Vertex-Based Simulation of Dry Foam

Master Thesis

Micky Kelager
[micky.kelager@gmail.com]

DIKU
Department of Computer Science
University of Copenhagen, Denmark

September 14, 2009

Abstract

Foam dynamics is gaining interests within Computational Fluid Dynamics and Computer Graphics. We believe that liquid foam, or froth, is the next evolution in the field of fluid dynamics. Froth is the natural phenomenon of bubbles that arise due to nucleation of gas in liquids. As the physics of foams in two dimensions is better understood this thesis begins the journey of physics-based foam dynamics in Computer Graphics with two-dimensional dry foam. A dry foam is a foam with little or no volume fraction of liquid. From the field of Computational Physics we present the kinetics and dynamics of foams including the vital equilibrium rules that foams comply with.

We have implemented a vertex-based method for two-dimensional simulation of dry foam. By design the solver is a nonlinear quasi-static Newton method. We deal with several dynamic processes which include topological changes, gas diffusion, and shear deformation. For animation purposes the numerical model is well behaved and stable and can converge even if the foam is locally ill-defined.

This thesis introduces the Ghost Bubble method, a novel contribution to the vertex-based foam model that allows foam simulations with free surfaces and dynamic boundary conditions. A number of improvements to stabilize the numerical model are discussed and implemented and convergence rate analyses are performed against the improved dry foam solver which indicate superlinearity.

Resumé

Skumdynamik vinder interesse indenfor Computational Fluid Dynamics og Computer Graphics. Vi tror på at flydende skum bliver den næste evolution indenfor væskedynamik. Flydende skum er det naturlige boble-fænomen der opstår grundet kimdannelse af gas i væsker. På grund af fysikken bag skum i to dimensioner er lettere at forstå så vil dette speciale begynde rejsen indenfor fysik-baseret skumdynamik i Computer Graphics med todimensionelt tørt skum. Tørt skum er skum med lidt eller ingen væske i dets volumen. Fra forskningsområdet Computational Physics præsenterer vi den kinetik og dynamik som skum består af, og dette inkluderer de vigtige ligevægtsregler som skum opfylder.

Vi har implementeret en punktbaseret simuleringsmetode til todimensionelt tørt skum. Processen er en ikke-lineær kvasistatisk Newton metode. Vi beskæftiger os med en række dynamiske processer som inkluderer topologiske ændringer, gasdiffusion og deformation. Modellen opfører sig animationsmæssigt pænt og stabilt, og den kan konvergere selv i situationer hvor skummet lokalt ikke er korrekt defineret.

Dette speciale introducerer Ghost Bubble-metoden, hvilket er et nyt bidrag til den punktbaseret skummodel, der tillader skumsimuleringer med frie og afgrænsede kantoverflader. En række forbedringer til at stabilisere den numeriske skummodel bliver diskuteret og implementeret, og analyse af konvergensrater udføres på den forbedret model, som indikerer superlinearitet.

Contents

Preface **1**

1 Introduction **3**
1.1 Motivation . . . 3
1.2 Two-dimensional dry foam . . . 4
1.2.1 Absence of external forces . . . 6
1.3 Thesis goals . . . 6
1.4 Thesis overview . . . 7

2 History of Foams and Bubbles **8**
2.1 Previous work in Computer Graphics . . . 8
2.2 Previous work in Computational Physics . . . 10

3 Physics of Foams **12**
3.1 Description of Foam . . . 12
3.1.1 Dry foam . . . 14
3.1.2 Surface tension . . . 15
3.1.3 Foam statistics . . . 15
3.2 Foam Equilibria . . . 16
3.2.1 The law of Laplace-Young . . . 16
3.2.2 Radius of curvature . . . 17
3.2.3 The laws of Plateau . . . 17
3.3 Topological Processes . . . 18
3.3.1 The T1 process . . . 18
3.3.2 The T2 process . . . 19
3.4 Coarsening . . . 20
3.4.1 Fick's law . . . 20
3.4.2 Von Neumann's law . . . 21
3.5 Rheology . . . 22
3.5.1 Shear strain . . . 22
3.6 Foam Collapse . . . 22

3.6.1 Kinetics of film thinning . . . 24
3.6.2 Film rupture . . . 24
3.7 Summary . . . 24

4 Vertex-Based Foam Model **26**
4.1 Foam Model Constraints . . . 26
4.2 Areas and Angles . . . 28
4.2.1 The vertex angle . . . 30
4.2.2 The cell area . . . 31
4.3 The Vertex-Based Dry Foam Model . . . 32
4.3.1 Mathematical model . . . 32
4.3.2 Numerical model . . . 33
4.3.3 Jacobian derivatives . . . 35
4.3.4 Foam coarsening . . . 36
4.4 Fixed Boundaries . . . 36
4.5 Visualization . . . 37
4.6 Summary . . . 37

5 Improvements **38**
5.1 Dynamic Boundaries . . . 38
5.1.1 The ghost bubble . . . 39
5.1.2 Ghost bubble boundary conditions . . . 40
5.2 Foam Dynamics . . . 41
5.2.1 Foam collapsing . . . 41
5.2.2 Shearing . . . 42
5.3 Stability issues . . . 42
5.3.1 Damped line search method . . . 42
5.3.2 Invalid vertex positions . . . 43
5.3.3 Inverted angles . . . 44
5.3.4 Vertex fluctuation . . . 45
5.3.5 Topological edge flipping . . . 46
5.4 Nonlinear Newton Simulation Method . . . 47
5.4.1 Numerical derivative step size . . . 47
5.4.2 Newton simulation algorithm . . . 48
5.5 Summary . . . 49

6 Implementation **50**
6.1 Foam Mesh . . . 50
6.1.1 Data structure . . . 50
6.1.2 Limitations . . . 51
6.2 Initial Foam Conditions . . . 52

6.2.1 Voronoi diagrams and foam networks 54
6.3 Topological Changes 54
6.3.1 The T1 process 55
6.3.2 The T2 process 57
6.3.3 The three-sided double-bubble problem 58
6.4 Dynamic Boundary Conditions 59
6.4.1 Ghost area and pressure 60
6.4.2 Ghost angle and position 60
6.4.3 Ghost matrix system 61
6.5 Foam Rendering 62
6.5.1 Drawing arcs 62

7 Evaluation and Analysis **64**
7.1 Stability analysis 64
7.1.1 Initial Voronoi stability 65
7.1.2 Mature foam stability 65
7.2 Sensitivity discussions 66
7.2.1 Surface tension 67
7.2.2 Permeability and diffusivity 67
7.2.3 Topological thresholds 68
7.3 Convergence rate analysis 69
7.3.1 Deviation means 69
7.3.2 Error plots 70
7.4 Performance 73
7.4.1 Time consumption 73
7.4.2 Memory consumption 75
7.5 Visual Comparisons 78
7.5.1 Weaire and Kermode 78

8 Results **79**
8.1 Dynamic foam boundary 79
8.2 Dry foam maturing 81
8.3 Dynamic foam features 81
8.3.1 Foam evolution 83
8.3.2 Shear deformation 83
8.3.3 Forced bubble collapse 83
8.3.4 Pressure differences 88
8.4 Dry foam structure in real life 88

9 Future Work **90**
9.1 Ghost Bubble Method 90
9.2 Diffusion 91
9.3 Two-Dimensional Dry Foam 92
9.4 Two-Dimensional Wet Foam 93
9.4.1 Plateau borders 93
9.4.2 The decoration theory 94
9.5 External Dynamics 95
9.5.1 Gravity 95
9.5.2 Two-way interaction 96
9.6 Numerical solver 96

10 Conclusion **97**
10.1 Obtained Goals 97
10.2 Contributions 98

Bibliography **99**

A Supplementary Material **103**
A.1 Source Code 103
A.1.1 The OpenTissue library 103
A.2 Downloadable Contents 104
A.2.1 Demonstrative foam application 104
A.2.2 Motion sequences 104

List of Figures

1.1 Beer froth 4
1.2 Milk-soap froth 5
1.3 Single three-dimensional foam layer 6

3.1 Foam elements in two and three dimensions 13
3.2 Foam length scales 14
3.3 The T1 process 18
3.4 T1 and Plateau's rule 19
3.5 The T2 process 20
3.6 Stress-strain relation for a liquid foam 23
3.7 Yield stress depends on foam wetness 23

4.1 Variables in the equilibrium process 27
4.2 The law of Laplace-Young illustrated 28
4.3 The radius of curvature 29
4.4 The vertex angle 31

5.1 The ghost bubble 39
5.2 Invalid vertex positions 43
5.3 Inverted vertex angle 45

6.1 Two-sided bubble 52
6.2 Initial foam mesh configuration steps 53
6.3 The T1 change process 56
6.4 The T2 change process 57
6.5 The three-sided double-bubble problem 59
6.6 The ghost bubble in action 60
6.7 Drawing an arc 62

7.1 Relaxation stability graph 66
7.2 Vertex convergence rate 71
7.3 Foam convergence rate 72

7.4 Time performance graph 74
7.5 Memory consumption graph 76
7.6 Visual comparison of dry foam 77

8.1 Dynamic foam boundary 80
8.2 Dry foam maturing 82
8.3 Foam evolution of 10000 cells 85
8.4 Shear deformation 86
8.5 Forced bubble collapse 87
8.6 The Beijing Water Cube 88
8.7 Pressure differences in colors 89

9.1 Ghost bubble interface foam merge 91
9.2 Plateau borders in a wet foam 93
9.3 Wet foam example from a decorated dry foam 94

Preface

This thesis has been written by Micky Kelager, DIKU[1], in conformity with the requirements for the degree of Master of Science at the University of Copenhagen. Supervisor has been Assistant Professor Kenny Erleben, DIKU.

Contributions

This thesis introduces the Ghost Bubble method, a novel contribution to the two-dimensional vertex-based foam model that will allow dry foam simulations with free dynamic surfaces and finite boundaries. Traditionally, numerical vertex-based liquid foam models require periodic boundaries but with the addition of the Ghost Bubble we can simulate any finite boundary foam sample.

In this thesis we also derive an improved nonlinear Newton dry foam solver that converges even though the foam may be locally ill-defined. The numerical model presented in this thesis has less than 1% total error even though it employs a first order finite difference method to solve the partial differential equations of the system. We analyze and present the convergence rate of the nonlinear Newton method which is a scientific contribution for this type of problem.

We present several stability and feature improvements to the vertex-based foam model that enrich the animation quality. Some physics-based but less physics correct features have been added for Computer Graphics purposes. It is our hope that this thesis will merge Computational Physics and Computer Graphics on the topic of liquid foam to define an exciting new area of physics-based foam dynamics.

Prerequisites

The work presented in this thesis caters for people interested in physics-based two-dimensional foam simulations and the physics and mathematics behind foam dynamics. The reader should have a common understanding in the fields of physics-based simulation, computational geometry, and computer graphics as the thesis will employ terms from

[1] Datalogisk Institut, Københavns Universitet $\sim$ Department of Computer Science, University of Copenhagen (Denmark)

these research areas. The thesis will in particular reference topics from the field of Computational Fluid Dynamics.

It is not essential to be familiar with the physics of foams or computational foam dynamics as these topics will be defined and discussed in the thesis.

Acknowledgments

At first, I would like to thank my supervisor Kenny Erleben for the graceful guidance and hours spent on discussing the theory and practice of foam dynamics. Thank you for your patience and tenacity and for helping me completing this thesis.

Thanks must also go to the employees and students in the Image Group at DIKU and the eScience Center for administrative help, scientific discussions, and for providing a nice research environment. Thanks to Kim Steenstrup Pedersen for the useful feedback and insight in statistics.

Thanks to Theo Engell-Nielsen and Carl Johan Lejdfors for proofreading.

Personal thanks to Vivi, and my father, Finn, for granting me sanctuary the week before the important milestone. Special personal thanks to Litha for the provided support and in particular during the long period of writing.

Finally, I would like to express my severe gratitude to my wife Tanja and my lovely children Magnus and Maya. It has really been a tough ride and I thank you deeply for allowing this to happen. I love you!

Micky Kelager
- an elder of a dying breed

DIKU, September 14, 2009.

Chapter 1

Introduction

"I'd definitely go for a foamy froth than a frothy foam, Cheers!"

- Micky Kelager

Over the last decade several methods to simulate liquid fluids have been presented in Computer Graphics and these include [Stam, 1999, Losasso et al., 2004, Losasso et al., 2006, Selle et al., 2005]. A common focus for the models within Computational Fluid Dynamics (CFD) has been on realism and today it can be a challenge to distinguish between real and simulated fluids. We believe that the next evolution in realistic fluid simulation lies within the subject of liquid foams, or froths. Froth is a natural phenomenon of bubbles that arise due to nucleation of gas in liquid fluids, e.g. the layer of bubbles that occurs when a beer is poured into a glass as depicted in Figure 1.1.

1.1 Motivation

Compared to the work conducted on foams within Computational Physics the amount of work conducted on foam dynamics in Computer Graphics is limited. In our opinion the most promising result to date to create liquid foam is the Lagrangian fluid dynamics approach by [Cleary et al., 2007]. Even though this method delivers convincing images of beer froths the method has issues with foam motion and dynamic behaviors. We assume that the foam animation problems in Computer Graphics are due to that fact that foams are treated as fluids and not as real foams. Due to this we reckon that the theoretic foundation of foam dynamics must be obtained from Computational Physics. With the knowledge of how physicists are working with existing liquid foam simulation models we believe that we, as computer scientists, can provide new ideas and results to the field of foam dynamics in Computer Graphics.

Figure 1.1: Beer froth appears on top of the beer when it is poured into a glass.

1.2 Two-dimensional dry foam

The three-dimensional foam case as illustrated with the milk-soap froth example in Figure 1.2 has been a big desire to work with. However, at first sight it became clear that the topic of foams is quite large and as a consequence of this observation the focus in this thesis has been redirected to the subject of two-dimensional foams alone. Another reason for this restriction is due to the fact that physicists mostly agree on the processes of two-dimensional foams and how such foams are behaving. It is also more practicable to verify theory with practice in two dimensions even though it is possible to investigate two-dimensional slices of a three-dimensional foam. Finally, two-dimensional foams can be created in real life and as such is not only of a theoretical foundation. Liquid foams constrained between two glass plates and single layers of three-dimensional foams are all behaving and considered as two-dimensional foams where the latter situation is depicted in Figure 1.3.

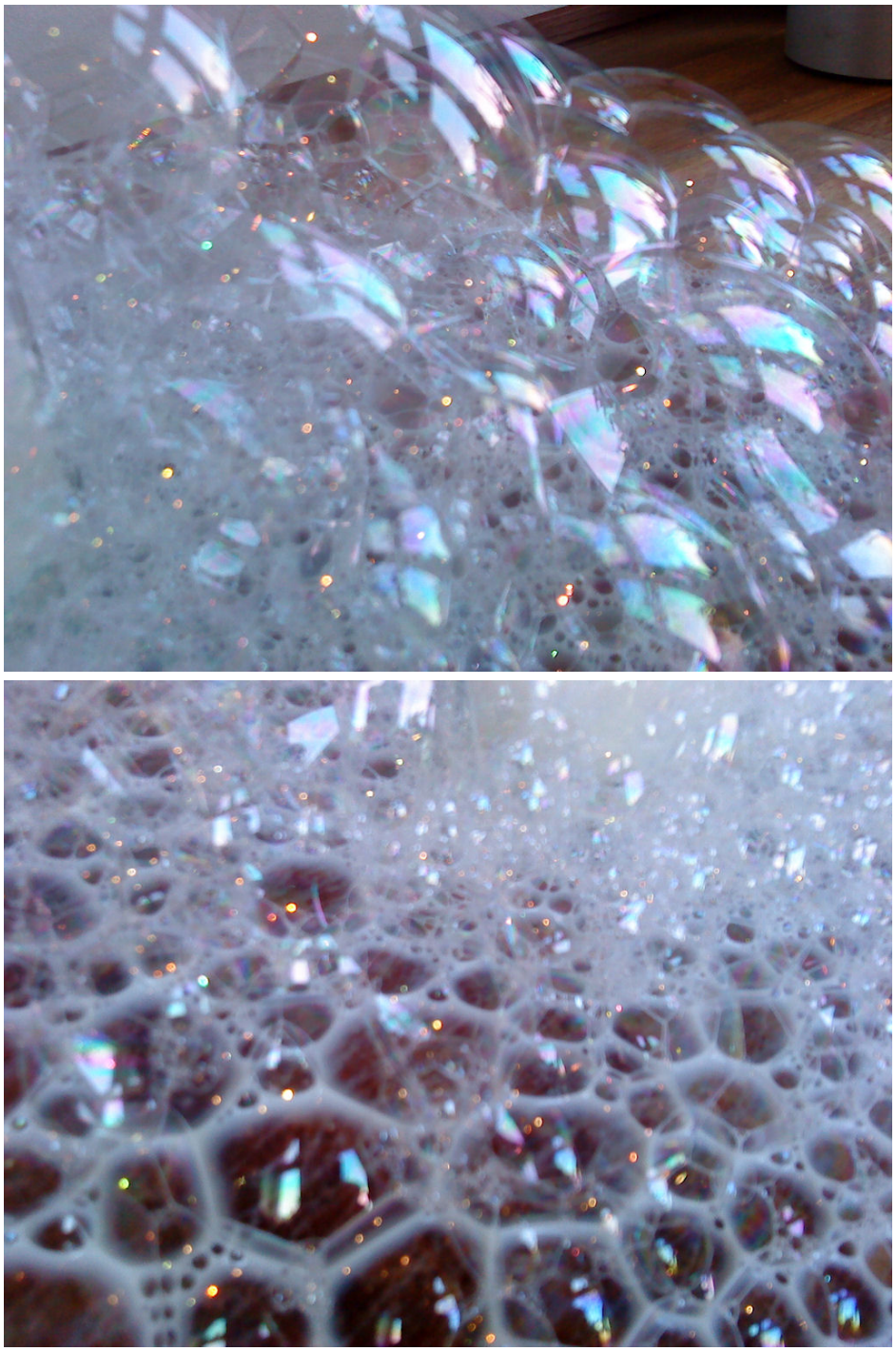


Figure 1.2: Milk-soap froth produced by blowing air through a straw into a milk-soap liquid mixture. In the lower photo the foundation of the foam structure is identical to that of a two-dimensional wet foam.

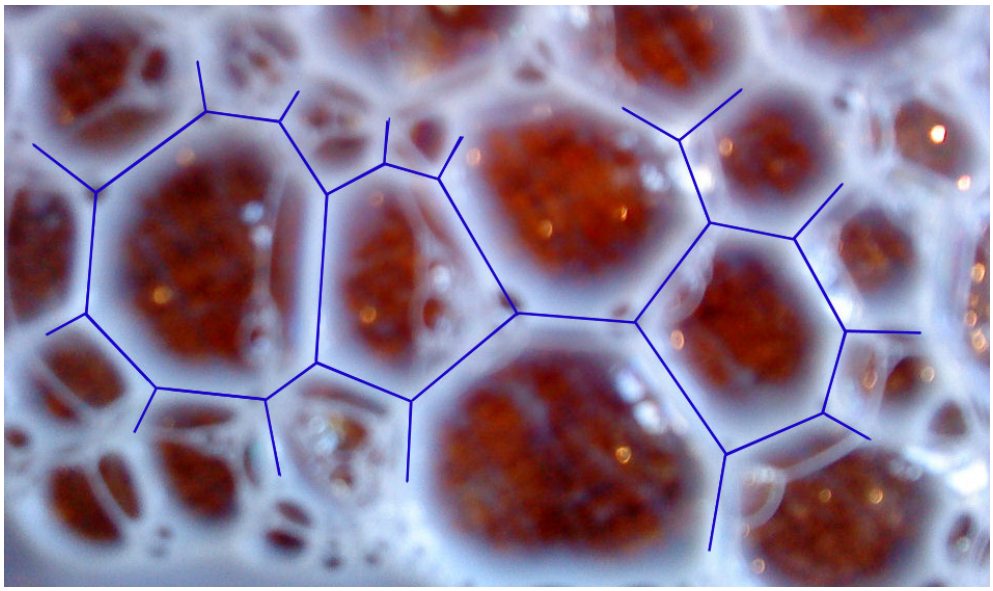


Figure 1.3: A single bubble layer from a three-dimensional liquid foam is behaving as a foam in two dimensions. Lines have been drawn on top of the wet foam to indicate the corresponding dry foam structure.

Additionally, only the physics of relatively dry, two-dimensional foams are well described by current models [Weaire et al., 2003] which is why this thesis specifically will focus on two-dimensional liquid, dry foams. It is somewhat contradictory to use the two terms *liquid* and *dry* about the same thing. Nonetheless, physicists distinguish between liquid and non-liquid foams. A liquid foam can be categorized from wet to dry, which depends on how much of its volume that actually is liquid.

1.2.1 Absence of external forces

We believe that it is important to state that this thesis will not handle external dynamics forces such as gravity and collision handling. However, we will treat the internal forces and dynamics of the dry foam, e.g. topological changes and gas diffusion, along with external interactive contributions within the limits of two-dimensional dry foam, e.g. shear deformation and bubble collapse.

1.3 Thesis goals

The purpose of this thesis is to learn and communicate the mathematically and physically foundations behind liquid, dry foams. With the gained insight in the physics of foams we seek to derive and implement a numerical model that can equilibrate and simulate the evolution of two-dimensional dry foams of various characteristics.

With the completion of this thesis we wish to have a better understanding of how liquid, dry foam behaves. This thesis should serve as a foundation for future work on liquid foam simulation in Computer Graphics.

1.4 Thesis overview

A brief overview of the rest of the thesis and the contents of the chapters are hereby presented.

2 History of Foams and Bubbles (p. 8) When conducting computer scientific work that includes multiple research fields it is a common approach to present the previous work done within the fields. This chapters delivers a brief overview about bubbles and foam simulations from Computer Graphics and Computational Physics.

3 Physics of Foams (p. 12) From Computational Physics we can get information about the physics of foams. In this chapter we discuss the kinetics and dynamics of foams that are required to understand and derive a dry foam model.

4 Vertex-Based Foam Model (p. 26) In this chapter we derive the mathematical and numerical models of a two-dimensional, vertex-based, dry foam simulator that will allow such model to be implemented.

5 Improvements (p. 38) Based on our implementation of the vertex-based dry foam model in this chapter we analyze the simulation behavior and discuss a number of suggestions to improve the numerical model.

6 Implementation (p. 50) This chapter is dedicated to practice and to our implementation of the two-dimensional dry foam simulator which also include the suggested improvements. Important details about the implementation are presented.

7 Evaluation and Analysis (p. 64) The improved numerical model is evaluated throughout this chapter. Attention is focused on convergence, stability, and performance and comparisons against other foam simulators are presented.

8 Results (p. 79) The results from our two-dimensional dry foam simulator are presented in this chapter. Still frames from actual dry foam simulations are depicted.

9 Future Work (p. 90) Additional work on liquid foams has begun and several new ideas have been developed. This chapter presents some of this work for the future to come and why we believe it is interesting.

10 Conclusion (p. 97) In this final chapter we conclude our work and recapitulates the obtained results in comparison with our initial goals and expectations.

A Supplementary Material (p. 103) A short appendix on how to obtain the additional materials as downloadable contents that accompany this thesis.

Chapter 2

History of Foams and Bubbles

"*The challenge of history is to recover the past and introduce it to the present.*"

- David Thelen

The topic of foams have been of interest within Computational Physics for a very long time and is now beginning to arouse interest within Computer Graphics as well. The focus and the results from the research fields are different in general. Computer Graphics has a greater interests in bubbles and foam simulations and how they are visualized while Computational Physics is more keen on trying to understand, describe, and measure all the different physics processes within a foam. The previous work of foams and bubbles within these fields in the past 30 years will briefly be presented in this chapter.

Many specific research oriented terms are used in this chapter. Still they are all common to Computer Graphics and Computational Physics. A vastly number of these terms will be described later in this thesis but the reader of this chapter is assumed to be familiar with the majority of the terms.

2.1 Previous work in Computer Graphics

In [Icart and Arquès, 1999] a single layer of foam bubbles, i.e. a two-dimensional foam, is modeled by iteratively adjusting the centers and radii of spherical bubbles to best fulfill a geometric measure that penalizes deviation from Plateau's laws. Film thickness is modeled from an empirical law such that interference patterns can be rendered. The method is best described as a geometric bubble-bubble interaction model that is solved using an iterative relaxation method.

In [Glassner, 2000a, Glassner, 2000b] an analytical method for bubble clusters of 2 and 3 bubbles is developed. An analytical solution to the placement and size of spherical bubbles is found from Plateau's laws. Hereafter constructive solid geometry (CSG)

operations are used to generate the geometry of the shared soap films and finally the bubble geometries are rendered using a shader that takes the Fresnel terms and interference patterns into account.

An explicit curvature-driven model is used in [Durikovic, 2001]. Bubble-bubble interactions are modeled using energy potential functions to penalize unwanted bubble configurations. A mass-spring model with explicit time integration is adopted. The modeling of surface tension is done using the Laplace-Young law to add "excess" pressure forces while at the same time pressure forces are added to model how bubbles repulse each other. Gas diffusion in the foam is not modeled. The computational mesh is fixed and no Plateau borders are modeled. In short bubbles are merely curvature driven sticky spherical objects. Results shown include scenes with up to 10 bubbles having clusters of up to 4-5 bubbles.

A bubble-bubble interaction method is presented in [Kück et al., 2002] that models bubbles as spheres with a particle system with no mass. Springs are added to model the bubble interactions. Bubble creation and film rupture are handled by adding and removing bubbles at random. No Plateau borders are modeled but instead special ray-tracing shaders are used to mimic the real foam structure. Results are shown with 700-3000 bubbles with simulation time from 4-28 seconds per frame.

A fluid animation technique is used in [Hong and Kim, 2003]. Here the volume-of-fluid method and the front-tracking method that have been developed in the field of Computational Fluid Dynamics are combined. Interfaces are directly adjusted to avoid numerical diffusion and yield mass conservation. The work addresses bubbles that are rising in liquid but does not deal with the problems of the thin shared films in bubble clusters. The method could be classified as using a hybrid Euler-Lagrangian representation of the bubbles in a liquid.

In [Iwasaki et al., 2004] a mass-spring system is used to set up a curvature-driven method similar to [Durikovic, 2001]. Film thickness is estimated from area weighted mass of the tessellated film surfaces. This yields a measure of how much the local mass is smeared out on the surrounding surface. No rupture, nor topological changes are modeled. Results include visualization of 1-2 bubbles. The main contribution in this work is real-time rendering method taking light interference into account.

In [Li and Volkov, 2006] a two-dimensional elastic model is extended with a pressure force model where pressure variation in external media is included to model buoyancy. This work describes how to add shape preserving bending forces and is related to bubble/-foam simulations in the sense that it describes a Lagrangian surface model with pressure forces.

In [Zheng et al., 2006] a continuous multiphase fluid simulator is used to handle the bubble dynamics. A novel regional level set method and a semi-implicit surface tension model is introduced. Bubble thickness is explicit modeled by a distribution function and the effect of drainage is included. Gas diffusion is still not modeled, nor is it very clear how to deal with boundary conditions on the velocity field. Results show 5-10 bubbles computed on grid sizes ranging from $30 \times 12 \times 15$ to $40 \times 20 \times 40$ in 2-20 seconds per frame for simulation.

For the first time in Computer Graphics we saw a virtual beer being poured into a

glass in [Cleary et al., 2007]. The method models liquid beer, bubbles in the liquid, and foam on top of the beer using the same Lagrangian model of smoothed particle hydrodynamics (SPH) which makes the interaction between the multiphase fluid very convincing. Gas diffusion is handled in an SPH-based manner. Bubble collapsing is handled similar to [Kück et al., 2002] except that a more probabilistic method is employed during collision handling of the bubbles of the foam. Due to the particle-based nature of the foam model it omits modeling the foam films and hense the importance of topological changes. Rheology and shearing of the bubbles are handled in a psudo-physical way by including explicit cohesive forces between the particles. However, the motion of the dry foam on top of the beer is too rigid and some of the realism is lost on this account.
The dynamic bubble models used in Computer Graphics are:

- Geometric methods (analytic and iterative relaxation-based) [Icart and Arquès, 1999, Glassner, 2000a, Glassner, 2000b].
- Lagrangian elastic models [Durikovic, 2001, Li and Volkov, 2006].
- Fluid simulator-based methods [Hong and Kim, 2003, Zheng et al., 2006].
- Bubble-bubble interaction models [Kück et al., 2002, Cleary et al., 2007].

Many methods in Computer Graphics are concerned with modeling film thickness such that the interference patterns can be shown during rendering. However, many of the dynamic processes in foam are ignored, e.g. gas diffusion, foam rupture, and topological changes. Lately, the work on bubbles and foams in Computer Graphics have begun integrating physically correct ideas [Cleary et al., 2007, Kim et al., 2007] which allow foams and bubble clusters to behave more convincingly.

2.2 Previous work in Computational Physics

In [Aboav, 1980] the attention is focused on the distribution of cells in a soap foam which is related in a smiliar way to the the grain growth in polycrystals. No actual simulations are performed in this work. Instead a real soap foam is constrained between two glass plates and photographed. The resulting images are employed to determine the various statistics about the foam distributions.

In [Kawasaki et al., 1989] the vertex model of two-dimensional grain growth is developed. The attention is primarily focused on distributions and statistics. What characterizes the vertex model is the approximation of the cell edges by straight lines for computational purposes. The vertex model was extended to three dimension in [Nagai et al., 1990] where the approximation of curved faces by spatial planar polygons makes it possible to simulate large numbers of cells.

In [Weaire and Kermode, 1983, Weaire and Kermode, 1984] a simulation method for two-dimensional dry soap foam is presented. This method models the real curvatures of the edges between vertices by the law of Laplace-Young and topological changes are

performed to the soap foam nerwork. The foundation of this thesis is based on the work by Weaire and Kermode.

In [Bolton and Weaire, 1991] a method to decorate a two-dimensional dry foam to create an equilibrated wet foam is treated. The work is extended to include Plateau borders in [Bolton and Weaire, 1992] for the two-dimensional dry foam model. The liquid foam model is the first of its kind to be implemented and simulated numerically.

In [Brakke, 1992] the Surface Evovler is presented which is a software solution that can minimize the energy of a surface subject to different constraints. The Surface Evovler can simulate two-dimensional grain growth with periodic boundary conditions but the real force of the software is to perform accurate computations of ordered three-dimensional foams.

In [Durian, 1995] a numerical bubble-bubble simulation model is used to investigate the deformation and flow of foam in the wet regime. In [Durian, 1997] the work is follow up to include the investigation of the rheological behavior paticularly from avalances.

In [Marsh et al., 1995] a curvature-driven vertex model for two-dimensional grain growth is presented which is an enhancement of [Kawasaki et al., 1989]. Vertex positions are updated at the same time accordingly to their velocities. Topological changes are performed if required, and the vertex velocities are re-computed to reflect the changes.

In [Weygand et al., 1999] the curvature-driven vertex model is extended to three-dimensional grain growth kinetics. It is found that two-dimansional sections of a three-dimensional simulation can be distinguished from a pure two-dimensional simulation.

In [Cox et al., 2006] the rhelogical flow of a two-dimensional foam is simulated and analysed. Lift and drag forces around a circular obstacle in the bulk of a foam is measured. The interactions between the foam and the obstacle is simulated using the Surface Evovler. The foam simualtion models used in Computational Physics are:

- Vertex models (typically for grain growth) [Kawasaki et al., 1989, Nagai et al., 1990].
- Curvature-driven dynamics model [Marsh et al., 1995, Weygand et al., 1999].
- Two-dimensional vertex-based liquid foam models [Weaire and Kermode, 1983, Weaire and Kermode, 1984, Bolton and Weaire, 1991, Bolton and Weaire, 1992].
- The Surface Evolver [Brakke, 1992, Cox et al., 2006].
- Bubble-bubble interaction models [Durian, 1995, Durian, 1997].

Common for the two-dimensional foam models in Computational Physics is the requirement of infinite networks which for numerical simulations imply periodic boundary conditions. Distribution functions and statistics are primary interests within Computational Physics where dynamical effects come secondary.

Chapter 3

Physics of Foams

"*It is wrong to think that the task of physics is to find out how Nature is. Physics concerns what we say about Nature.*"

- **Niels Bohr**

In this chapter we describe the physics of foams with focus on two-dimensional dry foams. If the reader is interested in the general topic of physics of foams she is urged to read [Weaire and Hutzler, 1999] which is a rich source about the general aspects of foams, including two-dimensional wet foams, three-dimensional foams, and some on magnetic and solid foams.

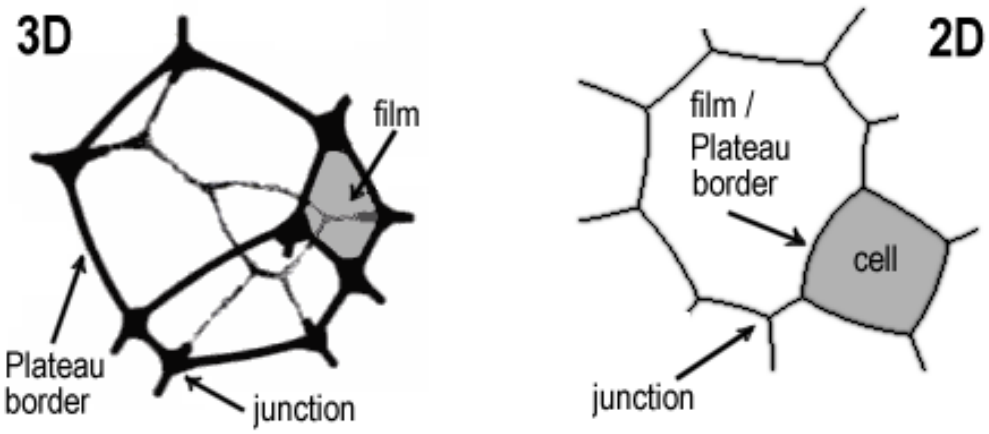
The chapter can be split up into two parts. The first part concerns the quantitative description and the structure of foam, where the last part concerns the dynamics of foam.

Part one - Foam structure:
3.1 Description of Foam (p. 12)
3.2 Foam Equilibria (p. 16)
3.3 Topological Processes (p. 18)

Part two - Foam dynamics:
3.4 Coarsening (p. 20)
3.5 Rheology (p. 22)
3.6 Foam Collapse (p. 22)

3.1 Description of Foam

A foam is a two-phase system in which bubbles of gas are enclosed by thin layers of a liquified substance. On a regular basis a foam is disordered which refers to the bubbles that are disordered and can vary greatly in size. Ordered foams usually do not occur in real life and is therefore a topic of theoretical research. Liquid-gas foam, or simply liquid foam, has a form that is categorized as *cellular* due to the fact that bubbles of gas form cells that are surrounded by liquid. Throughout the thesis we will be working with two-dimensional foam at a level where the foam can be structured into four elements.



(a) (b)

Figure 3.1: The different elements in the foam structure in (a) three dimensions, where a bubble (cell) from a wet foam is illustrated, and in (b) two dimensions (Part of (a) is taken from Figure 1.6 on page 6 from [Weaire and Hutzler, 1999]).

These elements have been illustrated in Figure 3.1 and include:

- Bubble (face or cell)
- Film (edge or side)
- Plateau border (border)
- Junction (vertex)

The terms in parenthesis are alternate terms of the same foam element. These alternatives will be employed a great deal through the thesis. In two-dimensional foams the Plateau borders coincide with the films as depicted on Figure 3.1(b).

This apparently complex two-phase material has a well-defined local structure which allows progress in predicting the dynamic properties of the foam. The different types of foam are in general being categorized into two parts. Below is a general definition of the two different types of foams:

Liquid - A liquid foam is a soft, highly deformable, substance that range from wet to dry dependent on how much of its volume is liquified. A special variant of liquid foam is *froth*. Froth usually designates the dense, wet foam that lies on top of a liquid, e.g. beer froth. By using a ferrofluid liquid forms can become ferromagnetic. A ferromagnetic froth evolves by the presence of a magnetic field, and react somewhat differently than a soap froth.

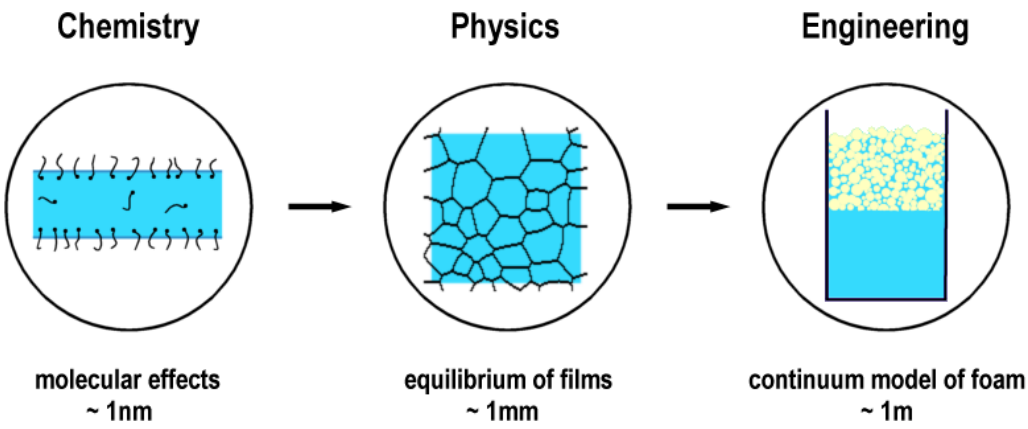


Figure 3.2: Different length scales for foam employed within different research areas. Computer Graphics is typically represented in the range from Physics to Engineering but the work in this thesis concentrates on foams defined on the length scale corresponding to the physics level, 0.1cm – 10cm (The illustration is a re-creation of Figure 1.15 on page 13 from [Weaire and Hutzler, 1999]).

Solid - A solid foam usually starts out as a liquid foam which will undergo a rapid solidification. Solid foams include metal foams, e.g. aluminum foam which is used for construction and polyurethane which is a polymer foam used for insulation purposes. Other types of solid foam include bread and natural sponges.

For the rest of this chapter we will only be concentrating on liquid foams with the same characteristics as soap froth.

Foam is a topic of research within several fields of the academia. The length-scale of foam range from microscopic to macroscopic as illustrated in Figure 3.2. Computer Graphics typically range from physics to engineering depending on the problem. The scope of this thesis is to work with foam on the physics level which corresponds to length-scales in the range of $1-100$mm and time-scales from seconds to tens of minutes.

3.1.1 Dry foam

The volume of a liquid foam always contains a fraction of liquid that primarily flows in the Plateau borders and junctions. The degree of wetness is expressed by the volume liquid fraction Φ_l of the foam or the gas fraction Φ_g. There is a linear dependency between the two fractions [Weaire and Hutzler, 1999] which is

$$\Phi_l = 1 - \Phi_g \,.$$

The volume of liquid in a foam categorizes the foam in the range of dry to wet where a dry foam is defined when the gas fraction is significant close to 100% ($\Phi_g \to 1$).

3.1.2 Surface tension

Surface tension is a property of a liquid surface. It is given by the symbol γ and it is has dimension force per unit length $\gamma = \frac{F}{l}$ [Weaire and Hutzler, 1999]. In the bulk of a liquid the attractive intermolecular forces will pull molecules equally apart. On the surface of the liquid the molecules will be pulled tangentially and inwards into the bulk. The missing liquid molecules beyond the surface are the main cause of the surface tension. Surface tension can also be interpreted as surface energy per unit area. The effect of surface tension will lower the surface energy, and thus minimize the surface area.

When *surfactants* are added to a liquid they will decrease the surface tension of the liquid. Surfactants are a contraction of *surface acting agents* and the main ingredient in commercial products like dishwasher detergents. A common misunderstanding about water and detergents is that it is the detergent itself that when added to water will increase the surface tension and thus allow bubbles to be created and prevented from rupture. Actually, the opposite is the case as it is the strong surface force caused by the surface tension that prevents normal water from creating stable foam bubbles[1]. Adding detergents to the water will decrease the surface tension and by that prevent the bubbles from being torn apart immediately.

An important contribution from surface tension to soap bubbles is that a bubble will always try to minimize the energy by reducing its surface area. The work δU it takes to create a new surface area δA is given by

$$\delta U = \gamma \delta A,$$

as the amount of molecules that is transferred from the bulk to the surface is proportional the the increase in surface area [Weaire and Hutzler, 1999].

When dealing with foam and bubbles the surface tension is sometimes used as the film tension σ which is twice the surface tension to account for the the two surfaces for the bubble film

$$\sigma = 2\gamma.$$

3.1.3 Foam statistics

In this subsection we will briefly present some commonly employed formulas in foam statistics. Foam statistics is heavily used in Computational Physics to classify and compare results from different foam samples.

In a cellular structure the numbers of cells C, faces F, and vertices V are linked by their topological structure and according to Euler's equation [Weaire and Hutzler, 1999] this relation yields

$$F - E + V = \chi, \tag{3.1}$$

where $\chi = 1$ for a two-dimensional plane.

[1] If you pour clean water into a container with a high flow velocity you will actually see bubbles that emerge. However, they are very unstable and last only for a fraction of a second.

In two dimensions a foam has distributions

$$p(A_b) = < \textit{cell area} >$$

and

$$p(n) = < \textit{number of edges} > ,$$

and for an infinite two-dimensional dry foam the average number of edges in all cells is precisely

$$\overline{n} = 6,$$

which also implies that on average a cell has six vertices due to (3.1).

The second moment of number of edges per bubble μ_2 is a good way to characterize different foam samples [Weaire and Hutzler, 1999] and is given by

$$\mu_2 = \sum_n (n - \overline{n})^2 p(n). \tag{3.2}$$

The mean cell area is

$$\overline{A}_b = \frac{1}{n} \sum_n A_n, \tag{3.3}$$

where the sum is over all cells, and likewise, the mean vertex angle is

$$\overline{\phi}_v = \frac{1}{n} \sum_n \phi_n, \tag{3.4}$$

where the sum is over all angles.

3.2 Foam Equilibria

Disregarding all dynamic processes this section presents the rules that can bring a foam into a quasi-static equilibrium.

3.2.1 The law of Laplace-Young

From Subsection 3.1.2 we know that a soap bubble will try the minimize the surface energy but the sole reason why a soap bubble will not shrink and decrease its surface area to zero is due to the surface tension from the two films is balanced against the excess pressure difference Δp between the surroundings and the inside of the bubble.

Assume that the bubble can shrink its initial radius r to radius $r - \delta r$ then the reduction in surface energy is given by [Weaire and Hutzler, 1999]

$$\delta U = 2\gamma 8\pi r \, \delta r, \tag{3.5}$$

and the work performed against the excess pressure then becomes

$$\delta U = \Delta p 4\pi r^2 \, \delta r. \tag{3.6}$$

Assuming that the bubble is in equilibrium we can equate (3.5) and (3.6) that yields

$$\begin{aligned} \Delta p 4\pi r^2\,\delta r &= 2\gamma 8\pi r\,\delta r, \\ \Delta p &= \frac{16\gamma\pi r\,\delta r}{4\pi r^2\,\delta r}, \\ \Delta p &= \frac{4\gamma}{r}, \end{aligned} \tag{3.7}$$

which is the law of Laplace and Young for the simple soap bubble.

The general form of the law of Laplace-Young for a given surface with H as the mean curvature is given by

$$\begin{aligned} \Delta p &= 2\gamma H \\ &= \gamma\left(\frac{1}{R_1}+\frac{1}{R_2}\right), \end{aligned} \tag{3.8}$$

where R_1 and R_2 are the principal radii of curvature for the surface. In two dimensions the local mean curvature for a dry foam becomes

$$\frac{1}{R_1}+\frac{1}{R_2}=\frac{2}{r}. \tag{3.9}$$

Inserting (3.9) into (3.8) yields the law of Laplace-Young for a dry foam in two dimensions

$$\Delta p = \frac{2\gamma}{r}, \tag{3.10}$$

with $\Delta p = (p_i - p_j)$ as the difference between two adjacent cells i and j.

3.2.2 Radius of curvature

The local radius of curvature of a foam film in two dimensions defines a circular arc between the two vertices, and is trivially given by the law of Laplace-Young (3.10) as

$$r = \frac{2\gamma}{\Delta p}. \tag{3.11}$$

If the pressures are identical the the radius becomes infinite, i.e. $\lim_{\Delta p\to 0} r = \infty$, and the circular arc of the film simply becomes a straight line.

3.2.3 The laws of Plateau

Plateau added a set of additional rules to the law of Laplace-Young. These rules are vital for foam equilibrium and to some extent sufficient to define an equilibrium configuration. The laws of Plateau include the following equilibrium rules [Weaire and Hutzler, 1999]:

A1 *For a dry foam the films can intersect only three at a time, and must do so with an angle of $\phi = 120°$. In two dimensions that applies to the lines which define the cell boundaries.*

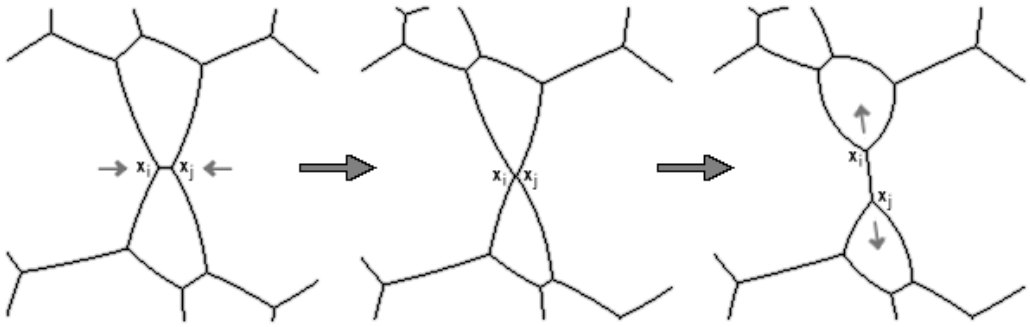


Figure 3.3: The T1 process on a dry foam. Upon approaching an unstable four-fold vertex, it will split into two stable three-fold vertices.

A2 *Again for dry foams we may assert that at the vertices of the structure no more than four of the intersection lines (or six of the surface) may meet, and this tetrahedral vertex is perfectly symmetric. Its angles all have the value* $\phi = cos^{-1}(-1/3) \approx 109.47°$*, sometimes called the Maraldi angle.*

B *Where a Plateau border joins an adjacent film the surface is joined smoothly, that is, the surface normal is the same on both sides of the intersection.*

With respect to the topic of this thesis equilibration rule **A1** is the only one of real interest. Equilibrium rule **A2** is only valid for tree-dimensional foams, where **B** is for wet foams explicitly.

3.3 Topological Processes

During equilibration and evolution of a dry foam the network will undergo different structural changes. These changes can be categorized into two topological processes that in the physics literature are referenced to as the T1 and T2 process [Weaire and Hutzler, 1999]. It has been observed that other topological restructurings in a foam are combinations of one or more T1 and T2 operations.

3.3.1 The T1 process

In the two-dimensional T1 process the topological change consists of exchanging, or flipping, the films in question as depicted in Figure 3.3. When two threefold vertices are moving towards each other the distance between the vertices becomes zero and the foam structure will be in an unstable configuration. In reality the two threefold vertices will merge and become one fourfold vertex which is unstable and quickly will split into two new threefold vertices. It is necessary to monitor the length of the distance between the

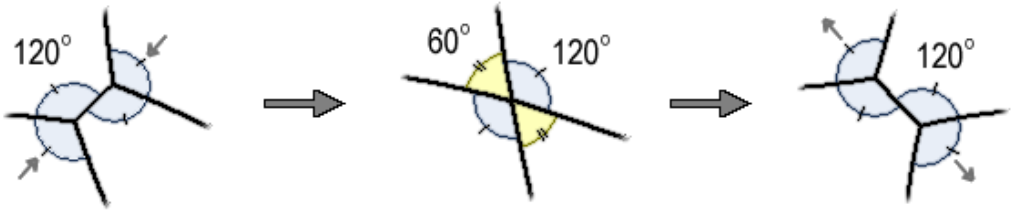


Figure 3.4: After the film-flip from a T1 operation the vertices are bound to move apart due to Plateau's 120°-rule.

vertices and when the distance become close to zero, or less than some threshold T_{dist}, the T1 process will occur. We set the condition for the T1 process to occur by the following

$$\mathbf{d} \cdot \mathbf{d} < {T_{\text{dist}}}^2 , \quad T_{\text{dist}} > 0, \tag{3.12}$$

where $\mathbf{d} = (\mathbf{x}_i - \mathbf{x}_j)$ is the displacement vector between the two threefold vertices of the film. Generally, the length of the film is larger than $\|\mathbf{d}\|$ as the film is curved due to Laplace-Young (3.10) but we will argue that the T_{dist} can be chosen to be sufficient close to zero that this difference in length can be neglected.

The default film flipping strategy for the T1 process is simply to place the film vertices perpendicular to the straight line $\mathbf{d}$ exactly when the T1 condition (3.12) is satisfied. If we assume that the two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ meet the 120°-rule of Plateau then due to the geometrical properties of the T1 operation the vertices will be forced to move apart which is illustrated in Figure 3.4. When the two vertices oppose each other all six vertex angles are approximately 120° each. Just before the T1 operation the distance between the vertices is close to zero and thus the angles between the opposing edges are approximately 60° each. As the T1 process flips the edge to be perpendicular to the edge before the T1 the two vertices are forced to move apart before before reaching equilibration.

3.3.2 The T2 process

The topological change in a T2 process consists of removing a three-sided cell as depicted in Figure 3.5. The T2 process occurs due to a gas diffusion process that will transport gas from the smaller cells to their larger adjacent ones. In the T2 process a three-sided foam cell will wither away and collapse into a single vertex. The position for the vertex that is created in the wake of the removal of a cell is set to be the average of the three equilibrated cell vertices. T2 processes take place during coarsening from diffusion of gas through the films. It is necessary to monitor the cell areas and perform a T2 operation once a cell area has become less or equal than some area threshold T_{area}. The T2 condition is given as

$$n_i = 3 \wedge A_i \leq T_{\text{area}} , \quad T_{\text{area}} \geq 0, \tag{3.13}$$

where n_i designates the number of films, or edges, of cell i and A_i is the current area of the cell.

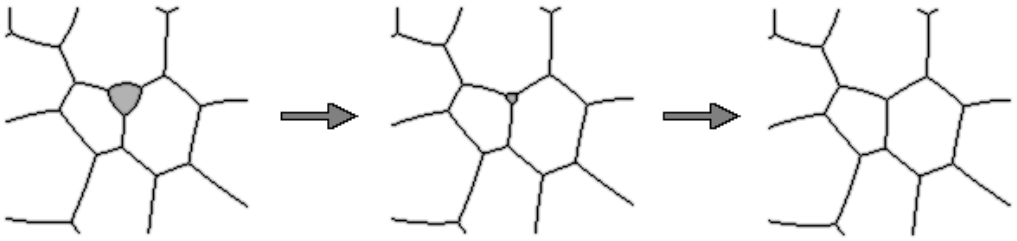


Figure 3.5: The T2 process for a dry foam. A three-sided cell that approaches zero area, due to gas diffusion, will perish.

It is not only three-sided cells that can perish. It has been observed that cells with more than three films have vanished [Weaire and Hutzler, 1999]. Such cells indeed collapse into a single vertex but it is not caused by the T2 process alone. A number of T1 operations are flipping the films on the vanishing cell until a three-sided cell is the outcome and then the normal T2 process finishes the process.

3.4 Coarsening

The process of incompressible gas that transfers through the thin films in a foam is called *diffusion*. Diffusion is driven by the pressure differences between adjacent cells across their common boundaries. Some cells will grow at the expense of others but in the end all cells will perish, due to the T2 process or film rupture which is described in Section 3.6. This evolution of a foam is called *coarsening* [Weaire and Hutzler, 1999]. The most common applied model for coarsening in an ideal two-dimensional dry foam is known as *von Neumann's law*.

3.4.1 Fick's law

Fick's first law [Zaikov et al., 1988] in n-dimensions takes the general appearance of

$$J = -D\nabla\psi, \tag{3.14}$$

where J is the *diffusion flux*, D is the diffusion coefficient or *diffusivity* of the concentration ψ. It states that diffusion will take place as long as there is a concentration gradient and additionally it states that diffusion moves matter from higher concentrations towards lower concentrations. The flux measures the amount of substance that will flow through a small area during a small time interval.

In cellular applications Fick's first law designates the rate of change in volume of a cell due to diffusion and is defined as

$$\frac{\partial V}{\partial t} = -\kappa A \Delta p, \tag{3.15}$$

where κ is the permeability constant of the membranes in the concentration, A is the surface area over which the diffusion is taking place, and Δp is the difference in pressure of the gas across the surface area. In two dimensions the permeability is the film conductance, or the ability for a gas to penetrate the thin liquid films.

3.4.2 Von Neumann's law

For a dry foam in two dimensions an important sum rule on the edges in every cell can be defined [Weaire and Hutzler, 1999]. For a cell with n edges where each edge tangent turns by $\frac{\pi}{3}$ on account of Plateau's equilibrium rules and each edge i has the length l_i and curvature r_i^{-1} we get

$$\sum_i \frac{l_i}{r_i} + n\frac{\pi}{3} = 2\pi,$$

which leads to the sum rule

$$\sum_i \frac{l_i}{r_i} = 2\pi\left(1 - \frac{n}{6}\right). \tag{3.16}$$

In two dimensions Fick's law (3.15) becomes the rate of change in area $\frac{\mathrm{d}A}{\mathrm{d}t}$ of a cell which in mathematical terms becomes

$$\frac{\mathrm{d}A_n}{\mathrm{d}t} = -\kappa\sum_j (p_n - p_j) l_j, \tag{3.17}$$

where the sum is over all the adjacent cells of the n-sided cell, l_j is the length of film j, and p_n, p_j are the pressures of the cell in question and the adjacent cell j, respectively. When inserting the law of Laplace-Young (3.10) into (3.17) we get

$$\begin{aligned}\frac{\mathrm{d}A_n}{\mathrm{d}t} &= -\kappa\sum_j \frac{2\gamma}{r_j} l_j \\ &= -2\gamma\kappa\sum_j \frac{l_j}{r_j},\end{aligned} \tag{3.18}$$

which by making use of the sum rule (3.16) yields

$$\begin{aligned}\frac{\mathrm{d}A_n}{\mathrm{d}t} &= -2\gamma\kappa 2\pi\left(1 - \frac{n}{6}\right) \\ &= -\frac{2\pi}{3}\gamma\kappa(6-n) \\ &= \frac{2\pi}{3}\gamma\kappa(n-6),\end{aligned} \tag{3.19}$$

which is known as *von Neumann's law*. This result shows that cells with more than six sides steadily enlarge while cells with less than six sides steadily decrease in area. One important aspect to note is that cells with six sides which is the mean number of sides, will not conduct to any diffusion process.

3.5 Rheology

Rheology is the study of how matter flows. Usually rheologists concern themselves with liquid matters, but large deformations in soft solids which reaches beyond elastic deformations are also considered rheological processes. Liquid foams are highly deformable and by their properties an ideal topic of rheology. Under certain conditions the same liquid foam can act as both a solid and a liquid. The phase transformations depend on the applied stress and the liquid fraction of the foam which is illustrated in Figure 3.6 and Figure 3.7 respectively.

For a dry foam low applied stress will turn it into an elastic solid with a well defined shear elastic modulus [Weaire and Hutzler, 1999]. At higher applied stress, yet still below the yield stress, the foam enters the plastic regime and topological rearrangements will occur. Beyond a certain yield stress the liquid foam will flow, like liquid fluid, as topological changes take place indefinitely.

Wet foams are going through the same phase transformations but the regime separations will happen at much lesser stress levels than for dry foams and hence the yield stress is also much lower for wet foams.

3.5.1 Shear strain

Two different types of linear shear can be applied to a liquid foam that will result in a linear stress-strain response [Weaire and Hutzler, 1999]. In two dimensions the simple, or normal, shear is defined by

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} 1 & \varepsilon \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3.20}$$

and extensional shear is defined by

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} (1+\varepsilon) & 0 \\ 0 & (1+\varepsilon)^{-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{3.21}$$

Each type of shear is defined by a direction and a strain parameter ε and shearing is carried out by modifying the dimensions x and y of the foam target. The 2×2 shear matrix in (3.21) has the property of conserving the foam target area which defines the equation as a two-dimensional shear deformation.

3.6 Foam Collapse

Liquid foams usually do not last very long. Over time they collapse due to the rupture of exposed films. A foam usually collapse from the outer regions and continue inwards but film ruptures do occur in the bulk.

The topological T2 process that was described in Subsection 3.3.2 (p. 19) is a special case of how a bubble will collapse. Foam collapsing can be caused by a number of phenomena but film thinning is one of the primary contributions to the process.

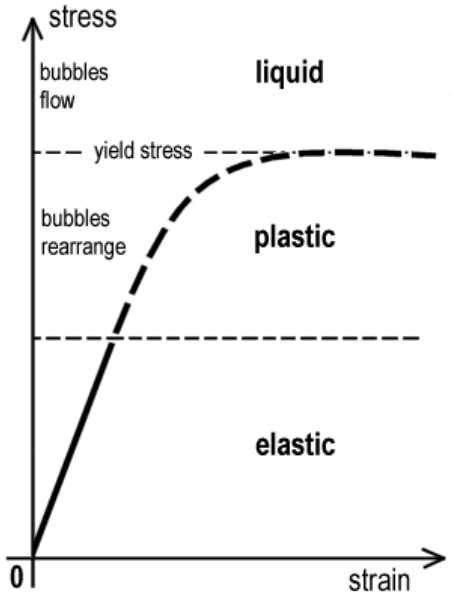


Figure 3.6: The stress-strain relation for a liquid foam (The illustration is a re-creation of Figure 1.12 on page 10 from [Weaire and Hutzler, 1999]).

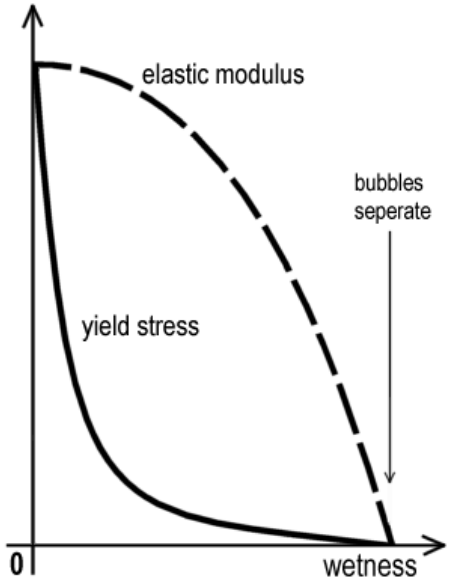


Figure 3.7: The yield stress of a liquid foam heavily depends on the wetness, or liquid fraction, of the foam. (The illustration is a re-creation of Figure 1.13 on page 11 from [Weaire and Hutzler, 1999]).

3.6.1 Kinetics of film thinning

The lifetime $\Delta\tau$ of thinning foam films can be estimated by

$$\Delta\tau = \int_{h_0}^{h_{cr}} \frac{dh}{v},$$

where h_0 is the initial thickness, h_{cr} is the critical thickness at witch the film ruptures, and v is the rate of film thinning [Exerowa and Kruglyakov, 1998].

Empirically, the relation between film thickness and time of film thinning can be described by

$$h = h_0 \exp(-\alpha\tau), \tag{3.22}$$

where h_0 is the initial thickness at which the time τ recording begins and α is an experimentally determined constant which depends on viscosity, pressure, and film radius.

Equation (3.22) gives a simple relation for the rate of film thinning and a linear dependence of film thickness on film lifetime. When the foam films become too thin the surface tension becomes too strong and will tear the films apart.

3.6.2 Film rupture

The films of a liquid foam will thin over time. For a wet foam a part of this thinning is due to drainage. Drainage is the process of liquid that flows through the Plateau borders due to gravity. Liquid that drains out of the foam is part of the wet foam equilibrium process which turns the wet foam into a dry foam. This thinning of foam films will ultimately lead to their rupture. However, drainage and film thinning are not alone to be responsible for the film rupture. Thin films can rupture due to dust or other environmental particles not to mention the total bubble and foam annihilation due to collision impact with obstacles or human interactions.

3.7 Summary

In this chapter the following important topics and results have been discussed:

- The main focus of this thesis rests on two-dimensional liquid, dry foam, such as soap froth.
- The second moment of number of edges per bubble is defined by

$$\mu_2 = \sum_n (n-\overline{n})^2 p(n).$$

- The local radius of curvature is immediately given from the law of Laplace-Young as

$$r = \frac{2\gamma}{\Delta p}.$$

- From the laws of Plateau we know that for a two-dimensional dry foam to be in equilibrium the films must intersect three at a time each at an angle of 120°.
- Two topological processes can occur in a foam during equilibration. These are the T1 process that corresponds to an edge flip in the foam network and the T2 process that reduces a three-sided cell into a single vertex.
- A two-dimensional dry foam can get into quasi-static equilibrium if it is continuously relaxed by applying the items above.
- Diffusion take place only through films of adjacent cells and conforms to the law of von Neumann (3.19) which is defined as

$$\frac{\mathrm{d}A_n}{\mathrm{d}t} = K(n-6)\,.$$

- Linear shear strain can be applied at low rates to a two-dimensional dry foam. Two types of shearing will result in a linear stress-strain response and these are the normal shear,

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} 1 & \varepsilon \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

  and the extensional shear,

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} (1+\varepsilon) & 0 \\ 0 & (1+\varepsilon)^{-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

- Over time foam film will undergo thinning and rupture. The foam will collapse in the outer regions and continue inwards.

Chapter 4

Vertex-Based Foam Model

"*If you can't explain it simply,*
you don't understand it well enough."
- Albert Einstein

In two dimensions the simplicity of the dry foam structure can be utilized to construct a mathematical model that can be simulated numerically to find a state of equilibrium. In this chapter we will derive the mathematical and numerical models for a two-dimensional dry foam simulator.

4.1 Foam Model Constraints

From a geometrical point of view only arcs of circles are required to define and represent the foam. From a physical point of view we know that pressure is associated with a cell and from the law of Laplace-Young (3.10) that the pressure difference between adjacent cells relates to the curvature of the arcs. From a computer scientific point of view we can use the vertex coordinates and the pressures to mathematically model functions A and ϕ that determine the cell area and the vertex angle

$$A(\mathbf{p},\mathbf{r}) \equiv A(p_i, p_j, p_k, x, y)$$

and

$$\phi(\mathbf{p},\mathbf{r}) \equiv \phi(p_i, p_j, p_k, x, y),$$

where $\mathbf{p}$ represents a vector that contains the three vertex cell adjacent pressures in counter clockwise order and $\mathbf{r}$ is the coordinates (x,y) of the vertex. In Figure 4.1 the variables that are included in the equilibrium process are drawn on to a selection of a dry foam network.

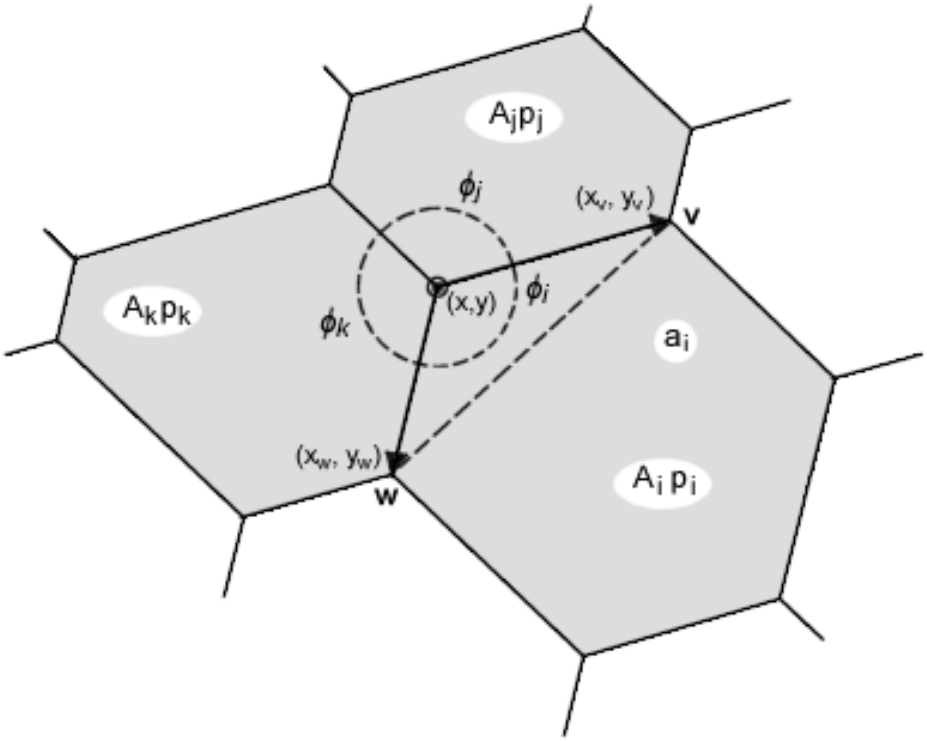


Figure 4.1: The different variables that are involved solving the equilibrium process.

When developing a foam system for the state of equilibrium we assume that viscous forces and inertial effects are negligible and that the gas that is enclosed by the cells is treated as incompressible, such that the foam network will minimize the surface energy of the soap film at all times [Weaire and Kermode, 1983].

From Subsection 3.2.3 (p. 17) we know that the 120°-rule from Plateau states that for a two-dimensional dry foam to be in equilibrium each vertex must be intersected by three and exactly three films and that the intersected films must meet exactly at $\phi = 120^\circ = \frac{2\pi}{3}$ radians. The vertex angles can be considered as constraints in the mathematical system and is defined as

$$C_\phi(\phi) = \frac{2\pi}{3} - \phi = 0, \quad \text{for all angles}. \tag{4.1}$$

The cell areas are likewise considered as constraints in the mathematical system. To solve the area constraints each cell will have a prescribed area which we call the *target area* A^{target} to fulfill the constraints

$$C_A(A) = A^{target} - A = 0, \quad \text{for all areas}. \tag{4.2}$$

A foam network can be brought to a state of equilibrium by solving the constraints defined by (4.1) and (4.2) but to do so we must be able to compute the cell areas and the vertex angles.

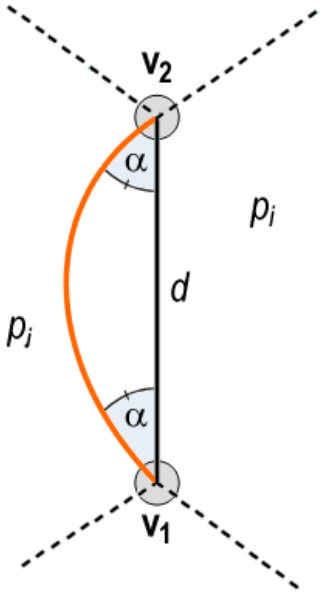


Figure 4.2: Illustration of the law of Laplace-Young. Here we see the curvature of an edge between two adjacent cells. The shared edge d is running between vertices $\mathbf{v}_1$ and $\mathbf{v}_2$.

4.2 Areas and Angles

As we work with a geometrical model the full vertex angles not only depend on the radii of curvature (3.11) but also on the Euclidean distances between film vertices.

In static equilibrium the law of Laplace-Young dictates the curved shape of the film geometry which is illustrated in Figure 4.2. Figure 4.3 shows the parameters we will use as an example in how to compute the required values. We seek to compute the angle α and the area A_A bounded by the arc region **A**.

The distance length d between the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ is determined by

$$\begin{aligned} d &= \|\mathbf{v}_2 - \mathbf{v}_1\| \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \end{aligned} \tag{4.3}$$

We assume that the curvature center $\mathbf{v}_c$ has been placed such that the line from $\mathbf{v}_c$ perpendicular to $(\mathbf{v}_2 - \mathbf{v}_1)$ splits the triangle **T** in two equal sized areas. From trigonometry we get

$$\sin\left(\frac{\theta}{2}\right) = \frac{d}{2}\frac{1}{|r|},$$

where r is the signed radius of curvature given by Equation (3.11). The angle θ thus yields

$$\theta = 2\sin^{-1}\left(\frac{d}{2|r|}\right). \tag{4.4}$$

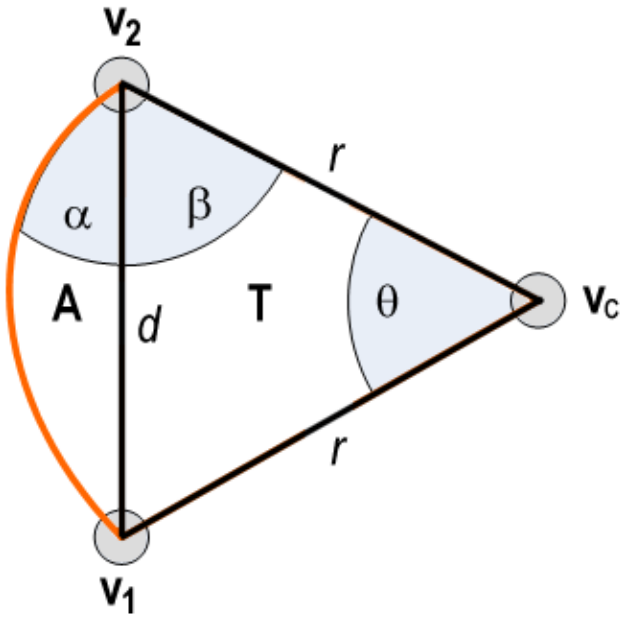


Figure 4.3: We introduce the virtual auxiliary point $\mathbf{v}_c$. This vertex is the center of the curvature circle and is not part of the froth network. The task is to compute the angle α and the area A_A of the arc region **A**.

Given the properties of $\mathbf{v}_c$ we know that the sum of the angles of the triangle **T** is

$$\pi = 2\beta + \theta\,, \tag{4.5}$$

and the angle β then yields

$$\beta = \frac{\pi - \theta}{2}\,. \tag{4.6}$$

The vectors going from the vertices to the center of curvature $(\mathbf{v}_1 - \mathbf{v}_c)$ and $(\mathbf{v}_2 - \mathbf{v}_c)$ respectively must all be normal to the circular arc due to

$$|r| \equiv \|\mathbf{v}_1 - \mathbf{v}_c\| = \|\mathbf{v}_2 - \mathbf{v}_c\|\,,$$

and from this observation it is given that

$$\frac{\pi}{2} = \alpha + \beta\,. \tag{4.7}$$

Inserting (4.6) into (4.7) we can now find the angle α as

$$\begin{aligned}\alpha &= \frac{\pi}{2} - \frac{\pi - \theta}{2} \\ &= \frac{\theta}{2}\,.\end{aligned} \tag{4.8}$$

The area of a full circle is $A_\circ = \pi r^2$ and hence the area of the pizza slice in Figure 4.3 is

$$A_\mathrm{P} = \frac{1}{2} r^2 \theta ,$$

and the area of the triangle **T** is conveniently written as

$$A_\mathrm{T} = \frac{1}{2} r^2 \sin\, \theta .$$

Finally, the area of the arc region **A** can be computed as

$$\begin{aligned} A_\mathrm{A} &= A_\mathrm{P} - A_\mathrm{T} \\ &= \frac{1}{2} r^2 \theta - \frac{1}{2} r^2 \sin\, \theta \\ &= \frac{r^2}{2} (\theta - \sin\, \theta) . \end{aligned} \tag{4.9}$$

Note that the equations in this subsection are all computed with respect to a specific cell. As r is the signed radius it is important that the sign of r has an influence on the computations of α and A_A for future processing. The arc angle (4.8) and arc area (4.9) are therefore changed to include the sign of the radius,

$$\alpha = \mathrm{sgn}(r) \frac{\theta}{2} , \tag{4.10}$$

and

$$A_\mathrm{A} = \mathrm{sgn}(r) \frac{r^2}{2} (\theta - \sin\, \theta) . \tag{4.11}$$

4.2.1 The vertex angle

Given a vertex $\mathbf{v}$ and its three adjacent vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ in counter clockwise order as illustrated Figure 4.4. Without any loss of generality let us assume that we seek to find the vertex angle ϕ_i. The two other vertex angles can be found from cyclic permutations of the equations we derive below.

The angle θ_i between the two vectors $(\mathbf{v}_1 - \mathbf{v})$ and $(\mathbf{v}_2 - \mathbf{v})$ can be computed by the means of the dot product as

$$\theta_i = \cos^{-1} \left(\frac{(\mathbf{v}_1 - \mathbf{v})(\mathbf{v}_2 - \mathbf{v}_j)}{\|\mathbf{v}_1 - \mathbf{v}\| \|\mathbf{v}_2 - \mathbf{v}\|} \right) .$$

We also need the two arc angles due to the pressure difference between the cells. Employing Equation (4.10) the two arc angles yields

$$\alpha_i = \mathrm{sgn}(r_i) \sin^{-1} \left(\frac{\|\mathbf{v}_1 - \mathbf{v}\|}{2\,|r_i|} \right) ,$$

$$\alpha_j = \mathrm{sgn}(r_j) \sin^{-1} \left(\frac{\|\mathbf{v}_2 - \mathbf{v}\|}{2\,|r_j|} \right) ,$$

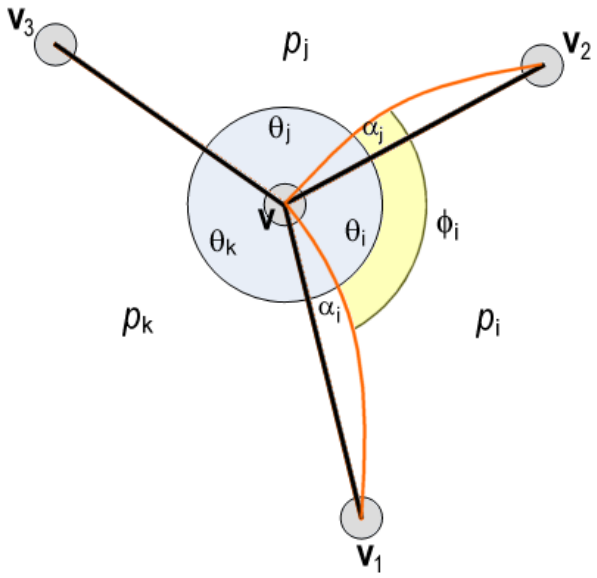


Figure 4.4: A vertex **v** has three adjacent vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. Note that subscripts on pressure and angle terms denote the respective cell. We seek to find the vertex angle ϕ_i.

with

$$r_i = \frac{2\gamma}{p_i - p_k} \quad \text{and} \quad r_j = \frac{2\gamma}{p_i - p_j}.$$

The full vertex angle is now the sum of all three angles which becomes

$$\phi_i = \theta_i + \alpha_i + \alpha_j. \tag{4.12}$$

4.2.2 The cell area

The total cell area is the sum of its signed arc areas and the main polygon. We need the area of the polygonal part and we need the signed area contribution from the circular arc regions around the polygon.

From linear algebra the two-dimensional determinant can be used to compute the signed area of the planar parallelogram spanned by two vectors. The determinant is defined as

$$\begin{aligned} \det(\mathbf{v}, \mathbf{w}) &\equiv \det \begin{bmatrix} \mathbf{v}_x & \mathbf{v}_y \\ \mathbf{w}_x & \mathbf{w}_y \end{bmatrix} \\ &\equiv \mathbf{v}_x \mathbf{w}_y - \mathbf{w}_x \mathbf{v}_y. \end{aligned} \tag{4.13}$$

The area will be positive if the rotation of **v** into **w** is a positive right-handed orientation. Employing the determinant the signed area of a triangle is $A_\triangle = \frac{1}{2}\det(\mathbf{p}_1 - \mathbf{p}_0, \mathbf{p}_2 - \mathbf{p}_0)$. If the points are provided in counter-clockwise order then the area is positive and if the points are provided on clockwise order the area will be negative.

In the following exposition we will focus at a cell i. For convenience we define the set of vertex indices of cell i in counter-clock-wise order as $I(i) = \{0,1,2,\ldots\}$. Let the number of indices be given by $K = |I(i)|$. The edge of film k of the cell $(\mathbf{v}_k, \mathbf{v}_{k+1})$ is a common edge between the cell and another cell which we will conveniently label with the index k. The signed area A_A of the region bound by edge k and the arc is given by (4.11) as

$$A_\mathrm{A}(k) = \operatorname{sgn}(r_k)\,\frac{r_k^2}{2}\left(\theta_k - \sin\,\theta_k\right),$$

with

$$\theta_k = 2\sin^{-1}\left(\frac{\|\mathbf{v}_{k+1} - \mathbf{v}_k\|}{2\,|r_k|}\right) \quad \text{and} \quad r_k = \frac{2\gamma}{p_i - p_k}.$$

The total area A_i of cell i now yields

$$A_i = \sum_{k=0}^{K-1} A_\mathrm{A}(k) + \frac{1}{2}\sum_{k=0}^{K-3} \det\left(\mathbf{v}_{k+1} - \mathbf{v}_0, \mathbf{v}_{k+2} - \mathbf{v}_0\right).$$

4.3 The Vertex-Based Dry Foam Model

We are now ready to derive the vertex-based foam model for the two-dimensional dry foam simulator. The model is a quasi-static method that can drive a two-dimensional dry foam network into equilibrium. A quasi-static method is a method that remains arbitrarily close to equilibrium at all times. Foam dynamics effects are simply added to the foam model by changing the state of the foam system, e.g. target area or vertex position such that the foam network deviates from an equilibrium. The purpose of the quasi-static simulation model is to bring the foam network back into equilibrium.

4.3.1 Mathematical model

Given a foam network with periodic boundaries we can then assume that each vertex has three films and three adjacent cells. We seek to solve the full foam system constraints defined by the area constraints in Equation (4.2) and the angle constraints in Equation (4.1).

If we assemble all the constraints into a matrix notation then the full constraint system, over all areas A_i and angles ϕ_i, can be written as

$$\begin{bmatrix} C_A(A_1) \\ C_A(A_2) \\ \vdots \\ C_A(A_n) \\ C_\phi(\phi_1) \\ C_\phi(\phi_2) \\ \vdots \\ C_\phi(\phi_m) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.14}$$

Arranging (4.14) into a function $F\colon\ \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ yields

$$F(\mathbf{x}) = \mathbf{0}, \tag{4.15}$$

with

$$\mathbf{x} = [p_1, p_2, \ldots, p_n, \mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_m^T]^T,$$

which takes the shape of a classic root-finding problem [Nocedal and Wright, 2006].

Besides Equation (4.15) the mathematical model also include the topological processes from Section 3.3 and von Neumann's gas diffusion law (3.19) which is discussed in Subsection 4.3.4.

4.3.2 Numerical model

To derive the numerical model from the basic mathematical root-finding problem (4.15) that was presented in the previous subsection we are considering the following situation. Given $\mathbf{x}^k \in \mathbb{R}^{n+m}$ for some iteration step k, for which

$$F(\mathbf{x}^k) \neq \mathbf{0},$$

we seek to find $\Delta\mathbf{x}^k$ such that

$$F(\mathbf{x}^k + \Delta\mathbf{x}^k) = \mathbf{0}. \tag{4.16}$$

By taking a first order Taylor series approximation of (4.16) around the current iterate $\mathbf{x}^k$ which is

$$F(\mathbf{x}^k) + \nabla F^T(\mathbf{x}^k)\Delta\mathbf{x}^k = \mathbf{0}, \tag{4.17}$$

we get a linear system of equations which can be solved using an iterative Newton method [Nocedal and Wright, 2006]. Rearranging (4.17) we can write the Newton step as

$$\nabla F^T(\mathbf{x}^k)\Delta\mathbf{x}^k = -F(\mathbf{x}^k). \tag{4.18}$$

If we substitute the known components using

$$\mathbf{b} = -F(\mathbf{x}^k)$$

and

$$\mathbf{A} = \nabla F^T(\mathbf{x}^k)$$

then (4.18) can be written as

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}, \tag{4.19}$$

which is the well-known structure of a matrix system of linear equations. To compute the unknown $\mathbf{x}$ we solve (4.19) for $\mathbf{x}$ in an iterative Gauss-Seidel manner like

$$\mathbf{x}^{k+1} = \mathbf{L}^{-1}\left(\mathbf{b} - \mathbf{U}\mathbf{x}^k\right),$$

where $\mathbf{L}$ and $\mathbf{U}$ are the lower and strictly upper triangular decompositions of $\mathbf{A}$, respectively, such that

$$\mathbf{A} = \mathbf{L} + \mathbf{U},$$

and where we have assumed that $\mathbf{A}$ is invertible, e.g. $\det(\mathbf{A}) \neq 0$.

From [Kermode and Weaire, 1990] we have been guided to focus on local readjustments of pressures and vertex positions for each vertex in succession rather than solving for the full system implicitly. Such a strategy is not at all unfamiliar for vertex-based methods, especially for interactive schemes in Computer Graphics [Jakobsen, 2001, Müller et al., 2004, Müller et al., 2007]. When solving the system iteratively it is important to realize that achieving a global equilibrium state for the foam network with absolute minimum energy is unlikely. Instead, a foam network in local equilibrium state can be obtained. We equilibrate a vertex by adjusting the coordinates x, y and the pressures of the three adjacent cells p_i, p_j, p_k. With the fundamentals of the quasi-static foam model governed by Equation (4.14) and Equation (4.18) the system is defined by five linear equations in five unknowns as

$$\begin{bmatrix} \frac{\partial A_i}{\partial p_i} & \frac{\partial A_i}{\partial p_j} & \frac{\partial A_i}{\partial p_k} & \frac{\partial A_i}{\partial x} & \frac{\partial A_i}{\partial y} \\ \frac{\partial A_j}{\partial p_i} & \frac{\partial A_j}{\partial p_j} & \frac{\partial A_j}{\partial p_k} & \frac{\partial A_j}{\partial x} & \frac{\partial A_j}{\partial y} \\ \frac{\partial A_k}{\partial p_i} & \frac{\partial A_k}{\partial p_j} & \frac{\partial A_k}{\partial p_k} & \frac{\partial A_k}{\partial x} & \frac{\partial A_k}{\partial y} \\ \frac{\partial \phi_i}{\partial p_i} & \frac{\partial \phi_i}{\partial p_j} & \frac{\partial \phi_i}{\partial p_k} & \frac{\partial \phi_i}{\partial x} & \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_j}{\partial p_i} & \frac{\partial \phi_j}{\partial p_j} & \frac{\partial \phi_j}{\partial p_k} & \frac{\partial \phi_j}{\partial x} & \frac{\partial \phi_j}{\partial y} \end{bmatrix} \begin{bmatrix} \Delta p_i \\ \Delta p_j \\ \Delta p_k \\ \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Delta A_i \\ \Delta A_j \\ \Delta A_k \\ \Delta \phi_i \\ \Delta \phi_j \end{bmatrix}. \tag{4.20}$$

Figure 4.1 depicts the parameters that are involved in the equilibration procedure. The right-hand side of (4.20) defines the set of differences which are identical to the differences in the area and angle constraints as

$$\Delta A_i \equiv C_A(A_i)$$

and

$$\Delta\phi_i \equiv C_\phi\left(\phi_i\right).$$

The 5×5 system matrix $\mathbf{A}$ is the Jacobian of the system. It contains all of the partial derivatives of the system. The last two rows of $\mathbf{A}$ describe the rate of change of the angles ϕ_i and ϕ_j. The reason why the third angle ϕ_k is not part of the Jacobian is simply because it can be computed from the two other angles as

$$\phi_k = 2\pi - \left(\phi_i + \phi_j\right), \tag{4.21}$$

4.3.3 Jacobian derivatives

The partial derivatives in $\mathbf{A}$ can be approximated numerically by using a numerical method such as the Finite Differences Method [Eberly, 2008]. In [Kermode and Weaire, 1990] it is claimed that the system is too hard to derive analytically. We will not dispute that claim but accept that the Jacobian derivatives must be determined numerically. The numerical method employed to compute the derivatives has an impact on the overall accuracy of the Jacobian.

When operating on spatial entities, such as coordinates, it is often favorable to employ central difference approximations where forward and backward approximations explicitly can distort the results. The first order central finite difference operator has a second order error term

$$\frac{\mathrm{d}f\left(x\right)}{\mathrm{d}x} = \frac{f\left(x+\Delta h\right) - f\left(x-\Delta h\right)}{2\Delta h} + O\left(\Delta h^2\right),$$

where Δh is the step size that defines how good the approximation is. The finite difference method for functions in multiple variables is a matter of approximating one variable at a time. Usually each variable is subject to its own finite step size but we will assume that we can find a valid step size candidate that fits all variables.

The partial cell area function derivatives can be approximated as

$$\frac{\partial A\left(\mathbf{p},\mathbf{r}\right)}{\partial p_i} \approx \frac{A\left(p_i+\Delta h, p_j, p_k, \mathbf{r}\right) - A\left(p_i-\Delta h, p_j, p_k, \mathbf{r}\right)}{2\Delta h},$$

$$\frac{\partial A\left(\mathbf{p},\mathbf{r}\right)}{\partial x} \approx \frac{A\left(\mathbf{p}, x+\Delta h, y\right) - A\left(\mathbf{p}, x-\Delta h, y\right)}{2\Delta h},$$

where the rest of the partial area derivatives are approximated exactly the same way. The partial vertex angle function derivatives are approximated using

$$\frac{\partial \phi\left(\mathbf{p},\mathbf{r}\right)}{\partial p_i} \approx \frac{\phi\left(p_i+\Delta h, p_j, p_k, \mathbf{r}\right) - \phi\left(p_i-\Delta h, p_j, p_k, \mathbf{r}\right)}{2\Delta h},$$

$$\frac{\partial \phi\left(\mathbf{p},\mathbf{r}\right)}{\partial x} \approx \frac{\phi\left(\mathbf{p}, x+\Delta h, y\right) - \phi\left(\mathbf{p}, x-\Delta h, y\right)}{2\Delta h}.$$

The numerical derivate step size Δh is discussed in greater details in Subsection 5.4.1 (p. 47).

4.3.4 Foam coarsening

To allow a foam network to evolve over time different kinds of dynamic foam processes can be applied to the foam system. Different types of foam dynamics have been discussed in Chapter 3 which among others includes coarsening due to diffusion. Other interesting dynamics processes are discussed later in Section 5.2 (p. 41).

The mathematical model for coarsening is given by von Neumann's Law and repeated here for convenience

$$\frac{\mathrm{d}A_n}{\mathrm{d}t} = K(n-6), \tag{4.22}$$

which defines the rate of change in cell area over time as a function of cell edges. This change in area can easily be integrated into the quasi-static equilibrium model. Recall from Equation (4.2) that each cell is constrained to reach its target area A^{target}. By updating the cell target area we can simulate the phenomenon due to gas diffusion as

$$A^{target} \leftarrow A^{target} + K(n-6), \tag{4.23}$$

where n is the number of edges of the cell. The time step factor has be neglected due to the quasi-static nature of the foam simulation.

4.4 Fixed Boundaries

One of the profound limitations of vertex-based foam models in Computational Physics [Weaire and Kermode, 1983, Kermode and Weaire, 1990] is the requirement of an infinite foam network. In practice the infinite foam translates into a finite network with periodic boundaries which makes it possible to implement and simulate the foam network numerically.

A significant reason for the periodic boundaries to assimilate an infinite network is that each vertex has exactly three adjacent cells which is required by the numerical model. Another reason for the periodic boundaries is due to Euler's law [Weaire and Hutzler, 1999] which demands that the average number of edges of all cells in an infinite network is precisely

$$\overline{n} = 6,$$

which is assumed for vast part of the theory behind two-dimensional foam, e.g. von Neumann's gas diffusion law (4.22).

In Computer Graphics one way to deviate from the periodic boundaries is to employ fixed boundaries. With fixed boundaries the vertices on the boundary are omitted from the equilibration process which will leave the impression that they are immovable, or fixed. If fixed boundaries are applied it is important to realize that gas diffusion on the boundary cells should also be omitted. Otherwise it is possible that the topological processes can violate the fixed boundary constraints and in worst case introduce local stability issues in the foam network.

Although fixed boundaries can be applied to the foam network to circumvent the periodic boundary condition it will also constrain the foam as an entity from any dynamic

movement which is not in the interests of any Computer Graphics application. We seek a solution that avoids the periodic and fixed boundary conditions alike and in the next chapter we will investigate such a solution.

4.5 Visualization

Paying attention to Computer Graphics it is mandatory that we can provide some kind of visual feedback not only to our selves but also to the viewers of any of the applications the foam simulator presented here might be a part of.

The vertex-based model that has been derived through this chapter operates on pressure differences which by (3.11) implies a radius of curvature when the adjacent pressures differ. A simple but convincing way to illustrate the dry foam network is to draw the films between adjacent vertices as curved lines or circular arcs. Optionally, the cells can be drawn using filled, colored triangles to provide the viewer with additional information, e.g. the pressure that is associated with each cell. Of course the cells can be drawn using any kind of shader but independently of the method we want to apply for rendering the cells we will need a way to draw arcs, in wireframe and in solid.

4.6 Summary

In this chapter the following important topics and results have been discussed:

- The mathematical foam model can be summarized to include
  1. Von Neumann's law
     $$\frac{\mathrm{d}A_n}{\mathrm{d}t} = K(n-6)\,.$$
  2. Topological processes and other foam state modifications
  3. The constraint equations as a root-finding problem
     $$F(\mathbf{x}) = \mathbf{0} \quad \text{with} \quad \mathbf{x} = [p_1, p_2, \ldots, p_n, \mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_m^T]^T\,.$$
- The numerical foam model can be summarized to include
  1. Von Neumann's law
     $$A^{target} \leftarrow A^{target} + K(n-6)\,.$$
  2. Foam state changes and foam mesh updates.
  3. Nonlinear quasi-static Newton method.

Chapter 5

Improvements

"*I am easily satisfied with the very best.*"

- **Winston Churchill**

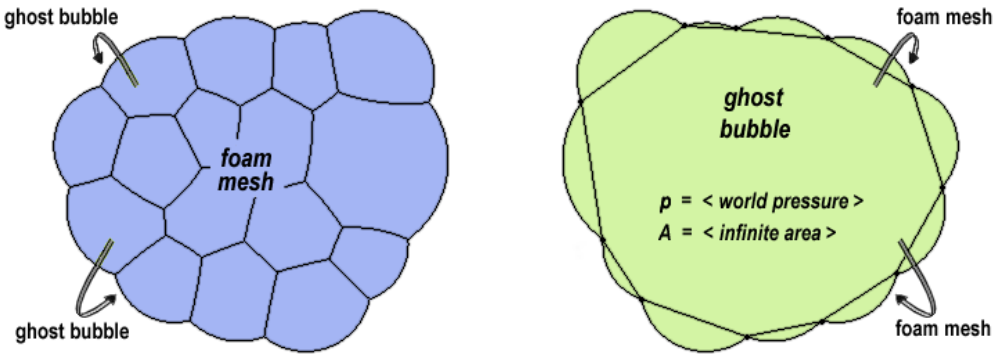
In this chapter we evaluate the behavior of the vertex-based foam model which has been implemented as it has been presented in Chapter 4. Based on experience gained from debugging and testing the basic implementation we suggest adjustments and extensions to correct and improve the stability and the visual outcome of the vertex model. The basic implementation with the improvements from this chapter is presented together in greater details in Chapter 6. Results from before and after the improvements are presented in Chapter 8.

5.1 Dynamic Boundaries

The fixed boundary approach that has been presented in Section 4.4 is a simple boundary condition for the finite foam network in the vertex model. The fixed boundary condition is employed as a mean to get around the periodic boundaries, that otherwise is necessary to work with the vertex model, but they do not allow the foam to evolve naturally.

In Computer Graphics we are interested in foam, or froth, as a contained entity. With the concept of free surfaces in CFD we will apply the concept of *free foam surface* to foam dynamics, which is a foam with finite and dynamic boundaries. The free foam surface is in contrast to the fixed foam surface which is the default boundary condition that appends our basic implementation of the vertex model. Applications of liquid foams with dynamic boundaries include all industrial production assets.

In Computer Graphics it is imperative to employ the dynamic boundary condition as the results of foam simulations with free, movable surfaces often interact with additional systems, e.g. froth that buoyants on a liquid as the example on Figure 1.1 illustrates.



(a) (b)

Figure 5.1: The ghost bubble. (a) At the **front** of the foam is the foam mesh defined. (b) The ghost bubble is defined on the **back** of the foam mesh where it reuses all existing boundary vertices in the foam and acts as a *mirrored* bubble that contains all normal bubbles in the foam mesh.

A foam with a freely movable surface can not have fixed boundaries and new dynamic boundary conditions must be applied to the vertex model to allow for free foam surfaces.

5.1.1 The ghost bubble

During the development of the foam model we have looked into several possibilities to handle the dynamic boundary conditions for a two-dimensional dry foam. The main challenge is to let the boundaries vanish and equally deal with the entire foam in the bulk as well as on the boundaries.

In one of the attempts we introduced the *ghost bubble* which is a new bubble that is created just before we are about to equilibrate a boundary vertex. The term *ghost* comes into play because firstly the ghost bubble should not be visible, or at least not rendered, and secondly because the ghost bubble is only present when it is required for the free foam surface. This is an analogue to the ghost fluid method [Fedkiw et al., 1999] for solving the fluid motions on the boundaries in the Eulerian grid. The main idea with the ghost bubble is to make sure that each vertex not only has three connecting films but also is surrounded by three bubbles. The three bubbles are mandatory to solve the system matrix of the vertex model in Equation (4.20) (p. 34). The ghost bubble was removed again once the boundary vertex was equilibrated. In theory the ghost bubble must not add any potential energy into the foam system. However, in practice the ghost bubble is required to have a pressure and an area before it can be used to solve the system matrix.

The idea with the ghost bubble led us to the *ghost bubble method* which is a novel approach[1] to handle dynamic boundary conditions on a finite sized foam. The method

[1]To the best of our knowledge no one has described an idea like this one to handle dynamic boundary conditions on a liquid foam.

adds free surfaces to the foam and is not limited to two dimensions. However, for the sake of clarity, and the premises of this thesis, we will discuss the idea solely in two dimensions. The ghost bubble method will add one ghost bubble to the foam system. To avoid adding additional vertices the ghost bubble reuses the already existing boundary vertices and is added onto the back of the foam mesh, as depicted on Figure 5.1.

5.1.2 Ghost bubble boundary conditions

Given the characteristics of the ghost bubble it should act as a passive auxiliary to the foam solver. We seek to model the ghost bubble as a natural element of the real world, i.e. a *world bubble*, that defines the atmosphere that surrounds the foam system. This results in the following two rules that the ghost bubble must comply with:

Area - The ghost bubble has infinite area $\Rightarrow$ No change in ghost bubble area.

Pressure - The ghost bubble has constant unity pressure $\Rightarrow$ No change in ghost bubble pressure.

Naturally, the gas diffusion process is taking place on the foam boundary as in the bulk of the foam. When gas diffuses through boundary films into the atmosphere the volume of the lost foam gas is insignificant compared to the gas volume of the *world*. From this argument it makes sense that the ghost bubble has constant area. The same argument holds for the constant ghost bubble pressure.

Ghost area and pressure

The ghost bubble boundary rules imply that

$$\Delta A_{\text{ghost}} = 0, \tag{5.1}$$

$$\Delta p_{\text{ghost}} = 0. \tag{5.2}$$

We seek to find a solution for $\mathbf{A}$ in (4.20) such that (5.1) is met. Extracting the *ghost*-index row of (4.20) yields

$$\frac{\partial A_{\text{ghost}}}{\partial p_i}\Delta p_i + \frac{\partial A_{\text{ghost}}}{\partial p_j}\Delta p_j + \frac{\partial A_{\text{ghost}}}{\partial p_k}\Delta p_k + \frac{\partial A_{\text{ghost}}}{\partial x}\Delta x + \frac{\partial A_{\text{ghost}}}{\partial y}\Delta y = \Delta A_{\text{ghost}}, \tag{5.3}$$

and if we insert (5.1) into (5.3) we get

$$\frac{\partial A_{\text{ghost}}}{\partial p_i}\Delta p_i + \frac{\partial A_{ghost}}{\partial p_j}\Delta p_j + \frac{\partial A_{\text{ghost}}}{\partial p_k}\Delta p_k + \frac{\partial A_{\text{ghost}}}{\partial x}\Delta x + \frac{\partial A_{\text{ghost}}}{\partial y}\Delta y = 0. \tag{5.4}$$

A quick observation of (5.4) suggests that setting

$$\mathbf{A}^T_{\text{ghost}} = \mathbf{0}, \tag{5.5}$$

will satisfy (5.4) independently of $\mathbf{x}$ where $\mathbf{A}^T_{\text{ghost}}$ designates the *ghost*-index row of $\mathbf{A}$ in (4.20).

Finding $\mathbf{A}$ such that (5.2) will be satisfied must be done implicitly as we seek to find $\mathbf{x}$. However, as we are dealing with a linear system a simple observation of (4.20) suggests that assuming that (5.2) is met then setting

$$\mathbf{A}_{\text{ghost}} = \mathbf{0}, \tag{5.6}$$

will not affect the outcome of $\mathbf{b}_{\text{ghost}}$ which is the *ghost*-element of the result vector $\mathbf{b}$ and where we similarly designate $\mathbf{A}_{\text{ghost}}$ as the *ghost*-index column of $\mathbf{A}$.

The results from (5.5) and (5.6) are interesting because dealing with the ghost bubble we can handle the dynamic boundary conditions by storing zeros in $\mathbf{A}$ for the row and the column that involve the ghost bubble.

Ghost angle and position

Recall that only two of the three angles are present (rows 4 and 5) in the system matrix $\mathbf{A}$ of (4.20) and that the third angle is given by (4.21). The linearity of the system to solve provides the flexibility to use any two of the three angles in $\mathbf{A}$ and $\mathbf{b}$. When a vertex to be equilibrated is on the boundary of the foam mesh we know that one of the bubbles will be the ghost bubble. When this is the case we use the angles given by the vertex in question and the two foam bubbles that is *not* the ghost bubble.

5.2 Foam Dynamics

In Subsection 4.3.4 (p. 36) the dynamic process of coarsening due to diffusion was discussed. In this section we will discuss the other dynamic processes that have been described in Chapter 3 (p. 12).

A basic form of foam collapsing can take advantage of the same mechanism employed for coarsening. That is, to modify the state variables for the foam system and let the quasi-static foam simulator bring the foam network back into equilibrium.

5.2.1 Foam collapsing

A cell can be removed due to a topological T2 operation if its current area A_i is less or equal than a specified threshold $T_{\text{area}} \geq 0$ (3.13),

$$A_i \leq T_{\text{area}},$$

but the above condition can be expanded to include the target area as well,

$$A_i \leq T_{\text{area}} \vee A_i^{target} \leq T_{\text{area}}. \tag{5.7}$$

To collapse cell i has now become a simple operation of setting

$$A_i^{target} = 0,$$

and next time the foam undergoes equilibration that cell will perish. The modification to (5.7) will also increase the stability in situations where the neighborhood of a cell is ill defined such that it prevents the cell from being equilibrated. If diffusion is applied then the target area of the ill defined cell will eventually become too low and the cell will be removed by the topological operations.

5.2.2 Shearing

Applying shear strain to the foam mesh is a straight forward operation that can be executed by modifying the vertex positions. New vertex positions due to simple and extensional shear are computed using Equation (3.20) and Equation (3.21), respectively. The shear mechanism works out of the box because the foam mesh will be equilibrated afterwards. However, the foam network must not be too elastically deformed in-between equilibrations as the foam network can deviate too greatly from its equilibrated state and introduce stability problems.

We model shear interactively by updating each vertex position $\mathbf{p}_i$ using a specified strain parameter ε with a reference point $\mathbf{p}_{\text{ref}}$ as

$$\mathbf{p}_i \leftarrow \mathbf{p}_{\text{ref}} + \mathbf{S}_\varepsilon \left(\mathbf{p}_i - \mathbf{p}_{\text{ref}}\right), \tag{5.8}$$

where $\mathbf{S}_\varepsilon$ is a 2×2 shear matrix as defined in Subsection 3.5.1 (p. 22).

5.3 Stability issues

Stability is the key to make the vertex-based foam simulation model usable. We have encountered several stability issues with the model and the majority of these instabilities are directed against the topological structure of the foam network. In this section we discuss how to cope with these instabilities.

5.3.1 Damped line search method

A known trick to stabilize simulation results obtained from line search methods is to damp the adjustments that are produced by taking the full line step [Nocedal and Wright, 2006]. The stabilization behind the damped line search method is equivalent to taking smaller time steps in explicit solvers. The convergence rate will decrease but there is an improved chance that convergence will be obtained which implies stability.

We introduce the linear damping parameter τ to adjust the result from Equation (4.20) (p. 34),

$$\hat{\mathbf{x}} = \tau\mathbf{x} = \tau\left(\mathbf{A}^{-1}\mathbf{b}\right), \qquad 0 \leq \tau \leq 1,$$

where $\tau = 1$ will perform the full adjustment while $\tau = 0$ will omit the adjustment completely. Setting $\tau \ll 1$ to be constant throughout the simulation will add relaxation to the foam solver. However, the value can still be too big and local reductions to τ can be required to ensure stability. Instead we let τ be a variable that can be locally modified to

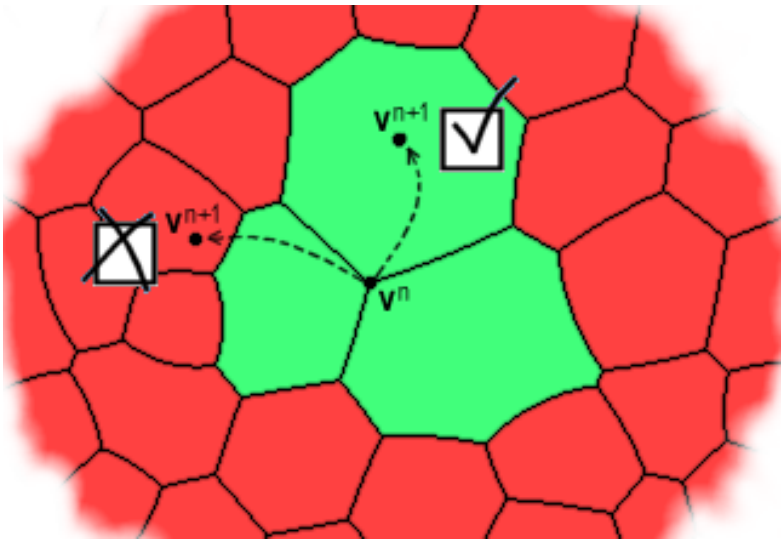


Figure 5.2: The updated vertex position can become invalid if it is displaced beyond the boundary defined by its three surrounding bubbles. Green cells indicate valid positions for vertex **v** and red cells (the rest of the foam) indicate invalid positions.

handle additional stability cases. Initially we set $\tau = \overline{\tau}$ with $0 \ll \overline{\tau} \ll 1$ as the damping constant.

5.3.2 Invalid vertex positions

New vertex position displacements produced by the foam solver can lead to invalid vertex positions. The problem is quite extreme in the case of new unequilibrated foam networks. Once the foam network has been equilibrated invalid vertex position displacements rarely happens. A vertex position displacement becomes invalid if it brings the vertex beyond the boundary defined by its three surrounding bubbles, as illustrated on Figure 5.2.

Invalid vertex positions are unstable because a part of the adjacent cell area, or all of it, can become negated. We know from the T2 condition (3.13) that if the current area of a cell is below the threshold then the cell will suffer a T2 process. Invalid vertex positions are also a visual problem as the topology will overlap itself and create a flawed foam mesh.

We seek to reduce a vertex position displacement **d** until the updated vertex position no longer is invalid. The procedure we employ is a so-called *backtracking* approach [Nocedal and Wright, 2006] which is a direct continuation of the line search method from Subsection 5.3.1. The backtracking line search algorithm reduces the damping parameter τ iteratively until the desired condition of stability has been achieved:

1. Compute updated vertex position from

$$\mathbf{u} = \mathbf{v} + \tau\mathbf{d}$$

2. Validate updated vertex position is inside one of the three surrounding cells

$$\begin{aligned}\textbf{inside} = &\ \text{PointInsideCell}(\mathbf{u}, \text{cell}_i) \\ &\vee \text{PointInsideCell}(\mathbf{u}, \text{cell}_j) \\ &\vee \text{PointInsideCell}(\mathbf{u}, \text{cell}_k)\end{aligned}$$

3. If **inside** is *false* then adjust τ as

$$\tau \leftarrow \frac{\tau}{2},$$

   and try again (jump to 1.).

The procedure loops until τ will ensure that the updated vertex position stays inside the surrounding cells. The function `PointInsideCell` checks if the provided position is inside the edges of the given cell which is not required to be convex. We will not provide a solution for such a function but merely refer to [de Berg et al., 2000] which includes several solutions for such computational geometrical problems. We will argue that using only straight lines for the edges, and not curved lines, will be sufficient as the the local radii of curvature in the foam bulk often is close to infinity . The only exception is boundary cells that are connected to the ghost bubble where the pressure differences usually are larger. However, `PointInsideCell` should always return *true* if the cell is the ghost bubble.

5.3.3 Inverted angles

The standard trigonometric cosine function that is employed to compute the vertex angles in Subsection 4.2.1 (p. 30) is subject to instability when we fail to realize that

$$\cos^{-1} \in (0..\pi),$$

which results in an angle between two vectors never can become greater than $\theta = 180^\circ$. Instability occurs when the angle between two edges in fact become grater than 180° as illustrated in Figure 5.3. We call such an angle for an inverted angle. A vertex with an inverted angle is unstable because the solver will be unable to equilibrate the vertex, due to the 120° angle constraints. We cannot guarantee that an inverted vertex angle will not be subject to the system matrix in (4.20) and for this reason it is best practice to remove the situation completely.

To cope with inverted angles we seek a method to find a correctly oriented angle, and the two-dimensional determinant that is defined in Equation (4.13) (p. 31) can be used for this purpose. For two given vectors $\mathbf{v}$, $\mathbf{w} \neq \mathbf{0}$, if $\det(\mathbf{v}, \mathbf{w}) < 0$ it indicates that the angle between the vectors will become inverted. We can now define a new function Θ that

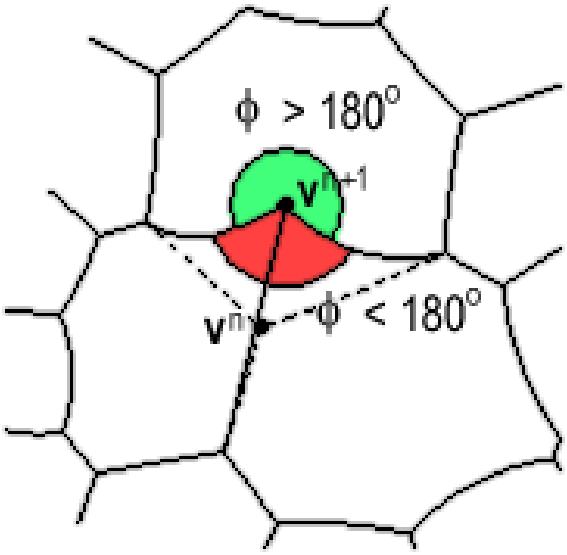


Figure 5.3: Special care must be exhibited when the angle is greater than 180°. The green angle $\phi > 180°$ for $\mathbf{v}^{n+1}$ is the correct angle but the red angle $\phi < 180°$ is computed by default.

computes the oriented angle between two vectors[2] as

$$\Theta(\mathbf{v},\mathbf{w}) = \begin{cases} \cos^{-1}\left(\dfrac{\mathbf{v}\cdot\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}\right) & \text{if } \det(\mathbf{v},\mathbf{w}) > 0 \\ 2\pi - \cos^{-1}\left(\dfrac{\mathbf{v}\cdot\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}\right) & \text{if } \det(\mathbf{v},\mathbf{w}) \leq 0 \end{cases}.$$

To avoid unstable inverted angles in the computations of the foam angle constraints the new Θ function must be employed to compute the correctly oriented angles. Equation (4.12) (p. 31) must be replaced with

$$\phi_i = \Theta(\mathbf{v},\mathbf{w}) + \alpha, \tag{5.9}$$

where α is the contributions from the two signed angles due to the curvatures.

5.3.4 Vertex fluctuation

From time to time the foam structure can become locally ill-defined, e.g. the solver cannot find a valid position for some vertex that ensures that all three angles are (close to) 120°. During the equilibration of such an ill-defined vertex the updated positions will fluctuate with each iteration. This position fluctuation typically implies divergence and when the

[2]The functionality of the inverse trigonometric method **atan2** is similar to the described Θ function. However, the method is normally only associated with programming languages.

solver reaches the maximum allowed amount of iterations the vertex can be far from equipoised than prior the equilibration procedure.

During tests and experiments we have observed that if fluctuation occurs it is best practice to skip or ignore the vertex in question and let the solver continue normally. Like other relaxation-based methods it is likely that the next time we encounter a previously ill-defined vertex it can now be equilibrated without any problems.

During equilibration the solver is not aware of the final vertex position. Thus, it is not usable to employ the positions as a mean to determine fluctuation. Instead, we will use the angles to measure fluctuation. When iterating through a single vertex equilibration it is possible to monitor the maximum angular displacement error O from the right-hand side of Equation (4.20) (p. 34) as

$$O_n = \max\left(|\Delta\phi_i|, |\Delta\phi_j|\right),$$

where n is the current iteration. If we are converting towards equilibrium then $O_n \to 0$, but if

$$O_n - O_{n-1} > 0,$$

then the solver is diverging away from equilibrium, and we choose to skip the vertex. Initially, we set $O_0 = \frac{2\pi}{3}$ which is the maximum possible displacement error.

5.3.5 Topological edge flipping

During the evolution of a foam network topological T1 processes will occur. We have witnessed situations where the outcome of a T1 edge flip immediately has trigged another T1 process. The problem is that after a T1 edge flip has occurred the vertices might still not be too far apart to prevent another T1 operation. The outcome is a continuous edge flipping that persists until the vertices has become separated enough to stop the T1 process.

The T1 flipping can in most situations be solved by equilibrating the flipped vertices immediately after the T1 operation. Due to the equilibration procedure we know from Subsection 3.3.1 (p. 18) that the vertices are bound to move apart which will prevent the T1 flipping. However, in rare occasions the T1 flipping may still occur. A local ill-defined network can prevent the vertices from being equilibrated after a T1 process which again can result in the T1 flipping. We solve this problem by introducing a T1-tag to each edge in the network. Initially, the T1-tag is not set and the T1 operation will set the tag on every flipped edge. This will prevent another T1 process from being performed if the T1-condition in Equation (3.12) (p. 19) is modified to

$$\mathbf{d}\cdot\mathbf{d} < {T_{\text{dist}}}^2 \ \wedge \ \textit{not}\ \text{T1-tagged}. \tag{5.10}$$

All T1-tags should be reset just before the entire foam equilibration procedure is commenced.

5.4 Nonlinear Newton Simulation Method

The matrix system in Equation (4.20) (p. 34) is a linear system but even though the mathematical equations within the system have been linearized they are not linearly dependent on each other, and because of this the problem we address is nonlinear. Newton's method that also is known as the Newton-Raphson method is a well-known iterative method to successively compute more accurate approximations to some function $f(\mathbf{x}) = 0$. The iterative Newton-Raphson method seems evident to apply as the foam simulation strategy.

To solve the system of linear equations we employ the method of *singular value decomposition* (SVD). The reason for the this method to solve the system is because SVD can work on nearly every type of matrix. We have not analyzed the system matrix thoroughly to conclude that other matrix solvers will be able to handle the problem but we know from Subsection 5.1.2 (p. 40) that when the dynamic boundary conditions are employed at least one diagonal matrix element will be zero. This fact alone suggests that the SVD method is a fairly good choice.

5.4.1 Numerical derivative step size

In Subsection 4.3.3 (p. 35) we discussed how to compute the partial derivatives of the system matrix numerically by employing first order central differencing approximations. The finite difference step size Δh has a direct influence on the accuracy of the numerical derivatives which again depends on the local configuration of the foam mesh. Upon assembly of the system matrix it is imperative that Δh is set to a value that ensures robustness of the matrix.

To validate the robustness of the matrix we employ an iterative step-size-halving strategy. If the absolute value of the largest element from the difference between two matrices assembled from Δh and $\frac{1}{2}\Delta h$ is below a prescribed positive threshold, we will argue that the system matrix assembled from the latest step size is satisfactory. The difference between the two matrices is

$$\Delta\mathbf{A} = \mathbf{A}_{\Delta h} - \mathbf{A}_{\frac{1}{2}\Delta h}, \tag{5.11}$$

where the largest absolute element value is defined as

$$e_{\max} = |\Delta\mathbf{A}|_{\max}.$$

The condition for using $\frac{1}{2}\Delta h$ as step size for the system matrix assembly is

$$e_{\max} < T_{\text{stop}}, \tag{5.12}$$

where $T_{\text{stop}} > 0$ is the convergence stop criterion threshold. If (5.12) is not met then we halve the step size as

$$\Delta h \leftarrow \frac{1}{2}\Delta h,$$

and test for convergence again from Equation (5.11). The process is iterative and continues until a safe Δh step size has been obtained.

5.4.2 Newton simulation algorithm

Before outlining the steps of the nonlinear Newton solver we will assume that we are working on an equilibrated, or nearly equilibrated, foam network. How to initialize the foam network is discussed in Section 6.2. Listed below are the items that implement a single Newton simulation step. Besides the various improvements from this chapter the algorithm also employs most of the information from Chapter 4 (p. 26) and some from Chapter 3 (p. 12).

Foam network adjustments

- For each cell in the network:
  - Adjust target area A^{target} due to diffusion by employing von Neumann's law Equation (4.23) (p. 36), or
  - Set $A^{\text{target}} = 0$ if the cell should be collapsed.
- For each vertex in the network:
  - Adjust position $\mathbf{p}$ due to shear strain (5.8).
- For each border in the network:
  - Reset T1-tag.

Foam network equilibration

For each vertex in the network:

- Check if any T2 condition is met in the three adjacent cells, and if so then
  - Perform T2 topological change on the first T2 valid cell and continue to next vertex, otherwise
- Check if any T1 condition is met in the three adjacent films, and if so then
  - Perform T1 topological change on the first T1 valid border and continue to next vertex, otherwise
- For each global vertex iteration n:
  - Compute the result constraint vector $\mathbf{b}$ and if $\|\mathbf{b}\| \to 0$ then stop the vertex iteration loop, otherwise
  - Assemble the Jacobian system matrix $\mathbf{A}$.
  - Compute the difference vector $\mathbf{x}$ employing the SVD matrix solver.
  - Update the relaxation coefficient τ using the different stability methods.
  - Update the vertex position $\mathbf{p}$ and the three adjacent cell pressures p_i, p_j, p_k using $\tau\mathbf{x}$.

5.5 Summary

In this chapter the following important topics and results have been discussed:

- The Ghost Bubble method which enables dynamic boundaries for the foam model. The ghost bubble is defined on the back of the foam mesh which allows it to move with the foam. The ghost bubble must maintain an infinite area and a constant pressure.
- The matrix system is modified to handle the dynamic boundaries by inserting zeros in the column and row elements which corresponds to the ghost bubble.
- A cell can be collapsed by enforcing a T2 operation using

$$A^{target} \leftarrow 0\,.$$

- The linear relaxation parameter τ has been introduced to damp the results produced by the matrix solver

$$\mathbf{x} = \tau\left(\mathbf{A}^{-1}\mathbf{b}\right), \qquad 0 \leq \tau \leq 1\,.$$

- The damping parameter τ can be modified to account for invalid vertex position.
- A function Θ was derived that computes the total angle in $(0..2\pi)$ between two non-zero vectors.
- Simulation details from the nonlinear Newton algorithm for two-dimensional dry foam were derived.

Chapter 6

Implementation

"*This is a simple matter in principle*
but quite an awkward exercise in practice."
- **Denis Weaire**

In this chapter details about the implementation of the vertex-based foam model will be revealed. The implementation includes the basic model that was derived in Chapter 4 and our improvements from Chapter 5. The source code is subject to future publications and is as such not yet public available. However, supplementary thesis material including a binary vertex-based foam application[1] that demonstrates the methods from this thesis is available from [Kelager, 2009]. See Appendix A for additional details about downloading and executing additional thesis contents.

The vertex-based, quasi-static equilibrium model for two-dimensional dry foam has been implemented in C++. Our implementation of the model is supported by the OpenTissue library [OpenTissue, 2009] which provides a flexible math system, powerful data structures, and easy debug rendering.

6.1 Foam Mesh

In this section we will describe the foam mesh and the associated data structure, along with some modifications to the data structure that was necessary to map the vertex-based foam model into it.

6.1.1 Data structure

A data structure that can hold information about vertices, edges, and faces in a way so they all are connected is required to represent the foam network. Such a data structure seems

[1] At the time of writing the vertex-based foam application is only available for win32 platforms.

Foam Element	Mesh Element	Mesh Traits
Junction, Vertex	Vertex	Position
Film	Half-Edge	Thickness
Border	Edge	T1 Tag
Bubble, Cell	Face	Target Area, Pressure, Ghost Tag
Foam	Mesh	Surface Tension, Diffusion Permeability, Topological Thresholds

Table 6.1: Mapping between the foam mesh and the polymesh, with assigned foam data traits.

reasonable close to the *doubly-connected edge list* data model [de Berg et al., 2000] and OpenTissue implements an open variant of this data structure called the *polymesh*. For a general description and a good definition of the doubly-connected edge list we recommend looking at the many details given in [de Berg et al., 2000].

The polymesh data structure employs standard terms for the elements it defines and compared to the default polymesh structure our vertex-based model has been relying on an additional set of terms. We have mapped the foam elements into the appropriate polymesh elements. Besides storing the geometric and topological information the polymesh also offers attribute data type traits to store additional application specific data. As we are mapping the foam model into the polymesh data structure it is important that physical foam data can be assigned to the foam network. We have assigned a set of foam data traits to each of the mesh elements. The mapping between the polymesh and our foam mesh with our foam traits are given in Table 6.1.

6.1.2 Limitations

The OpenTissue polymesh data structure has been designed to take advantages of applications that employ standard two-dimensional meshes. Several validation methods are applied in creation and modification of a mesh. One thing that is a problem is that it is impossible to create a face with less than three vertices. In general, this limitation is both healthy and sane, and it prevents applications of the polymesh from performing illegal topologically operations. However, this limitation also prevents our implementation from forming a bubble with only two vertices as illustrated in Figure 6.1.

Although the polymesh data structure do not support two-sided bubbles, and the behavior of the implementation is unknown if this situation occurs, it has been necessary to handle the initial three-sided double-bubble problem, which subsequently results in a two-sided bubble. The three-sided double-bubble problem is discussed in Subsection 6.3.3.

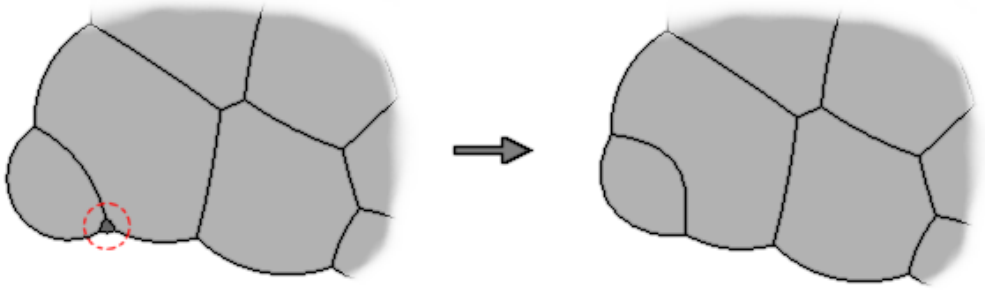


Figure 6.1: At the lower-left two bubbles are connected to each other, both of which consist of three vertices. After a T2 process has eliminated the small bubble, a two-sided bubble should in theory be left behind.

6.2 Initial Foam Conditions

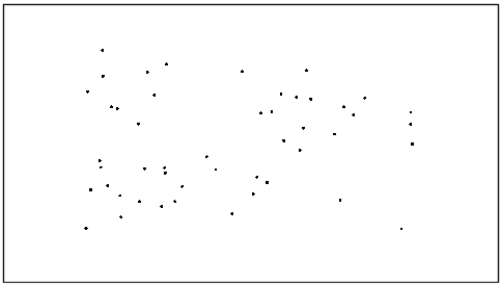
In this section we define the initial procedures that are required to mature a dry foam network. A *mature* foam network is a foam network that has been completely, or nearly, equilibrated. In contrast to a mature foam network we will employ the term *immature* about a newly created foam network that deviates significantly from equilibrium.

Before we begin simulating the foam network it is imperative to have some initial, well-defined data to work with. Often in cases similar to this one the simulator is only as stable as the initial data allows. To prevent implementing a large amount of solutions to handle problematic data in various degenerated cases we have chosen only to support a certain type of input data; *a set of planar points*. Once a set of points has been provided the application will handle the rest of the data initialization automatically. Additionally, the program supports automatic generations of random planar points. The user just need to specify how many points the application should create.

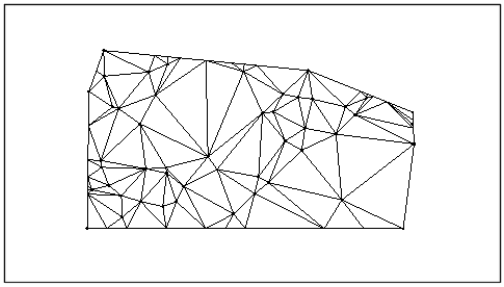
In [Weaire and Kermode, 1984] the authors claim that the network from a Voronoi diagram provides an excellent initial foam condition. They are working with the exact same type of two-dimensional foams as we are, except for periodic boundaries, and it will make sense to employ the same procedure to construct the initial foam network.

Below is a list that contains the different items that our implementation will execute to process an initial foam network condition:

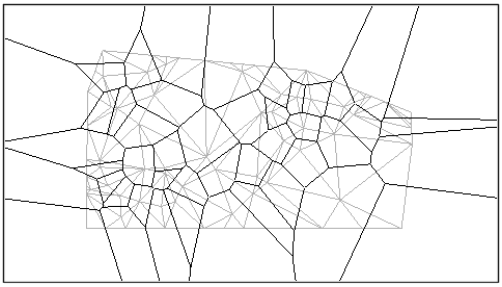
1. Initialize a set of planar input sites/points
2. Construct the Delaunay triangulation from the input points
3. Compute the Voronoi diagram (the dual of the Delaunay triangulation)
4. Clean up Voronoi diagram to meet our initial foam mesh requirements
5. Equilibrate the foam mesh



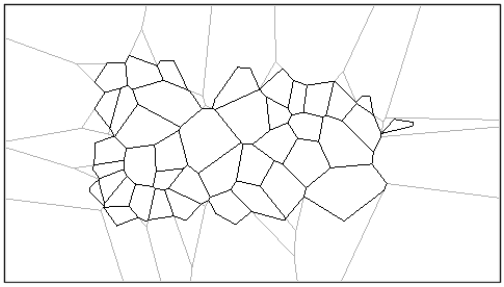
(a) Random generated point sites.



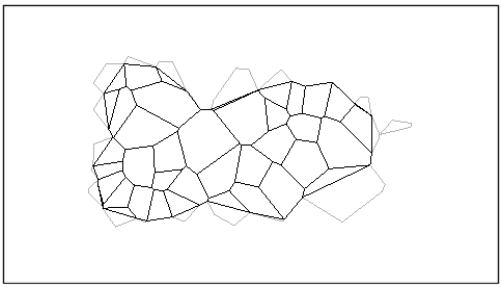
(b) Delaunay triangulation created from the initial sites.



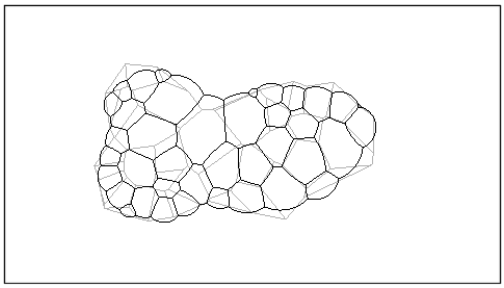
(c) Voronoi diagram computed as the dual of the Delaunay triangulation.



(d) Voronoi diagram is cut to remove all infinite edges.



(e) Cut Voronoi diagram is corrected to initialize the foam mesh.



(f) Foam mesh is equilibrated and ready for simulation.

Figure 6.2: Initial foam mesh steps. (a) A number of random generated point sites in the plane. (b) A standard Delaunay triangulation is created using the random point sites as input. (c) The dual of the Delaunay triangulation is computed that results in an infinite Voronoi diagram. (d) All the infinite edges are cut away and cells are merged to form a finite Voronoi diagram. (e) All vertices with less than three edges are removed and invalidated vertices are re-connected to form a valid foam mesh. (f) The initial foam mesh is equilibrated to meet all the equilibrium rules.

In Figure 6.2 the different foam initialization steps are depicted. Generation of random points is trivially implemented. The Delaunay triangulation computed from a finite set of points is less trivial but can be accomplished by the means of circumscribed circles [de Berg et al., 2000]. OpenTissue provides functionality to compute the Delaunay triangulation of any given set of two-dimensional points and will consequently not be discussed any further here. The dual graph of a Delaunay triangulation is the Voronoi diagram which can be constructed by connecting all the adjacent centers of the Delaunay circumcircles. To the keen reader that is interested in Computational Geometry we recommend looking into [de Berg et al., 2000] where additional details about the construction of Delaunay triangulations and Voronoi diagrams are treated in great details.

6.2.1 Voronoi diagrams and foam networks

The construction of a Voronoi diagram provides a useful starting condition for the foam simulations but neglects the all-important film curvature. However, this can be dealt with by running a couple of equilibration steps before beginning the foam simulation process. The transformation from initial foam mesh to equilibrated foam mesh is illustrated in Figure 6.2(e) and Figure 6.2(f), respectively.

Traditionally, the Voronoi diagram has infinite edges going away from the boundary as depicted in Figure 6.2(c). In Figure 6.2(d) the infinite edges are cut off at the expense of leaving vertices with valency 2. We then iterate counter clock-wise around the boundary and removes all vertices with valency < 3 as they are not valid foam vertices. This procedure is carried out by traversing from an invalid vertex along the boundary until a valid valency 3 vertex is found in both outgoing directions. The first two valid vertices are then connected and all the invalid vertices, and their connected edges, in between are removed from the foam mesh. The result is illustrated in Figure 6.2(e) which now has become a valid but immature foam network.

To mature the foam network all cell target areas are initially assigned to their current polygonal area. A value of pressure is equally assigned to every cell in the foam network including the ghost bubble. In all our simulations we have been using the value of pressure unity. At this point all areas will be fully equilibrated but in general the angles will not be. We perform several foam network equilibrate iterations as described in Subsection 5.4.2 and once the complete foam network is more than 95% equilibrated we designate it as a mature foam network.

6.3 Topological Changes

From a Computer Graphics' point of view the two topological processes that can occur while equilibrating the foam mesh correspond to an edge-flip and a face-collapse in the polygonal mesh data structure. These topological changes can be implemented by employing standard mesh modifier operations on the foam mesh data which is a huge advantage.

With performance in mind it would be beneficial to write pointer manipulations directly on the data structure for changing the topology but by employing the existing atomic operations it has been faster to implement and more convenient to ensure the correctness and robustness.

The general strategy is as follows. We record the state of the current topology, create new required features, and modify the recorded state information. Next we delete all topological features that are subject to changes and finally re-create them based on the modified state information.

6.3.1 The T1 process

To illustrate the T1 topological process we will go through an example as illustrated in Figure 6.3 on how the process can be implemented by employing simple atomic polymesh operations. The T1 process can be described by four steps:

Record state For each of the four faces that are involved in the T1 process remember the vertices in ccw-order. This situation is depicted in Figure 6.3(a), and the vertex lists yield,

$$\begin{aligned} L_{\mathrm{F}_1} &= \{\mathbf{B}, \mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7, \mathbf{v}_8\}, \\ L_{\mathrm{F}_2} &= \{\mathbf{B}, \mathbf{A}, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5\}, \\ L_{\mathrm{F}_3} &= \{\mathbf{A}, \mathbf{v}_{11}, \mathbf{v}_1, \mathbf{v}_2\}, \\ L_{\mathrm{F}_4} &= \{\mathbf{A}, \mathbf{B}, \mathbf{v}_8, \mathbf{v}_9, \mathbf{v}_{10}, \mathbf{v}_{11}\}. \end{aligned}$$

Modify recorded state Create the two new vertices **C** and **D** perpendicular to the line segment $(\mathbf{B} - \mathbf{A})$ as illustrated in Figure 6.3(b). Modify the recorded vertex lists while respecting the ccw-order of the vertices. Replace **A** and **B** with the correct combinations of **C** and **D**. The modified vertex lists then become

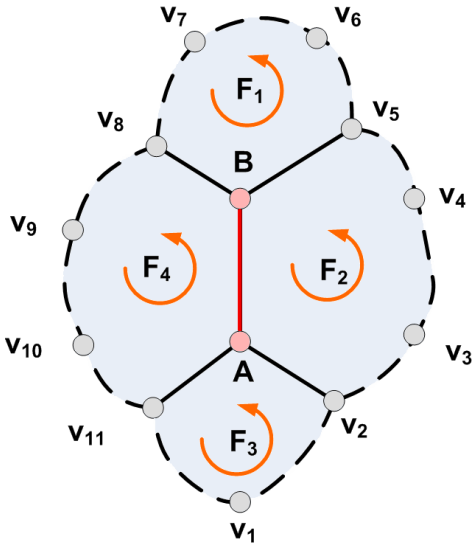
$$\begin{aligned} L_{\mathrm{F}_1} &= \{\mathbf{C}, \mathbf{D}, \mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7, \mathbf{v}_8\}, \\ L_{\mathrm{F}_2} &= \{\mathbf{D}, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5\}, \\ L_{\mathrm{F}_3} &= \{\mathbf{D}, \mathbf{C}, \mathbf{v}_{11}, \mathbf{v}_1, \mathbf{v}_2\}, \\ L_{\mathrm{F}_4} &= \{\mathbf{C}, \mathbf{v}_8, \mathbf{v}_9, \mathbf{v}_{10}, \mathbf{v}_{11}\}. \end{aligned}$$

Delete features Delete the four faces involved F_1, F_2, F_3, and F_4, and the old vertices **A** and **B**. The result is depicted in Figure 6.3(c).

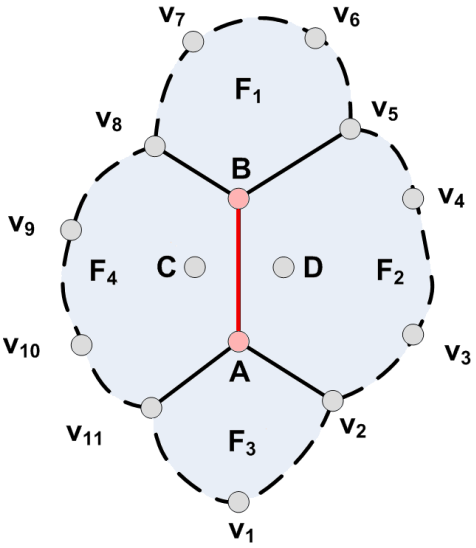
Recreate features Create new features using the modified vertex lists. The final result is illustrated in Figure 6.3(d) where the four new faces have new numbers, i.e. F_5, F_6, F_7, and F_8.

The positions of the two new vertices are placed perpendicular to the middle of the old ones. The middle position is

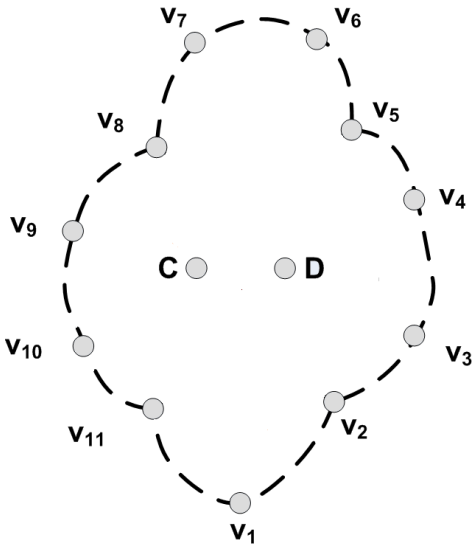
$$\mathbf{p}_{\mathrm{mid}} = \mathbf{A} + \mathbf{d}, \tag{6.1}$$



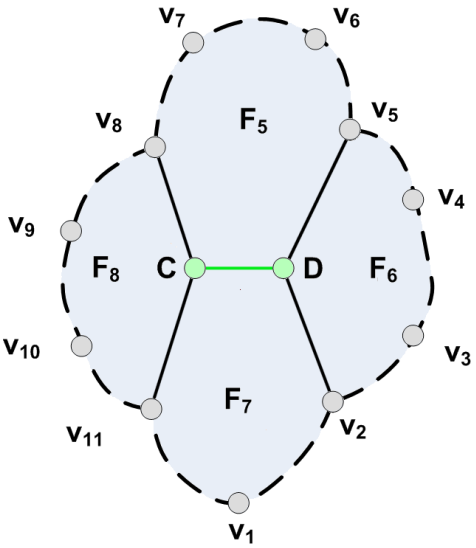
(a) Create vertex handle lists for each face.



(b) Add two new nodes that corresponds to the flipped state. Modify the vertex handle lists by replacing occurrences of nodes A and B with corresponding occurrences of nodes C and D.

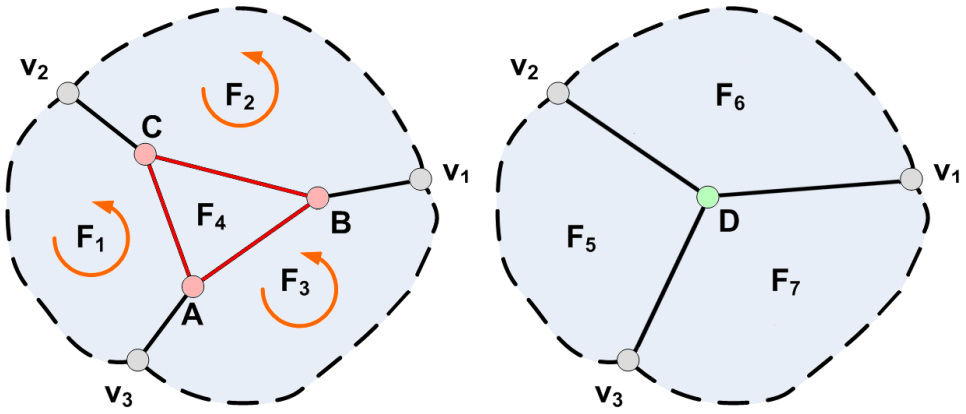


(c) Delete all faces and any isolated vertices.



(d) Use the modified vertex handle lists to create new faces corresponding to the new topological state.

Figure 6.3: Illustration of how the T1 process can be performed using atomic mesh operations.



(a) The face $F4$ is being collapsed.

(b) Any sub-pattern of the nodes A, B, and C in (a) is replaced by the single new node D.

Figure 6.4: Illustration of how the T2 process can be performed using simple atomic mesh operations.

with

$$\mathbf{d} = \frac{1}{2}(\mathbf{B} - \mathbf{A})\,. \tag{6.2}$$

The positions are computed as

$$\mathbf{p}_\mathrm{C} = \mathbf{p}_\mathrm{mid} - \hat{\mathbf{d}}, \tag{6.3}$$

$$\mathbf{p}_\mathrm{D} = \mathbf{p}_\mathrm{mid} + \hat{\mathbf{d}}, \tag{6.4}$$

where

$$\hat{\mathbf{d}} \equiv (\mathbf{d}_y, -\mathbf{d}_x)\,. \tag{6.5}$$

During the implementation it became clear that some care is necessary when replacing the old vertices with the two new ones. However, the same patterns appear to which the following replacement rules can be placed:

$$\mathbf{AB} \mapsto \mathbf{C}, \tag{6.6}$$

$$\mathbf{BA} \mapsto \mathbf{D}, \tag{6.7}$$

$$\mathbf{A} \mapsto \mathbf{DC}, \tag{6.8}$$

$$\mathbf{B} \mapsto \mathbf{CD}. \tag{6.9}$$

6.3.2 The T2 process

Given the details from the T1 process in the previous subsection the T2 process is handled nearly the same way. The following example make use of Figure 6.4.

Record state For the four faces that are involved in the T2 process remember the vertices in ccw-order for the three faces that will be recreated,

$$\begin{aligned} L_{\mathrm{F}_1} &= \{\mathbf{A}, \mathbf{C}, \mathbf{v}_2, \mathbf{v}_3\}, \\ L_{\mathrm{F}_2} &= \{\mathbf{C}, \mathbf{B}, \mathbf{v}_1, \mathbf{v}_2\}, \\ L_{\mathrm{F}_3} &= \{\mathbf{B}, \mathbf{A}, \mathbf{v}_3, \mathbf{v}_1\}. \end{aligned}$$

Modify recorded state Create the new vertex **D** and modify the vertex lists accordingly. Each succession of vertices from the T2 cell is simply replaced with the new vertex, as

$$\begin{aligned} L_{\mathrm{F}_1} &= \{\mathbf{D}, \mathbf{v}_2, \mathbf{v}_3\}, \\ L_{\mathrm{F}_2} &= \{\mathbf{D}, \mathbf{v}_1, \mathbf{v}_2\}, \\ L_{\mathrm{F}_3} &= \{\mathbf{D}, \mathbf{v}_3, \mathbf{v}_1\}. \end{aligned}$$

Delete features Delete all four faces involved F_1, F_2, F_3, and F_4, and the old vertices **A**, **B**, and **C**.

Recreate features Create new features using the modified vertex lists. The final result is illustrated in Figure 6.4(b) where the three new faces have new numbers, i.e. F_5, F_6, and F_7.

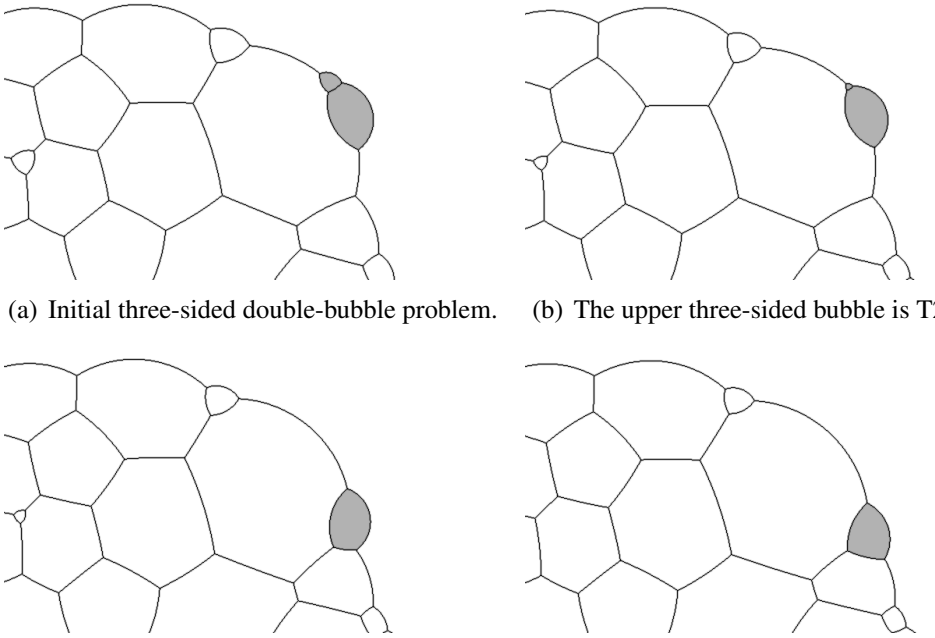
The position of the new vertex is created as the average of the three old ones and yields

$$\mathbf{p}_{\mathrm{D}} = \frac{1}{3}(\mathbf{A} + \mathbf{B} + \mathbf{C}). \tag{6.10}$$

6.3.3 The three-sided double-bubble problem

When two three-sided bubbles are sharing the same film we call that configuration a *three-sided double-bubble*. Examples of three-sided double-bubbles are depicted in Figure 6.1 and Figure 6.5(a). If we assume that the foam network undergoes gas discussion the three-sided double-bubble is affected by the polymesh limitation as mentioned in Subsection 6.1.2. One of the three-sided bubbles will perish due to a T2 operation and leave the other bubble ill-defined. The polymesh data structure do not allow for two-sided bubble to be created. On one hand we cannot allow a T2 change to occur on a three-side double-bubble but on the other hand we cannot just ignore the situation either as local instability will occur to the foam mesh.

We move away a bit from physical realism to find an alternate way to handle this problem. For the sake of stability and the quality of the foam animation we will slide the three-sided double-bubble along the outgoing film of the largest of the two bubbles as this bubble will perish last. When the bubble approaches the forthcoming adjoining vertex it



(a) Initial three-sided double-bubble problem.

(b) The upper three-sided bubble is T2 ready.

(c) Before the T2 bubble is eliminated both three-sided bubbles are pushed along the outgoing film of the largest bubble.

(d) The remaining three-sided bubble has been equilibrated.

Figure 6.5: The three-sided double-bubble problem is handled in a non-physically correct way. However, animation-wise it serves the method well and increases stability of the simulation.

will ultimately suffer from a T1 process, and be converted into a four-sided bubble. The remaining three-sided bubble can now safely undergo the T2 process and perish. This strategy to handle the three-sided double-bubble problem is visualized in Figure 6.5.

The foam network transformation for the three-sided double-bubble is not physically correct as it is merely a neat animation trick to handle the situation. However, this trick is not just randomly chosen. The behavior is closely related to the physical behavior of a single metastable two-sided cell in a dry foam [Weaire and Kermode, 1983].

6.4 Dynamic Boundary Conditions

The ghost bubble method and the associated mathematically boundary conditions are presented in Section 5.1 (p. 38). To illustrate how the dynamic boundary conditions can be applied to the existing matrix solver we use Figure 6.6 as reference. In Figure 6.6 **v** is the vertex to be equilibrated. For this vertex **v** bubble j is the ghost bubble, leaving bubbles i and k as actual foam bubbles.

In the following subsections we will discuss how to perform the necessary changes to

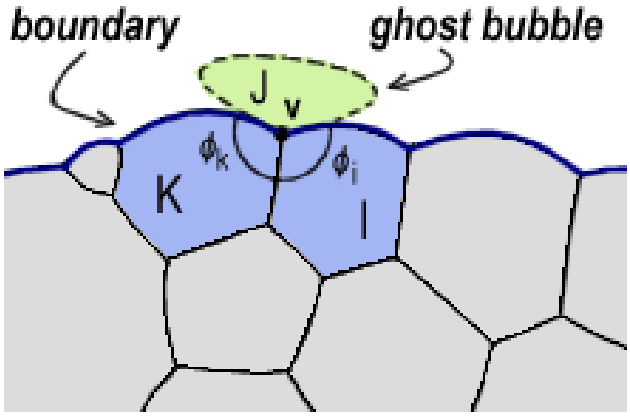


Figure 6.6: The Ghost Bubble Method in action.

the system matrix **A** to take the ghost bubble boundary conditions into account.

6.4.1 Ghost area and pressure

From Equation (5.5) and Equation (5.6) we know that to handle the dynamic boundary conditions mathematically we store zeros in the *ghost*-index row and column of the system matrix **A**. Focusing on the first three, area-based rows of **A** while using the example in Figure 6.6, where index $j = ghost$, the system matrix becomes

$$\mathbf{A} = \begin{bmatrix} \frac{\partial A_i}{\partial p_i} & 0 & \frac{\partial A_i}{\partial p_k} & \frac{\partial A_i}{\partial x} & \frac{\partial A_i}{\partial y} \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\partial A_k}{\partial p_i} & 0 & \frac{\partial A_k}{\partial p_k} & \frac{\partial A_k}{\partial x} & \frac{\partial A_k}{\partial y} \\ - & 0 & - & - & - \\ - & 0 & - & - & - \end{bmatrix}, \tag{6.11}$$

where the symbol "−" means that this part of the matrix does not concerns areas.

6.4.2 Ghost angle and position

From Subsection 5.1.1 (p. 39) we know that any two of the three vertex angles can be employed in the linear matrix system (4.20). Again turning our attention towards the example in Figure 6.6 the ghost bubble is the j-index. We now use the i-indexed and k-indexed angles as the last two, angle-based rows of **A** and the system matrix becomes

$$\mathbf{A} = \begin{bmatrix} - & 0 & - & - & - \\ 0 & 0 & 0 & 0 & 0 \\ - & 0 & - & - & - \\ \dfrac{\partial \phi_i}{\partial p_i} & 0 & \dfrac{\partial \phi_i}{\partial p_k} & \dfrac{\partial \phi_i}{\partial x} & \dfrac{\partial \phi_i}{\partial y} \\ \dfrac{\partial \phi_k}{\partial p_i} & 0 & \dfrac{\partial \phi_k}{\partial p_k} & \dfrac{\partial \phi_k}{\partial x} & \dfrac{\partial \phi_k}{\partial y} \end{bmatrix}, \tag{6.12}$$

and likewise where the element symbol "−" means that this part of the matrix does not concerns angles.

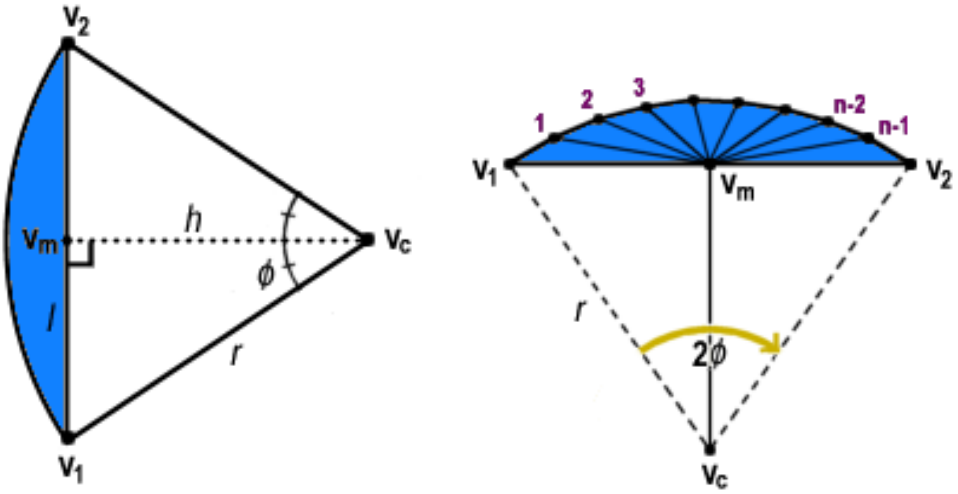
6.4.3 Ghost matrix system

Using Figure 6.6 as a running example on how to implement the matrix system when dealing with the ghost bubble we can now combine the results of (6.11) and (6.12) into the full matrix system which then becomes

$$\begin{bmatrix} \dfrac{\partial A_i}{\partial p_i} & 0 & \dfrac{\partial A_i}{\partial p_k} & \dfrac{\partial A_i}{\partial x} & \dfrac{\partial A_i}{\partial y} \\ 0 & 0 & 0 & 0 & 0 \\ \dfrac{\partial A_k}{\partial p_i} & 0 & \dfrac{\partial A_k}{\partial p_k} & \dfrac{\partial A_k}{\partial x} & \dfrac{\partial A_k}{\partial y} \\ \dfrac{\partial \phi_i}{\partial p_i} & 0 & \dfrac{\partial \phi_i}{\partial p_k} & \dfrac{\partial \phi_i}{\partial x} & \dfrac{\partial \phi_i}{\partial y} \\ \dfrac{\partial \phi_k}{\partial p_i} & 0 & \dfrac{\partial \phi_k}{\partial p_k} & \dfrac{\partial \phi_k}{\partial x} & \dfrac{\partial \phi_k}{\partial y} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \Delta A_i \\ 0 \\ \Delta A_k \\ \Delta \phi_i \\ \Delta \phi_k \end{bmatrix}. \tag{6.13}$$

The assembly of the system matrix is performed for each vertex in turn it is now a matter of detecting if one of the bubbles in fact is the ghost bubble. Recall from Subsection 6.1.1 that a bubble is tagged as being the ghost bubble if this is the case. If it turns out that the ghost bubble is among the three vertex bubbles we omit all of the computations for the numerical derivatives for this bubble and assign zeros to the appropriate elements in the matrix.

Note, in a previous attempt we computed the total area of the ghost bubble each time a vertex was on the boundary which not only was incorrect since the foam is rarely convex but also computational expensive. All the ghost bubble computations was then wasted because the results were overwritten with zeros. Also note that it is really important to make sure the result vector **b** and the rows of the system matrix **A** are synchronized, especially the angle indices should follow the same pattern when skipping the ghost bubble.



(a) Vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ and the radius r are known. Standard trigonometry can be employed to compute $\mathbf{v}_c$ and the angle ϕ.

(b) The arc can be drawn piecewise linearly by rotating $\mathbf{v}_1$ around $\mathbf{v}_c$ clockwise in n steps with $\theta = \frac{2\phi}{n}$ per step. To draw a filled arc the midpoint $\mathbf{v}_m$ is included to draw triangles.

Figure 6.7: We way we draw an arc. (a) The symbols used in the drawing process are depicted. (b) To draw the discrete arc in wireframe line segments are drawn from $\mathbf{v}_1$ to $\mathbf{v}_2$ by rotating around $\mathbf{v}_m$. To fill the discrete arc the midpoint $\mathbf{v}_m$ is simply added to the line segments to form triangles.

6.5 Foam Rendering

In Section 4.5 (p. 37) we briefly discussed that a convenient way to visualize the foam is to render the cell films as circular arcs. By default OpenTissue employs OpenGL for rendering but unfortunately no drawing method exists in OpenGL that supports rendering of arcs both in wireframe and in solid[2]. We now show how arcs can be drawn manually in either wireframe or as filled triangles using standard OpenGL drawing methods.

6.5.1 Drawing arcs

To draw an arc the idea is to approximate it by a number of linear line segments between two given points. This can be done by rotating one of the points towards the other one along the arc that is defined by the radius. As an arc is a partial circle we need to find the center of the full circle before we can rotate the point and thus draw the line segments.

[2]The OpenGL utility gluPartialDisk method can be used to render an arc alone in wireframe although some algebraic transformations must be carried out before the vertex model output data can be used directly. However, it is not fit to draw a solid arc as the whole pizza slice will be drawn in this case and not only the arc top.

In the following we refer to Figure 6.7(a). We seek to compute the rotation point, or center of the circle, $\mathbf{v}_c$, and the angle ϕ, which is half the rotation angle. The midpoint between the two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ is given by

$$\mathbf{v}_m = \frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)\,. \tag{6.14}$$

Half the vector between $\mathbf{v}_1$ and $\mathbf{v}_2$ is

$$\begin{aligned}\mathbf{l} &= \frac{1}{2}(\mathbf{v}_2 - \mathbf{v}_1) \\ &= \mathbf{v}_m - \mathbf{v}_1\,,\end{aligned} \tag{6.15}$$

which has the length

$$l = \|\mathbf{v}_m - \mathbf{v}_1\|\,. \tag{6.16}$$

From trigonometry the angle ϕ yields

$$\phi = \sin^{-1}\left(\frac{l}{r}\right), \tag{6.17}$$

and the height h is computed[3] as

$$h = r\cos\phi\,. \tag{6.18}$$

The center of the circle can now be computed as

$$\mathbf{v}_c = \mathbf{v}_m + h\frac{\mathbf{u}}{\|\mathbf{u}\|}, \tag{6.19}$$

with

$$\mathbf{u} = \hat{\mathbf{l}} = (\mathbf{l}_y, -\mathbf{l}_x)^T, \tag{6.20}$$

where we have exploited the fact that we are working in two dimensions and orthogonality is a simple operation.

To compute any point $\mathbf{p}$ on the circular arc we rotate the point $\mathbf{v}_1$ around $\mathbf{v}_c$ as

$$\mathbf{p}(\alpha) = \mathbf{R}(-\alpha)\,(\mathbf{v}_1 - \mathbf{v}_c)^T + \mathbf{v}_c\,, \qquad 0 \le \alpha \le 2\phi\,, \tag{6.21}$$

where $\mathbf{R}$ is the 2×2 rotation matrix defined as

$$\mathbf{R}(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}. \tag{6.22}$$

The reason for the negation of α in (6.21) is due to the fact that $\mathbf{R}$ rotates counter clockwise as it is the positive orientation in a right hand system but we want to rotate clockwise from $\mathbf{v}_1$ to $\mathbf{v}_2$. Different points $\mathbf{p}(\alpha)$ and $\mathbf{p}(\alpha+\delta)$ on the arc can be computed incrementally to draw line segments and ultimately draw the arc in wireframe. A solid arc is obtained by including the midpoint (6.14) and then draw filled triangles instead of line segments as depicted on Figure 6.7(b).

[3]The height h can also be computed using Pythagoras, $h = \sqrt{r^2 - l^2}$

Chapter 7

Evaluation and Analysis

"It requires a very unusual mind
to undertake the analysis of the obvious."

- Alfred North Whitehead

The implementation has been compiled into an application that can demonstrate all the parts of the vertex-based, two-dimensional, dry foam model that has been derived through the thesis. The application has not been written with optimization in mind. The applied policy-based programming paradigm is not the way to go if high performance and high frame rates are the main achievements. What we can achieve is good and stable simulation results.

In this chapter we will conduct several tests to evaluate the convergence rate and performance of the vertex-based foam model in the given application. We will finalize the chapter by performing visual comparisons of our own results against results from other foam simulators.

7.1 Stability analysis

Stability is vital for the vertex-based foam model to converge. It is not likely that the foam network will converge if the relaxation parameter, see Subsection 5.3.1 (p. 42), is set too high $\tau \rightarrow 1$. This is especially the case during initialization from Voronoi diagram to an equilibrated and mature foam network.

In the following we will measure the stability of the foam model by performing a number of tests. The measurements are solely based on visual validation. As soon as instability occurs a test is interrupted and marked as unstable.

τ	Voronoi Stability [%]	Diffusion Stability [%]
1.0	5	80
0.9	8	90
0.8	16	100
0.7	33	100
0.6	48	–
0.5	72	–
0.4	79	–
0.3	83	–
0.2	94	–
0.1	99	–

Table 7.1: The distribution of stability based on τ. 100 executions per τ-value have been performed for the Voronoi stability tests but only 10 executions for the diffusion stability tests. For all test executions 10 iterations per vertex has been used.

7.1.1 Initial Voronoi stability

To get an overview on the foam stability as a function of τ we have executed a series of initialization tests of random Voronoi diagrams with different values of τ. Each Voronoi diagram contained approximately 200 cells on average and was tested 100 times per τ value using 10 iterations per vertex.

Based on the experiments with the model the expected outcome of the stability measurements is $0.1 \leq \tau < 0.2$. Table 7.1 shows the results which has been plotted onto graphs in Figure 7.1. Paying attention to the values it is clear that when performing the initial equilibration from a Voronoi diagram to foam mesh then

$$\tau_{\text{init}} = 0.1\,.$$

It is true, as can be witnessed on the graph in Figure 7.1, that setting $\tau_{\text{init}} = 0$ yields a 100% stability but doing so will also result in divergence as the system will be idling.

7.1.2 Mature foam stability

Continuing the τ-stability analysis after the initial equilibration of the foam mesh brings us to the foam simulation of a mature foam network where the foam undergo diffusion followed by the mandatory equilibration process. Starting with a random mature foam mesh of approximately 200 cells we ran 10 executions per τ-value for about $100-200$

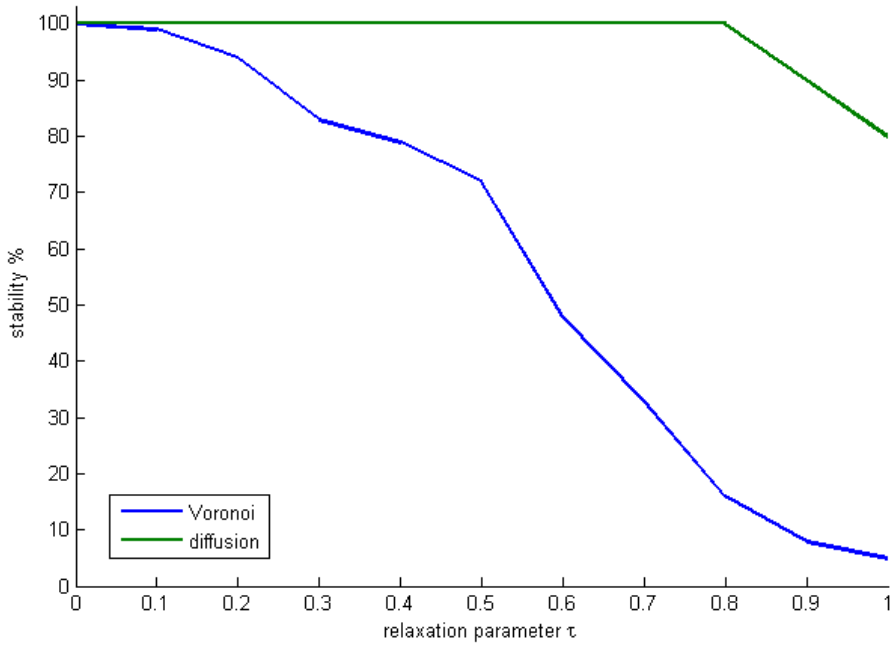


Figure 7.1: The results from the stability tests from Table 7.1.

frames. If the simulation became unstable during equilibration it was stopped and marked as such.

We expect that the value of the relaxation parameter can be quite high $\tau \geq 0.5$ as the model seems quite stable as soon as it has been equilibrated. In Table 7.1 the results are shown next to the initialization stability results, and the graph is visualized in Figure 7.1. It became clear already after a couple of simulations that τ could be set to a high value. The simulations can safely use

$$\tau_{\text{sim}} = 0.7\,.$$

We realize that choosing $\tau_{\text{sim}} = 0.7$ might have been done prematurely as we have only performed 10 tests per τ-value. Nonetheless, we ran several hundred frames per test without any stability issues.

7.2 Sensitivity discussions

In the previous section we analyzed the stability of the line search damping parameter τ but other parameters have been introduced to the foam model as well. We will briefly discuss the parameters that were introduced in Chapter 3 (p. 12), their influences, and ranges.

7.2.1 Surface tension

The surface tension parameter γ basically models the surface normal forces in the liquid of the foam films. Surface tension has the dimension of force per unit area, or force per unit length in the two-dimensional case,

$$\gamma = \left[\frac{F}{l}\right],$$

and the SI units is often given milli-Newton per meter or dynes per centimeter,

$$1\,\frac{mN}{m} = 1\,\frac{dyne}{cm} = 0.001\,\frac{N}{m}.$$

Surface tension is used in the law of Laplace-Young that relates cell pressure with the radius of curvature as

$$\Delta p = \frac{2\gamma}{r}. \tag{7.1}$$

The surface tension must be greater than zero,

$$\gamma > 0,$$

before (7.1) is valid and makes sense. Increasing the surface tension will not render the equilibrium process unstable as the pressure is balanced against the surface tension and will be increased accordingly. Film fracture is not included in our implementation of the dry foam model. Otherwise increasing the surface tension would have a significant impact on the stability of the films as the forces due to surface tension will tear the foam films apart as the surface tension is reaching the limit of normal water at $20\,^{\circ}C$[1] which is

$$\gamma \approx 73\,\frac{mN}{m}.$$

Surface tension is also included in the computation of the diffusivity, or gas diffusion constant, K together with the permeability constant κ in von Neumann's diffusion law as

$$K = \frac{2\pi}{3}\gamma\kappa, \tag{7.2}$$

and this is discussed next.

7.2.2 Permeability and diffusivity

The permeability κ for a foam describes the ability of penetration of gas through the liquid films. Together with the surface tension γ the permeability is the main factor of (7.2). From von Neumann's law which yields

$$\frac{\mathrm{d}A_n}{\mathrm{d}t} = K(n-6), \tag{7.3}$$

[1]The surface tension of water can be looked up in books on material science and many places on the web. We have googled the provided result.

it is noticeable that the diffusivity or diffusion constant K takes the general form of a spring constant with (7.3) as the one-dimensional Hookean spring. The normal stability issues that are in evidence when K becomes too large are not as important in this situation as the foam in time is bound to perish and therefore spring oscillations are non-existing.

The diffusivity K has the dimensions length squared per unit time

$$K = \left[\frac{l^2}{t}\right],$$

which has the SI units of

$$1\ \frac{mm^2}{s} = 10^{-6}\ \frac{m^2}{s}.$$

K must be non-negative

$$K \geq 0,$$

where $K = 0$ applies that gas diffusion has been omitted. From the previous subsection we know that $\gamma > 0$ which applies that

$$\kappa \geq 0.$$

We have evaluated the implemented the foam model by running several hundred simulations with different number of cells in different area sizes while varying κ from $\kappa < 10^{-3}$ to $\kappa > 10^{3}$ without any technical instabilities. When a high value of κ is employed the foam is collapsing very rapidly which almost behaves like plain water foam, i.e. a very short foam lifespan.

7.2.3 Topological thresholds

Throughout the thesis we have not been discussing the distance and area dimensions we are working with but in Section 3.1 (p. 12) we stated that the scope of the thesis was defined to be on the length-scales of $1 - 100$mm. The topological changes are executed when their conditions are met. For a T1 process the condition threshold is T_{dist} and it was defined in (3.12) and later updated in (5.10). Similar for a T2 process the condition threshold is T_{area} and it was defined in (3.13) and later extended in (5.7).

For the sake of generalization we assume that the threshold are given in unit length for distances and unit squared length for areas. From the definitions of the threshold we know that they must be greater than zero which make sense as are we are working with length dimensions and the thresholds have been introduced to avoid numerical issues when values become zero. On the other hand the thresholds should not be chosen to be too large as this will contradict their purpose and provoke topological changes prematurely. Based on our experiments and tests we will argue that the thresholds can be set safely to be in the range

$$0.01 \leq T_{\text{dist}}, T_{\text{area}} \leq 0.25,$$

and these ranges also prevent the topological changes from creating unstable situations. We have chosen $T_{\text{dist}} = 0.05$ and $T_{\text{area}} = 0.05$ which have proven to be stable values for the topological thresholds in all simulations.

7.3 Convergence rate analysis

It is time to focus on the convergence rate for the nonlinear Newton method. We will measure the deviations from absolute equilibrium at vertex level and at foam level. We assume there will be a significant difference between the number of iterations that are required to bring a single vertex into absolute equilibrium and the number of iteration it will take to bring every vertex into equilibrium simultaneously.

7.3.1 Deviation means

From Subsection 3.1.3 (p. 15) we can measure the mean cell area and the mean vertex angle over all cells and vertices. To learn something about the convergence of the Newton method and the system in general we will measure the error as the mean deviation from equilibration.

Let the mean cell area deviation be given by

$$\overline{A}_\varepsilon = \frac{1}{n}\sum_n |\Delta A_n|,$$

where the sum is over all cells, and let the mean vertex angle deviation by given by

$$\overline{\phi}_\varepsilon = \frac{1}{3|V|}\sum_n |\Delta \phi_n|,$$

where the sum is over all vertex angles and V is the set of all vertices. The two differences have been defined in Subsection 4.3.2 (p. 33) but repeated here for convenience

$$\Delta A_n = A_n^{target} - A_n,$$

$$\Delta \phi_n = \frac{2\pi}{3} - \phi_n.$$

To compare the mean deviations against the absolute means which are

$$\overline{A} = \frac{1}{n}\sum_n A_n^{target},$$

and the absolute vertex mean as Plateau's 120°-rule, and to unify the results we define the absolute percentage equilibration of cell area as

$$A_\% = 1 - \frac{\overline{A}_\varepsilon}{\overline{A}},$$

and the absolute percentage equilibration of vertex angle yields

$$\phi_\% = 1 - \frac{3\overline{\phi}_\varepsilon}{2\pi}.$$

For the total percentage equilibration for the whole foam network we simple take the mean of the absolute percentage equilibria,

$$T_\% = \frac{A_\% + \phi_\%}{2}.$$

7.3.2 Error plots

To measure the deviation from absolute equilibrium we will plot the errors onto graphs for further analysis. The convergence rates tests of the nonlinear Newton method will be split into two parts, one at vertex level, which will measure the pure Newton method, and one at foam level. Plotting the deviation results at logarithmic scale the shape of the graphs can be mapped to a convergence rate directly.

Newton method convergence rate

For the deviation errors at vertex level we will use a newly created foam mesh with random noise applied to all vertices which employ that neither angles nor areas are near equilibrium. We will measure the deviation from absolute equilibrium of 12 random selected vertices, 6 bulk vertices and 6 boundary vertices. The vertices are all chosen not to be in the adjacent neighborhood of any of the other vertices, i.e. a vertex is not to be located in any of the cell perimeters of the other vertices. It is only the chosen vertices that will be equilibrated in turn using 1 iteration per vertex with a constant $\tau = 0.7$ which was found to be a stable value in Section 7.1 for a mature foam. The absolute deviation from equilibrium are then recorded and plotted onto convergence graphs.

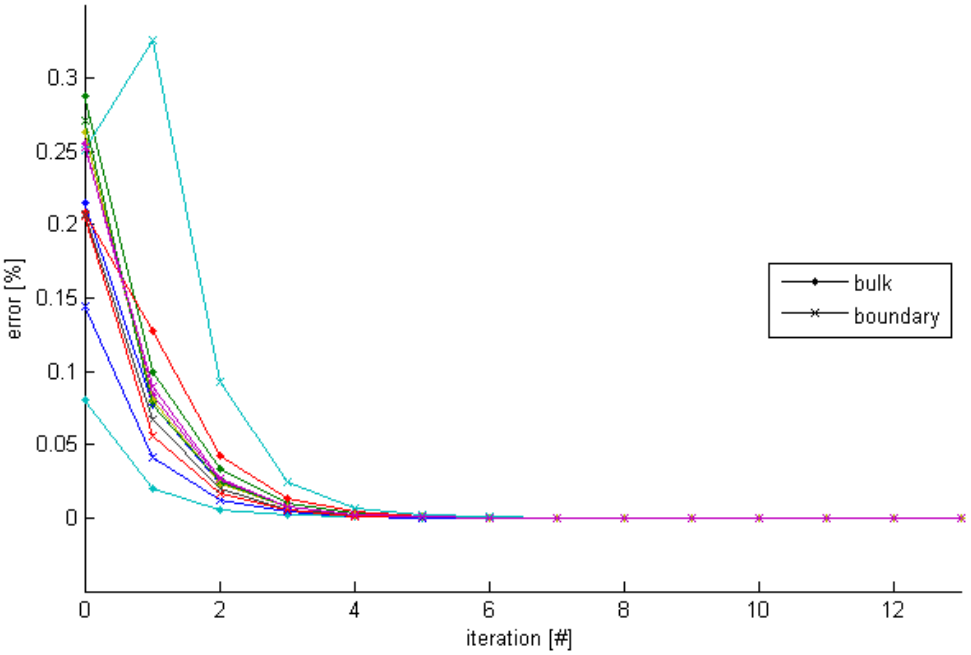
From [Nocedal and Wright, 2006] it is stated that the nonlinear Newton line search method at best can have quadratic convergence rate. Experiences with our nonlinear Newton solver do not indicate quadratic convergence but we expect that the convergence rate for the solver will be superlinear. The results are depicted in Figure 7.2 where the secondary axis indicate the error in percentage $(0..1)$. At first the graphs in Figure 7.2(b) show linear convergence but with a convergence constant of $\lambda \geq 0.5$ which can be verified in Figure 7.2(a).

The topmost graph in Figure 7.2 actually increases the error after the first iteration. This is a boundary vertex that likely has been ill-defined locally. Still the solver managed to bring the vertex into equilibrium.

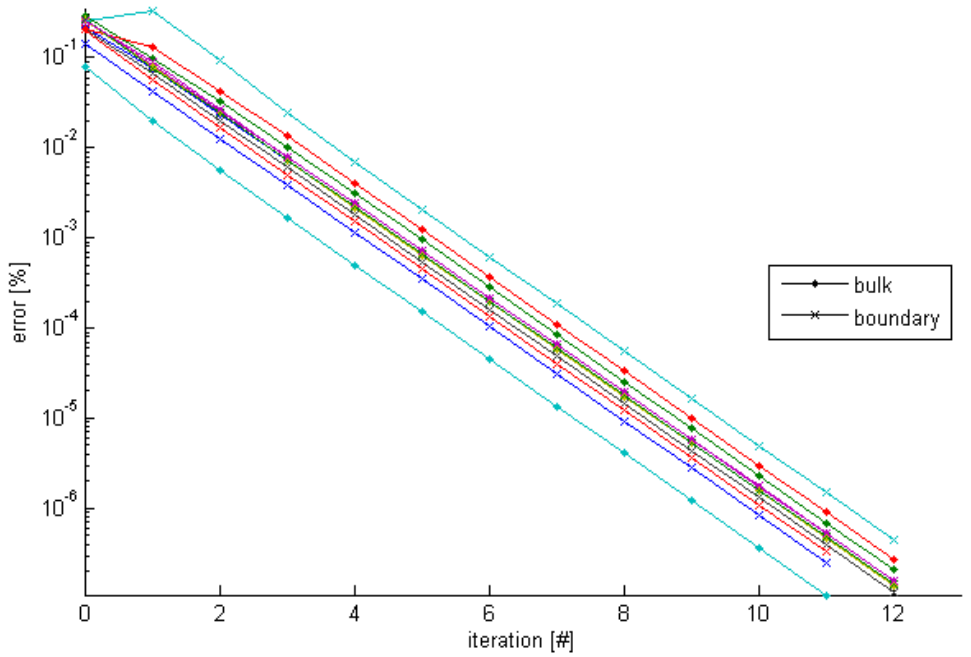
Foam system convergence rate

From Figure 7.2(a) we can determine that using 10 iterations each independent vertex will always be fully equilibrated. When all vertices are equilibrated in succession a total state of equilibrium for the foam network cannot be guaranteed because each vertex potentially can bring its adjacent vertices away from their own local equilibrium. This is a well-known behavior for any relaxation-based method. To let the whole foam network converge to equilibrium is a matter of iterating through all vertices continuously until the total foam equilibrium has been reached.

Initially, we will diffuse a foam network with 200 cells and add some random noise to all the vertices to approximately get an area and angle error of 10%. After each foam iteration we monitor the total deviation from absolute equilibrium for the whole foam network and plot the results. As the Newton method for a single vertex has linear convergence rate we will expect that the convergence rate for the foam solver similar is close to

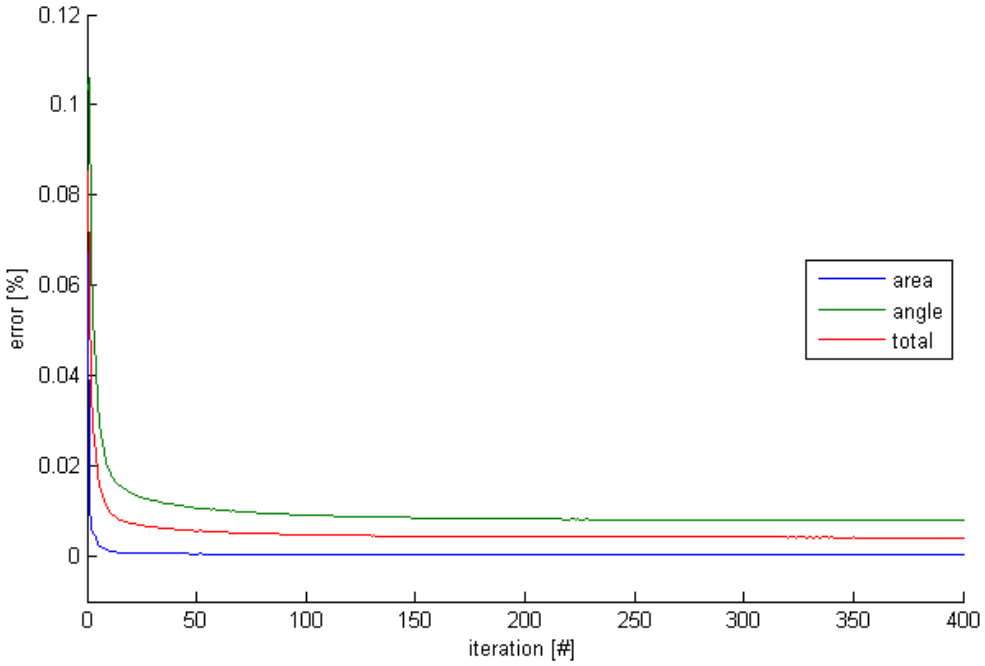


(a) Vertex errors from absolute equilibrium.

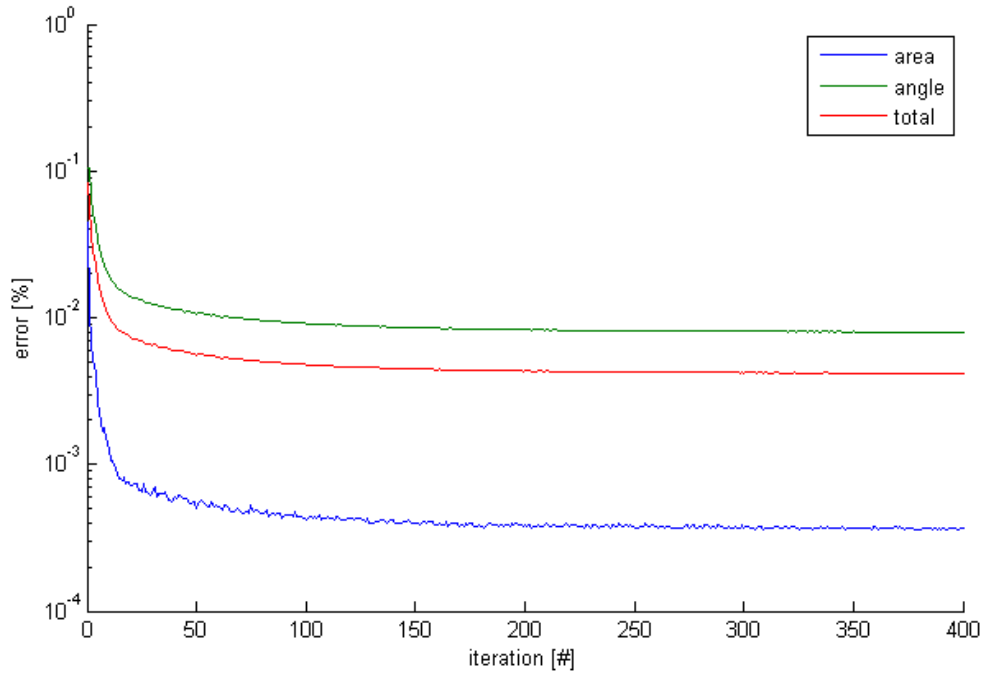


(b) Same data set as (a) but in logarithmic scale.

Figure 7.2: Error plots of 6 isolated bulk vertices and 6 isolated boundary vertices. In each equilibration iteration we used $\tau = 0.7$.



(a) Foam network errors from absolute equilibrium.



(b) Same data set as (a) but in logarithmic scale.

Figure 7.3: Error plots a foam with 200 cells (and 398 vertices). For each foam iteration we used 10 iterations per vertex with $\tau = 0.7$.

linear.

In Figure 7.3 the results are depicted. The graphs show that the foam solver will never converge to total equilibrium but instead it looks like the total graph converges to another limit just below 1% deviation from total equilibrium. This limit tells us that the chosen solver might not be good enough to get the foam into total equilibrium. However, the limit can also be an evidence of an amount of small numerical imprecisions that build up a larger error. From Figure 7.3(b) we have the resemblance of a superlinear convergence rate for the whole system solver.

7.4 Performance

Evaluation of performances is interesting because it can reveal if parts of the implementation is not performing correctly as otherwise expected. Thus, the performance rates will not be compared to those of an interactive application in Computer Graphics. We will measure the time and memory performance by running the foam application and monitor computation time and memory consumption.

According to Euler's equation (3.1) there is a linear dependency between the foam features, e.g. an increase in cells always applies a linear increase in vertices and edges, but whether the increase in memory and time is linear dependent on the amount of cells in foam network is to be investigated.

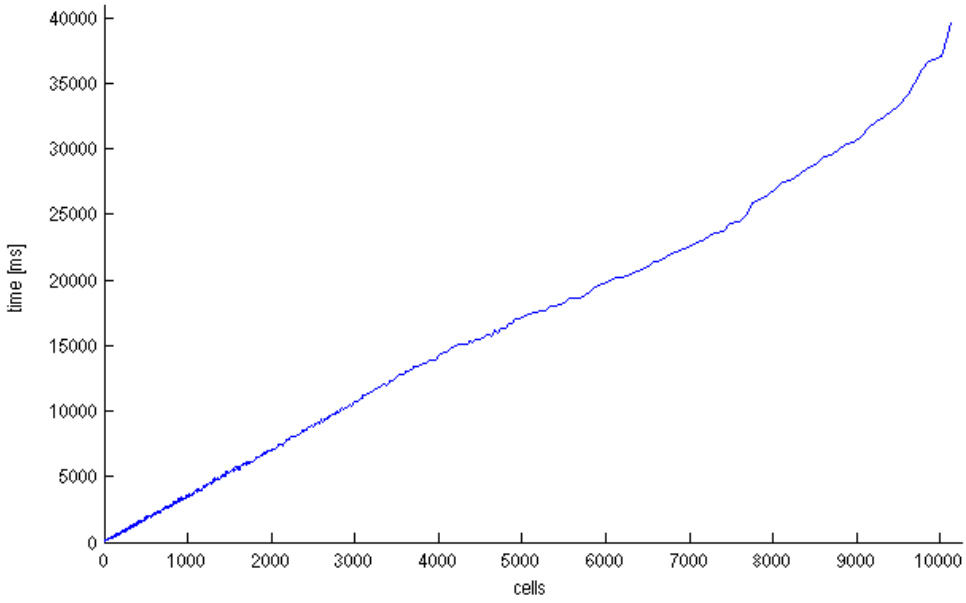
7.4.1 Time consumption

The time performance test will measure the time spent on each simulation step of a mature foam network that is allowed to evolve over time. The gas diffusion is employed as a mean to make changes to the whole network and to measure simulation time as a function of the number of cells. The initial foam network is built from random sites, which after the Voronoi network correction process ends up with just above $10,000$ cells. The foam network is initially equilibrated to $T_{\%} \approx 99$.

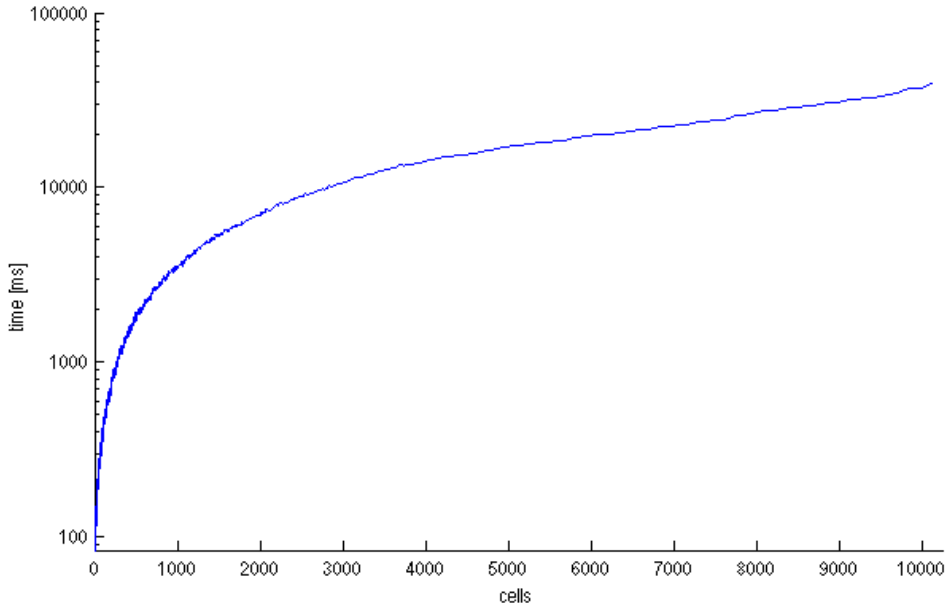
The foam simulation will use 10 iterations per vertex with a relaxation parameter $\tau = 0.7$ and with this Newton configuration the simulation will in general equilibrate the foam within $T_{\%} = 99.2 \pm 0.2$. The expectations from the time performance will be that the time to execute a single simulation will scale linearly as the number of cells are increasing. As we start with a newly equilibrated foam network with approximately $10,000$ cells, we will actually monitor the time performance as the number of cells decrease over time.

The simulation times are shown in Figure 7.4. From the shape of the graph in Figure 7.4(a) it is tempting to state, besides a small deviation, that the performance time is linearly dependent on the amount of cells in the foam network. However, Figure 7.4(b) reveals that from approximately $8,000$ cells the the graph has a small exponential growth that no longer implies linearity.

The reason for the exponential growth is a bit hidden. In reality the graph time span should be read from right to left, as we are evolving the foam network by employing



(a) Performance time measured for an evolving foam network.



(b) Same data set as (a) but in logarithmic scale.

Figure 7.4: Time performance graph for a mature foam network evolving over time. Initial cells is 10k with a frame time of 39.5s. The last frame contains 11 cells and took 84ms. Throughout the foam evolution $\tau = 0.7$ with 10 iterations per vertex.

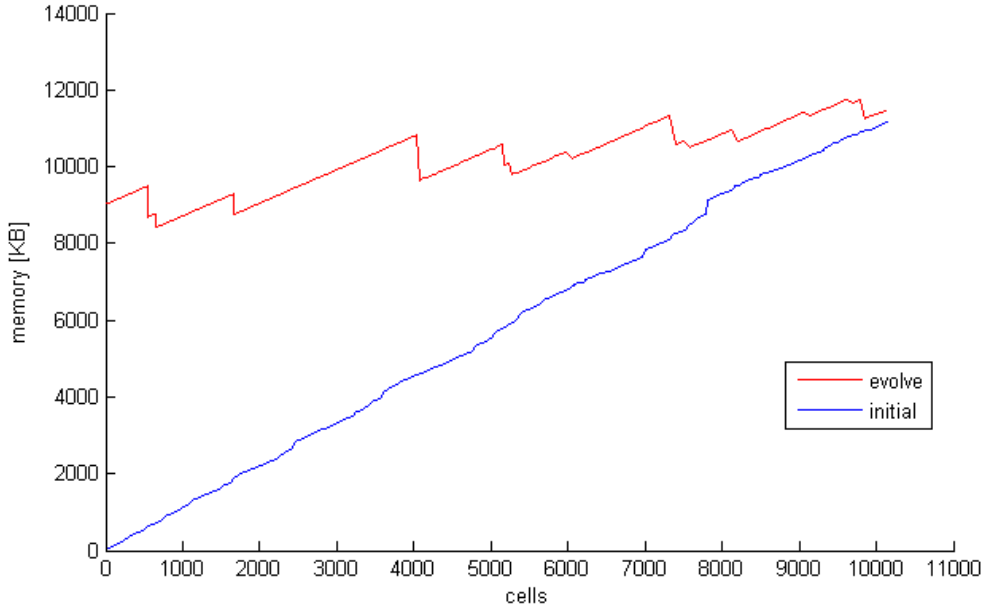
coarsening. When the evolution of the foam begins the system contains many small-sized cells in area and vertices, all of which imply mandatory topological changes. During the first 50 Newton iterations the foam approximately decreases from $10,000$ to $8,000$ cells. Compared to a normal vertex that undergoes equilibration, a topological change requires additional work, e.g. maintenance of data containers, additional equilibration of newly added vertices, etc. This rapid development in the foam network, at the beginning of the foam evolution, is the cause of the time increase.

7.4.2 Memory consumption

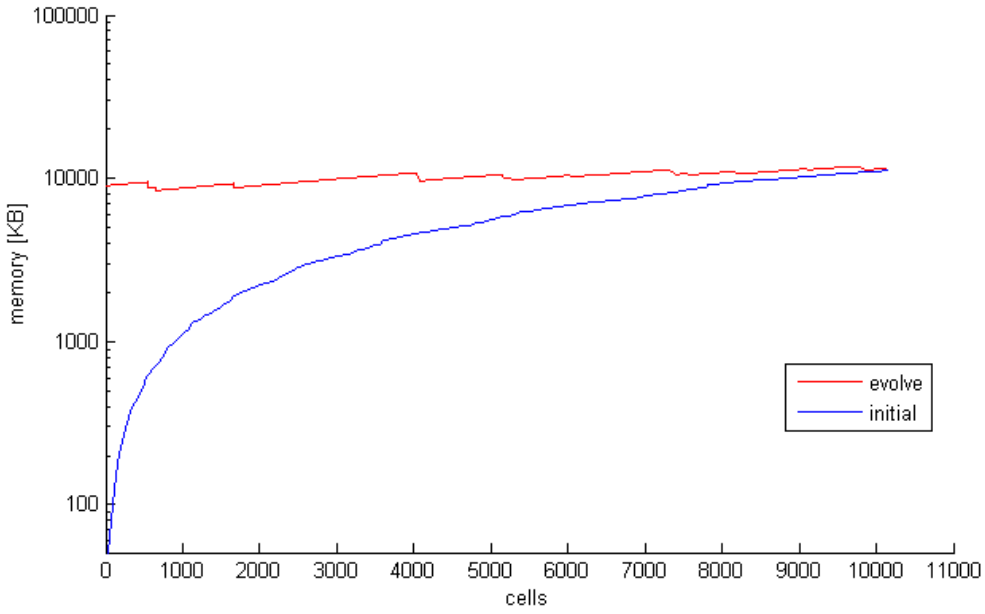
Like with the time performance measurements we expect that the memory consumptions will scale linearly with the number of cells. The same foam network and simulation as described in Subsection 7.4.1 above is used for these memory measurements. Keep in mind that the foam network is evolving and will decrease in number of cells over time, which likewise should imply a decrease in memory consumption.

The memory measurements from the foam evolution are shown in Figure 7.5 (the red graph). It is clear the the memory performance does not linear decrease as the cells decrease. As a matter of fact the memory performance seems highly irrational, which is shown in the graph as a sawtooth pattern. However, a valid explanation exists for this behavior. The implementation heavily relies on the *Standard Template Library* (STL) that today is a permanent extension to the C++ programming language. Among other things STL provides a large set of data containers, all of which employ individual memory allocator strategies. Many such STL data containers have been employed in our vertex-based foam implementation, and in OpenTissue as well. To cope with the requirements of fast data insertions, removal, and look up in the various containers, memory handling is not a priority. The reason for the appearance of the graph can be excused for this very reason. Memory is simply not deallocated from the containers each time a data element is removed. We have not investigated why the sawtooth pattern appears, but a rough estimate could be that it is related to all the topological operations that removes several features, after which most are added again. On the other hand we can verify that memory is not increasing beyond the memory consumption that has been allocated at the time of initialization.

To get a more accurate image of how the true memory consumption scales with the number of cells, it has been necessary to perform another type of performance test. Random foam networks will be generated with approximately an increasing 50 cells and then equilibrated to $T_{\%} \approx 92$. The allocated memory is then measured and the results have been integrated into Figure 7.5 (the blue graph). From the shape of this graph we can state that the memory consumption is linearly dependent on the amount of cells in the foam network, which also is supported by Figure 7.5(b) as the graphs seem to flatten.

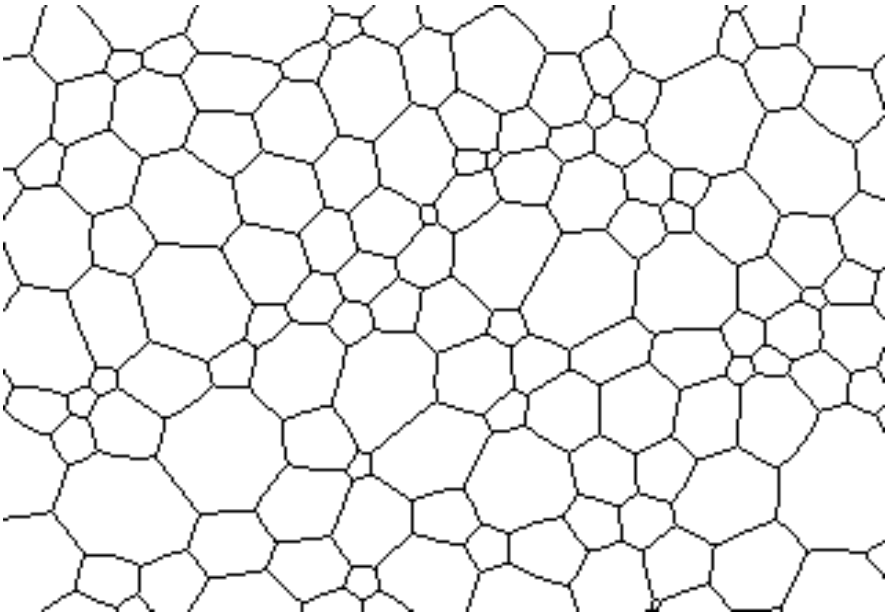


(a) Memory consumption measured for an evolving foam network.

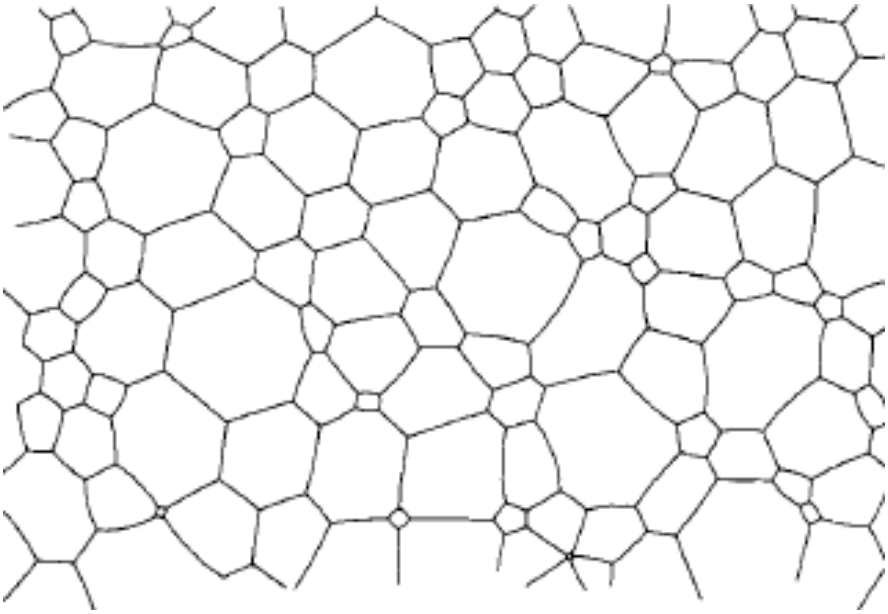


(b) Same data set as (a) but in logarithmic scale.

Figure 7.5: Memory consumption graph for the same mature foam network evolving over time as Figure 7.4 (red), and the memory consumptions of newly equilibrated foam networks with increasing number of cells (blue).



(a) Thesis method.



(b) Weaire and Kermode method.

Figure 7.6: Visual comparison of dry equilibrated foam networks between (a) our dry foam model derived in this thesis and (b) the soap froth model from Weaire and Kermode (The image in (b) is taken from [Bolton and Weaire, 1991], Fig. 1, p. 796.).

7.5 Visual Comparisons

The two-dimensional, vertex-based model for dry foam simulation that has been derived in this thesis is based on the work by Weaire and Kermode [Weaire and Kermode, 1983, Weaire and Kermode, 1984, Kermode and Weaire, 1990]. To make sure that our model is at least as good as the original work, or to find out if there are any differences we will compare our results with theirs.

The Surface Evolver [Brakke, 1992] is a software solution to minimize the energy of a surface subject to different constraints. It supports two-dimensional grain growth with periodic boundary conditions which in every sense is similar to our work. We have compared the animation quality of our foam simulator against an animation produced by the Surface Evolver. The compared animations can be found in [Kelager, 2009]. We believe that the quality of the animations produced by our foam simulator stays more true to real foam than the Surface Evolver especially during T1 topological changes where the Evolver animations converge from a T1 too slowly.

7.5.1 Weaire and Kermode

A simple visual comparison between a still frame produced from an equilibrated, mature dry foam network will be performed. We expect that there should not be too significant a difference between the two images, although we are clearly not able to employ the exact same foam network.

The result from the visual comparison is depicted in Figure 7.6. Our image shows resemblance to the one from Weaire et al. The film radii of curvature in Weaire et al. are slightly more curved which might be is a result of a larger surface tension.

Chapter 8

Results

"*Don't worry when you are not recognized,*
but strive to be worthy of recognition."

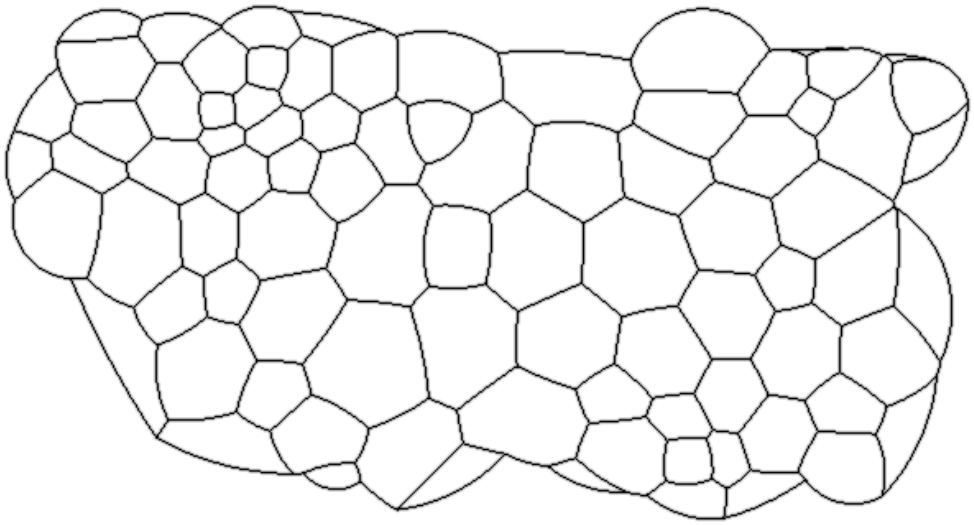
- Abraham Lincoln

We have obtained different kinds of results from the vertex-based dry foam simulator which we will present in this chapter. The foam results include the improvements to the dry foam model, all the discussed features, and the results from the evaluations. In the end we briefly look at an example of the usage of two-dimensional dry foam.

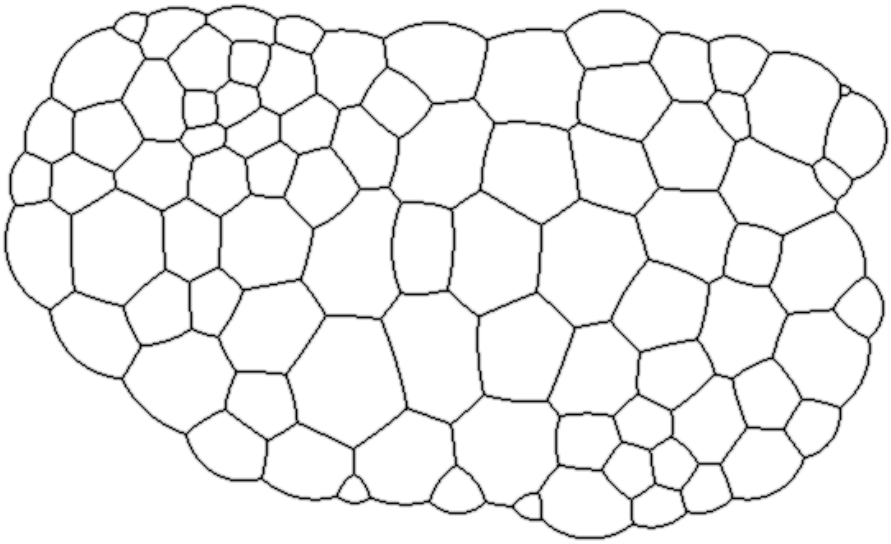
In Chapter 5 (p. 38) we discussed several improvements to the vertex-based dry foam model. The majority of the improvements are related to improving the stability of the numerical model but some are related to Computer Graphics and in this chapter we present these visual improvements. Additional images and movies are available as downloadable contents. See Appendix A for details and accessibility.

8.1 Dynamic foam boundary

The basic dry foam model was implemented with fixed boundaries due to our wish to accommodate Computer Graphics and employ the foam as a distinct entity rather than a foam with periodic boundaries. In Subsection 5.1.1 (p. 39) we introduced the Ghost Bubble method that among other things enhances the dry foam model with dynamic foam boundaries. In Figure 8.1 we have shown the a foam sample with fixed boundary condition and with dynamic boundary condition that comes in the wake of enabled the ghost bubble. Figure 8.1(a) shows a boundary that is highly invalid in agreement with the laws of Plateau but with the addition of the ghost bubble the boundary can be perfectly equilibrated as shown in Figure 8.1(b). The ghost bubble method also benefits greatly in foam dynamics situations which will be shown later on.



(a)



(b)

Figure 8.1: An example of how the dynamic foam boundary enhances the dry foam. In (a) a foam with fixed boundaries which will not be able to equilibrate its boundaries which both will be and look wrong. In (b) the same foam but with dynamic boundaries that enables the free foam surface boundary condition and allows the complete foam to equilibrate and evolve naturally.

We could have chosen not to visualize the vertices and edges along the fixed boundary and this would have removed the most severe visual problems. However, there would still be a number of non-boundary edges that would be left alone without any connections. These edges could be removed as well but then just as many vertices would be left with only two edges which again yields an invalid foam structure visually.

8.2 Dry foam maturing

In Section 6.2 (p. 52) we discussed how to mature a newly created foam network and by employing this procedure we get results as the one shown in Figure 8.2. The figure shows that our dry foam model can mature a newly created foam. We have tested the credibility of the maturing mechanism and in more than 95 % of the initial equilibration of any random sized foam networks the foam model succeeded in maturing the foam networks.

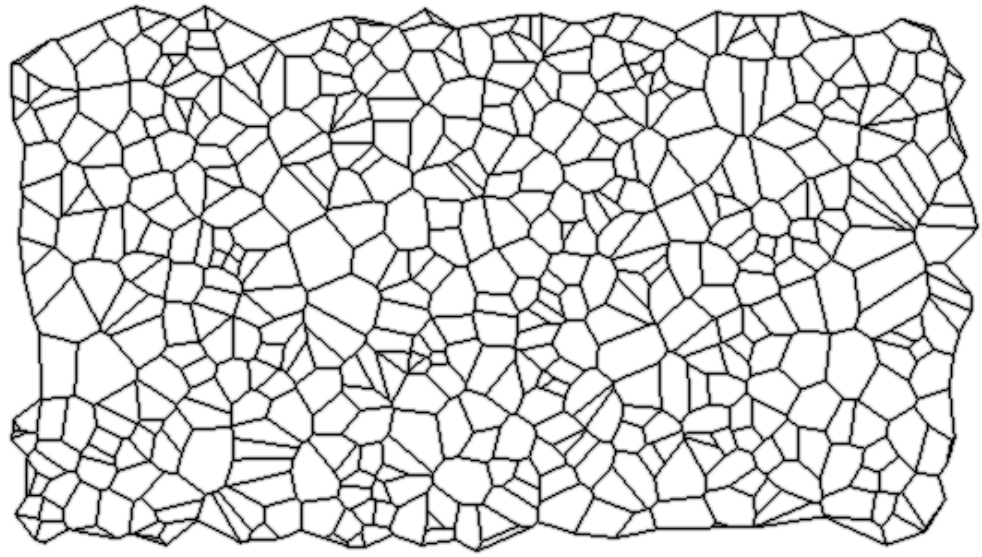
In the approximate last 5 % of the cases where the maturing procedure fails in equilibrating a newly created foam network the stability issues always began on the boundary where some vertex was very ill-defined. The typical behavior from a failing maturing procedure is that the first couple of position displacements of an ill-defined boundary vertex will bring the vertex too far away from the foam. This often results in a massive increase in cell area and from here on the solver never converges. We believe that the problem arises with the instability as described in Subsection 5.3.2 (p. 43) because the ghost bubble acts as the entire foam and by that any boundary vertex is allowed to be displaced away from the foam to infinity.

8.3 Dynamic foam features

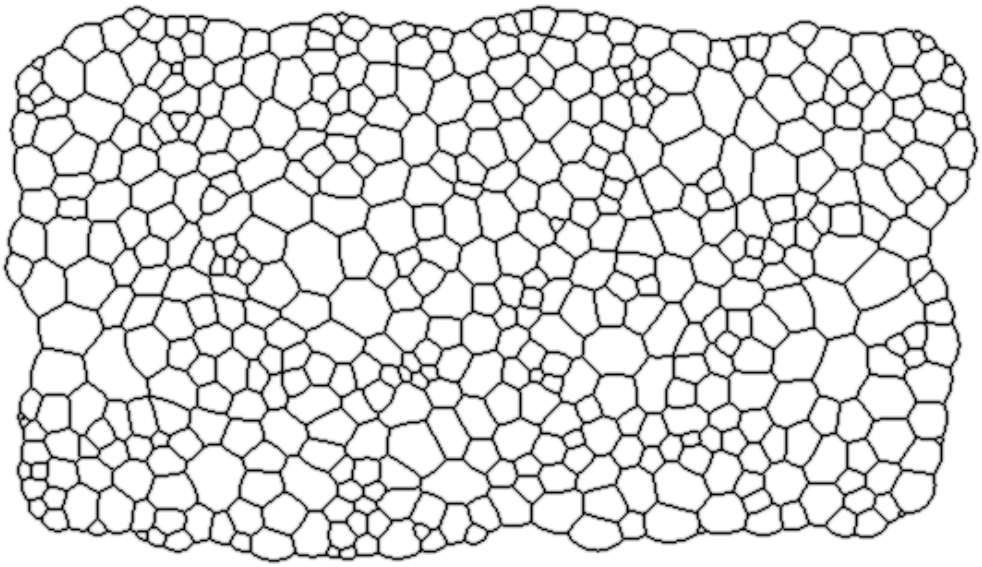
In this section we will show off the dynamic features that have been implemented in our dry foam model. The features include the topological processes due to equilibrium and the internal dynamic processes, e.g. gas diffusion, and interactive dynamic processes, e.g. bubble collapse and foam deformation.

We will test the topological operations implicitly during the results produced from other dynamic foam processes. The majority of T1 topological processes are performed during the foam maturing procedure. Once a foam has been matured T1 processes mostly occur in connection with T2 processes as bulk cells in general have more than three sides and will initiate T1 processes just before a native T2 process can be executed. The T1 processes that occur in connection with a T2 process are usually normal T1 processes as vertices approach each other, but it can also be the artificial T1 processes that are produced from a forced T2 process as a cell target area has become less than the area threshold.

A normal T1 process is shown in Figure 3.3 (p. 18) which actually is from a coarsening sequence produced by our foam simulator. The case of a topological T2 process on a three-sided cell is shown in Figure 3.5 (p. 20). If our dry foam simulator is unable to handle the T1 and T2 processes correctly then it will not be possible to show any results



(a)



(b)

Figure 8.2: A dry foam is matured by equilibrating (a) a Voronoi diagram in 500 cells generated from random points into (b) a completely equilibrated foam network with $A_{\%} = 99.993\%$ and $\phi_{\%} = 99.878\%$.

from the supported dynamic foam features without a severe set of instabilities. Based on the following results we claim that with the implementation of the stability improvements from Section 5.3 (p. 42) the solver can handle any of the supported topological processes satisfactory.

8.3.1 Foam evolution

The gas diffusion process which is discussed in Subsection 4.3.4 (p. 36) will evolve the foam network by changing the target area of the cells. The process can be slow like a real-life foam. Especially for a foam with many cells all of which have a very small area. The permeability constant operates uniformly on area and as the foam evolves the mean cell area is increasing. Thus, as the mean area becomes larger the animation creates the feeling of an idle gas diffusion which is not the case as the gas diffuses constantly. To enhance the animation it might be necessary to increase the permeability constant as the mean cell area is increased.

Figure 8.3 shows four frames from a complete foam evolution with initially $10,000$ cells. The number of T2 operations performed to such a foam sample is large. Recall from Subsection 7.4.1 (p. 73) that approximately $2,000$ T2 operations were executed during the first 50 foam iterations. This supports our claim that the foam simulator can handle T2 topological changes satisfactory.

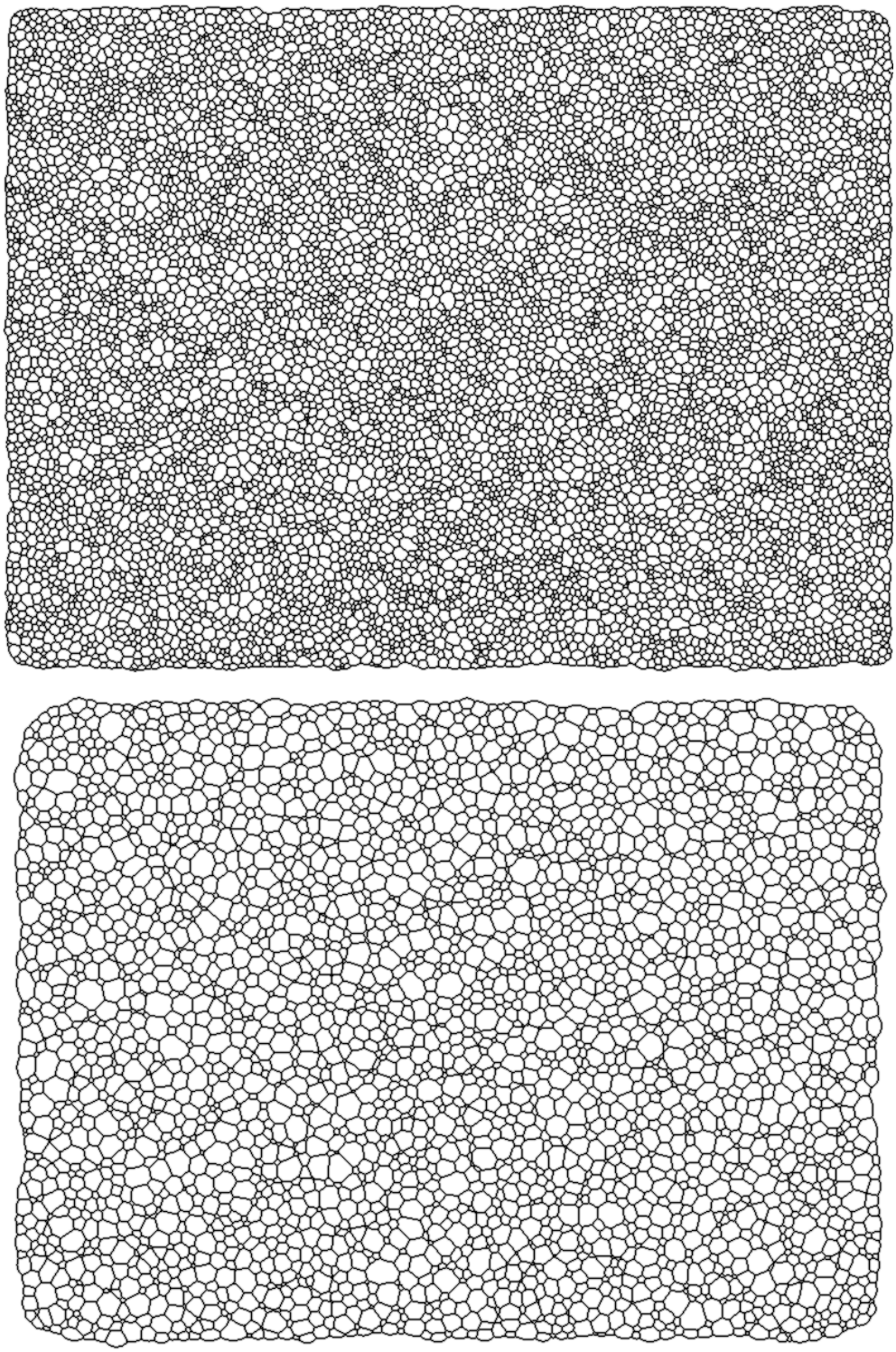
8.3.2 Shear deformation

A dry foam sample can become a highly deformable solid when performing simple and extensional shear operations. In Figure 8.4 extensional shear is performed on a dry foam sample. We have disabled gas diffusion to focus on the effect of shear deformation alone. When the foam is being deformed slowly and controlled it shears within the elastic regime while the deformation is large or the foam becomes too stressed plastic deformation sets in. This is in perfect agreement with the rheology of dry foam as discussed in Section 3.5 (p. 22).

Topological T1 changes are executed in great numbers during shear deformation. For the test example in Figure 8.4 that consists of 66 cells we registered 58 T1 operations while performing the extensional shear. This supports our claim that the foam simulator can handle T1 topological changes satisfactory.

8.3.3 Forced bubble collapse

To illustrate the manual foam bubble collapse, which is equivalent to the real-life situation when a bubble manually is being punctured by a finger, we have matured a random foam mesh of 66 cells. The dynamic gas diffusion process has been deactivated to make sure that the only bubbles that will disappear are the ones that have been selected for manual bubble collapsing. The result of a sequence of bubble collapses is shown in Figure 8.5 where cells of 4, 5, and 6 edges have been targeting for deletion.



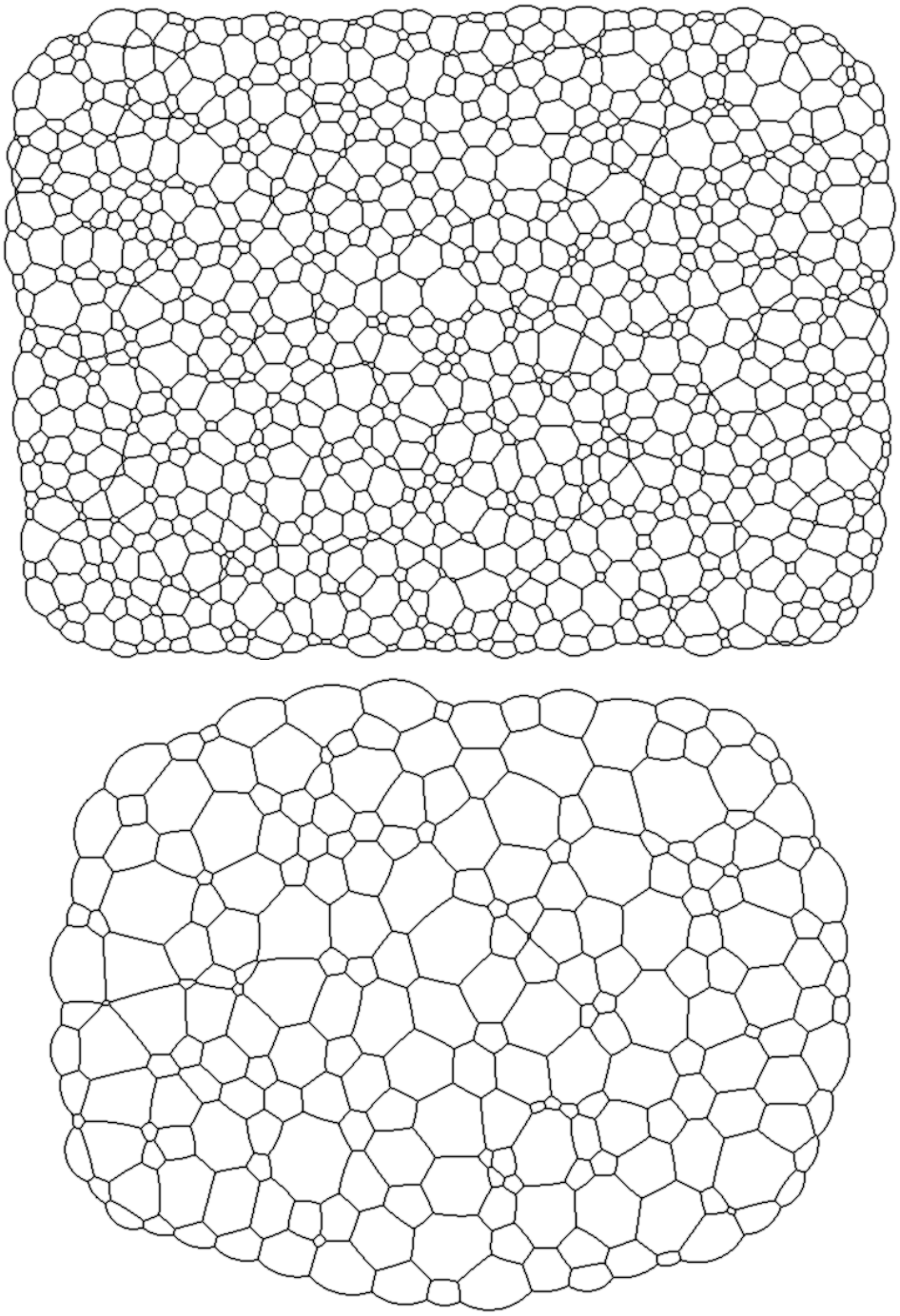


Figure 8.3: A foam system is evolving with initially $10,000$ cells. The time gap between each shown frame is approximately squared.

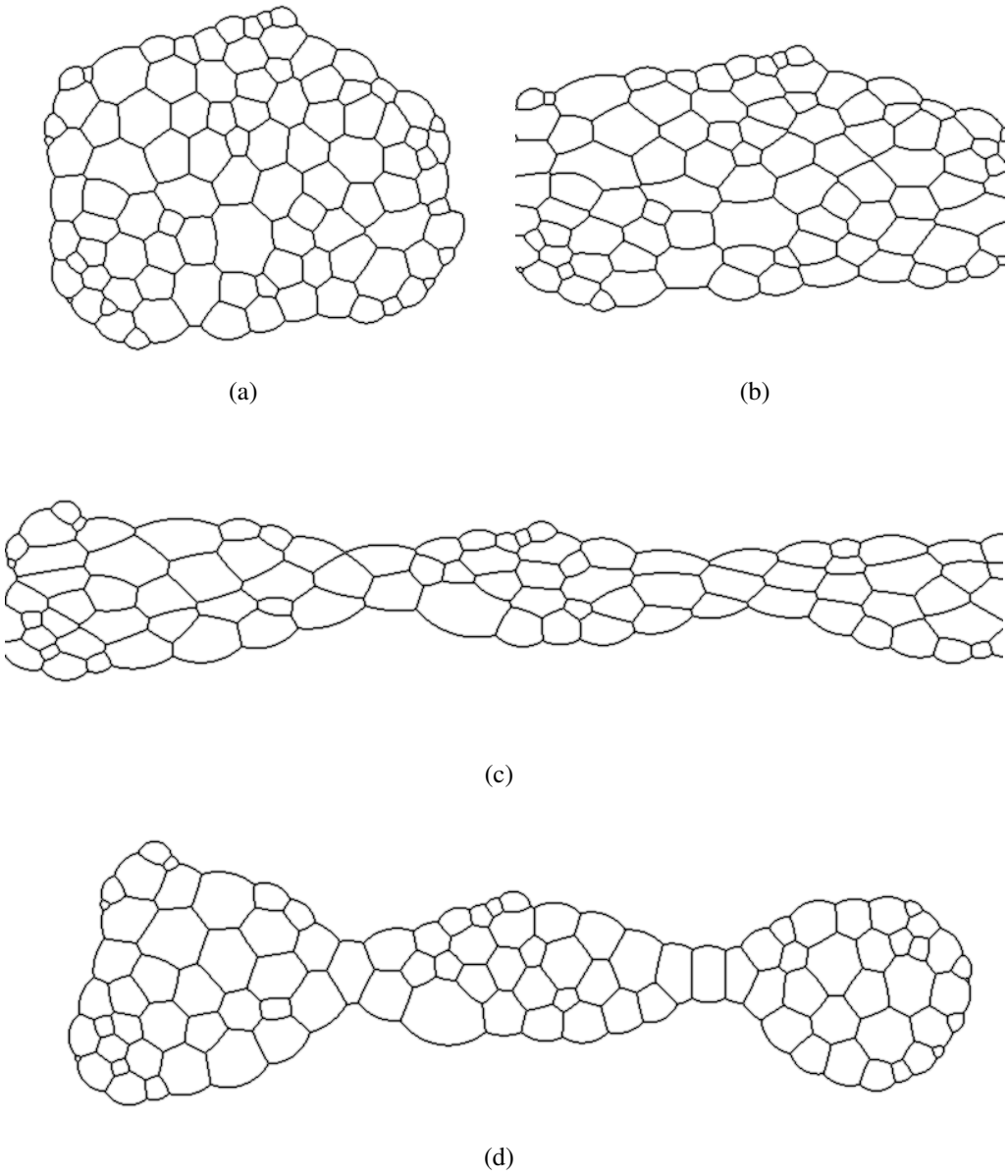


Figure 8.4: A dry foam can become a highly deformable solid when shearing is performed to the foam with the gas diffusion disabled. (a)-(c) Extensional shearing is performed to the foam that expands it horizontally while shrinking vertically. (d) The foam is released and starts to recover by contracting itself horizontally.

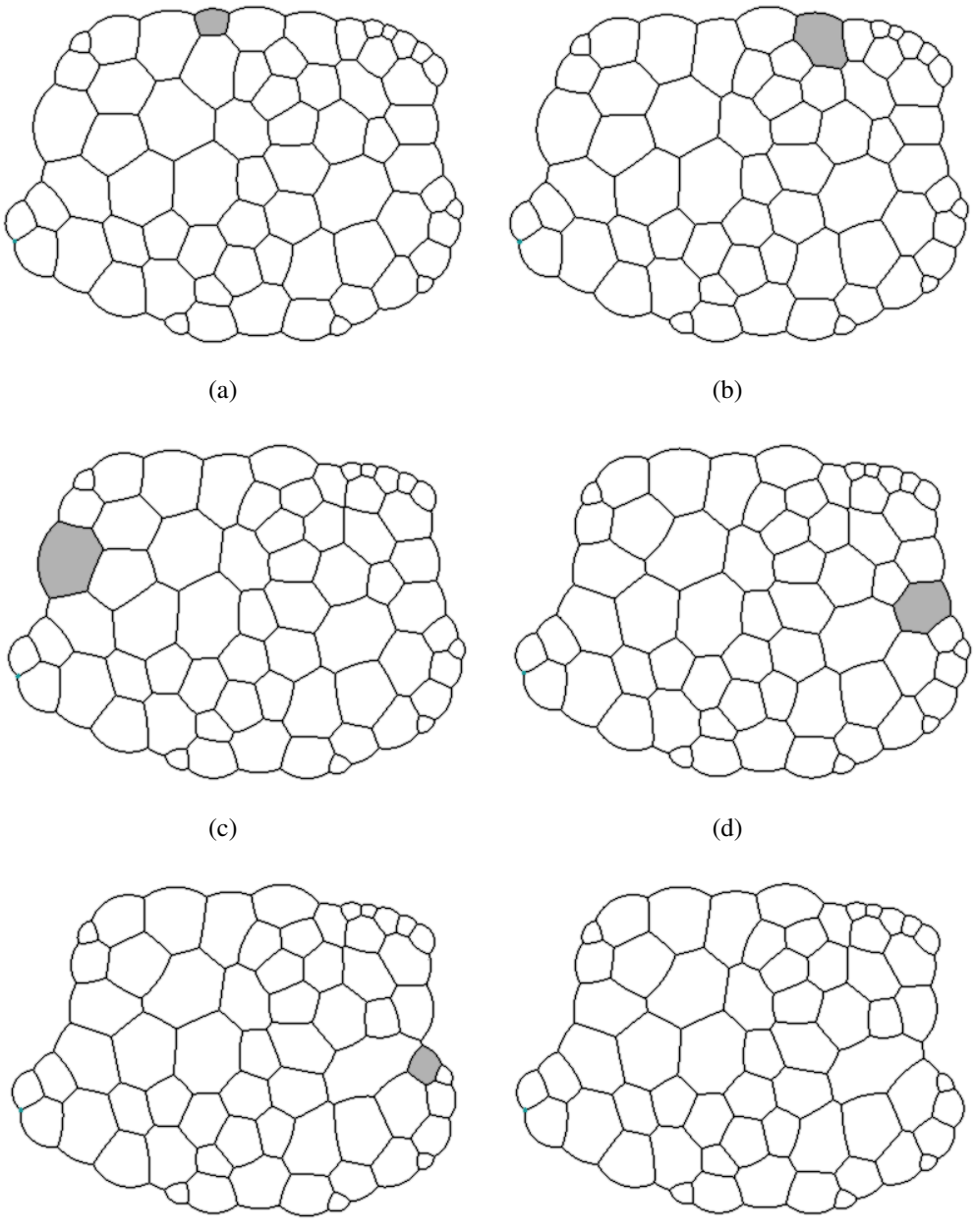


Figure 8.5: Forced bubble collapse is performed on the dynamic foam boundary. Between each subfigure the marked cell is next to be collapsed.

Figure 8.6: The Beijing National Aquatics Center, or Water Cube, in China in its final construction days (photo is taken from www.beijingolympicsfan.com).

Even though we have not shown any results of bulk cells collapsing the method works exactly the same as for boundary cells. Recall from Subsection 5.2.1 (p. 41) that the bubble mechanism works out of the box by forcing a T2 topological operation upon a cell by setting the target area to zero.

8.3.4 Pressure differences

The foam mesh can be rendered using flat shaded polygons. How we render the filled circular arcs is discussion in Section 6.5 (p. 62). Figure 8.7 shows a large example of how the pressure differences are distributed throughout a mature foam. Note that cells with equal pressure tend to group or make chains and the chains form walls that almost makes the image look like a labyrinth.

8.4 Dry foam structure in real life

The Beijing National Aquatics Center in China, also known as the Water Cube, was built for the swimming competitions of the 2008 Summer Olympics. The wall and roof structure of the Water Cube is based on the Weaire-Phelan solution to the Kelvin problem [Weaire and Hutzler, 1999]. Even though the Weaire-Phelan structure is three-dimensional it is still only the surfaces of the building that are constructed completely and these are two-dimensional. The surface foam structure is based on the vertex model which employs straight lines as edges [Kawasaki et al., 1989]. The Water Cube in its normal daylight environment is shown in Figure 8.6 during the final days of construction.

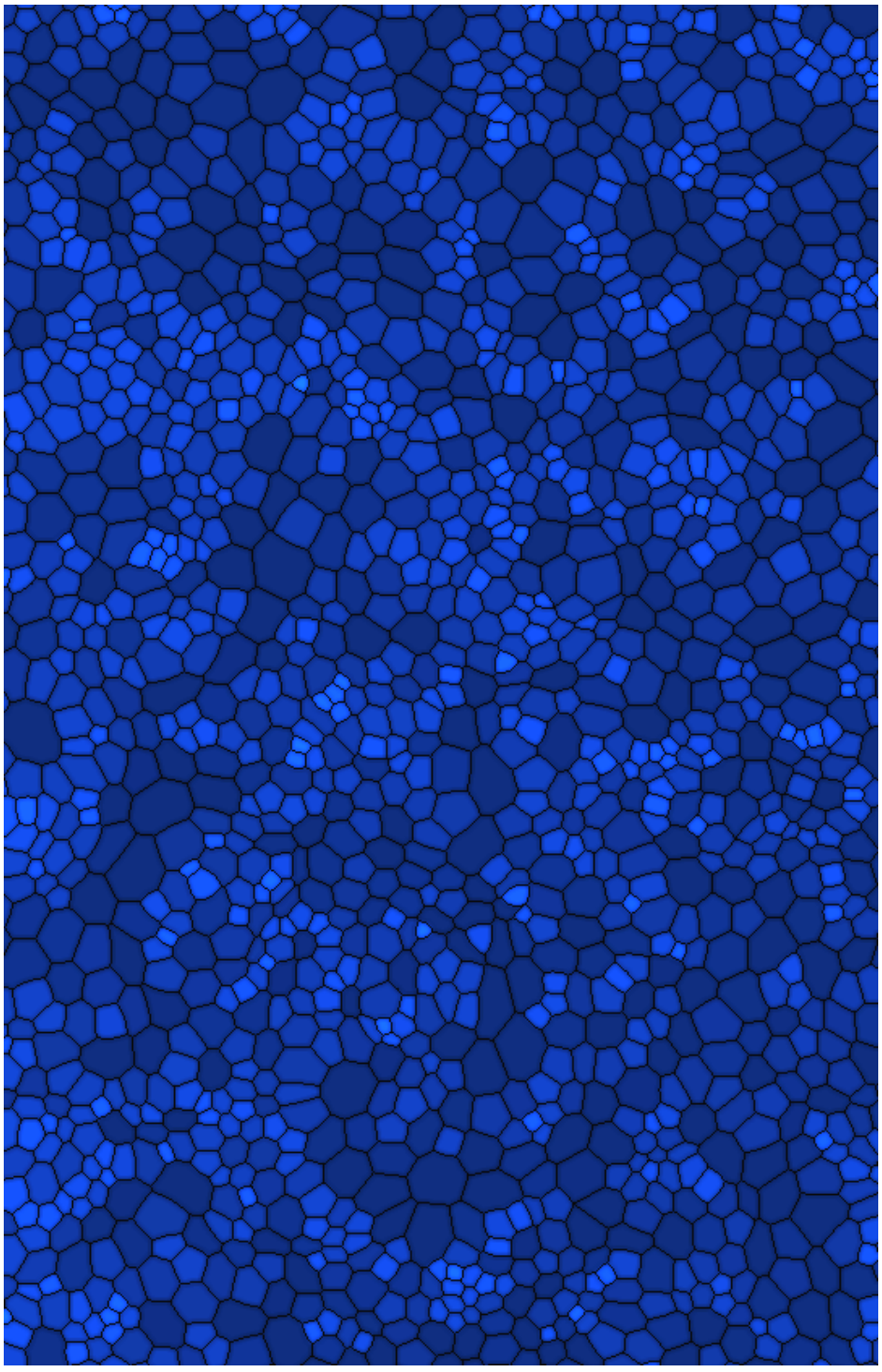


Figure 8.7: The different cell pressures are visualized in flat shaded, blue-toned polygons. The colors have been exaggerated to make the difference more visible.

Chapter 9

Future Work

"*The best way to predict the future is to invent it.*"

- **Alan Kay**

In this chapter we will highlight some of the interesting work we are currently focusing at within foam dynamics. Additional foam-based research has been conducted already while other areas are still in the planning phase. Equally for everything that is described in this chapter is that time alone has prevented us from finalizing the work.

Being computer scientists working in the field of Computer Graphics it would seem appropriate to expand the two-dimensional, vertex-based foam model to include the third dimension. However, a number of challenges is introduced by this expansion. The surfaces involved are not spherical in shape and a general mathematical description of the bubbles is yet to be given [Weaire and Hutzler, 1999]. We believe that before we add a new dimension, with an accompanying tetrahedral-based foam model, we should continue working with liquid foam in two dimensions to catch up with the areas within liquid foams that have been left out of this thesis.

9.1 Ghost Bubble Method

The ghost bubble method presented in this thesis is a novel approach to handle the dynamic boundary conditions for two-dimensional dry foams. In its presented context it is *glued* to the back of the foam mesh to let it move together with the foam which consequently allows the foam mesh to evolve freely. This is a typical Lagrangian approach. However, one of the main reasons why the ghost bubble exists at all is because the original vertex-based model requires the employment of periodic boundaries. Periodic boundaries imply that every vertex will have exactly three adjacent bubbles and three films. During the derivation of the ghost bubble method we proved that this requirement can be modified

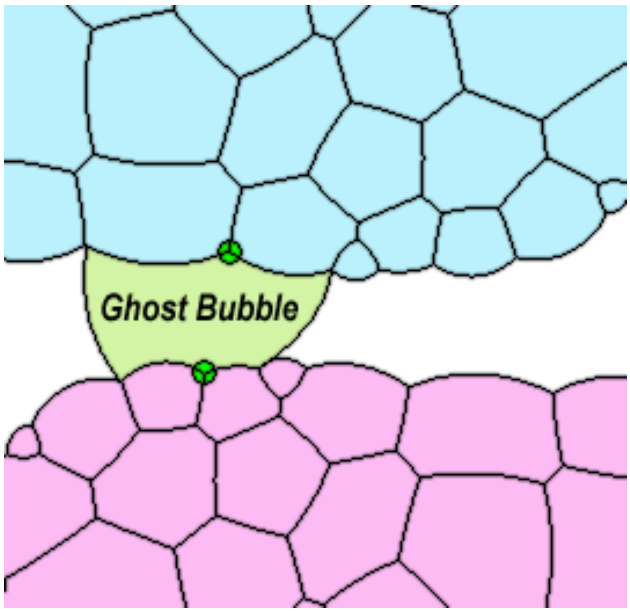


Figure 9.1: New potential applications for the ghost bubble method. In this case the ghost bubble can act as a interface bubble between two foam system to allow a smooth merge.

as we did with the presented dynamic foam boundaries. The ghost bubble is not contributing to anything by its presence in the foam. Thus, a future performance improvement will be to remove it completely and handle the dynamic boundaries simply be detecting if we are equilibrating a vertex on the mesh boundary.

Another application for the ghost bubble method is to create small ghost bubbles when operating on the boundary to handle multiple foam systems. In this case the ghost bubbles will act as interface bubbles that can enable a smooth merge between the foams as illustrated on Figure 9.1. We know from the foam collapsing procedure presented in Subsection 4.3.4 (p. 36) that the target area of the ghost bubble can be set to zero, where topological operations will remove the ghost bubble elegantly and thus merge the two foams.

9.2 Diffusion

In this thesis foam coarsening due to gas diffusion has been modeled by the means of von Neumann's law. This model is the result of how gas permeates on average in an infinite large foam network. We have only been working with finite foam networks with up to $10,000$ bubbles, as depicted on Figure 8.3, which is finite in size.

Von Neumann's law is a generalization of Equation (3.17), which is repeated here for

the sake of convenience,

$$\frac{\mathrm{d}A_n}{\mathrm{d}t} = -\kappa \sum_j (p_n - p_j) l_j .$$

We believe that this diffusion model will exhibit a better animation result in cases where the average mean of edges < 6 which is the configuration for all the working examples in this thesis. The rate of change in area handles each film individually and makes sure that larger three-sided cells diffuse more gas than smaller three-sided cells (with less film lengths) which is not the case with von Neumann's law.

9.3 Two-Dimensional Dry Foam

Many subsystems have been presented throughout this thesis to handle two-dimensional dry foam but we are left with a couple of challenges that which work has not been able to solve completely. These challenges are all ongoing work in the wake of this thesis and include:

- *Two-sided bubbles* - The polymesh data structure that has been employed for the foam mesh do not support cells with less than three vertices and films, and situations like the isolated two-sided double-bubble, as illustrated in Figure 6.1, must be handled in a non-physical correct manner as our implementation otherwise will fail.

- *Perishing foam* - The vertex-based foam model and the foam mesh together are not sufficient to handle a foam that can perish completely. If the number of bubbles < 4 then the current foam model will fail.

- *Film rupture* - The foam collapsing method is physically incorrect but it has been implemented for animation purposes. The correct way is to simulate the film thinning mechanism that occurs on the boundary and in the bulk of the foam. Foam thinning in the bulk can rupture films and merge adjacent cells into one. This process is highly active in a real-life liquid foam.

- *Creating foam* - This thesis only model an already created foam. It can equilibrate the foam and evolve a mature foam over time but it cannot spawn new foam cells. We have already shown that the implementation support removal and recreation of foam cells which imply that to introduce new cells into the foam network seems possible. Being able to spawn foam cells other interesting simulation ideas can be obtained, e.g. froth that is created from pouring a liquid into a container.

- *Foam visualization* - The wireframe and flat shaded solid cell visualizations of the foam networks are likely only to be accepted in the academia. For the purpose of Computer Graphics the presented visualizations are insufficient. Even though the foam is two-dimensional it is possible to lift the bubbles to adopt a three-dimensional look and feel. We are working on bubble-based screen-space distortion effects using fragment shaders. While fragment shaders are being employed

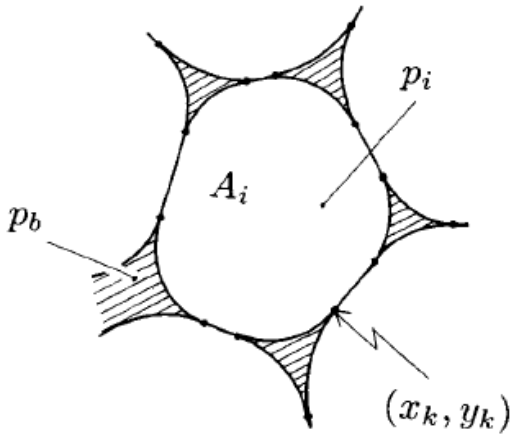


Figure 9.2: The Plateau borders all have constant pressure p_b which is treated similar to pressures in the cells (The illustration is taken from [Weaire and Hutzler, 1999], Figure 2.6, page 25).

we can gain additional output results, such as procedural soap foam texture maps. To apply the idea of wet foam on top of our dry foam mesh we will apply a modified version of the *Single-pass wireframe rendering* technique [Bærentzen et al., 2006] which will comply with the visualization of real-life wet foam as seen in Figure 1.3 (p. 6).

9.4 Two-Dimensional Wet Foam

A natural extension to the work conducted in this thesis is to add liquid into the system to wet the foam and include the liquid fraction Φ_l into the mathematically vertex-based model. When introducing wetness into the foam model it becomes necessary to model Plateau borders and real Plateau junctions.

9.4.1 Plateau borders

In a two-dimensional wet foam the Plateau borders are smoothly joined to the adjacent films as depicted in Figure 9.2. This is Plateau's rule **B** for wet foams from Subsection 3.2.3 (p. 17). For a wet foam it becomes necessary to model the pressures in the Plateau borders similar to the cell pressures. The pressure in the Plateau borders is lower than the pressure of the gas in the cells which are depicted by positive curvatures. The radius of curvature of a cell-border edge is computed using Laplace-Young as

$$r = \frac{\gamma}{p_i - p_b}, \tag{9.1}$$

where p_b is the pressure in the Plateau border which is assumed to be constant throughout the foam. Compared to Equation (3.11) (p. 17) the surface tension γ is not doubled in (9.1)

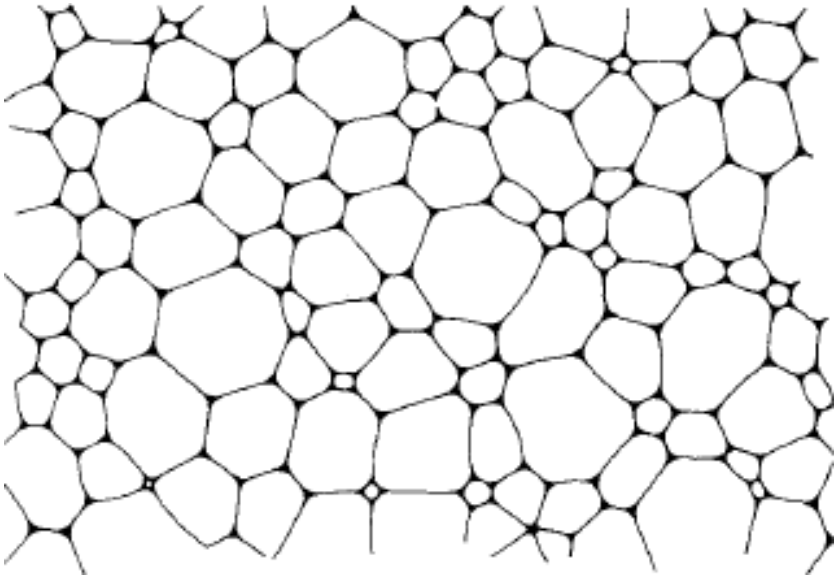


Figure 9.3: A decorated version of Figure 7.6(b) (p. 77) that has been converted from a dry two-dimensional foam into an equilibrated wet foam (The image is taken from [Bolton and Weaire, 1991], Fig. 4, p. 799).

because the Plateau borders entirely consist of liquid.

A simulation of a two-dimensional wet foam can be implemented by expanding the two-dimensional dry foam to include Plateau borders. New tangent constraints must be added to the mathematical model that will enforce that where a cell-cell arc meets a Plateau border all the arcs incident at that point must have the same tangent. These constraints are required to meet Plateau's rule **B**. Each vertex from the dry foam model is replaced by threefold Plateau borders. For a wet foam the T1 topological process becomes a two-step procedure of a border coalescence followed by a border separation as the emerging four-fold vertex becomes unstable.

9.4.2 The decoration theory

It is possible to decorate the structure of an equilibrated dry foam into the wet foam counterpart by employing the decoration theorem [Bolton and Weaire, 1991] which states:

> *Any two-dimensional dry foam structure can be decorated by the superposition of a Plateau border at each threefold vertex, to give an equilibrated wet foam structure, provided these Plateau border do not overlap.*

Converting a dry foam into a wet foam will only be convincing to illustrate the structure of wet foam as single images. Any motion sequences will still reveal that the behavior of the foam is strictly that of a dry foam. The effect of decorating an equilibrated dry foam into a wet foam is depicted in Figure 9.3.

Foam Element	Mesh Element	Mesh Traits
Junction, Vertex	Vertex	*Position*, Velocity, Acceleration, Mass
Foam	Mesh	Damping

Table 9.1: Additional foam mesh traits are required to numerically integrate velocity and position over time.

9.5 External Dynamics

The mathematical model presented in Chapter 4 (p. 26) is a quasi-static equilibrium model which implies that any kinds of dynamics has been omitted from the model except for the dynamic diffusion process. This section presents some ideas to introduce various kinds of external dynamics into the model. Employing dynamics makes interactivity possible in a natural and convincing way.

9.5.1 Gravity

The foam model is a vertex-based model which can be treated as a particle system. Expanding the junction traits with a couple of additional members will allow for Lagrangian position-based dynamics. See Subsection 6.1.1 (p. 50) for details about the data structure and data traits. Table 9.1 lists the additional traits to the foam mesh that are required to expand the foam model with a basic particle system.

Introducing external forces, e.g. gravity, into the dynamic model is a matter of adding the external force contributions to the particle accelerations $\mathbf{a}_i$ as

$$\mathbf{a}_i^n = \mathbf{g} + \frac{\mathbf{F}_i^n}{m_i}, \tag{9.2}$$

where $\mathbf{F}_i$ indicates additional external forces, besides the gravitational acceleration $\mathbf{g}$, to particle i and m_i is the mass of the particle. To advance the particle velocity $\mathbf{u}_i$ and position $\mathbf{v}_i$ over time Δt we integrate the values numerically, e.g. by employing the semi-implicit Euler integrator as

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \Delta t \mathbf{a}_i^n, \tag{9.3}$$

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t \mathbf{u}_i^n, \tag{9.4}$$

where the n-superscripts indicate the state of time.

Experiments have revealed that gravity can have a significant impact on the foam structure when handling collisions in such way that it prevents the foam from maintaining its quasi-static equilibrium. The force due to gravity is simply too large compared to the equilibrium forces. Care should be taken when including additional forces into the foam system. The dynamic processes must not be allowed to dominate the foam structure. A usable idea is to lower the gravity which is a convincing way of implementing the natural buoyancy of the foam.

9.5.2 Two-way interaction

If the vertex-based foam model is extended with position-based dynamics the whole system can be employed for quite many intriguing features. One obvious application is to let the vertex model interact with a Lagrangian fluid system, like the smoothed particle hydrodynamics approach [Kelager, 2006], to enable two-way fluid-foam interactions.

The smoothing kernels and radii in the SPH system can be modified to work in two dimensions. Due to the mass difference between the two dynamic systems it should primarily be the foam that gets affected by the SPH particles in the two-way collision handling. However, due to drainage, film rupture, and cell collapsing the foam system should relocate particles to the SPH system. Likewise, heavy turbulence and aggressive collision handling within the SPH system should also relocate particles to the foam system which will allow new foam cells to emerge.

9.6 Numerical solver

Using the parameter values as suggested throughout Chapter 7 (p. 64) we can simulate a foam network interactively at $\geq$ 10 FPS if the form network consists of no more than approximately 100 cells on an Intel Core2 Duo CPU at 2.0 GHz with 3.0 GB of system memory. Of course this number will increase on a more highend CPU. However, the fact remains that the presented foam solver will not be able to simulate tens of thousands of cells that in real life only add up to a small-sized foam, e.g. the nucleated foam after a regular soda has been poured into a glass.

It is possible to get a higher frame rate from the vertex-based foam simulator. The amount of iterations can be decreased if we are willing to sacrifice the convergence rate and the quality of the simulation results. However, while the number of solver iterations can be decreased it is possible to balance the damping parameter τ to make up for the lost iterations. The quality may suffer and the foam simulation can become unstable if τ becomes too large which was discussed in Section 7.1 (p. 64). Decreasing the amount of iterations will also make the foam more elastic. This is the same artifact as seen in other relaxation-based solvers, e.g. in position-based cloth dynamics.

The way we have designed the nonlinear numerical Newton method makes it possible to parallelize the method and allow for multithreaded simulations. An important restriction for such a multithreaded foam solver is that we must make sure that only vertices that do not explicitly affect other vertices are simulated in parallel, and this is familiar from our requirement from Subsection 7.3.2 (p. 70).

Chapter 10

Conclusion

"*But all bubbles have a way of bursting*
or being deflated in the end."

- **Barry Gibb**

In this thesis we have built up a foundation based on Computational Physics to create a vertex-based foam model for Computer Graphics. The model is defined to work in two dimensions and the mathematics behind the method exploits the fact that it is merely a two-dimensional model. Additionally, the model is a liquid foam model but we did restrain ourselves only to work with the dry limit of the liquid model. This has helped us focus on the basics of the physics of foam that was required to understand and control the two-dimensional dry foam model.

10.1 Obtained Goals

We believe that we have succeeded in understanding the physics behind the complex structure and behavior of liquid foams. We have derived a mathematical model for two-dimensional dry foam and we have shown how to implement the model numerically. Based on implementation experiences, the gained insight in the numerical model, and general knowledge from Computer Science we have suggested a number of improvements to increase both stability and animation quality of the dry foam model.

We are not blindfolded and conclude that there is no room for improvements. We realize that the presented foam model can be improved in several ways which is reflected in our ongoing research and future work.

With the completion of this thesis we find that we have succeeded in communicating the physics behind liquid, dry foams and how such foams behave dynamically. From the previous work on foam simulations in Computer Graphics we believe that we have proven

the importance of employing physics-based foam dynamics to improve the overall quality of the liquid foam motion and behavior.

Lastly, it is our hope that the work in this thesis can serve as a foundation for additional and future work on liquid foam simulations within Computer Graphics.

10.2 Contributions

The main contributions of this thesis are:

- The Ghost Bubble method for mathematically stable dynamic boundaries that allows foam simulations with free surfaces and finite boundaries. When developing a physics-based foam model for Computer Graphics it is vital that such model can serve the requirement of a distinct, renderable entity. With the addition of the Ghost Bubble to the foam model we can now simulate any finite foam sample and interact with it to create interesting situations.

- An improved nonlinear Newton method for a two-dimensional vertex-based dry foam model that is stable and will converge even though vertices can be locally ill-defined. The numerical model has less than 1% in total error from absolute equilibrium which is a respectable achievement compared to common first order finite difference methods that have employed to solve partial differential equations.

- Convergence rate evaluation and analysis of the two-dimensional dry foam solver which, to the best of our knowledge, never has been published before of this this type of problem. The improved nonlinear Newton method has superlinear convergence rate at best. Taken into account the number of linearizations, including the damping coefficients, that have been added to improve the stability of the method the convergence rate is fairly good.

- Several stability and feature improvements to the vertex-based foam model to enrich the animation quality of the dry foam simulator. Some physics-based but less physical correct features have been added for Computer Graphics purposes, including a physics-based strategy to handle three-sided double-bubbles given that the underlying mesh data structure do not allow faces with only two vertices.

Geez, I'm all dried out o.O
lets go guzzle that brewski!

Bibliography

[Aboav, 1980] Aboav, D. A. (1980). The arrangement of cells in a net. *Metallography*, 13:43–58.

[Alexandrescu, 2001] Alexandrescu, A. (2001). *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional.

[Bærentzen et al., 2006] Bærentzen, J. A., Nielsen, S. L., Gjøl, M., Larsen, B. D., and Christensen, N. J. (2006). Single-pass wireframe rendering. ACM SIGGRAPH Sketches.

[Bolton and Weaire, 1991] Bolton, F. and Weaire, D. (1991). The effects of plateau borders in the two-dimensional soap froth. I. decoration lemma and diffusion theorem. *Philosophical Magazine Part B*, 63(4):795–809.

[Bolton and Weaire, 1992] Bolton, F. and Weaire, D. (1992). The effects of plateau borders in the two-dimensional soap froth. II. general simulation and analysis of rigidity loss transition. *Philosophical Magazine Part B*, 65(3):473–487.

[Brakke, 1992] Brakke, K. A. (1992). The surface evolver. *Experimental Mathematics*, 1(2):141–165. http://www.susqu.edu/brakke/evolver/evolver.html.

[Cleary et al., 2007] Cleary, P. W., Pyo, S. H., Prakash, M., and Koo, B. K. (2007). Bubbling and frothing liquids. *ACM Transactions on Graphics*, 26(3):97–1–97–6.

[Cox et al., 2006] Cox, S. J., Dollet, B., and Graner, F. (2006). Foam flow around an obstacle: Simulations of obstacle-wall interaction. *Rheologica Acta*, 45(4):403–410.

[de Berg et al., 2000] de Berg, M., M, van Kreveld, Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition.

[Durian, 1995] Durian, D. J. (1995). Foam mechanics at the bubble scale. *Physical Review Letters*, 75(26):4780–4783.

[Durian, 1997] Durian, D. J. (1997). Bubble-scale model of foam mechanics: Melting, nonlinear behavior, and avalanches. *Physical Review E*, 55(2):1739–1751.

[Durikovic, 2001] Durikovic, R. (2001). Animation of soap bubble dynamics, cluster formation and collision. *Comput. Graph. Forum*, 20(3).

[Eberly, 2008] Eberly, D. (2008). Derivative approximation by finite differences. Geometric Tools, LLC.

[Exerowa and Kruglyakov, 1998] Exerowa, D. R. and Kruglyakov, P. M. (1998). *Foam and Foam Films: Theory, Experiment, Application*, volume 5 of *Studies in interface science*. Elsevier, Amsterdam, New York.

[Fedkiw et al., 1999] Fedkiw, R., Aslam, T., Merriman, B., and Osher, S. (1999). A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492.

[Glassner, 2000a] Glassner, A. (2000a). Soap bubbles: Part 1. *IEEE Comput. Graph. Appl.*, 20(5):76–84.

[Glassner, 2000b] Glassner, A. (2000b). Soap bubbles: Part 2. *IEEE Comput. Graph. Appl.*, 20(6):99–109.

[Hong and Kim, 2003] Hong, J.-M. and Kim, C.-H. (2003). Animation of bubbles in liquid. *Comput. Graph. Forum*, 22(3):253–262.

[Icart and Arquès, 1999] Icart, I. and Arquès, D. (1999). An approach to geometrical and optical simulation of soap froth. *Computers & Graphics*, 23(3):405–418.

[Iwasaki et al., 2004] Iwasaki, K., Matsuzawa, K., and Nishita, T. (2004). Real-time rendering of soap bubbles taking into account light interference. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 344–348, Washington, DC, USA. IEEE Computer Society.

[Jakobsen, 2001] Jakobsen, T. (2001). Advanced character physics. In *Game Developers Conference Proceedings*, pages 383–401. CMP Media, Inc.

[Kawasaki et al., 1989] Kawasaki, K., Nagai, T., and Nakashima, K. (1989). Vertex models for two-dimensional grain growth. *Philosophical Magazine Part B*, 60(3):399–421.

[Kelager, 2006] Kelager, M. (2006). Lagrangian fluid dynamics using smoothed particle hydrodynamics. Department of Computer Science, University of Copenhagen.

[Kelager, 2009] Kelager, M. (2009). Vertex-based simulation of dry foam DLC. Glowing Goo Research. http://www.glowinggoo.com/foam.

[Kermode and Weaire, 1990] Kermode, J. P. and Weaire, D. (1990). 2D-FROTH: a program for the investigation of 2-dimensional froths. *Computer Physics Communications*, 60(1):75–109.

[Kim et al., 2007] Kim, B., Liu, Y., Llamas, I., Jiao, X., and Rossignac, J. (2007). Simulation of bubbles in foam with the volume control method. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 98, New York, NY, USA. ACM.

[Kück et al., 2002] Kück, H., Vogelgsang, C., and Greiner, G. (2002). Simulation and Rendering of Liquid Foams. In *Proc. Graphics Interface*, pages 81–88.

[Li and Volkov, 2006] Li, L. and Volkov, V. (2006). Inflatable models. *J. Comput. Sci. Technol.*, 21(2):154–158.

[Losasso et al., 2004] Losasso, F., Gibou, F., and Fedkiw, R. (2004). Simulating water and smoke with an octree data structure. *SIGGRAPH 2004, ACM TOG*, 23:457–462.

[Losasso et al., 2006] Losasso, F., Shinar, T., Selle, A., and Fedkiw, R. (2006). Multiple interacting liquids. *SIGGRAPH 2006, ACM TOG*, 25:812–819.

[Marsh et al., 1995] Marsh, S. P., Masumura, R. A., and Pande, C. S. (1995). A curvature-driven vertex model for two-dimensional grain growth. *Philosophical Magazine Letters*, 72(6):429–434.

[Müller et al., 2007] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118.

[Müller et al., 2004] Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. (2004). Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Nagai et al., 1990] Nagai, T., Ohta, S., Kawasaki, K., and Okuzono, T. (1990). Computer simulation of cellular pattern growth in two and three dimensions. *Phase Transitions*, 28(1–4):177–211.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, second edition.

[OpenTissue, 2009] OpenTissue (2009). Open source project, Generic algorithms and data structures for rapid development of interactive modeling and simulation. http://www.opentissue.org.

[Selle et al., 2005] Selle, A., Rasmussen, N., and Fedkiw, R. (2005). A vortex particle method for smoke, water and explosions. *SIGGRAPH 2005, ACM TOG*, 24:910–914.

[Stam, 1999] Stam, J. (1999). Stable fluids. In *SIGGRAPH 1999 Conference Proceedings*, pages 121–128.

[Weaire and Hutzler, 1999] Weaire, D. and Hutzler, S. (1999). *The Physics of Foams*. Oxford University Press, New York.

[Weaire et al., 2003] Weaire, D., Hutzler, S., Cox, S., Kern, N., Alonso, M. D., and Drenckhan, W. (2003). The fluid dynamics of foams. *Journal of Physics: Condensed Matter*, 15(1):S65–S73.

[Weaire and Kermode, 1983] Weaire, D. and Kermode, J. P. (1983). Computer simulation of a two-dimensional soap froth - i. method and motivation. *Philosophical Magazine Part B*, 48(3):245–259.

[Weaire and Kermode, 1984] Weaire, D. and Kermode, J. P. (1984). Computer simulation of a two-dimensional soap froth - ii. analysis of results. *Philosophical Magazine Part B*, 50(3):379–395.

[Weygand et al., 1999] Weygand, D., Bréchet, Y., Lépinoux, J., and Gust, W. (1999). Three-dimensional grain growth: A vertex dynamics simulation. *Philosophical Magazine Part B*, 79(5):703–716.

[Zaikov et al., 1988] Zaikov, G. E., Iordanskii, A. P., and Markin, V. S. (1988). *Diffusion of electrolytes in polymers*. New concepts in polymer science. VSP.

[Zheng et al., 2006] Zheng, W., Yong, J.-H., and Paul, J.-C. (2006). Simulation of bubbles. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 325–333, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Appendix A

Supplementary Material

This appendix briefly explains the supplementary material that accompany this thesis. The enclosed DVD contains a `readme.txt` file that explains the contents of the DVD.

A.1 Source Code

The complete source code is copyright the author of this thesis. The complete source code of the derived vertex-based foam model from this thesis along with the demonstrative foam application are included on the enclosed DVD for references. The DVD also includes the OpenTissue library [OpenTissue, 2009] that has been used to built the vertex-based foam application.

At the time of writing the source code is not yet public available as it is subject to future publications. However, to get access the source code please contact the author of this thesis about the possibility to obtain the source code.

The implementation of the foam model is written in the programming language C++ applying the programming paradigm called *policy-based programming* which makes heavy use of templates [Alexandrescu, 2001]. It can be a challenge for a compiler to generate code that has been optimized for efficiency using the policy-based programming technique but at the same time it is often faster to prototype new ideas without spending time on implementing new things. The code has been written as an extension of OpenTissue and the file structure reflects this design.

A.1.1 The OpenTissue library

The OpenTissue library **revision 5055** is the approved revision that has been employed to build the vertex-based foam application. This revision of the OpenTissue library has also been included on the DVD for convenience. Please address the OpenTissue website [OpenTissue, 2009] for details about the library including installation and compiling.

The implementation of the arc drawing method as described in Section 6.5 (p. 62) has been added to OpenTissue and is therefore not a part of the foam implementation.

A.2 Downloadable Contents

A website has been created to include additional materials, all of which are freely downloadable contents that accompany this thesis. Please go to the following site to access the **Vertex-Based Simulation of Dry Foam** downloadable content (DLC):

glowinggoo.com/foam

The contents of the site are self-explanatory but some additional comments about the material are presented here.

A.2.1 Demonstrative foam application

A binary win32 application that contains the implementation of the vertex-based foam model can be freely downloaded from the thesis DLC web site. The foam application can be built on any OpenTissue supported platform using any OpenTissue supported compiler. Here is a very brief guide to the foam application (on any platform):

- Left mouse + left/right mouse movement will change the number of random sites.
- Pressing 'r' will automatically generate a new foam mesh based on sites.
- Pressing 'e' will equilibrate the foam mesh once.
- Pressing 's' will simulate (diffuse + equilibrate) the foam mesh once.
- Pressing <space> will toggle simulate on/off.
- CTRL + Left mouse + left/right mouse movement will perform simple shear.
- CTRL + Left mouse + up/down mouse movement will perform extensional shear.
- Right mouse will pop up the controls menu for addition features.

A.2.2 Motion sequences

All the motion sequences (movies) that are accompanying this thesis have been created from the output of the demonstration. It is not possible to save frames or movies with the distributed win32 foam application. The motion sequences have all been encoded in the XVID/MPEG-4 format and encapsulated in the AVI media container. Any recent multimedia player can replay the foam movies as long as an MPEG-4 supported codec is installed on the system, e.g. DivX, Xvid, etc.