{n+1} \leq T_{m-f}$ then
22:            $State_i \leftarrow Phase\ change_{m-f}$
23:        else if $T_i^{n+1} \geq T_{e-c}$ then
24:            $State_i \leftarrow Phase\ change_{e-c}$
25:        end if
26:    case $Gas$
27:        if $T_i^{n+1} \leq T_{e-c}$ then
28:            $State_i \leftarrow Phase\ change_{e-c}$
29:        end if
```

Next, we increment the latent heat using Algorithm 4. Let us define temperature as sensible heat, or the heat we can measure with a thermometer. Latent heat, in turn, is the energy used to perform the phase change. We illustrate this in Figure 4.1. In the initial (low temperature) state, a small amount of energy is stored. As the temperature increases (sensible heat), the accumulated heat also increases. When the body meets a critical temperature (melting and boiling points), the accumulated heat (latent) still increases, but the temperature remains constant. This means that the sensible heat stopped increasing.

After gaining enough energy, i.e., after accumulating enough latent heat, the material reaches its latent heat of melting/vaporization threshold and transitions to the next state of matter. Then, the sensible heat starts increasing again. This cycle is repeated in any of the phase changes and works in both ways. Notice that the latent heat amount required to melt some material is rarely the same that is needed to boil.

To handle states and transitions, we store the current and last states depending on the particle temperature, and the critical temperature values (sub-index $m - s$ and $e - c$ mean $melting - solidification$ and $evaporation - condensation$, respectively):

$$State = \begin{cases} Solid & T_i < T_m, \\ Phasechange_{m-s} & T_i = T_m \\ Fluid & T_m < T_i < T_e \\ Phasechange_{e-c} & T_i = T_e \\ Gas & T_i > T_e \end{cases} \tag{4.4}$$

Here we covered all the possible states of our particles and, if a given particle is shifting its state, the temperature must be constant. The model saves the latent heat of each phase change independently: L_{m-f} for the latent heat of melting and L_{e-c} for the latent heat of evaporation.

Our heat scheme works in the following way: when the phase change is occurring, which is to say $State = Phase\ change_{m-f}$ or $State = Phase\ change_{e-c}$, latent heat starts increasing or decreasing based on Equation 4.3, and then we enforce latent heat values to be within boundaries $[0, L_{threshold}]$. Other cases check if future temperatures are out of the current state and assign a phase change identifier to the particle.

4.2 Shifting Phase

Based on Figure 4.1, we analyze each of the phases that the materials undergo during the phase-change phenomena.

4.2.1 Solid Phase

In this phase, using a distance constraint (see Equation 3.21), we consider a solid as a deformable object with very high stiffness.

Stiffness: the Young's modulus ($E = \sigma/\epsilon$) defines the linear relationship between applied stress (σ) and associated deformation (ϵ) (MCNAUGHT; MCNAUGHT, 1997). The elastic properties of solids can be introduced easily by extending the standard distance constraint (see Subsection 3.5.2). The more the temperature increases, the more the solid body dilates. Consequently, E must decrease as it represents the solid's rigidity. We propose Equation 4.5, where using a sigmoid function we associate the stiffness parameter with the temperature change:

$$E = \frac{2E_0}{1 + e^{\Delta T}} \tag{4.5}$$

where E_0 is the Young's modulus of the material, a given property mostly obtained empirically. Also, we establish the initial distance between particles depending on the linear coefficient of expansion (e) and ΔT. This setup admits both dilation and contraction, depending on the direction of the heat change:

$$\Delta d_0 = e \Delta T d_0 \tag{4.6}$$

Hence, in the Equation 4.7, we obtain the final position correction for each couple of particles by adding Equations (4.5) and (4.6) to the original corrections (See Equation 3.22), with the XPBD approach.

$$\begin{aligned} \Delta \mathbf{x}_1 &= -E \frac{w_1}{w_1 + w_2 + \tilde{\alpha}} \left(|\mathbf{x}_{1,2}| - d_0 \left(1 + e\Delta T\right) - \lambda\tilde{\alpha} \right) \\ \Delta \mathbf{x}_2 &= +E \frac{w_2}{w_1 + w_2 + \tilde{\alpha}} \left[|\mathbf{x}_{1,2}| - d_0 \left(1 + e\Delta T\right) - \lambda\tilde{\alpha} \right] \end{aligned} \tag{4.7}$$

where d_0 is the initial distance.

4.2.2 Solid – Liquid Phase

In this transition phase we find solid and liquid particles, created and destroyed connections between particles. Depending on the gain or loss of energy, we see the melting and solidification phenomena, respectively:

- **melting**: solid particles have distance constraints to conserve the solid's shape. Each particle represents a node in the solid, and each constraint involves two particles. As a particle can admit more than one constraint, if its latent heat reaches the critic heat, all constraints containing this particle are deleted.

- **solidification**: in contrast with melting, the solidification (or freezing) does the opposite process. Here, we create new constraints based on a support radius rs_{max} (not necessarily the same used with the SPH kernels). When two neighboring fluid particles are decreasing their latent heat, they are candidates for solidification. If both meet the critical solidification value, they generate a new distance constraint. This process stops as soon as the particle is out of the $Phase\ change_{m-f}$ status. Notice that, for a particle to solidify, its last state must be ***fluid***. Additionally, if a particle reaches its maximum number of constraints nC_{max}, a distance constraint between particles will not be created.

Solid Manager: Algorithm 5 is introduced to manage the life cycle of distance constraints.

Algorithm 5 Solid Manager

```
1: for all Distance Constraint Particles i,j do
2:     if $State_i = Phase\ change_{m-f}$ & $State_j = Phase\ change_{m-f}$ then
3:         Delete Distance Constraint i,j
4:     end if
5: end for
6: for all Particles i do
7:     if $State_i = Phase\ change_{m-f}$ then
8:         for all Neighboring Particle j do
9:             $Cond\ 1 \leftarrow State_j < Phase\ change_{m-f}$
10:            $Cond\ 2 \leftarrow \mid x_i - x_j \mid < r_{max}$
11:            $Cond\ 3 \leftarrow nC_i < nC_{max}$
12:            if $Cond\ 1$ & $Cond\ 2$ & $Cond\ 3$ then
13:                new Distance Constraint i,j
14:            end if
15:        end for
16:    end if
17: end for
```

4.2.3 Liquid Phase

In this phase, our particles have fluid behavior. Here we use the density constraint seen in Subsection 3.5.3. The density of the fluid remains constant, while the viscosity varies. Smaller is the temperature, more viscous the fluid will be, while higher temperatures will decrease the fluid viscosity.

Viscosity: to damp out non-physical oscillations, we use the XSPH viscosity approach of Schechter and Bridson due to its low cost and easiness to manipulate in contrast with the classic XSPH (SCHECHTER; BRIDSON, 2012). In this manner, we damp out the noise when updating velocities:

$$\mathbf{v}_i^{next} = \mathbf{v}_i + \mu \sum_j (\mathbf{v}_i - \mathbf{v}_j) W(\mathbf{r} - \mathbf{r}_j, h) \tag{4.8}$$

where μ is tunable. We use this parameter in further sections to modify viscosity.

4.2.4 Liquid – Gas Phase

Phase-changes between liquid and gas generates the phenomena of evaporation and condensation. To model these phenomena, we modify the parameters of density and gravity.

Density: to obtain vaporization, we modify the initial rest density ρ_0 in the density constraint (see Section 3.5.3) multiplying it by a scaling function β. Doing this, convection behavior similar to water boiling arises when latent heat is increasing. Equation 4.9 represents an inverse sigmoid function:

$$\beta(\gamma) = 1 - \frac{1}{1 + \left(1 + \frac{12}{L_{e,threshold}}\right)^{\gamma}} \quad \star\,. \tag{4.9}$$

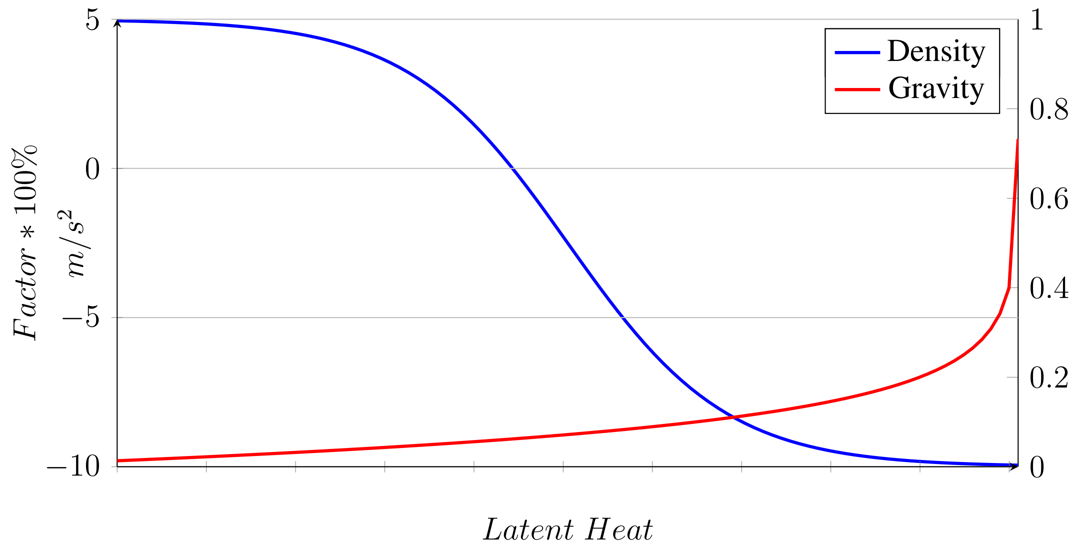
with $\gamma = \frac{L_{e,threshold}}{2} - L_e$. The behavior of Equation 4.9 is shown in Figure 4.4.

Gravity: vaporization simulation is performed by setting the particle gravity as a function of the latent heat. The function g (shown in Equation 4.10) keeps the particle gravity unchanged in smaller values of latent heat and, as it increases, gravity turns to positive, producing the vaporization effect (see Figure 4.4).

$$g(\omega) = \frac{10.8}{\ln\left(\frac{L_{e,threshold}+0.02}{0.02}\right)} \ln(\omega) - 9.8 \quad \star\,. \tag{4.10}$$

with $\omega = \frac{L_{e,threshold}+0.02}{L_{e,threshold}+0.02-L_e}$. The behavior of Equation 4.10 is shown in Figure 4.4.

Figure 4.4: These functions allow the simulation of the ebullition and evaporation effects. Blue: (right scale) the density factor decreases as latent heat increases, providing a natural convection effect. Red: (left scale) gravity remains low in almost 90% of the phase change, letting the convection effect occur. Gravity arises when the density is low, generating the evaporation effect.



Source: the author.

4.2.5 Gas Phase

The gas is a compressible fluid and is modeled using the equations presented on Subsection 3.5.3. Adding a non-physical characteristic, which is to make the particles affected by a positive constant gravity ($1m/s^2$), result in an effect where the particles of gas are floating in the air. The density and gravity parameters remain constant.

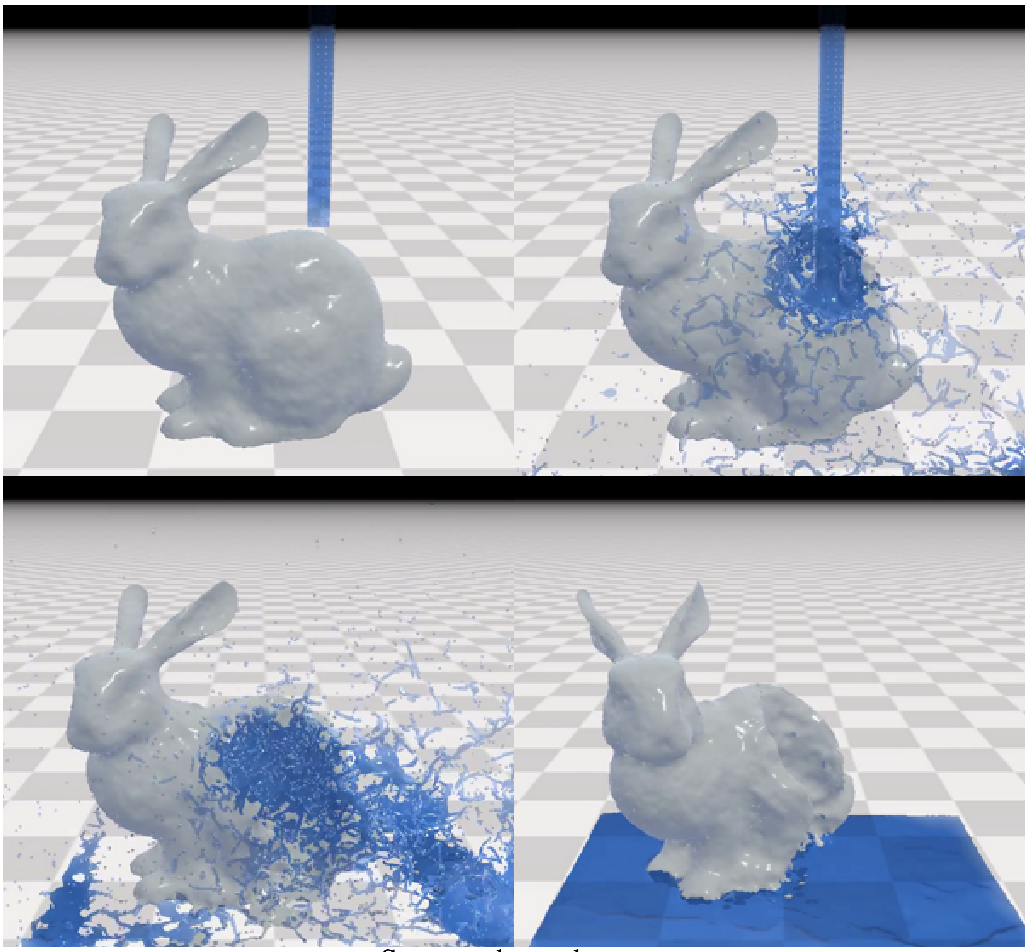
4.3 Results

Results achieved show the melting, solidification, evaporation, and condensation transitions between phases of the matters. In addition, dilation and convection effects are also visualized during the simulations. We implemented our model in C++ using the Eigen library. We used a 3.40 GHz i5-4670 CPU with 16GB RAM to generate all results here presented. The whole implementation is sequential on the CPU for the sake of simplicity and reproducibility. Besides, we used post-processed rendering to better isolate the software elements for analysis. The Nvidia Flex environment was used for graphics rendering. To assess the capabilities of our method, we developed scenarios that cover the effects of all phase transitions.

4.3.1 Simulations

Solid and Fluid Interactions: when a cold solid is thrown into a warm liquid, or vice-versa, the heat-transfer phenomena proceeds to eventually modify the phase of some of the particles in both the liquid and the solid, e.g. the interaction between ice and warm water. A higher amount of energy is present in the hot liquid particles. Then, energy is transported from the liquid, let us say, water, to the cold solid, melting it and decreasing the sensible heat present in the remaining water.

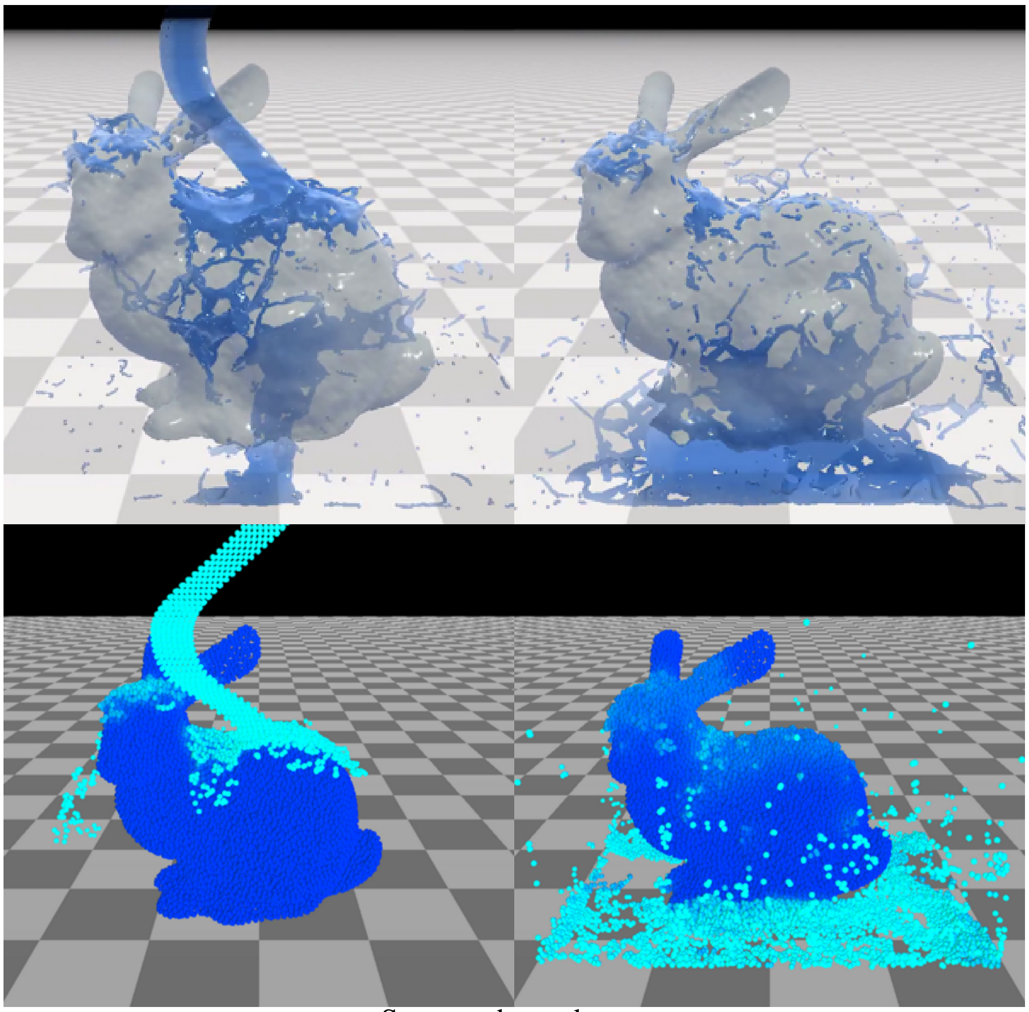
Figure 4.5: Melting simulation: Hot water drops over an ice-bunny that melts when hot particles contact with cold particles. Interestingly, some liquid particles cool enough to freeze when in contact with the very cold ice particles.



Source: the author.

Figure 4.5 depicts a scenario where some amount of hot water falls over an icy bunny, melting partially in the regions affected with enough energy to break the local solid constraints. When the falling particles do not have enough energy to break the constraints, however, a different phenomenon takes place. In the scenario of Figure 4.6, instead of melting the bunny, some liquid particles eventually solidify when they lose enough heat.

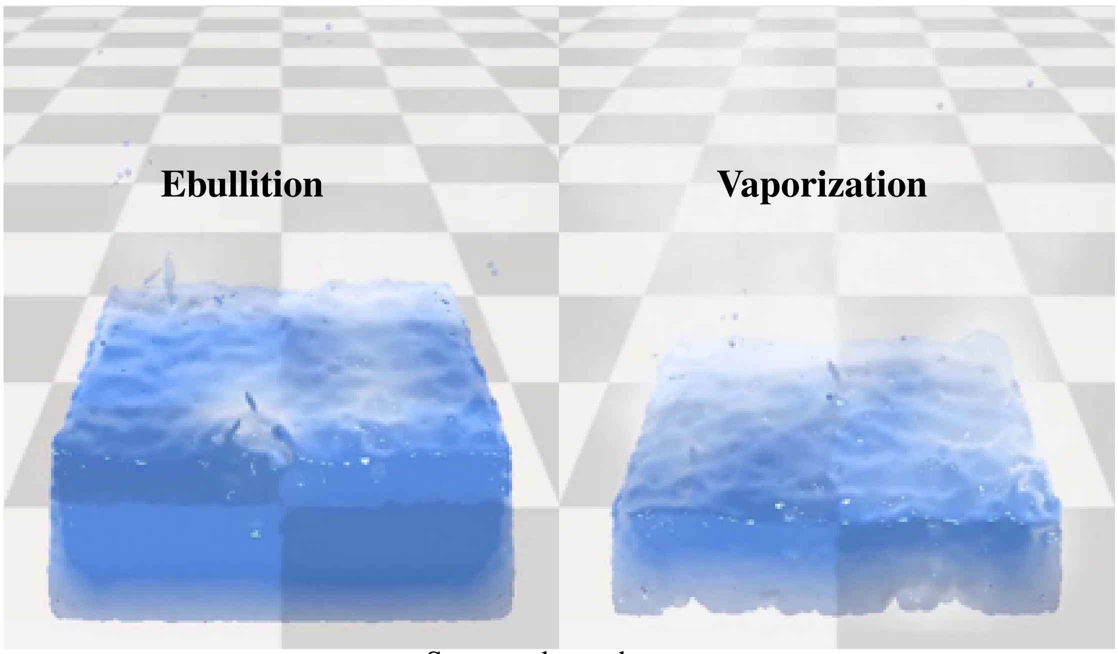
Figure 4.6: Solidification Simulation: Warm water falls over the icy bunny but the fluid particles have not enough heat to melt it. Then some fluid particles lose enough of their heat to freeze.



Source: the author.

Vaporization & Condensation: in a similar way to the convection scenario, we use the σ function to simulate vaporization. The buoyancy of gas particles depends on their latent heat of evaporation. The more latent heat of evaporation a particle has, the more buoyancy it experiences. Other works such as the ones proposed by Muller et al. (2005) and Macklin and Muller (2014) (MÜLLER et al., 2005b) suggest the introduction of air particles in the free domain to generate buoyancy and let the gas particles raise. Such approach could help to better model the fluid-air interaction in terms of heat transfer. However, its computational efficiency is questionable under some circumstances. Figure 4.7 shows our evaporation process in action: fluid particles start a convective-like effect and when enough energy is available, buoyancy effects begin raising these particles based on Equation 4.10.

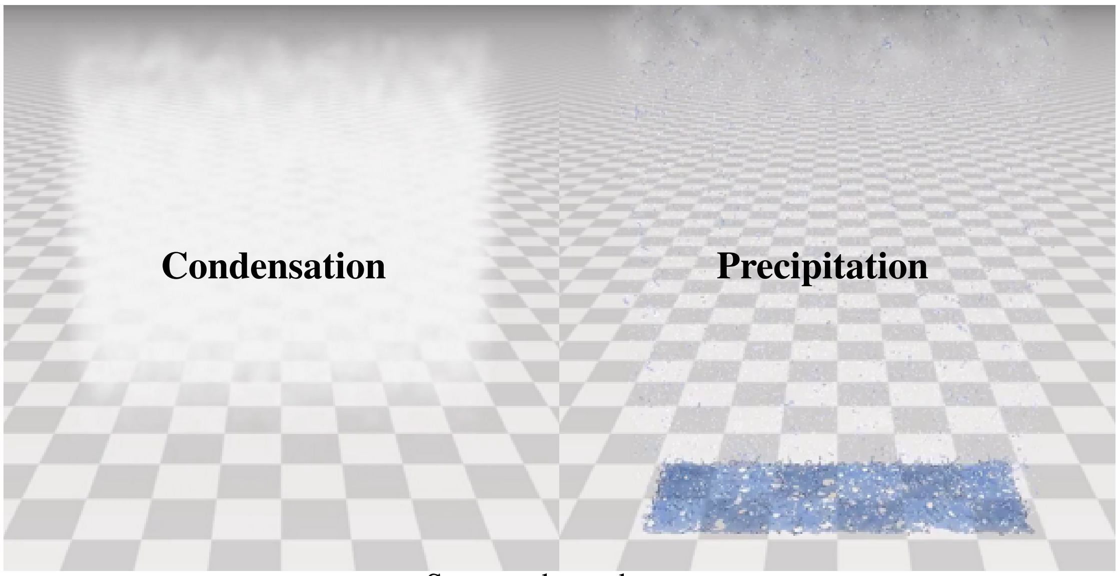
Figure 4.7: Vaporization Simulation: with 50k particles, demonstrates the effect of vaporization combined with convection that is typical of boiling water.



Source: the author.

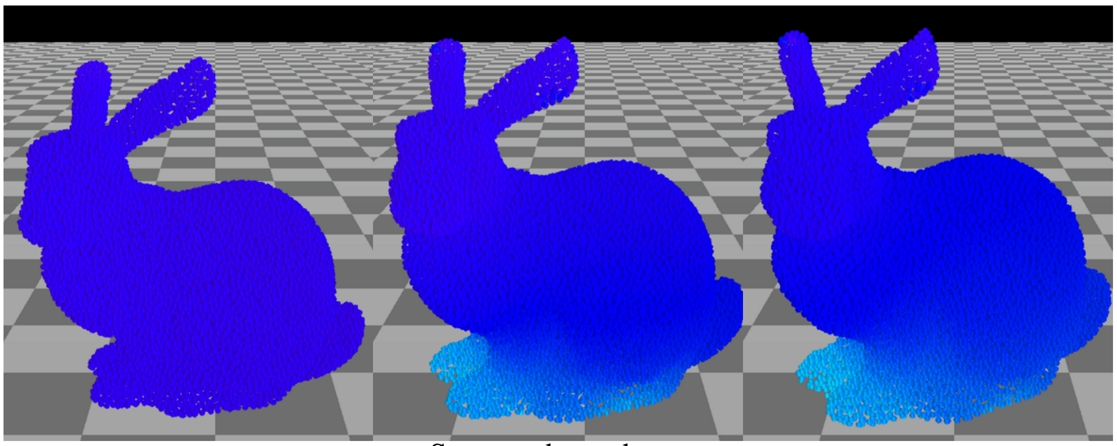
Figure 4.8, show the condensation effect. The simulation starts with vapor particles in the upper side of the container. As they meet a very cold surface at the top of the box, vapor particles gradually transition to liquid and fall back.

Figure 4.8: Condensation Simulation: With 26k particles, condensation illustrates the cloud formation and precipitation (rain).



Source: the author.

Figure 4.9: Dilation effect: from the left to the right, images show how the particles are distancing each other, causing the 3D bunny model to change its shape and size while gaining heat from the ground. This effect occurs before the solid reaches its transition state (solid–liquid).

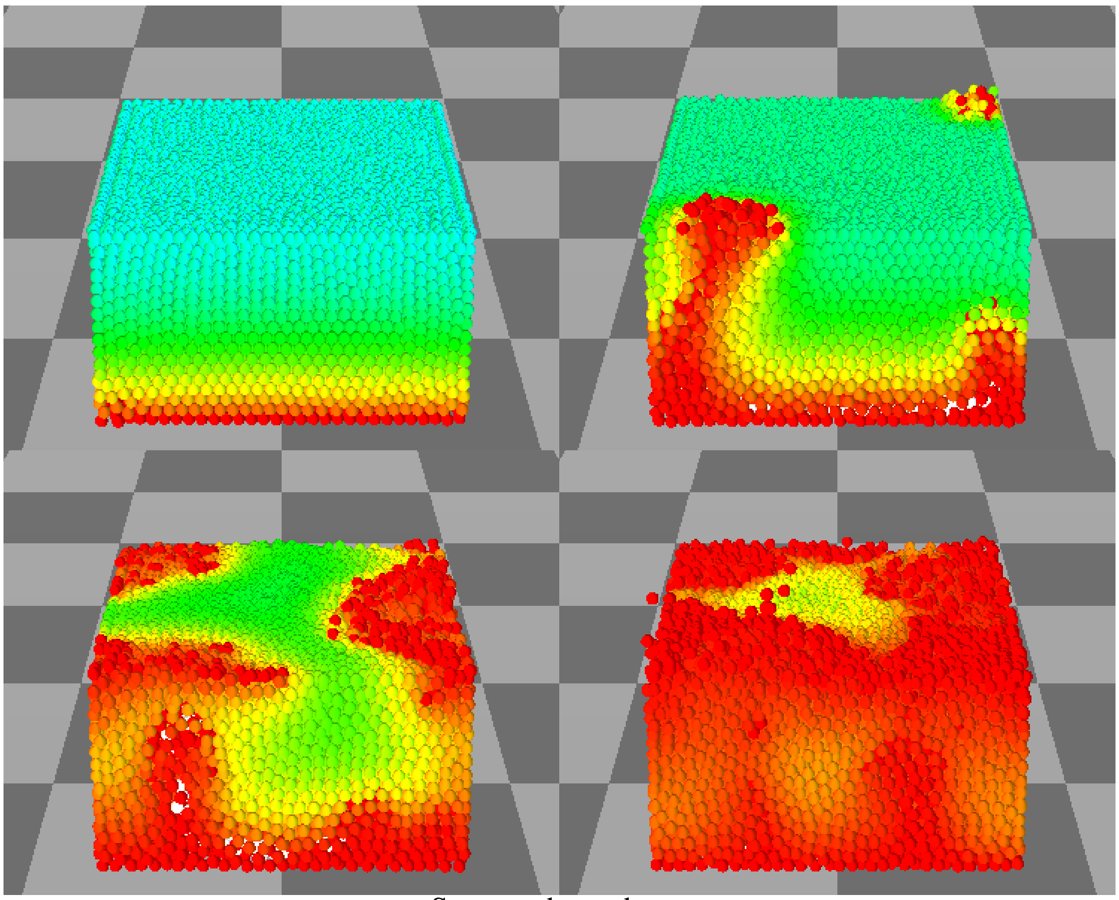


Source: the author.

Dilation Effect: modifying the distance and stiffness of the distance constraints, we reproduce a dilation effect on solids, as shown in Figure 4.9. Equation 4.6 shows the linear relationship between distance and temperature. The higher the temperature, the particles separate more and more. This distance variation can cause instability in the simulation, so we also modify the stiffness as a function of temperature (see Equation 4.5) – the higher the temperature is, smaller will be the stiffness. These adaptations allow the solid object to alter its initial shape, thus achieving the dilation effect.

Convection Effect: the use of sigma and logarithmic functions allows us to simulate convective effects on fluids. In a boiling water scenario, the change in density due to the increase of latent heat causes particles with higher energy to move up and particles with less energy to move down. In Figure 4.7 a boiling effect is shown in the top frame. When lower particles absorb enough energy, they rise to the surface and eventually vaporize. Figure 4.10 shows explicitly how particles with lower density and higher temperature arises from the bottom to the surface of the container.

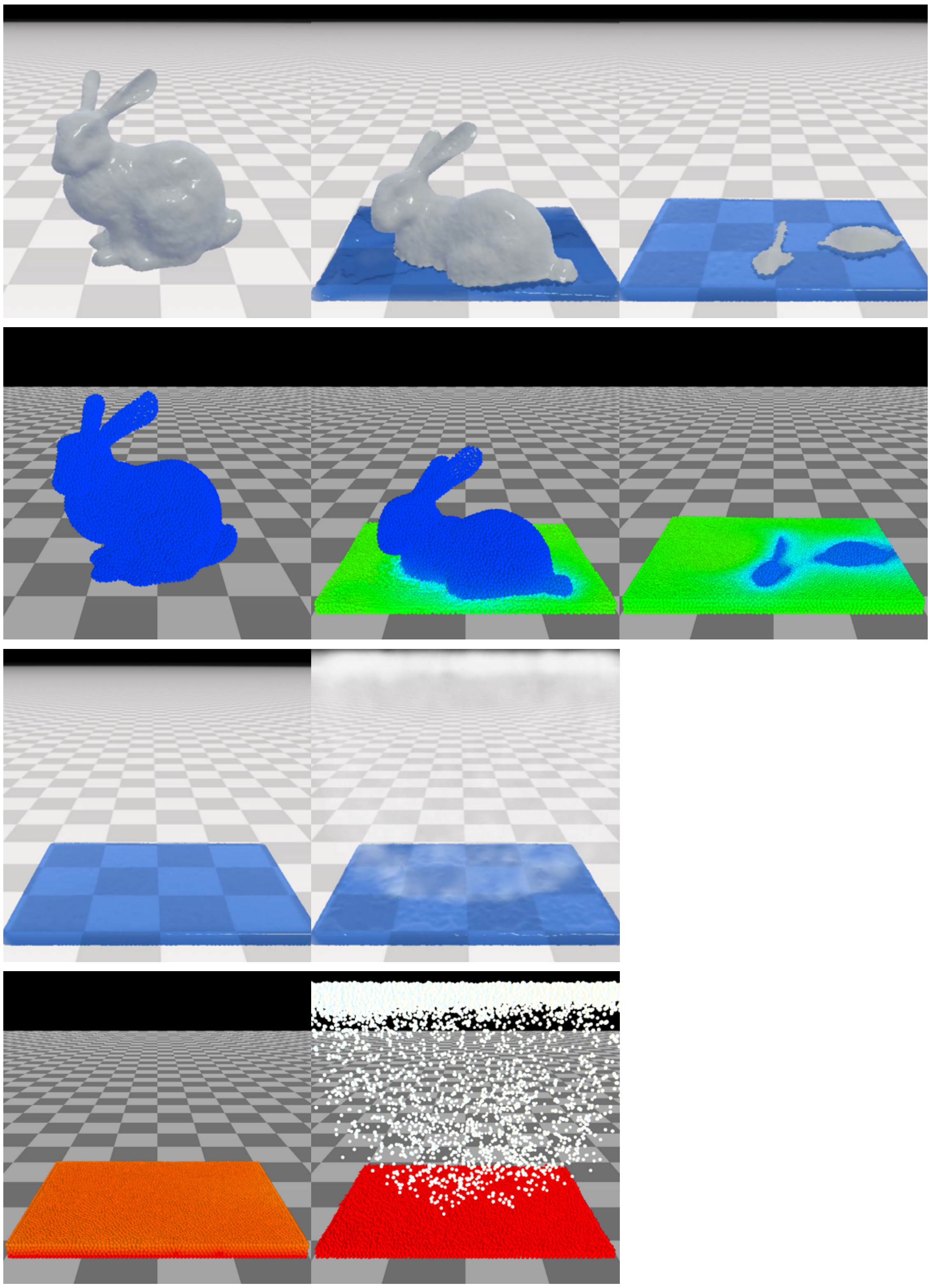
Figure 4.10: Convection effect: this scenario shows some frames where we initially see a liquid with temperature $10°C$ (upper left), that is heated from the bottom. The following frames show how the liquid with more heat rises to the surface and the liquid with less heat drops. These movements are characteristics of convection. This effect occurs before the liquid reaches the liquid–gas transition phase.



Source: the author.

Full Simulation: Figure 4.11 illustrates the melting and vaporization processes with 27k particles. We set a fixed high temperature on the floor where the solid-cold bunny is placed sitting. When a particle meets the latent heat threshold, our constraint manager breaks all the connections among this particle and its neighbors. When the particles change their states to liquid, the viscosity constraint is activated. Later, the transition to gas completes the process. This scenario shows one way of the whole process, involving three states: solid, liquid, and gas, as well as the two phase changes.

Figure 4.11: Full Simulation: melting and evaporation process involving three states with 27K particles. In this scenario, the temperature increases from the bottom, and the bunny starts to melt. After that, all particles are liquid and continue to heat. Finally, liquid particles switch to gas when the latent heat of evaporation is reached.



Source: the author.

Table 4.2: Average times spent per frame for the calculus of neighbor search, constraint projection, viscosity, heat transfer, as well as the solid manager (in milliseconds) for the different scenarios. For the final animation (not running in real time), we considered 64fps.

Simulation Scenario	Number particles	Total (ms)	Neighbor search	Constraint projection	Viscosity	**Heat transfer**	**Solid manager**
Heat transfer (Fig. 4.3)	18k	59.43	6.38	45.047	3.82	3.68	3.66×10^{-2}
Vaporization (Fig. 4.7)	18k	60.65	8.64	44.50	3.31	3.64	3.79×10^{-2}
Condensation (Fig. 4.8)	20.4k	64.35	15.46	41.33	3.51	3.46	3.96×10^{-2}
Melting (Fig. 4.5)	34.6k	409.66	20.37	353.01	6.10	6.32	22.89
Solidification (Fig. 4.6)	33.5k	218.46	16.66	179.32	4.67	5.23	11.66
Full Simulation (Fig. 4.11)	27.1k	97.39	14.35	71.71	4.35	5.13	1.06

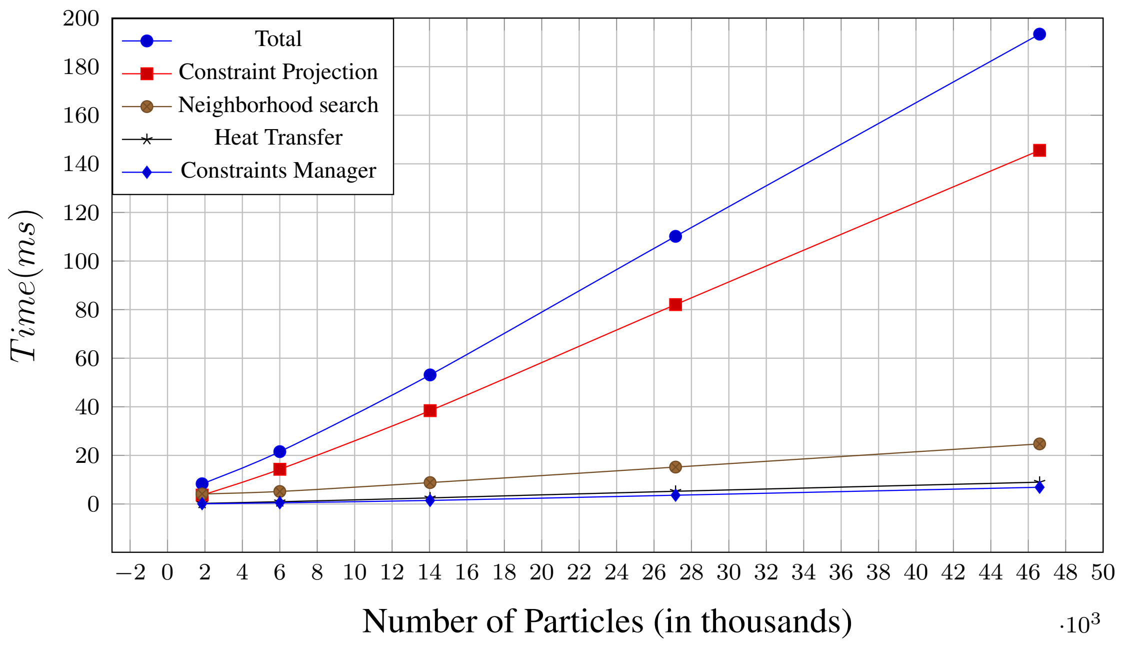
4.3.2 Discussion

Performance Analysis: Table 4.2 shows the performance of our approach in seven different scenarios, summarizing the computational impact for each functionality. *Total* shows the total average time per simulation step. For this performance test, we computed 8 PBD internal iterations per step. The table shows that the constraint projection takes longer than all other functions. As a reminder, constraint projection is a fundamental step in PBD. An interesting point here is the relatively low impact of our methods, the heat-transfer model, and the constraint manager, on the overall simulation. Heat transfer computation takes less than 8.8% of the total time in the worst-case scenario (**Heat transfer** row). Excluding that scenario, the mean value is even lower: 5.64%. Besides, our *constraint manager* takes even less time than heat transfer. The execution time was under 1% of the total cost of a timestep. These results indicate that the additional computational cost imposed by our methods is barely noticeable concerning the original framework without phase changes at all. Our method was implemented based on Jan Bender's sequential code (BENDER; MÜLLER; MACKLIN, 2017), available on his official website[1].

Particle number analysis: We analyzed how the computation time grows with the number of particles. Fig. 4.12 compares the average computation time per step of simulation for each functionality. For this test, we took the melting demo and varied the number of particles in the bunny model. The chart displays a linear behavior for times concerning particles number. Visibly, constraint projection takes most of the total time of each

[1]http://www.interactive-graphics.de/

Figure 4.12: The computational cost of each function according to the number of particles in the model. Notice that the cost for the Heat Transfer and the Constraint Manager are fairly lower than the Constraint Projection (PBD solver) and also grow much slower when the number of particles increases.

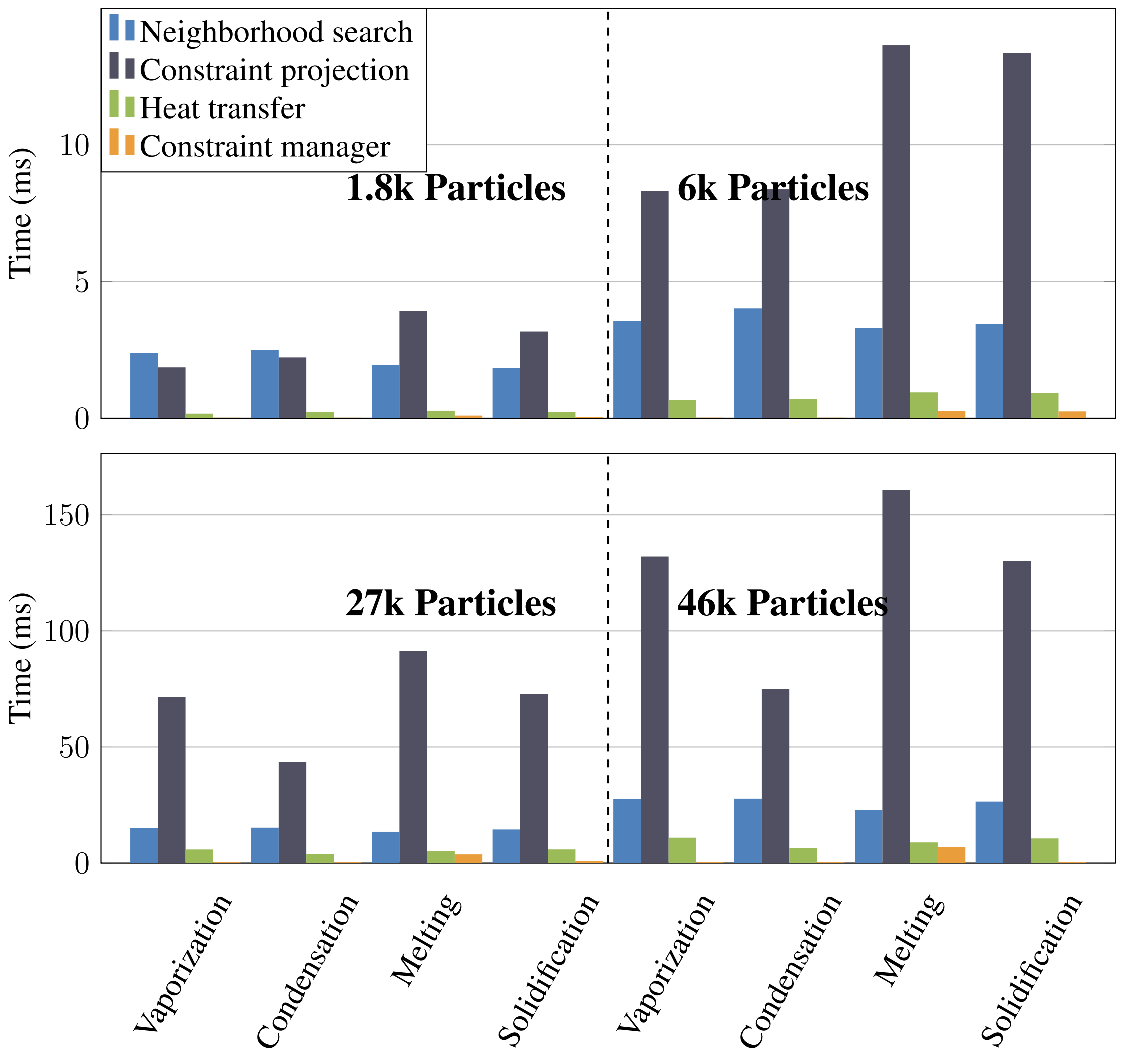


Source: the author.

timestep. Heat transfer, constraint manager, and neighborhood times search times also grow linear with the number of particles, but their growth factor is much lower. This indicates that the percentage of phase-shifting cost in relation to the total cost tends to decrease for larger numbers of particles.

In Figure 4.13, we compare the computation time among simulations when they have the same number of particles. At a low number of particles, the neighborhood search is more expensive than the constraint projection (for Vaporization and Condensation). On the other hand, among the other scenarios and particle configurations, the constraint projection takes more time than the timestep total. Despite that, even with a greater number of particles, the neighborhood search time remains stable, and the constraint manager has no noticeable influence in vaporization and condensation effects. The hardest simulation was melting, which took more time independently of the particle number, these simulations where our constraint manager had the worst behavior. Nevertheless, the constraint manager has a lower impact on the whole simulation, even at a large number of particles.

Figure 4.13: Time spent per task in simulations with different numbers of particles, considering that the number of particles is constant in each simulation.



Source: the author.

Parameters and Method Stability: Table 4.3 shows the principal variables used in the simulations. We mention most of the parameters in this document, and we will address these parameters regarding the stability and values of the methods we selected for the simulations.

First, the final value of the distance constraint relies on Young's Modulus E_0 and Linear coefficient of expansion e (listed on section 3.1). The value of E_0 must be within the 0 to 1 range because the temperature based E uses this value and an inverse sigma function so that it acts as a damper. Larger values of E_0 lead to instabilities in the solid phase from the first timestep. The main difference between E_0 and α from XPBD is the variation regarding the current temperature of the particles. Likewise, the parameter e is constrained to the range from 0.0 to 0.01. When set to 0, there is no dilation generated from temperature, and the bigger this value, the faster a body dilates and contracts. Values above 0.01 cause instabilities because of the velocity of contraction and dilation.

Table 4.3: List of parameters with their values used for simulations

Notation	Description	Range	Value
E_0	Initial Young's modulus	$(0 - 1]$	1
e	Linear coefficient of expansion	$[0.0 - 0.01]$	0
nC_{max}	Maximum constraint number	$4 - 10$	4
rs_{max}	Maximum radius for solidification	$[2r - 4r]$	0.055
h	SPH Support radius	-	4r
c	heat capacity	-	1000
$T_{melting}$	Melting Temperature	-	0
$T_{evaporation}$	Evaporation Temperature	-	100
L_{m-s}	Latent Heat melting - solidification	-	333
L_{e-c}	Latent Heat evaporation - condensation	-	2257
k	Thermal conductivity	-	Depends Demo
ϕ	diffusion of particle	$[10 - 270]$	Depends Demo
μ	viscosity	$0.025 - 1$	0.025
r	Particle radius	$0.01 - 0.025$	0.025
ρ_{solid}	density of solid particle	$[700 - 1000]$	900
ρ_{fluid}	density of fluid particle	$[700 - 1000]$	1000
ρ_{gas}	density of gas particle	$[700 - 1000]$	700
m	mass of particle	$[0.0064 - 0.07]$	$0.8 * 8r^3 * \rho_{solid}$
α	Compliance	$[0 - 1]$	$0.16e - 9$

The next set of parameters affects the freezing behavior. nC_{max} controls the maximum number of constraints that one can attach. Although any value above 0 could be set, the number of constraints highly affects the consistency of solids. Particles with less than four distance constraints are quite deformable. Particles with more than ten constraints turn the simulation unstable because our constraint manager does not create distance constraints with a predefined structure. So particles within the maximum radius for solidification rs_{max} and solidification temperature will be connected. Moreover, rs_{max} values under $2r$ will not create any new constraint due to the minimum distance between particles. Values above $4r$ are suitable to simulate. Still, the behavior of the new solid is somewhat unpredictable: distance constraints very different within the same body causes inconsistency when the body is contracting or dilating.

The third set of parameters in Table 4.3 are regarding the thermal properties of the material. Those values mainly affect the time for the heat to move from a particle to another. Heat capacity (c) directly influences the amount of latent heat in a given particle (Equation 4.3). So, this value changes the velocity of phase change. We used a constant value of 1000 in our simulations, but changing this value will not affect the simulation stability. Values for $T_{melting}$, $T_{evaporation}$, L_{m-f} and L_{e-c} were constant as well, and we used values corresponding to water. Since in real-world $T_{melting} < T_{evaporation}$, to meet this inequality is the only restriction to set those values. An incorrect selection of threshold temperatures will lead to artifacts because particles with distance constraints will arise, compromising the stability of the model. Moreover, L_{m-f} and L_{e-c} together with c define the time that a phase change takes in the simulation. Again, an incorrect value selection will not affect the stability, but it is recommended to use real-world values to obtain a simulation within a reasonable phase change time. Thermal conductivity (k) and ϕ are closely related in the sense that those variables define the heat transfer velocity on each of the schemes: Explicit or Cleary and Monaghan. Selecting the correct values for either of the models should cause the same behavior, but the Cleary model advantage is the capability to work with different materials interactions. The final value depends on the k value of each neighbor particle, while in the explicit model, ϕ is the same for each neighbor particle. Thus, it is not the selection of heat model but the parameter of each model that affects the behavior of the simulation.

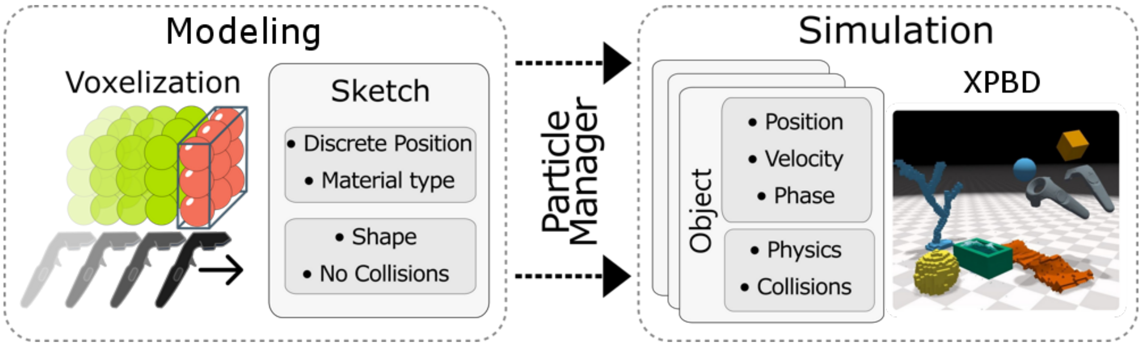
5 SKETCHING DYNAMIC OBJECTS IN IMMERSIVE VIRTUAL REALITY

In this chapter we present a novel immersive sketching application to create objects with different materials. Our solution allows physically-based interactions between objects in real-time using Extended Position-Based Dynamics (XPBD). It is possible to create expressive dynamic-sketches involving several types of physical behaviors such as rigid solid bodies, liquids, gases, soft solids, and clothes. As an example, it is possible to change the flow of a river by adding or digging into the ground while the river is flowing, or even to create beautiful waterfalls using the same methods. The application also allows the user to see in real time the interaction of recently created soft objects or even cloths with the existing environment, including other sketched objects. This chapter uses some results and paragraphs from our paper entitled *Phys-Sketch: Sketching 3D Dynamic Objects in Immersive Virtual Reality* (TICONA et al., 2019).

5.1 System Overview

The particle-based virtual reality proposed tool has two main modules. The first is the *Sketching* module, where the user sketching 3D objects. Users can select material types, shapes, and sizes of the brush of his/her choice. During this stage, the 3D objects are unprovided of physical properties to maintain control and stability of the simulation. For performance and integration reasons, the 3D objects are represented as a set of spheres (see Figure 5.1 left). This process is called *Voxelization*, which will be explained further on the Sub-section 5.1.1.

Figure 5.1: Our approach constantly switches between two main states for each new body in the scene; Sketching and simulation. The first state is used to create new elements into the scene by placing new particles as long as the user moves the controller through space. Dynamic behavior is given to the bodies on the second stage depending on the properties of each particle. The dynamic of the previously created bodies is not stopped when the user starts drawing a new body.

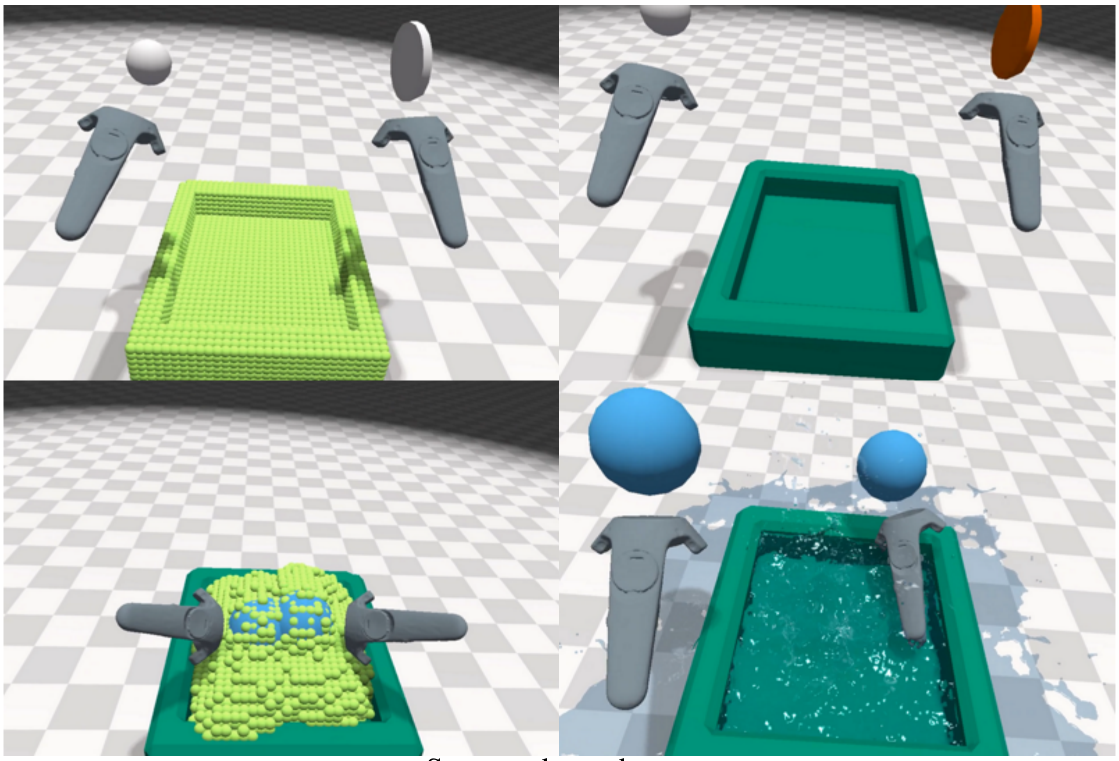


The second is the *Simulation* module the physics is assigned to the 3d object (see Figure 5.1 right). In this stage, the animated objects interact between them, with user and environment. The Position-Based Dynamic (PBD) solver for the simulation (see Section 3.2) is adapted and integrated with our *Sketching* module. During the experience, the user can switch between the stages every time he draws a new body. In addition to users can touch, stretch, grasp, and move them as desired. Figure 5.2 shows step by step as creating a 3D object.

We built the tool over the standard position-based dynamics framework (see Section 3.2). The next algorithm 6 gives a general view of one timestep of our simulation. After the main PBD calculations, we update both velocities and positions of the virtual tool using the external tracking system (see Section 5.1.4). When the user creates a new object (Line 5, from Sketching to Simulation stage), we capture the path traveled by the control and the material which was chosen by the user. Introduced the data (position, velocity, and phase) in the function *CreateParticles* (Line 6, Create the available materials Section 5.1.3), and depending on the material chose, computes the parameters based in the location particles. This is made between lines 8 and 15 (mass center, relative positions, cluster mass center, cluster relative positions, and density).

On line 16, once we have the material properties, the particles are moved to the main vector, and then we sync (Line 17) the new vector with the running simulation to avoid memory conflicts. If the user is not creating an object but modeling it (Line 18, Sketching stage), here we use the position and velocity of the tool to create a soft path (Line 19, Section 5.1.2). Having this path we can proceed to create or remove particles

Figure 5.2: Step by step of Sketching-simulation stages: Upper-left image shows the sketching stage where particles are in the absence of any physic behavior to make more accessible the composition of the body. The upper-right image shows the same object after the sketching stage; the new 3D object is now a solid and has physic properties. Lower-left shows the creation of a fluid body, and the brush is blue. Note that the already created objects stills in the simulation stage. In the last image, we assign fluid properties to the new particles. Both bodies are interacting in the same scene.



Source: the author.

using algorithms 7 and 8 (see Equation 5.7). Next, in lines 20-23, we define the boolean option to perform. If the selected material is *Solid*, *Soft* or *Fluid*, we create the voxelized particles which are candidates to become an object (Line 21). If the material does not correspond to one of the mentioned before, we delete the particles in the path (Line 24). The process ends in line 25 when we finally render the vector P_{shape}.

5.1.1 Voxelization

The manner as the particles are created plays a fundamental role in the simulation: placing more than one particle at the same position could generate overshoots at the first timesteps. To solve this issue, we use voxelization in the sketching stage, where each particle represents a voxel. The environment divided into a 3D grid $M \in R3$. Subsequently, M is initialized with -1 values. When placed a body into the scene, the values

Algorithm 6 Main Loop

```
for all Particles do (Timestep Starts)
    PBD Core calculations
    $P_{Tool} \leftarrow UpdatePositions$ (Section 5.1.4)
    $V_{Tool} \leftarrow UpdateVelocities$ (Section 5.1.4)
    if $NewObject$ then
        $createParticles(P_{shape}, MaterialType)$
        switch $MaterialType$ do
            case $Solid$ (Sub Sec. 3.5.1 Solids)
                $calculateMassCenter$ (Fig. 3.8)
                $calculateRelativePositions$ (Fig. 3.8)
            case $Deformable$ (Sub Sec. 3.5.2 Soft, Clothes)
                $calculateClusterMassCenter$ (Fig. 3.10)
                $calculateClusterRelativePositions$ (Fig. 3.10)
            case $Fluid$ (Sub Sec. 3.5.3 Fluids)
                $CalculateDensity$ (Fig. 3.11)
        $addParticles$
        $sync$
    else
        $P_{path} \leftarrow InterpolationPosition(P_{tool}, V_{tool})$
(Sec. 5.1.2)
        if $Material$ then
            $P_{shape} \leftarrow Voxelization(P_{path}, typeShape)$ (Alg. 7)
        else
            $particleRemotion(P_{path})$ (Alg. 8)
        end if
    end if
    $draw(P_{shape})$
    PBD Variable updating
end for
```

of M changed to 0. This proceeding is in algorithm 7. Finally, to remove particles, the algorithm 8 is applied. We sync the coordinates of the space and the coordinates of our binary 3D grid using the following relations:

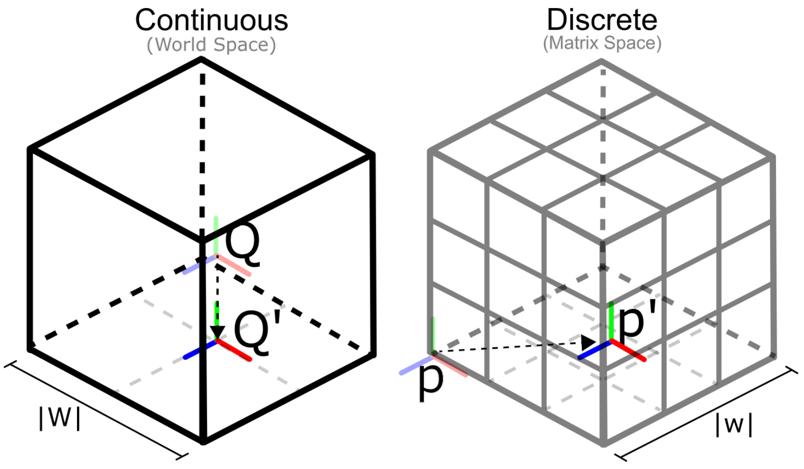
$$p' = p - c \tag{5.1}$$

$$Q' = Q - C \tag{5.2}$$

$$\frac{p'}{Q'} = \frac{|w|}{|W|} \tag{5.3}$$

where p is a positive integer of the 3D grid, p' the virtual coordinate system (VCS), translated by the center grid (c), and Q is the (Physic) World Coordinate System (WCS). Similarly, (C) is the center of the tracking system. $|w|$ and $|W|$ are the width of the 3D virtual grid and the physical workspace, respectively. Consequently, this gives us the Correlation

Figure 5.3: Space Map: World space coordinates coming from the immersive environment starts at a given point in the middle of the workspace, we translate that point *q* to the floor *q'*, and coordinates from matrix starts at (0,0,0) point *p*. Translate a point to the middle of the total size of the matrix *p'* to homogenize both spaces.



Source: the author.

5.3.

Algorithm 7 Voxelization

```
1: procedure VOXELIZATION($P$)
2:     $p = discretization(P)$ (Using Equation 5.4)
3:     if $p \in M$ & $M[p] < 0$ then
4:         $P' = interpolation(p)$ (Using Equation 5.5)
5:         $M[p] = index$
6:         $Particles[index] = P'$
7:         $index++$
8:     end if
9: end procedure
```

Moreover, to translate world positions to the discrete space (VCS), we use the Equation 5.4, while for bringing back the particles to the world space (WC), we use the Equation 5.5.

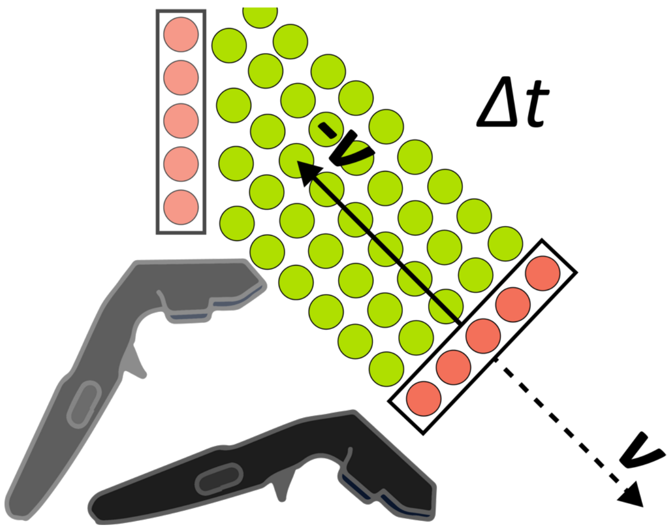
$$p = (Q - C)\frac{|w|}{|W|} + c \tag{5.4}$$

$$Q = (p - c)\frac{|W|}{|w|} + C \tag{5.5}$$

Algorithm 8 Remove Particles

```
1: procedure PARTICLEREMOTION($P$)
2:     $p = discretization(P)$ (Using Equation 5.4)
3:     if $p \in M$ & $M[p] \geq 0$ then
4:         $indexRemove = M[p]$
5:         $M[p] = -1$
6:         $removeParticle(indexRemove)$
7:     end if
8: end procedure
```

Figure 5.4: Particle placement: Particles are placed along the opposite direction of the velocity in each timestep to fill the gaps produced by the controllers travels.



Source: the author.

5.1.2 Particles Interpolation

Brush movement tend to be fast during the simulation, it creates non-connected curves in the space. To avoid holes in the sketch, the brush shape is used as extrusion plane in each timestep. To do this, we calculate the velocity vector oposed to the controller movement. Secondly we find the traveled distance ($||V||\Delta t$), and next we calculate how many particles (i_{max}) fit in that distance (see Equation 5.6).

$$i_{max} = Round(\frac{||V||\Delta t}{d_{particle}}) \tag{5.6}$$

Moreover, we divide this distance ($||V||\Delta t$) by the diameter of a particle ($d_{particle}$) to get the maximum number of particles (i_{max}) to project along the opposite velocity direction. Finally, in Equation 5.7 we proceed to calculate the exact positions of the new particles.

$$P'(i) = P - i\frac{V}{||V||}d_{particle} \qquad \forall i \in [0, i_{max}] \tag{5.7}$$

At the end of this stage, we have position information, but particles are not dynamic so far. However, in the next section when we discuss particle dynamics through our particle management.

5.1.3 Particle Manager

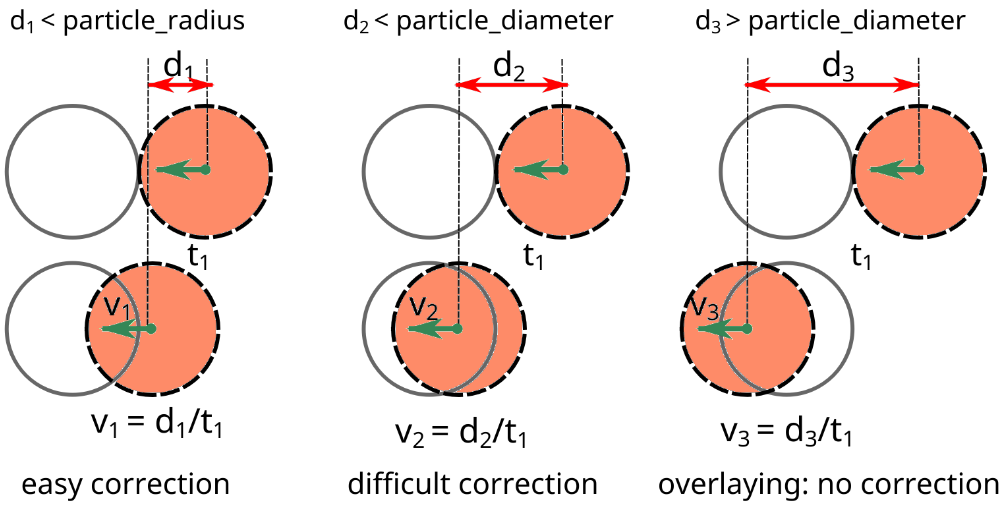
Having the particles in the space is enough to draw, and create a wide variety of artworks, but not enough to create animations, in this step we make particles behave according to their material properties. The first action we take after the voxelization is to initialize the particles: the phase is chosen according to the user input (Solid, Soft body, Cloth, Fluid). Each type of body has different requirements to interact in the scene, the constraints for each kind of body (see Sections 3.5):

5.1.4 Collision

As our model is built over Flex (by Nvidia), we based the collision detection on the *Flex* Model so, our system detects only particle-particle but not particle-mesh collision. That limitation forces us to model everything in the scene with particles, including the tools (controllers). To handle solid-fluid collisions we modify the radius of solid particles constrained to $R_{Solid} < R_{Fluid}$, using this we prevent possible leaks of fluid particles through thin groups of solid particles. Controller-objects collisions are managed differently: First, the controller mesh is filled with zero mass particles, this to avoid gravity effects on the interaction particles.

Collisions particle-particle remains as usual, but the dynamic of those particles is given by the external tracking and not by the simulation dynamics. The collision system remains stable under normal conditions, but there is a maximum velocity where the particles would pass through an object because of the simulation timestep, as this value is considerably high, it is not an issue. Collision detection works well with a velocity no higher than the following condition: $velocity < particle\_diameter/timestep$. This is obtained from the analysis of the cases shown in Figure 5.5. We consider the scenario where the particles' diameter is $10mm$ and the timestep is $1/60s$. With this configuration, we would have the maximum velocity for simulation as $60cm/s$. We could reduce the time step, but that would increase the computational cost, since it would take more time to compute a frame, preventing the real-time simulation.

Figure 5.5: Displacements of a particle (for $d1, d2andd3$ distances) towards another static particle in an timestep t_1. The left image shows a slight collision between particles. The middle one shows a collision that almost reaches the superposition of the particles, in both cases the position can be corrected. And finally the right image shows how one particle passes the other. In this case the restriction cannot be corrected.



Source: the author.

5.1.5 Rendering

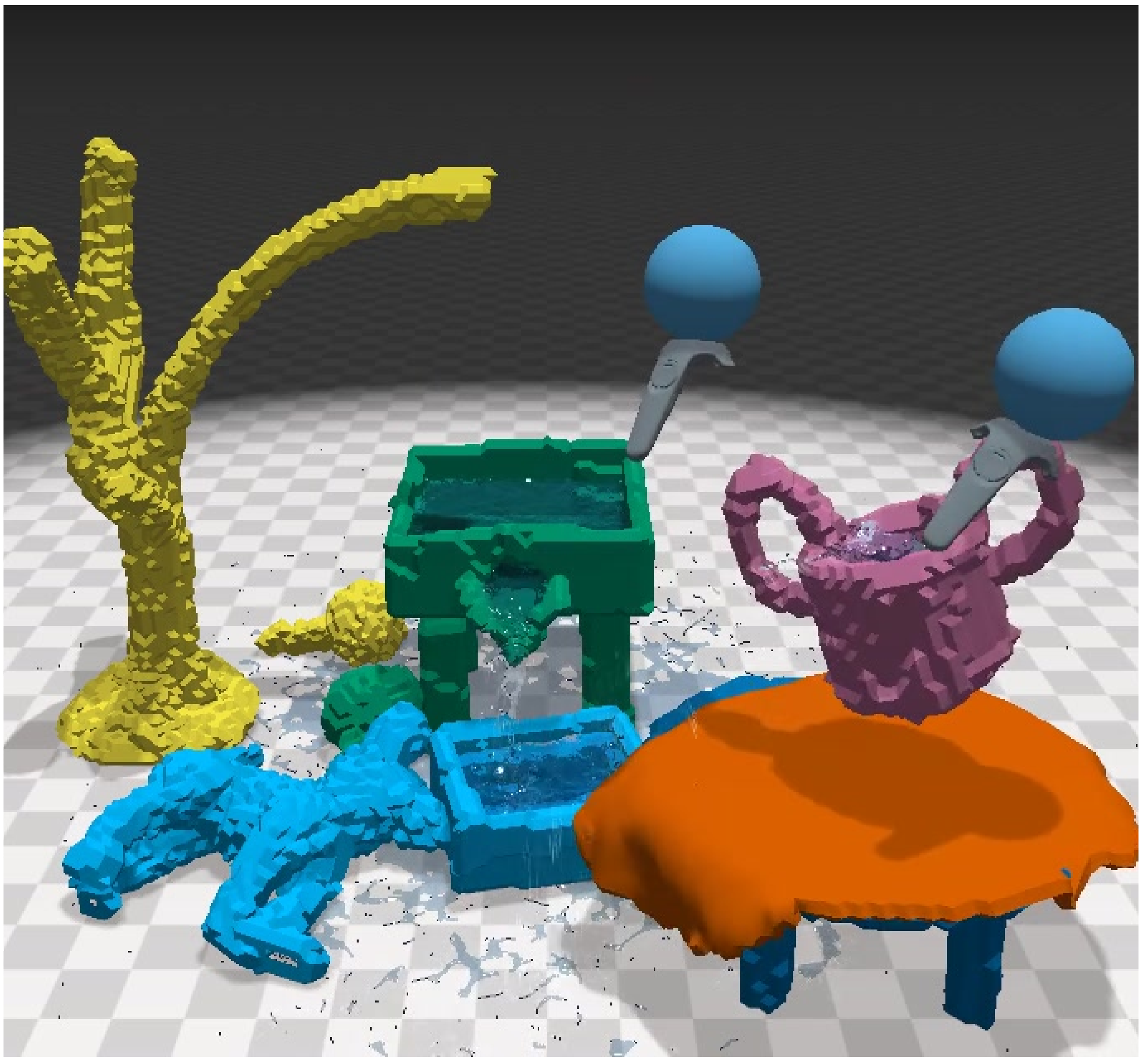
To Fluid: Fluid rendering was done using Anisotropic Kernels because this is the standard system used by Flex. This approach renders the objects based on the neighboring particles, namely performing Principal Component Analysis (PCA) over the neighbors. More information about this method could be found in the Yu & Turk paper published in 2013 (YU; TURK, 2013).

To Rigid and Soft bodies: As our model uses voxelization to arrange particles through space, marching cubes (MC) algorithm (LORENSEN; CLINE, 1987) is a straightforward way to get a conceptual visualization of the objects. During the simulation, we know from the neighborhood search, which particles are on the surface of the object, so we apply MC only on outer particles, reducing computational cost.

5.2 User Interface

To create an object within our framework, the user must follow two steps: First sketch the body and then giving it physical behavior. Although these two stages are well

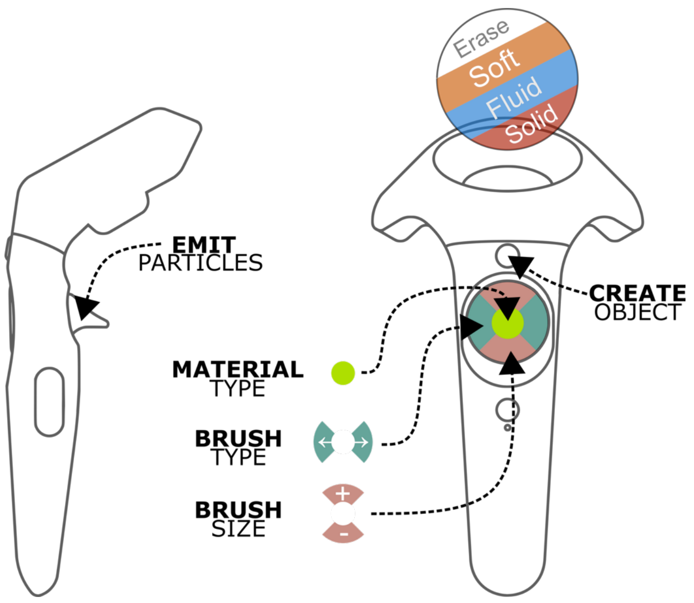
Figure 5.6: Different Objects interacting in the same VR scene and also with the user controllers. Maximum 80k particles were used.



differentiated for the user when creating an object, the simulation keeps always running on background, and newly created objects are smoothly added to the simulation stack.

This separation of stages is necessary to make the sketching easier as long as the user can last as much as he wants. Without this differentiation the newly introduced particles would start falling since its creation, making it harder to create consistent sketches. In the first moment, the generated particles do not have any animation nor rendering (Except the primitive sphere). At this moment, the user needs to start generating particles by using the brush. The brush is the central user-environment interaction tool, which has three main functions: Create, delete and move objects. Though the brush is a platform-independent generic tool, in this case, we have represented the virtual brush as an HTC VIVE controller wich holds the shape and color of the material on its top. Figure 5.8 shows both shapes and material type availability, the user can easily select the object

Figure 5.7: Our method uses the HTC VIVE Controllers to track the user's hands. The image describes the possible interactions while sketching. The circle over the control represent the feedback received by the user while sketching, the colors of this sphere (or chosen shape) defines the behavior that will be given to the new body

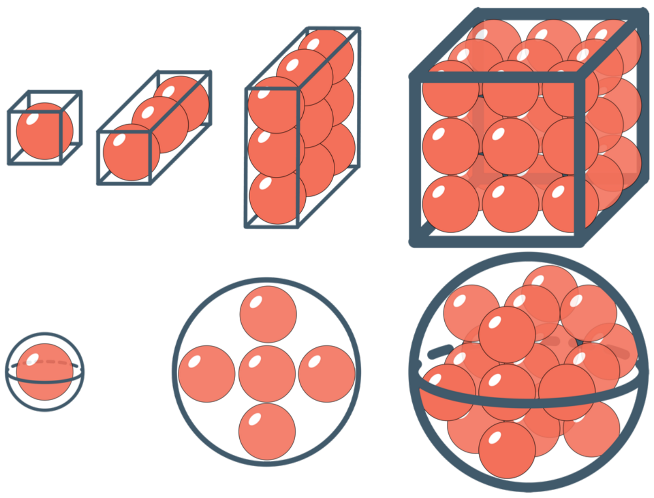


Source: the author.

properties through the controller buttons. Every time the user changes a property, visual feedback is provided on the top of the controller (As shown in Figure 5.7). The amount of particles introduced in the environment is given by the size and shape of the brush: New particles are evenly introduced (See Sub Section 5.1.1) within the volume of the chosen form. The method to create objects in the scene is the movement of the brush through space, the travels of the tool create paths conformed of particles, and these paths are the future objects. In this way, to create a solid body, the user must manually fill each part of the object.

Particle creation can be done with both hands at the same time without overlap conflicts because we correct particle positions on the go. Two hands sketching is free, and any hand can create or delete particles. Some issues could happen when the brush moves so fast, for instance some gaps in the path could appear and make it looks discontinuous, but the stability of the sistem is guaranted. Figure 5.7 show the possible interaction through the controller: To emit particles the user must press the trigger and hold while the sketching, when the trigger is released no particles will be created but the object will remain static until the button *Create Material* is pressed. Between these two interactions, the user can modify the brush shape and size as well as the material type using the circular trackpad. Pressing the center of the trackpad changes the material type, while the vertical

Figure 5.8: Brush tool: Basic particle creation shapes, On the top: The shapes created starting from a cube (Particle, line, plane, sphere). on the bottom: Shapes created starting from a sphere (Particle, circular plane, sphere)



Source: the author.

axis the brush size and the horizontal the brush type.

Movement through the scene is allowed during all the simulation, enabling users to create entire compositions instead of only modeling individual objects. Even though the virtual space is not limited, there are some physical and computational constraints to the movement through the scene; User can move as long as the wires of the HMD let him move. Also, the size of the compositions is limited to the number of particles allowed in the scene, around 80k particles are easily managed in real-time simulations.

5.3 Results

The sketching system was implemented and tested in a Dell workstation with an i7 3.2Ghz Processor, 16Gb Ram, and NVIDIA GeForce GTX 1070 Graphic Card, running Windows 10 and CUDA 9.2. All images and videos used in this paper were generated using HTC Vive headset and controllers for visualization and interaction (see Figure 5.9). However, the sketching system is also compatible (and indeed was tested) with Oculus Rift headset and touch controllers. We also successfully tested Leap Motion for interaction.

5.3.1 Expressiveness and Immersive Experience

We invited two artists to informally test the immersive sketching application to create some sketches using the tools presented in Section 5.1. Both had no previous experience with VR. The test was divided into three stages and artists were free to spend as many time as they wanted in each stage:

1. Training stage: The system was introduced to the participants, letting them become familiar with the controls, the virtual tools, and the virtual environment. The system was presented as a tool for 3D sketching in virtual environments, but capable to simulate physic behaviors. So, they first need to choose a material and a shape; Here the brush was introduced to them. We shown them how to change between shapes, sizes, and materials. To conclude this stage, we requested them to do it themselves drawing a free sketch.

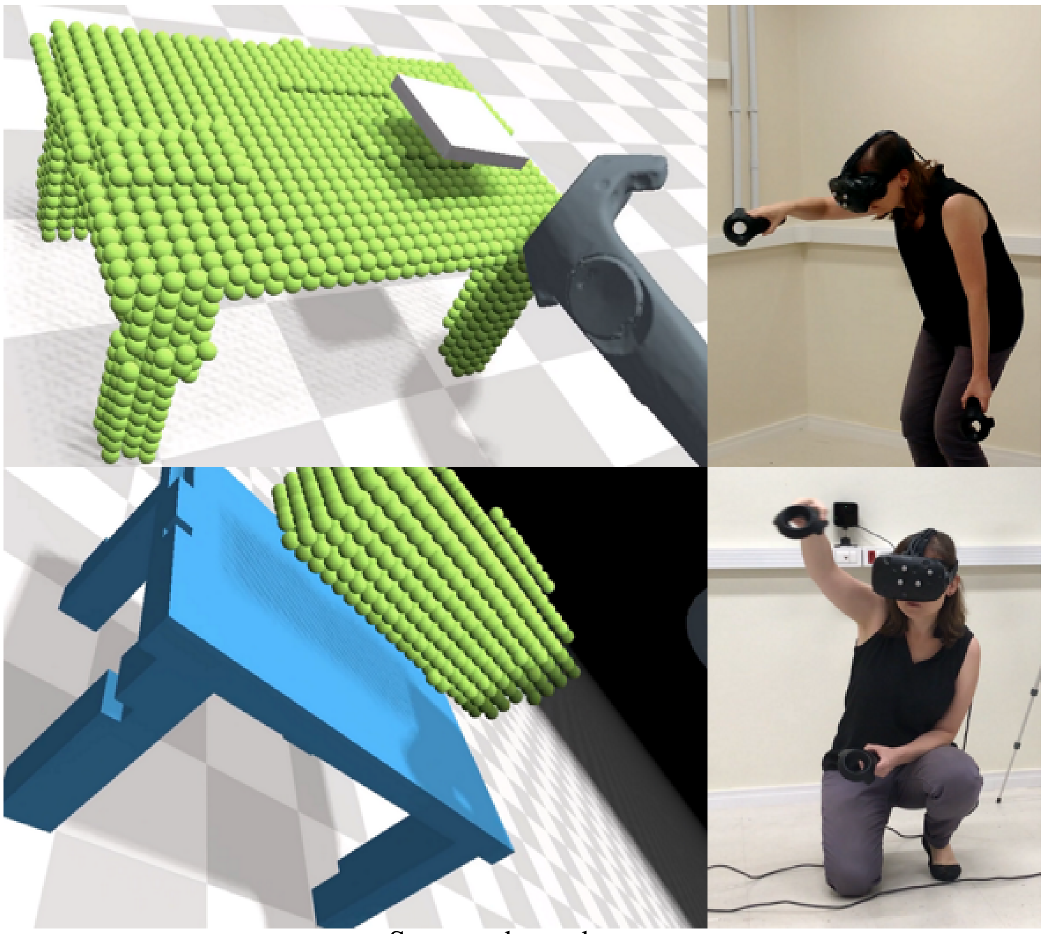
2. Drawing stage: The artists were asked to draw a table and a container to be filled with water. Then, users were requested to interact with the objects they previously created.

3. Expression stage: The artists were asked to sketch something original. For this stage, the gravity was set to zero. The sketched objects still reacts according to the materials they are made, but can freely float in the air, what can be seen as an interesting feature for artistics expression.

The experiments took approximately 40 minutes, where 15 min were spend for training and 25 min for executing the given tasks. Precisely, the first artist spent 40 minutes on the application while the second one spent 35 minutes. None of them reported any symptom of cyberseekness, and both were very comfortable in the virtual environment.

The artists executed a lot of gestures with their arms, also walking and jumping in the real world. During all the process, they changed their positions – walking and crouching – looking for a better point of view to continue their sketches. An interesting behavior we perceived in both was that they moved in such a way to avoid to collide or even to cross the virtual objects. From observing this behavior and as the result of an informal post-test interview, we can conclude they experienced a high sense of presence in the virtual environment.

Figure 5.9: User creating objects as part of the drawing stage. Upper image: User is creating the table and flattening the surface by erasing particles. Lower image: Having created the table, our user proceeds to create a water recipient on the table.

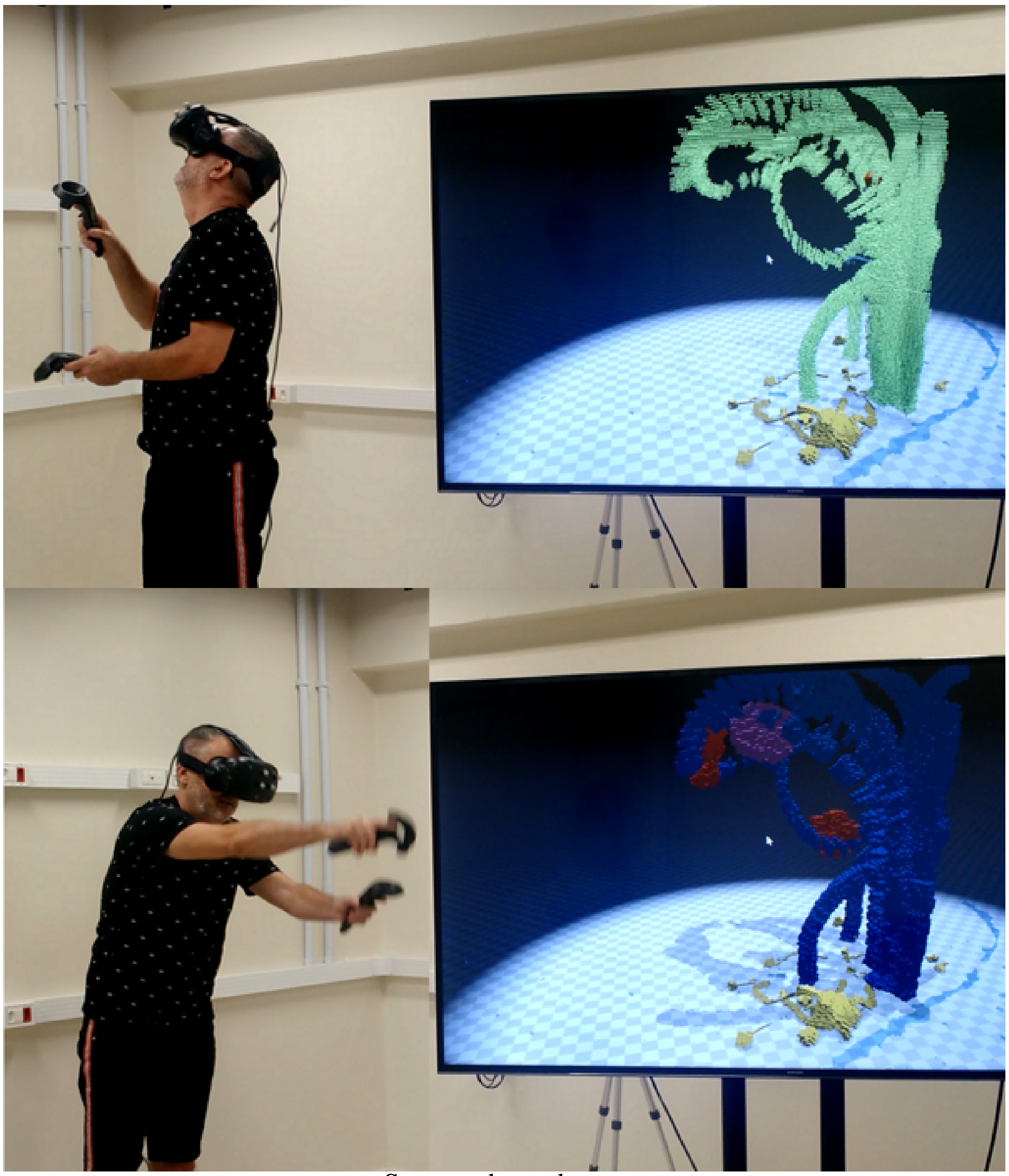


Source: the author.

Subjects also highlighted the comfort of working with physical objects referring to the soft bodies and water behavior. A drawback reported by one artist was the lack of a strong haptic feedback. Currently, our model only conveys vibrotactile feedback by means of the VIVE controllers when the controller strikes an object in the scene. In the expression stage, the subjects reported an improvement in the experience due to zero gravity. Even though this behavior is non-physical it let them draw objects without building supports.

Another comment of the users is that the best strategy to sketch is using the dominant hand to create particles and the other hand to do different actions, like erasing, for instance. As a suggestion, they highlighted the potential of mixing different materials into a single object, for example, to allows the creation of a solid object with a soft part. They also mentioned that the behavior of the objects is credible, but the rendering of solid and soft objects must be improved. Finally, they also stated that the choice of the color and

Figure 5.10: User sketching objects as part of the expression stage. Upper image: User is building a kind of tree. Lower image: User is interacting with his sketches by touching it.

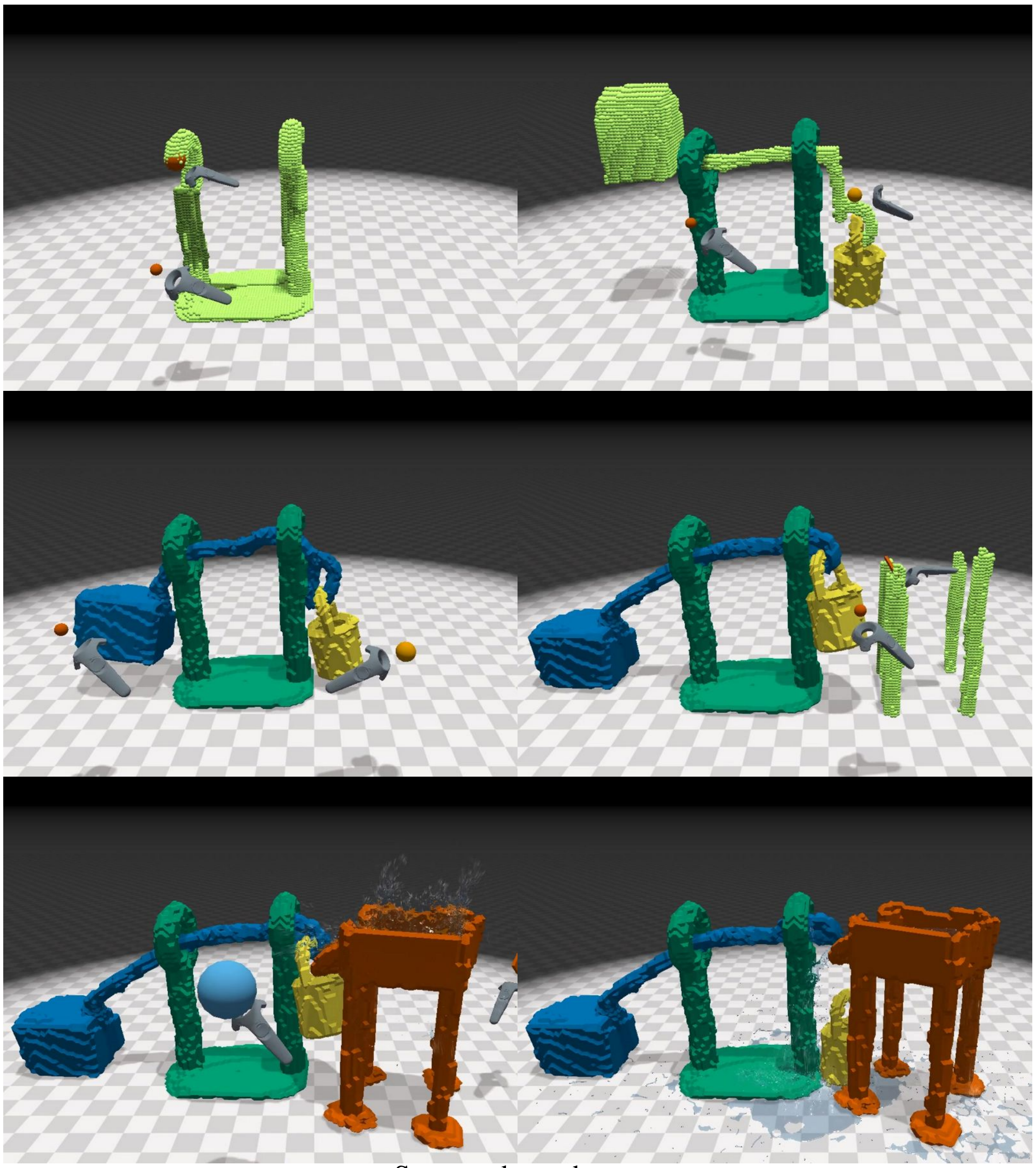

Source: the author.

texture is relevant in the artistic process. However, despite these suggestions for future improvements, they were very excited to keep using the sketching system to create new scenarios and artistic installations. Figures 5.9 and 5.10 show some frames of the users sketching their artistic installations.

5.3.2 Interaction with Physics

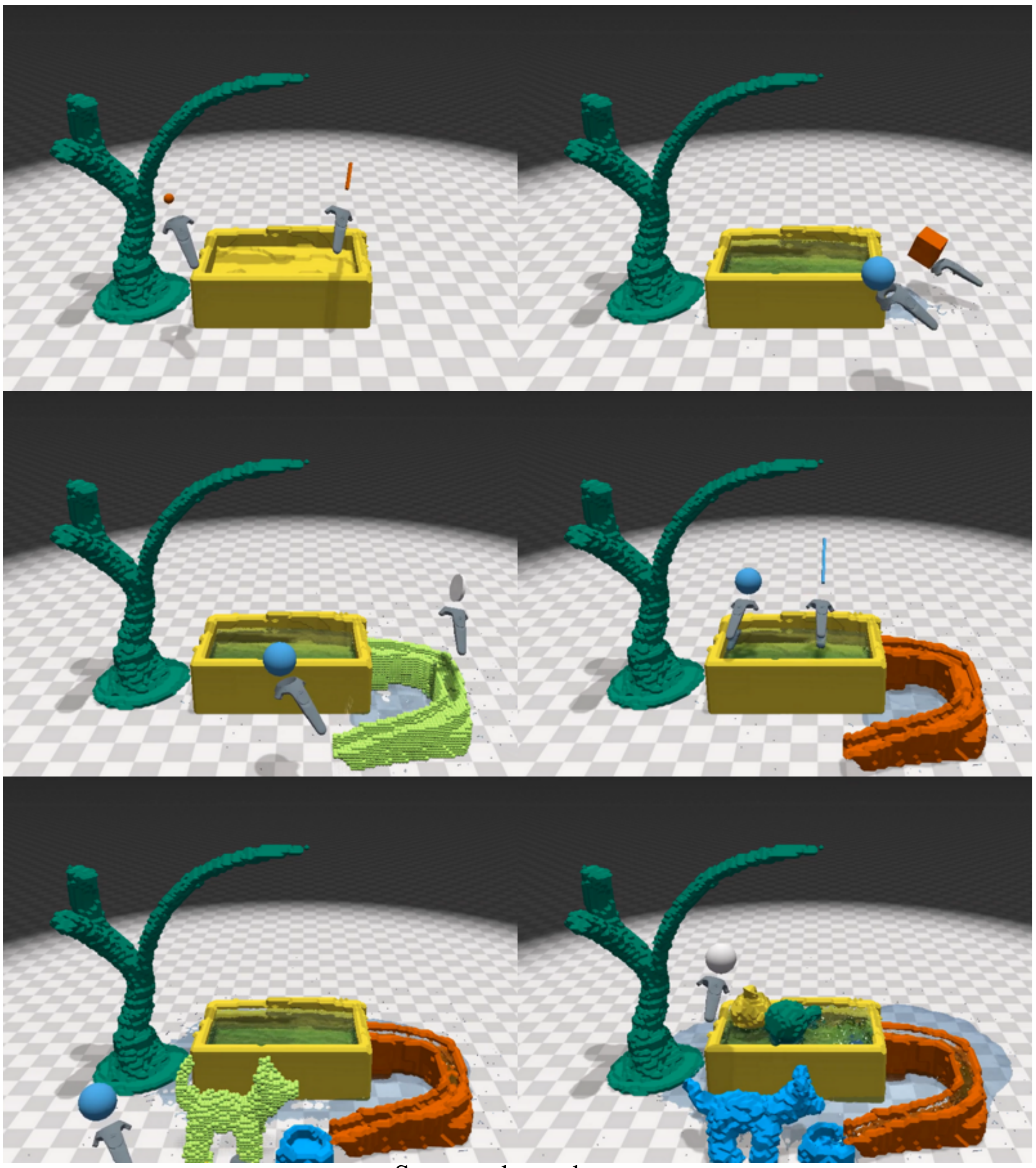
In this work we proposed and prototyped a particle-based system to sketch and simulate virtual objects with physical behavior in a VR environment. Results shown the performance achieved is sufficient to support 3D interaction and fluid animation of the objects. Figures 5.11 and 5.12 show step by step interacting with physics.

Figure 5.11: In these sequence images we show how our tool allows us to use physics during simulation. This figure shows a pail, a bucket and a rope that joins the pail and the bucket, during the simulation we see how the bucket that has more weight lifts the pail, then we create a container of water that will fill the pail and then match the weights. Maximum 45k particles were used.



Source: the author.

Figure 5.12: In this example we placed a container with a small hole (not visible) in the scene, this reveals the fluid behavior: as long as the water leaks, it flows through the path, to finally fill our dog’s bowl. Maximum 30k particles were used.



Source: the author.

6 CONCLUSION

In this work, we presented two contributions. First, we implemented a real-time physically-based method to simulate different types of objects, including the phase-change phenomena. Then, based on this implementation, we proposed and tested an immersive tool for modeling and interacting with objects with different behaviors, including solids, soft objects and fluids.

We presented a full Lagrangian method to simulate bidirectional phase-change materials as a function of temperature. We modeled the solid, liquid, and gas states using XPBD and SPH. In addition, we simulated the following transitions: melting, solidification, vaporization, and condensation. We also introduced modifications to the density, viscosity, and distance constraints, along with the parameters of rigidity and gravity to simulate the thermal phenomena.

A latent heat model was also implemented to drive the transitions between the different states. A heat transfer model was adapted to comply with the law of thermodynamics that defines the heat transfer flow direction. As our work is temperature-based, we need to obtain latent heat from temperature change. Introducing a fully energy-based model, although more computationally intensive, would be a more physics-based way to represent phase change and thermal phenomena in general. The proposed model, nevertheless, covers heat transfer by contact and, in our implementation, imitates convection.

Regarding the particle-based tool to model and simulate virtual objects with physical behavior in a VR environment, results shown the performance achieved is sufficient to support 3D interaction and fluid animation of the objects.

The informal experiments with target users have shown promising results. The users, even if they did not have any previous experience with VR, felt comfortable to move and interact in the virtual environment proposed. They also did not report any cybersickness symptoms during the 40 minutes each one spent fully immersed. Regarding the scenes sketched until now, they seem exciting and sufficient to demonstrate the expressive power of the sketching system proposed.

Notice that the rendering of the objects has a “legolized” appearance. An improvement of the render by smoothing the shapes would, we suppose, significantly improve the user experience. The marching cubes method generates smoother forms when the voxels are smaller. However, smaller voxels will demand more precision from the artists to sketch, as well as more computer power to simulate the behavior of the object.

6.1 Limitations

This work produced good results that are reflected in the published articles. However, we also must take into account the following cases and limitations:

- in the phase-change phenomena simulation, the number of particles is a limitation for the framework solver, simulations up to 8k particles are interactive.
- our solution does not consider the influence of air particles. Its presence would allow to extract the heat of our 3D models with high temperature. In addition it would produce an external force based on pressure that would affect all 3D models, especially fluids and soft objects.
- our collision detection algorithm does not detect particles moving in very high speeds (see Section 5.1.4). This limitation would lead to overshooting objects or causing instability in the simulation. It can also affect the interaction of users with the 3D models, that must be careful to avoid making quick or sudden movements. Otherwise, they may need to deal with inconsistencies that do not occur in real life.
- gas simulation is done with the same technique used to simulate liquids. The difference is that we convert their gravity from $-9.8m/s^2$ to $1m/s^2$, allowing the particles to rise. This strategy is not physically-based.
- regarding the virtual tool, the system allows the user to create and modify an object. However, once it is concluded and we start to sketch another object, we cannot go back and modify this object again
- because of the predefined size of the particles, the user is not able to sketch detailed object. In order to avoid this limitation, we need to reduce the particles' radius, which will increase the simulation cost.

6.2 Future Works

As future work, we can mention the following improvements:

- to include the pressure parameter in the simulation framework to make simulations more realistic, also allowing transfers between solid–gas (deposition, sublimation)

- to include heat transfer by radiation, which consists in transmitting heat from the source outwards in all directions. This would allow transmitting heat without the need for physical contact. In this way, we would be simulating the three ways of transmitting heat (by contact, convention, and radiation)
- to improve the gas simulation with new methods in SPH or PBF capable of handling smoke particles without the need for filling all the domains with atmospheric particles since it is computationally expensive. In this topic, the work of Ren et al. (REN et al., 2016) is worth to be explored
- to consider the influence of the air to improve the simulations results
- to provide haptic and force feedback to the immersive sketching tool. This stimulus may improve the performance of digital artists when sculpting or modeling their artwork
- to allow the modification of existing objects within the sketching tool. This can be easily implemented by using more sophisticated data structures, such as octrees, to store the objects during a sketching session
- to improve the rendering for soft and rigid bodies to give a better look to 3D models
- to conduct formal user tests with users
- to integrate our change-phase framework within the virtual reality sketching tool.

REFERENCES

AKINCI, N. et al. Versatile rigid-fluid coupling for incompressible SPH. **ACM Transactions on Graphics**, v. 31, n. 4, p. 1–8, 2012. ISSN 07300301.

ARORA, R. et al. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In: **Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2018. (CHI '18), p. 185:1–185:15. ISBN 978-1-4503-5620-6. Available from Internet: <http://doi.acm.org/10.1145/3173574.3173759>.

BARAFF, D.; WITKIN, A. Large steps in cloth simulation. In: **Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM, 1998. (SIGGRAPH '98), p. 43–54. ISBN 0-89791-999-8. Available from Internet: <http://doi.acm.org/10.1145/280814.280821>.

BECKER, M.; IHMSEN, M.; TESCHNER, M. Corotated sph for deformable solids. In: CITESEER. **NPH**. [S.l.], 2009. p. 27–34.

BECKER, M.; TESCHNER, M. Weakly compressible sph for free surface flows. In: EUROGRAPHICS ASSOCIATION. **Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation**. [S.l.], 2007. p. 209–217.

BENDER, J.; ERLEBEN, K.; TRINKLE, J. Interactive simulation of rigid body dynamics in computer graphics. **Computer Graphics Forum**, v. 33, n. 1, p. 246–270, 2014. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12272>.

BENDER, J. et al. Position-based simulation of continuous materials. **Computers & Graphics**, Elsevier, v. 44, p. 1–10, 2014.

BENDER, J.; MÜLLER, M.; MACKLIN, M. Position-Based Simulation Methods in Computer Graphics. In: ZWICKER, M.; SOLER, C. (Ed.). **EG 2015 - Tutorials**. [S.l.]: The Eurographics Association, 2015.

BENDER, J.; MÜLLER, M.; MACKLIN, M. A survey on position based dynamics, 2017. In: **Proceedings of the European Association for Computer Graphics: Tutorials**. Goslar Germany, Germany: Eurographics Association, 2017. (EG '17), p. 6:1–6:31. Available from Internet: <https://doi.org/10.2312/egt.20171034>.

BERNDT, I.; TORCHELSEN, R.; MACIEL, A. Efficient surgical cutting with position-based dynamics. **IEEE computer graphics and applications**, IEEE, v. 38, n. 3, p. 24–31, 2017.

BIAN, C.; XIAO, S.; LI, Z. A unified simulation framework for water phase transition based on particles. In: **Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry**. New York, NY, USA: ACM, 2018. (VRCAI '18), p. 2:1–2:8. ISBN 978-1-4503-6087-6. Available from Internet: <http://doi.acm.org/10.1145/3284398.3284419>.

BOUAZIZ, S. et al. Projective dynamics: Fusing constraint projections for fast simulation. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 33, n. 4, p. 154:1–154:11, jul. 2014. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2601097.2601116>.

BRACKBILL, J. U.; KOTHE, D. B.; RUPPEL, H. M. Flip: a low-dissipation, particle-in-cell method for fluid flow. **Computer Physics Communications**, Elsevier, v. 48, n. 1, p. 25–38, 1988.

BRIDSON, R.; MARINO, S.; FEDKIW, R. Simulation of clothing with folds and wrinkles. In: **ACM SIGGRAPH 2005 Courses**. New York, NY, USA: ACM, 2005. (SIGGRAPH '05). Available from Internet: <http://doi.acm.org/10.1145/1198555.1198573>.

CHEN, C.-W. et al. Ontlus: 3d content collaborative creation via virtual reality. In: SPRINGER. **International Conference on Multimedia Modeling**. [S.l.], 2018. p. 386–389.

CLEARY, P. W.; MONAGHAN, J. J. Conduction modelling using smoothed particle hydrodynamics. **Journal of Computational Physics**, Elsevier, v. 148, n. 1, p. 227–264, 1999.

DEUL, C.; CHARRIER, P.; BENDER, J. Position-based rigid-body dynamics. **Computer Animation and Virtual Worlds**, Wiley Online Library, v. 27, n. 2, p. 103–112, 2016.

DEUL, C. et al. Direct position-based solver for stiff rods. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2018. v. 37, n. 6, p. 313–324.

DOMARADZKI, J.; MARTYN, T. Improved particle-based ice melting simulation with sph air model. Václav Skala-UNION Agency, 2014.

EROGLU, S. et al. Fluid sketching—immersive sketching based on fluid flow. In: IEEE. **2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)**. [S.l.], 2018. p. 475–482.

FACEBOOK. **Quill**. 2016. Available from Internet: <https://quill.fb.com/>.

FARROKHPANAH, A.; BUSSMANN, M.; MOSTAGHIMI, J. New smoothed particle hydrodynamics (sph) formulation for modeling heat conduction with solidification and melting. **Numerical Heat Transfer, Part B: Fundamentals**, Taylor and Francis, v. 71, n. 4, p. 299–312, 2017. Available from Internet: <https://doi.org/10.1080/10407790.2017.1293972>.

GAO, Y. et al. A novel fluid-solid coupling framework integrating flip and shape matching methods. In: **Proceedings of the Computer Graphics International Conference**. New York, NY, USA: ACM, 2017. (CGI '17), p. 11:1–11:6. ISBN 978-1-4503-5228-4. Available from Internet: <http://doi.acm.org/10.1145/3095140.3095151>.

GAO, Y. et al. An efficient heat-based model for solid-liquid-gas phase transition and dynamic interaction. **Graphical Models**, v. 94, p. 14 – 24, 2017. ISSN 1524-0703. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1524070317300589>.

GOOGLE. **Tilt Brush**. 2016. Available from Internet: <https://www.tiltbrush.com/>.

GRAY, J. P.; MONAGHAN, J. J.; SWIFT, R. Sph elastic dynamics. **Computer methods in applied mechanics and engineering**, Elsevier, v. 190, n. 49-50, p. 6641–6662, 2001.

HOCHSTETTER, H.; KOLB, A. Evaporation and condensation of sph-based fluids. In: **Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation**. New York, NY, USA: ACM, 2017. (SCA '17), p. 3:1–3:9. ISBN 978-1-4503-5091-4. Available from Internet: <http://doi.acm.org/10.1145/3099564.3099580>.

IHMSEN, M. et al. A parallel sph implementation on multi-core cpus. **Computer Graphics Forum**, v. 30, n. 1, p. 99–112, 2011. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01832.x>.

IHMSEN, M. et al. SPH Fluids in Computer Graphics. In: LEFEBVRE, S.; SPAGNUOLO, M. (Ed.). **Eurographics 2014 - State of the Art Reports**. [S.l.]: The Eurographics Association, 2014. ISSN 1017-4656.

JAKOBSEN, T. Advanced character physics. In: IO INTERACTIVE, COPENHAGEN DENMARK. **Game developers conference**. [S.l.], 2001. v. 3, p. 383–401.

KIM, Y. et al. Canvox: High-resolution vr painting in large volumetric canvas. **arXiv preprint arXiv:1704.02724**, 2017.

KOSCHIER, D. et al. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In: JAKOB, W.; PUPPO, E. (Ed.). **Eurographics 2019 - Tutorials**. [S.l.]: The Eurographics Association, 2019. ISSN 1017-4656.

KUGELSTADT, T.; KOSCHIER, D.; BENDER, J. Fast corotated fem using operator splitting. **Computer Graphics Forum**, v. 37, n. 8, p. 149–160, 2018. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13520>.

KUGELSTADT, T.; SCHÖMER, E. Position and orientation based cosserat rods. In: EUROGRAPHICS ASSOCIATION. **Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation**. [S.l.], 2016. p. 169–178.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In: ACM. **ACM siggraph computer graphics**. [S.l.], 1987. v. 21, n. 4, p. 163–169.

MACHUCA, M. D. B. et al. Multiplanes: Assisted freehand vr sketching. In: ACM. **Proceedings of the Symposium on Spatial User Interaction**. [S.l.], 2018. p. 36–47.

MACKLIN, M.; MÜLLER, M. Position based fluids. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 32, n. 4, p. 104:1–104:12, jul. 2013. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2461912.2461984>.

MACKLIN, M.; MÜLLER, M.; CHENTANEZ, N. Xpbd: position-based simulation of compliant constrained dynamics. In: ACM. **Proceedings of the 9th International Conference on Motion in Games**. [S.l.], 2016. p. 49–54.

MACKLIN, M. et al. Unified particle physics for real-time applications. **ACM Transactions on Graphics (TOG)**, ACM, v. 33, n. 4, p. 104, 2014.

MCNAUGHT, A. D.; MCNAUGHT, A. D. **Compendium of chemical terminology**. [S.l.]: Blackwell Science Oxford, 1997.

MIAO, Y.; XIAO, S. Particle-based ice freezing simulation. In: ACM. **Proceedings of the 14th ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry**. [S.l.], 2015. p. 17–22.

MONAGHAN, J. Smoothed particle hydrodynamics and its diverse applications. **Annual Review of Fluid Mechanics**, v. 44, n. 1, p. 323–346, 2012. Available from Internet: <https://doi.org/10.1146/annurev-fluid-120710-101220>.

MONAGHAN, J. J. Simulating free surface flows with sph. **Journal of computational physics**, Elsevier, v. 110, n. 2, p. 399–406, 1994.

MONAGHAN, J. J. Smoothed particle hydrodynamics. **Reports on Progress in Physics**, IOP Publishing, v. 68, n. 8, p. 1703–1759, jul 2005. Available from Internet: <https://doi.org/10.1088%2F0034-4885%2F68%2F8%2Fr01>.

MÜLLER, M.; CHENTANEZ, N. Solid simulation with oriented particles. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 30, n. 4, p. 92:1–92:10, jul. 2011. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2010324.1964987>.

MÜLLER, M. et al. Meshless deformations based on shape matching. **ACM transactions on graphics (TOG)**, ACM, v. 24, n. 3, p. 471–478, 2005.

MÜLLER, M. et al. Particle-based fluid-fluid interaction. In: **Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation**. New York, NY, USA: ACM, 2005. (SCA ’05), p. 237–244. ISBN 1-59593-198-8. Available from Internet: <http://doi.acm.org/10.1145/1073368.1073402>.

MüLLER, M. et al. Position based dynamics. **Journal of Visual Communication and Image Representation**, v. 18, n. 2, p. 109 – 118, 2007. ISSN 1047-3203. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1047320307000065>.

NEALEN, A. et al. Physically based deformable models in computer graphics. **Computer Graphics Forum**, v. 25, n. 4, p. 809–836, 2006. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2006.01000.x>.

PAN, J. et al. Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics. **Computer Animation and Virtual Worlds**, Wiley Online Library, v. 26, n. 3-4, p. 321–335, 2015.

REN, B. et al. Fast sph simulation for gaseous fluids. **The Visual Computer**, Springer, v. 32, n. 4, p. 523–534, 2016.

SALAZAR, S. V. et al. Heat-based bidirectional phase shifting simulation using position-based dynamics. **Computers & Graphics**, Elsevier, v. 76, p. 107–116, 2018.

SCHECHTER, H.; BRIDSON, R. Ghost SPH for animating water. **ACM Transactions on Graphics**, v. 31, n. 4, p. 1–8, 2012. ISSN 07300301.

SEBASTIAN, A.; SAMI, S. **GPU-based clay simulation and ray-tracing tech in Claybook.** 2018. Available from Internet: <https://www.claybookgame.com/,https://www.secondorder.com/>.

SEO, J. H.; BRUNER, M.; AYRES, N. Aura garden: Collective and collaborative aesthetics of light sculpting in virtual reality. In: ACM. **Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems**. [S.l.], 2018. p. Art12.

SOLENTHALER, B.; PAJAROLA, R. Predictive-corrective incompressible sph. In: ACM. **ACM transactions on graphics (TOG)**. [S.l.], 2009. v. 28, n. 3, p. 40.

STAM, J. Nucleus: Towards a unified dynamics solver for computer graphics. In: IEEE. **2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics**. [S.l.], 2009. p. 1–11.

STOMAKHIN, A. et al. A material point method for snow simulation. **ACM Transactions on Graphics (TOG)**, ACM, v. 32, n. 4, p. 102, 2013.

STOMAKHIN, A. et al. Augmented mpm for phase-change and varied materials. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 33, n. 4, p. 138:1–138:11, jul. 2014. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2601097.2601176>.

TICONA, J. A. et al. Phys-sketch: Sketching 3d dynamic objects in immersive virtual reality. In: SPRINGER. **Computer Graphics International Conference**. [S.l.], 2019. p. 119–130.

WEILER, M.; KOSCHIER, D.; BENDER, J. Projective fluids. In: **Proceedings of the 9th International Conference on Motion in Games**. New York, NY, USA: ACM, 2016. (MIG '16), p. 79–84. ISBN 978-1-4503-4592-7. Available from Internet: <http://doi.acm.org/10.1145/2994258.2994282>.

WEILER, M. et al. A physically consistent implicit viscosity solver for sph fluids. **Computer Graphics Forum**, v. 37, n. 2, p. 145–155, 2018. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13349>.

YU, J.; TURK, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. **ACM Transactions on Graphics (TOG)**, ACM, v. 32, n. 1, p. 5, 2013.

APPENDIX A — LIST OF PUBLICATIONS

Articles with distinct contributions to Human-Computer Interaction, Virtual Reality, and Physically-based simulations were submitted and accepted throughout our research. The following is a complete list of our publications up to this time:

Journal Papers:

- **Heat-based bidirectional phase shifting simulation using position-based dynamics.**
  SALAZAR, S. V.; **TICONA, J. A.**; TORCHELSEN, R.; NEDEL, L.; MACIEL, A.
  2018, Computers & Graphics.

Conference Papers:

- **Phys-Sketch: Sketching 3D Dynamic Objects in Immersive Virtual Reality.**
  TICONA, J. A.; SALAZAR, S. V.; TORCHELSEN, R.; NEDEL, L.; MACIEL, A.
  2019, Computer Graphics International Conference.
- **Evaluation of Visual, Auditory and Vibro-Tactile Alerts in Supervised Interfaces.**
  SOUZA, G. A.; TORRES, L. A.; DANI, V. S.; SALAZAR, S. V.; **TICONA, J. A.**; MACIEL, A.; NEDEL, L.
  2018, 20th Symposium on Virtual and Augmented Reality (SVR).

Short Papers:

- **3DAthlon: 3D Gestural Interfaces to Support a 3-Stage Contest in VR.**
  GRANDI, J. G.; DEBARBA, H. G.; FRANZ, J.; OLIVEIRA, V.; **TICONA, J. A.**; SOUZA, G. A.; BERTI, I.; SALAZAR, S. V.; NEDEL, L.; MACIEL, A.
  2018, IEEE Conference on Virtual Reality and 3D User Interfaces (VR).

Demonstrations:

- **3DUI Contest. - Team CG-UFRGS**
  GRANDI, J. G.; DEBARBA, H. G.; FRANZ, J.; OLIVEIRA, V.; **TICONA, J. A.**; SOUZA, G. A.; BERTI, I.; SALAZAR, S. V.; NEDEL, L.; MACIEL, A.
  2018, The 9th annual IEEE 3DUI Contest.

APPENDIX B — RESUMO EM PORTUGUÊS

B.1 Contexto

A simulação da mudança de estado e a interação em realidad virtual são dois abordagens que queremos integrar neste trabalho. A mudança de estado consiste em que modifica as propriedades do material, envolve processos como solidificação, fusão, evaporação e condensação e é diretamente dependente da pressão e da temperatura. O esboço 3D em realidade virtual é uma forma de interação imersiva, que consiste em definir a forma, escolher o material e a ferramenta certa para apoiar a criação de um objeto 3D para permitir que os usuários personalizem seus protótipos.

Nosso trabalho apresenta a exploração e adaptação dos métodos de Dinâmica Baseada em Posição extendida (XPBD) e Hidrodinâmica de Partículas Suavizadas (SPH) para simular objetos nas suas três estados da matéria (sólida, líquida e gasosa). Posteriormente, simulamos as transições entre os estados da matéria usando a temperatura e o calor latente. Em seguida, propomos e implementamos uma ferramenta de realidade virtual que nos permite esboçar objetos 3D usando os materiais modelados. O resultado deste trabalho é a integração entre simulação e interação em realidade virtual, onde os usuários são capazes de criar objetos viscosos, rígidos e deformáveis e, ao mesmo tempo, testar o seu comportamento físico.

B.2 Modelagem Termodinâmica

Apresentamos uma combinação de métodos baseados em partículas que cobrem todos os estados da matéria, PBD para representar objetos sólidos e flexíveis, e SPH para fluidos(liquido e gás) adicionalmente incorporamos propriedades físicas para os objetos como rigidez, densidade, gravidade, viscosidade. Em seguida, combinamos partículas com um modelo de temperatura baseado em SPH para rastrear o fluxo de calor entre eles. Usamos a densidade para acoplar todos os objetos em um mesmo cenário. Em seguida, usamos um modelo de transferência de calor para variar a temperatura e o calor das partículas. O nosso modelo depende diretamente da variação da temperatura e do calor latente, que são dois parâmetros que modificam as propriedades físicas dos objetos. Para induzir uma mudança de temperatura, utilizamos partículas de contorno, que possuem a temperatura como propriedade, mas nenhum movimento.

B.3 Esboçando objetos dinâmicos em realidade virtual imersiva

Creamos um novo aplicativo de esboço envolvente para criar objetos com diferentes materiais. Nossa solução permite interações com e entre objetos em tempo real usando XPBD. É possível criar esboços dinâmicos expressivos envolvendo diversos tipos de comportamentos físicos, como corpos sólidos rígidos, líquidos, gases, sólidos moles e roupas. Por exemplo, é possível alterar o fluxo de um rio adicionando ou cavando o solo enquanto o rio está fluindo, ou mesmo criar belas cachoeiras usando os mesmos métodos. O aplicativo também permite que o usuário veja em tempo real a interação de objetos flexíveis recentemente criados ou mesmo tecidos com o ambiente existente, incluindo outros objetos esboçados.

B.4 Conclusões

Os parâmetros de rigidez, distância, viscosidade, densidade e gravidade geralmente têm valores constantes durante a simulação; entretanto, variamos os seus valores dependendo da mudança de temperatura e do calor latente. Propomos um gerenciador sólido para criar e destruir restrições de distância durante a transição sólido-líquido (fusão e solidificação). Para alterar os valores dos outros parâmetros continuamente, definimos funções suavizadas derivadas das funções sigmoide e logaritmo, e isso torna as transições estáveis.

Também integramos os métodos de simulação dentro da realidade virtual para oferecer uma ferramenta de modelagem utilizando uma estrutura de grade que permite controlar a inserção e o apagamento de partículas, evitando sobreposições durante a simulação. Para isso, propomos um gerenciador de partículas para definir o comportamento em função do tipo de material em uso.

Demonstramos a versatilidade e estabilidade dos métodos interativos utilizados, alcançando resultados visuais plausíveis. Na ferramenta proposta, os usuários podem interagir diretamente com as simulações em tempo real. Ele permite-nos criar ferramentas poderosas como a replicação de fenômenos naturais e modelagem 3D em VR usando diferentes tipos de materiais, com um alto grau de imersão, salvaguardando o comportamento físico.