Technical Report 2004-1-003

Towards Physics-based Models in Medicine, Animation and Rendering

Roman Ďurikovič

June 22, 2004

The Department of Computer Software
The University of Aizu
Tsuruga, Ikki-Machi, Aizu-Wakamatsu City
Fukushima, 965-8580 Japan

The technical reports are published for early dissemination of research results by the members of the University of Aizu. The completed results may be submitted later to journals and conferences for publication.

Technical Report 2004-1-003

Title:
Towards Physics-based Models in Medicine, Animation and Rendering

Authors:
Roman Ďurikovič

Key Words and Phrases:
elastic models, growth modeling, Lyndenmayer-systems, soap bubble dynamics, glare effects, real-time rendering, metallic effects.

Abstract:
This report addresses the author contribution and demonstrates how physics-based modeling (PBM) offers the power and generality needed in many medical, computer animation and computer rendering applications. I will demonstrate through a series of examples and illustrations, how PBM methods can be applied to address difficult, real world problems.

The author presents a large variety of methods and associated experiments in medical graphics, animation and rendering that help the reader better understand the presented material. I addition special emphasis has been given on the development of techniques with interactive performance.

Report Date:
6/22/2004

Written Language:
English

Any Other Identifying Information of this Report:
Submitted as habilitation to Comenius University, Bratislava, Slovak Republic, 2004

Distribution Statement:
First Issue: 10 copies

Supplementary Notes:

Computer Graphics Laboratory
The University of Aizu
Aizu-Wakamatsu
Fukushima 965-8580
Japan

Preface

During the past few years, physics-based animation has emerged as an important new approach to computer graphics and modeling. Although physics-based modeling (PBM) is inherently a physical and mathematical subject, the models involved are simplified to such extend that they can be calculated in real-time. One important aspect of the physical-based animation is the user interaction, therefore physical models should enable us to convert the inputs from user interaction to certain parameters in the model, such as external forces, ...

This habilitation addresses the author contribution and demonstrates how PBM offers the power and generality needed in many medical, computer animation and computer rendering applications. I will demonstrate through a series of examples and illustrations, how PBM methods can be applied to address difficult, real world problems.

The author presents a large variety of methods and associated experiments in medical graphics, animation and rendering that help the reader better understand the presented material. I addition special emphasis has been given on the development of techniques with interactive performance.

Chapter 2, demonstrates an interesting medical application of physical model embedded into a formal language called Lyndenmayer-system, to simulate the growth of human organs. This theory stems from the 1996 PhD thesis of the author in the Department of Engineering at the Hiroshima University, Hiroshima, Japan, under the supervision of Professors Nakamae, Nishita and Yamashita. It was the first thesis in PBM to address the problems in reconstruction of shape organs in medical graphics. Later this theory was significantly advanced by formal languages to simulate the growth processes, since the author joined Hiroshima University as postdoctoral fellow.

Since the author joined as an Assistant Professor the Software Department of the University of Aizu, Aizuwakamatsu, Japan in April 1999, the author has conducted most of his research in PBM techniques with applications to medical graphics, computer animation and rendering. The rest of the chapters was done in joint work with Dr. Konstantin Kolchin and master student Shinya Abe. Application of PBM in computer animation is a hot topic in research community and it still gains the popularity. Chapter 3, describes the animation of soap bubbles dynamics followed by Chapter 4 introducing the methods needed to do the real-time animation of single phase liquids. PBM methods in computer rendering are used for quite a long time to simulate the light transport and material properties. Application of PBM into a Perception of a strong illuminator can be correctly described by PBM as shown in Chapter 5. Modern trends in computer rendering utilize the graphics rendering units (PGU) to calculate the light transport physical model in real-time. Such methods produce physical ra-

diance values as opposed to faking techniques and approximate lighting models (e.g. Phong model, Cook-Torance model). Chapter 6 demonstrates the usage of PBM and GPU to rendering of metallic paints and their optical effects mostly observed on far east Asian lacquer-ware.

In each chapter a new concept or technique is introduced. At the beginning of each chapter a brief description of related work by other researches to clarify the differences between author's method and their research is included.

Contents

Chapter 1

Introduction

The development and use of Physics-Based Modeling (PBM) methods by many researchers has made it possible to address successfully difficult problems in computer vision, for instance estimation of human body motion, computer graphics, for instance modeling visco-elastic materials, and medical imaging for instance visualization of heart motion, that were not possible with purely geometric and kinematic techniques.

PBM methods utilize geometry, kinematics, dynamics, material properties and optical properties of materials in order to model physical objects and their interactions with the physical world. All the properties including the physical world can evolve in time thus forming complex dynamic system with all well known problems to mathematical community that must be challenged in order to find the stable solution to the system. As opposed to purely geometric models, physics-based models incorporate additional constraints such as material properties, external interaction forces that are very useful in both modelling and simulation applications.

A unique feature of PBM is that it provides a unified methodology for modeling the shape, motion and optical properties of surfaces in one dynamic model. The methodology can model the shape of rigid, articulated and deformable models.

This habilitation presents a Physics-Based Modeling (PBM) framework for 3D shape modeling and growth simulation, animation thin-film and liquids, and optical effects simulation with applications to medical graphics, computer animation and computer rendering. The framework addresses a variety of difficult modeling problems common to the above fields, through the development of deformable models and techniques for animating their shape, motion and interactions. I addition the visual models for perception of strong illuminators and illuminated surfaces and techniques for changing the material properties were developed.

The shapes of many natural objects undergo complex motions that are non-rigid and may be subject to various constraints. Animal bodies, for instance, produce surprisingly complex motions, not only as a consequence of their articulated skeletons, but also because of soft tissue deformations due to muscle action and gravitational effects. In such cases, identifying an animal's constituent parts and their shape is a very challenging problem. The field of geometric modeling has developed methods for representing object shapes, but pure geometry be-

comes insufficient when one faces problem of realistically animating deformable objects and their physical interactions.

Medical graphics (e.g. embryologists) applications face challenging problems to animate and estimate the complex shapes and motions of internal organs such as the digestive system, brain and stomach during the development of a human embryo.

Initially, Ďurikovič et al. in their work on "Reconstructing a 3D structure with multiple deformable solid primitives. Computers & Graphics, 21(5), pages 611-624, 1997" present a class of deformable objects whose behaviors are determined not only by their geometric structures, but also by Lagrangian mechanics principles involving mass, damping, and internal strain energies. The geometry of these models supports both global deformations parameters which represent the global shape features and local deformations which capture the shape details. The differential equations of motion that govern the dynamics make the models constrained by the externally applied forces. External forces may be derived from input data, may arise from interactions, may arise from force fields in simulated environments, or may be applied interactively by the user. The methodology was for the first time demonstrated at largest computer graphics conference in the world SIGGRAPH 1996 and consequently publicized by several international TV companies.

In subsequent chapters we extend the above methodology to develop

1. deformable models that are controlled by a growing skeleton (e.g. articulated figure) which can automatically change the topology,

2. Lyndenmayer system (formal automata system) that can interact with physical world,

3. methods for modeling thin-film interactions and topological changes,

4. methods for modeling liquid phenomena,

5. methods for modeling the human eye perception of strong illuminators including the simulation of corona and glare effects on human pupil and camera stop,

6. models of bi-directional reflectance distribution function (BRDF) capable of simulation of complex optical properties in real-time using the GPU.

Depending on the application requirements, we may employ two types of simplifications to our framework. The first at the model level where we ignore certain higher-order physical effects. The second is at the level of the numerical techniques used to solve the discretized Lagrange equations of motion. These simplifications allow the interactive time simulation of our models including continuous display of results.

1.1 Examples of PBM and their Applications

We will now illustrate the spectrum of modeling problems that fall within the scope of this habilitation through a series of examples from medical graphics, computer animation and computer rendering.

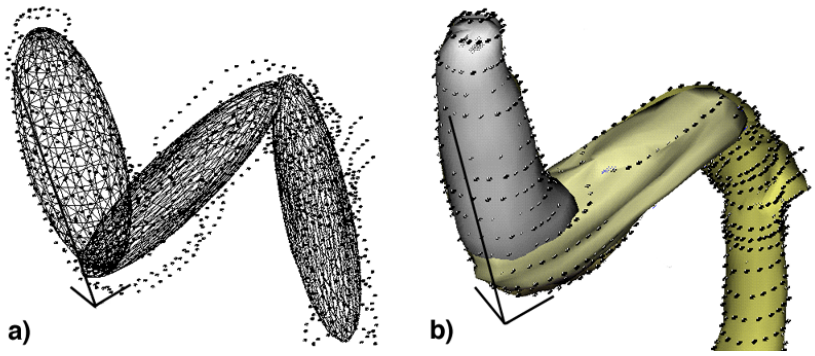

Figure 1.1: Stomach of an embryo reconstructed from range data with deformable models.

1.1.1 Medical Graphics

We have developed a new class of deformable models whose global deformations are controlled by growing skeleton in dynamic environment. We provide the control of kinematic skeleton by means of procedural method (Lindenmayer-system) to dynamically generate the global and local shape changes. Using these new models we have been able to analyze the shape changes and nonlinear motion during the organ development.

Fig. 1.1 shows three deformable parts fitting to the range data of an embryo stomach. The reconstructed shape of the stomach consist of union of three parts.

In Chapter 2 we present a new direction in medical applications which stems from our recent efforts to combine the PBM methods and physiology modeling to simulate the growth process.

Understanding relationships between anatomy, physiology and development is critical to the education of medical professionals. The computer system provides a unique opportunity to investigate these relationships by allowing researches to interact with simulated growth of human embryo.

In Fig. 1.2 we shows the development of a stomach simulated with proposed deformable model with L-system skeleton taking into account positional changes of surrounding organs. Meta-balls where used to obtain a smooth appearance of the shape. The indicated stages represent 28, 35, 70, 84, 140, 252 days of an animated sequence.

1.1.2 Computer Animation

The deformable models are useful for variety of computer graphics applications. Because they are dynamic, our models are well suited to physics-based animation tasks. Such application is illustrated on Fig. 1.3 which shows PBM of the elastic soap bubbles modelled by using a spring system. Bubbles can undergo bubble creation by blowing a wind into a straw, bubble collision by forming clusters and common surface, collision with other objects forming a hemispheres, interaction with external forces (e.g. wind) and bubble destruction. Results related to the soap bubble dynamics were presented at the largest computer graphics conference in the Europe EUROGRAPHICS 2001.

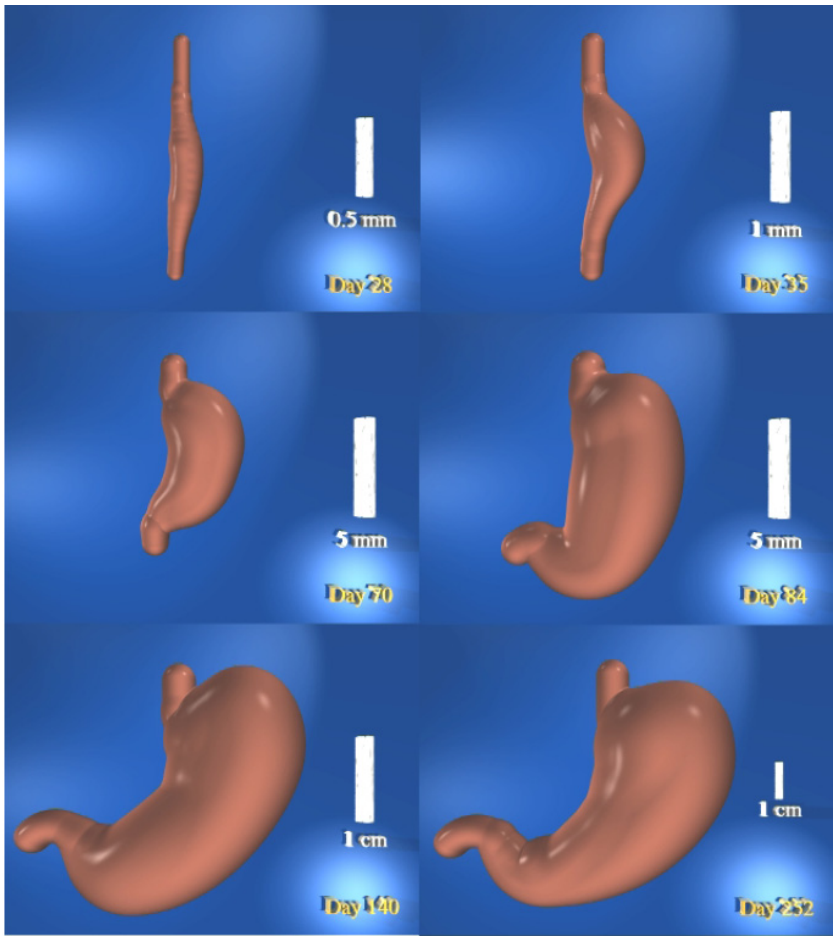

Figure 1.2: Development of a stomach with 28, 35, 70, 84, 140, 252 days of an animated sequence.

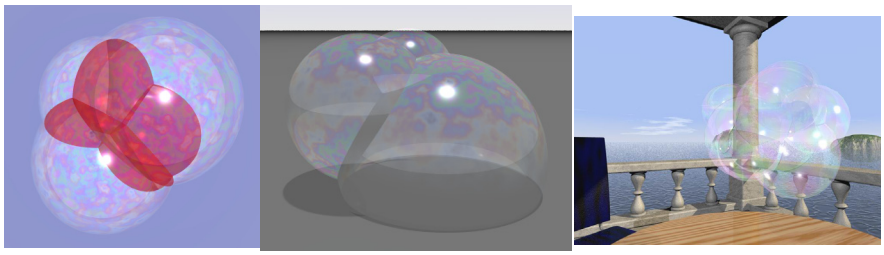

Figure 1.3: Soap bubbles. Left: Common surface formed during bubble contacts shown in red color. Center: Bubbles colliding with a plane forming hemisphere. Right: Cluster of 15 bubbles.

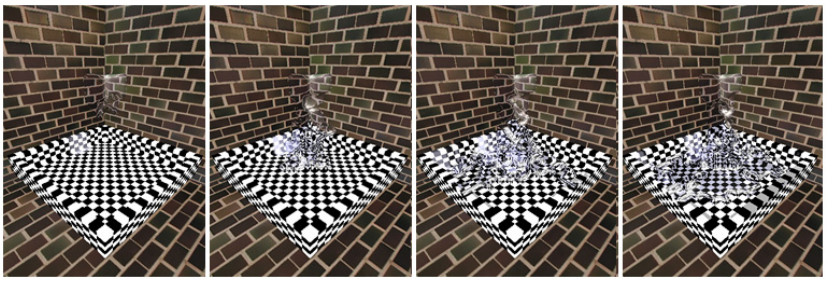

Figure 1.4: Dynamic liquid colliding with pool.

Figure 1.5: Virtual fountain in front of University of Aizu.

So far presented PBM examples have dealt with modeling phenomena involving multiple elastic objects. In Chapter 4 we present a PBM method for medeling a different class of physical phenomena in particular liquids. The frames in Fig. 1.4 show a fountain splash colliding with the ground. The liquid motion was calculated at the 20x30x20 grid. Optical effects such as reflection, refraction and caustics are simulated on fly on GPU.

Fig. 1.5 shows multiple fountains synchronized with a sound.

1.1.3 Rendering

Rendering is process that creates synthetic images for a given scene. A fundamental problem in rendering is to calculate the luminance values for each point of a projection plane by solving the so called rendering equation. Rendering equation is an integral equation describing the light transport in the scene. Depending on the application we can simplify the description of illuminator, optical properties of material or numerical calculation of rendering equation itself. GPU can be used to speed up the calculation at different levels. Current GPU offers also float calculation therefore high dynamic range images can calculated and at the final step the image is mapped to the low range of color displays. PBM can be used, for example to represent light distribution around the illuminator, reflection function at surface point, or when describing other

Figure 1.6: Corona effect created by a digital camera and PBM of a camera corona.

physical effects like scattering and wave properties of the light. The rendering community say that the method is physics-based when it uses a model derived from physics theory (not just faking) and when it outputs physically correct luminance values.

Fig. 1.6 shows the PBM of the light scattering within the camera and film emulsion creating bloom and corona effects. The same effects can be observed directly by a human eye when looking directly to the strong illuminator.

Example shown in Fig. 1.7 demonstrate the PBM applied on to the BRDF representation in such way that rendering of surface with full measured BRDF can be done in real time, including textures and different mixtures of materials used.

Figure 1.7: Real-time rendering of Japanese lacquer-ware using gold, copper, silver metal and reg pigments.

Chapter 2

Growth Simulation of Digestive System using Function Representation and Skeleton Dynamics

We present a new direction in medical applications which stems from our recent efforts to combine the PBM methods and physiology modeling to simulate the growth process.

Understanding relationships between anatomy, physiology and development is critical to the education of medical professionals. The computer system provides a unique opportunity to investigate these relationships by allowing researches to interact with simulated growth of human embryo. Recently, computer graphics techniques have developed sufficiently to enable creating realistic virtual environments. The enormous success of flight simulators to train pilots and driver simulators to train drives make biomedical researches feel that such systems can be developed for biomedical applications, such as for surgical training and embryological study. Additional advantage is that a virtual embryo can serve to demonstrate different pathological cases during the development.

The aim of this research is to link growth models that express mechanical behavior with a visually and structurally accurate anatomical model at certain time of development.

Computer based medical modeling has a long history in the bioengineering community, for example the necessity to develop the 3D reconstruction methods from cross-sections came from this community, originally. While there is a growing body of multimedia medical instruction programs, most of them are based on simple geometrical models. Some recent, surgical training programs use the elastic models for organ representation. However, none of the above attempts model graphically the development of organs.

Since our deformable contours and meshes allowed topology changes thai proved to be very effective in 3D reconstruction of mouse embryo organs. The straightforward idea is to use the reconstructed models for animation of organ development. Although physics-based simulation is guaranteed to generate

natural looking motion, it provides user with very little control over the result produced. In kinematically controlled animation, the user is responsible for every aspect of the generated motion; however, no one can grantee that results are physically correct. In the technique proposed here we take advantage of the strengths of both techniques. We provide the control of kinematic skeleton by means of procedural method (Lindenmayer-system) to dynamically generate the global and local shape changes. There were not the successful simulation of organ growth prior to publication of this research by the author at EUROGRAPHICS 1998 conference. Initially the method was developed to simulate the stomach growth of human embryo later it was extended to the brain and digestive system growth.

The method presented in this chapter has been successfully used to simulate the development of human embryo organs, particularly stomach, brain, and digestive system. The new methodology discussed here is combination of growth model defined by an Lindenmayer-system (L-system) with skeleton of the organ represented as physical spring model. L-systems is a formal language where each symbol has several parameters, that can be used to sent the growth signals. Although, the L-systems are widely accepted by mathematical biologists as good candidates for growth models, they are very deterministic with difficulty to set the appropriate initial parameters. I overcome the problem of setting the initial parameters by embedding the L-system model into the physical environment, while mutual interaction between model and environment is guaranteed.

Shape representation is another problem that we solve here. Methodology of implicit function defined by their skeletons is used. Composition of different organ parts is done by set theoretic operations defined by real functions, hence comes the term "Function representation".

I demonstrate the method by development of digestive system, publications that demonstrate the modeling of other organs can be found in references.

The embryologists found the realistic human organ models and animations of development necessary for their studies. The main idea of this chapter is a methodology producing a realistic animation of development by combining the L-system growth model with a physical model. The skeleton of a digestive system is a line skeleton with a tree structure. Therefore, its growth in length can be simulated by an algebraic L-system which controls the growth of skeleton segments. The global deformations of the skeleton due to the gravity and the lack of space in abdominal cavity are simulated by a dynamics of skeleton segments. The movements that have no physical reasons such as looping are implemented by external forces applied on links controlling the organ movement in space. The convolution surfaces generated by skeletons define the final shape for growth animation. The entire system consists of two steps: First, the actual number of skeleton segments and the length of each skeleton segment is calculated from growth functions, second, the skeleton deformation in space is updated based on dynamics.

2.1 Introduction

The shape and structure of a human embryo digestive system change significantly over a short period of time. It bends, twists and creates many loops during the development. Embryologists therefore strive to visualize the move-

ment and shape changes over time. In this chapter we will discuss how to model the outer shape and the shape metamorphosis during the growth of some human embryo organs, particularly in the digestive system. Controlling the shape metamorphosis between two mesh objects become a problem when they have different topology and geometry. To create the realistic-looking human organ models and to generate the animations demonstrating the growth process require an appropriate methodology. The aim of this chapter is to present a methodology based on the functional representation and convolution surfaces [48, 3].

Ďurikovič et al. [66] have developed a system in which organs and the environment are separate processes, information from each being transmitted using communication modules. The methodology, which is based on algebraic L-systems, was demonstrated on a growth model of human embryo stomach. Method requires setting many initial parameters, not an easy task to do for the complex growth motions and bindings of a digestive system. The proposed work in this chapter significantly decreases the effort required to design of L-systems which can simulate the dynamic growth of a digestive system.

Using implicit surfaces generated by skeletons for growth animation of smooth surfaces seems to be a very good idea, since they can be stored in a very compact way.

Semi-automatic reconstruction methods based on implicit iso-surfaces generated by skeletons that can be used for noisy scattered points of medical organs were studied by Tsingos et al. [65]. The basic idea of their method is the minimization of the distance between the data point and iso-surfaces, during a series of skeleton refinements. Their skeleton consists of disconnected data points while we would prefer a continuous skeleton which suites the animation purposes better.

Several attempts have been made using computer graphics to visualise biological growth. Methods for modeling botanical trees were developed to generate natural tree images [4, 7]. These methods directly model the growth process using statistical data, such as the angle between trunk and branch. An L-system formalism was proposed by Lindenmayer [34], and the method has been used as a general framework for plant modeling. An L-system with several extensions was extensively described by Prusinkiewicz and Lindenmayer [54], and the extensions allow for such factors as context-sensitivity, random variations, and branch cutting. An expansion of the L-system to handle the interaction between plants and their environment has also been developed [33, 53]. A recently proposed method considers the plant and its environment as two separate models with information flowing between them in both directions [52]. Another interesting extension of L-systems is a behavioural L-system capable of animation the autonomous actors by external tactile and behavioral forces [42].

What are our requirements for the growth animation? First, the object should be capable of animating in other words it should have a control skeleton with intuitive shape representation during the growth. Second, shape changes and movements should be smooth i.e. shape function as well as the skeleton elongations and deformations should have a continuous first derivative. Lastly, the movement should correspond to real growth as is assumed by the community of embryologists.

The L-system is an excellent method for modeling botanical growth, as demonstrated in many publications [22, 52]. However, the method cannot be directly applied to human organ development. Trees or plants monotonically

increase their length and thickness of branches. In contrast, the growth process of human organs is more complicated. In the early stages of human organ development, a number of significant changes occur, making it very difficult to simulate organ growth. Generally, organs have no tree structure and their growth is not a monotone process. For example, in stomach and intestine development, different combinations of rotation, bending and twisting take place. These difficulties are unique to the modeling of human organ growth; they are not encountered in the modeling of tree or plant growth. Although, attempts to bend the plant leaves and apexes have been made in plant formation [40], they do not allow the bending transformation of the entire plant structure.

We propose a method that takes the advantage of positional changes in time to efficiently control the growth of organ skeleton. To simulate the complex bends of small and large intestines we propose to use the spring model of a skeleton. The rest length of each spring increases (decreases) according to the growth functions. The dynamic simulation then determines the current position of each skeleton segment (spring) in space while taking into account the space constraints and user-defined external forces. This approach gives the user ability to control such growth movements which cannot be explained by the physical dynamics of neighboring organs.

The structure of the chapter is as follows. First, we describe the biological and physiological development of the intestine system. Section 3, defines the skeleton, describes the skeleton growth based on L-systems and defines the growth functions. Next section describes the skeleton dynamics using a spring model. Section 5, shows how to create the shape from skeleton using the skeleton-based convolution cylinders. Function representation is used to handle the blending of physiological parts. Next, section show the results followed by conclusions.

2.2 Biological Development of the Intestine System

Development of the midgut is a complicated series of events generally involving three phases herniation, reduction and fixation [59].

1. *Herniation.* The development of the primary intestinal loop is characterized by rapid elongation. As a result of the rapid growth and expansion of the liver, the abdominal cavity temporarily becomes too small to contain all the intestinal loops, and they enter the extraembryonic coelom. Coinciding with growth in length, the primary intestinal loop rotates counterclockwise approximately 90°, shown in left of Fig. 2.1.

2. *Reduction.* The intestine returns rapidly to the abdomen at 9 weeks while continuing to rotate about remaining 180°, as shown at the center of Fig. 2.1. The large intestine likewise grows considerably in length and it descends on the right side of the abdominal cavity.

3. *Fixation.* At the beginning of the 6^{th} month and continuing until after birth, the colon comes to adopt its final position, shown on the right of Fig. 2.1.

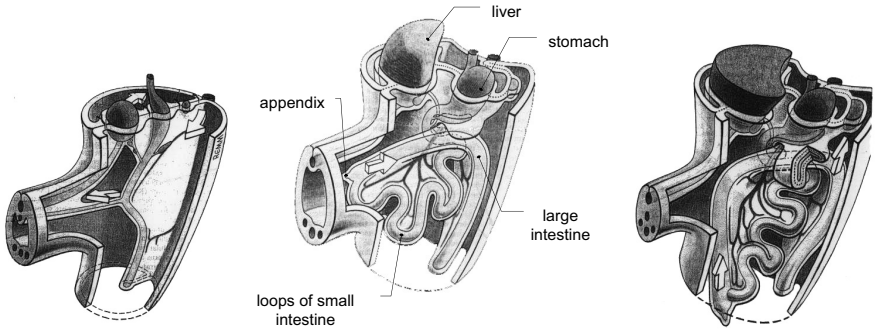

Figure 2.1: The major development stages of fetal gut at 28 (herniation), 70 (reduction), and 120 (fixation) days of development.

2.2.1 Physiological Development of the Intestine System

Every organ is built up from a huge number of cells, each growing with different properties, such as speed and direction of growth. Since, it is not worth of effort to simulate the entire organ as a cellular structure we have to reduce the number of cells without compromising to any great degree such organ properties as volume, surface area, topology, speed of growth in specific areas, and the like. To approximate the shape of an organ while considering the speed and direction of cell growth at the same time, we group the entire set of cells into a number of cylindrical bunches (clusters). Thus, the skeleton of the organ is defined by a chain of linear segments passing through the cluster centers, see Fig. 2.2. Organ growth can then be modeled by the growth of the line skeleton, and variations in shape thickness during the growth process can be captured by variations in cylinder size. When a cylinder changed in size, it was understood that the organ cells grew in the directions emanating from the cluster center. Similarly, when the skeleton segment underwent changes in length, it was understood that the cells included in two adjoined clusters grew in directions parallel to this segment.

Taking into account the above statements, the organs were divided into physiological parts having different speed and direction of growth to suite the animation purposes. The physiological parts of the intestine system are shown in Fig. 2.3 and marked I, II and III for stomach, marked IV for small intestine, marked VII for large intestine, marked V for appendix, and marked VI for vitteline duct.

2.2.2 Measurements of Shape Changes

Several statistical measurements have been made to specify the shape changes which serve as a starting point for the design of growth functions. Measurements should be taken for every physiological part, shown in Fig. 2.3, at every major developmental stage. We obtained several photographs of mouse and human embryos from Medical School of Hiroshima University, Japan. The atlas of embryology [59] contains hand-drawn pictures and photographs of human embryo organs ordered by age. For the purpose of this study the human embryo models with age from range of 28 - 113 days were used. The pictures were scanned, stored in binary form and measured. Both the drawing of chain skeletons and

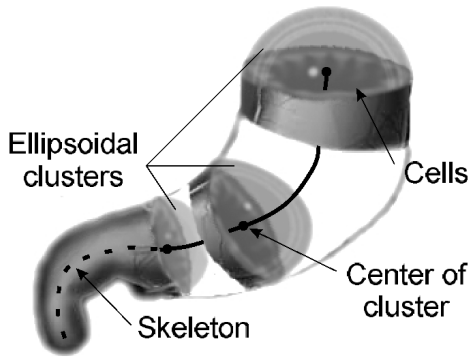

Figure 2.2: Skeleton of the organ and the clusters.

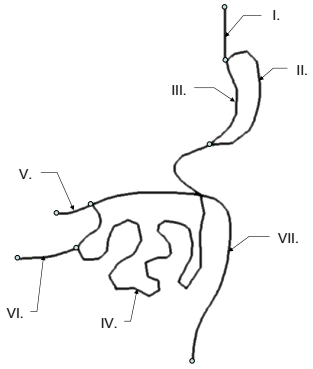

Figure 2.3: Physiological parts represented with line skeleton.

measuring of photographs were done by hand. While refering to Langman's embryology [59] we collected data that are shown in Table 2.1. For each available embryo age (developmental stage) of large intestine its mean thickness and skeleton length are listed. Statistical data for a human embryo stomach have already been summarized by Ďurikovič *et al.*[66].

The organs, at this preprocessing stage, were divided into physiological parts having different speeds and directions of growth to suite the animation purposes.

2.3 Skeleton

The topology of the digestive system is expressed by a tree structure and the development of the tree-like structure can be easily modeled with an algebraic L-system [40, 66]. An L-system formalism was proposed by Lindenmayer [34], and the method has been used as a general framework for plant modeling. The L-systems are extended to by introducing continuous global time control over the productions, stochastic rules for the capture of small variations, and explicit functions of time used to describe continuous aspects of model behavior, in addition to differential equations.

Table 2.1: Shape measurements of large intestine, physiological part no. IV.

Embryo age (day)	Length (mm)	Thickness (mm)	
28	2.76	0.30	herniation
49	15.58	0.45	
58	19.49	0.52	
70	21.77	0.61	reduction
83	24.04	0.82	
113	28.62	1.00	fixation

In some cases it is convenient to describe continuous behavior of the model using explicit functions of time rather then differential equations. For example, global shape transformations varying over time require a large and complicated system of differential equations, while only few explicit functions of time are sufficient for the description of these transformations.

2.3.1 L-system and Turtle Graphics

L-system is a rewriting system based on an *alphabet*, an *axiom* and a set of rewriting rules or *productions* that rewrites the modules according to the production rules at each iteration step. The rewriting process is initiated by an axiom, for example

$$\omega : A(1)B(3).$$

The production that $A(x)$ should be changed to the module $B(x+1)$ with supporting branch $A(x)$ and condition $x < 4$ is represented as

$$p1 : A(x) : x < 4 \rightarrow B(x+1)[A(x)].$$

Consequently, the first derivation step may have the form:

$$A(1)B(3) \Rightarrow B(2)[A(1)]B(3).$$

Modules in the derived string can be related to structural elements in the growing form. To create a graphical model, the derived string of modules is scanned sequentially and modules are interpreted as commands to a turtle. At any point within the string, the *turtle state* is characterized by a position P and a coordinate system $\vec{U}$, $\vec{L}$ and $\vec{H}$ that indicates the turtle's *up*, the *left*, and the *heading* direction, see Fig. 2.4. The coordinates of these vectors can be accessed using *query* modules of the form $?U(x, y, z)$. Some modules have reserved meaning, for example modules $+(\theta)$ and $\&(\theta)$ rotate the turtle counter-clockwise around vector U and L, respectively.

2.3.2 Skeleton Growth

From now on we will assume that the line segment from skeleton can be represented by a module F with two parameters l and τ, which indicate its current length and local time. The following L-system table in Fig. 2.5 shows an example of the growth of a line skeleton with omitted branches. It describes the behavior of the apical segment. Segments will grow within the local time in-

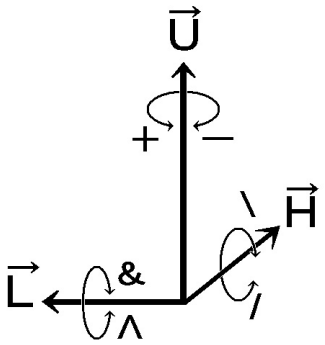

Figure 2.4: Local coordinate system $(\vec{U}, \vec{L}, \vec{H})$ and the rotation syntax.

table Skeleton {
$F(l, \tau)$:
  if $l < l_{th}$ & $\tau \in [\tau_s, \tau_e]$
    solve $\frac{dl}{dt} = g_{l_{\max},p}(l)$, $\frac{d\tau}{dt} = 1$ (a)
  if $l = l_{th}$ & $\tau \in [\tau_s, \tau_e]$
    produce $F(kl, \tau)F((1-k)l, \tau)$ (b)
}

Figure 2.5: L-system table controlling the elongation of skeleton.

terval $[\tau_s, \tau_e]$. The growth of segments causes the first order continuity of the entire skeleton length. Upon reaching the threshold length l_{th}, the segment subdivides into segment F of length kl and a shorter apex with length $(1-k)l$, given by production Fig. 2.5b. The logistic growth function $g_{l_{\max},p}(l)$ describes the increasing length of segment according to a global time t (see Fig. 2.5a).

The recursive application of L-system results in a string of symbols where each symbol represents a linear skeleton segment and parameters determine its length. The total number of symbols is the same as that of skeleton segments.

2.3.3 Skeleton Global Bending

Let us consider the problem of global bending of a tubular shape defined by its central skeleton during its growth over time t. Let s denote the arc-length distance of a skeleton point from the origin of this skeleton then the skeleton at time t_i can be approximated by a parametric curve $\mathcal{P}_i(s)$, $\mathcal{P}_i : \mathcal{R}^+ \to \mathcal{R}^3$. In our implementation we used a cubic interpolation spline with the global parameter s.

Assuming that several key skeletons for a given time ticks $t = t_0, \ldots, t_n$ are known and that they are approximated with curves $\mathcal{P}_i$ we can define the function $\mathcal{P}(s, T)$ that will interpolate the set of curves $\mathcal{P}_i$ in time domain. We can use the cubic interpolative spline for this step, too.

Thus function $\mathcal{P}(s, t) : \mathcal{R}^+ \times \mathcal{R}^+ \to \mathcal{R}^3$ defines the shape of a skeleton along the spatial coordinate s at time t. The explicit function $\mathcal{P}$ can then be used in the L-system in such a way that the heading direction H is always identical to

the tangent vector of parametric curve $\mathcal{P}(s,T)$ for given s and T.

The following L-system table in Fig. 2.6 shows an example of the skeleton bending according to the explicit function of time. The initial structure consists of apex A defined in axiom. The parameter s of the apex represents the current position of the turtle, measured in arc-length distance from the origin, parameters t and τ are the global system time and age of the module, respectively. The production rule (a) creates an organ axis as a sequence of cylindrical segments of length Δs. Apex grows in length and generates segments F with length Δs during the time interval $[\tau_s, \tau_e]$. The process will stop when the actual length s' of all segments is equal to the length l of the skeleton parametric curve. Rotation modules $+\Omega_U$ and $\&\Omega_L$ in this production depend on current turtle position and global time t, they are followed by the query modules $?L$ and $?U$ for future calculation of angles.

Production rule (a) from *SkeletonBending* table in Fig. 2.6 will be repeated recursively till local time is equal to τ_e while table *AngleCalculation* will run as a last step just before the graphical representation. Here the rotation angles are calculated from the tangent vector of function $\mathcal{P}$ at turtle position s and time t (line b). Projection of tangent vector on to the current turtle U vector gives the rotation angle around L axis after conversion to radians (lines c and d). Similar idea is used for module $\&\Omega_U$ giving rotation around axis U, see Fig. 2.7.

After application of both tables the turtle heading direction H is always in the direction of the tangent vector to the arc-length parametric curve of function $\mathcal{P}(s,T)$ at current time T. Parameter Δs should be set such that we have enough cylindrical segments to approximated the high curvature bends.

table SkeletonBending {
Axiom: A(0,0,0)

$A(s,t,\tau)$:
  $s' = s + \Delta s$
  if $s' \leq l$ & $\tau \in [\tau_s, \tau_e]$ (a)
    produce $+\Omega_U(s',t)?L(l_x,l_y,l_z)\&\Omega_L(s',t)?U(u_x,u_y,u_z)F(\Delta s,\tau)A(s',t,\tau)$
}

table AngleCalculation {
#define $K = 57.29$ /* radians to degrees */

$\&\Omega_L(s,t) > ?U(u_x,u_y,u_z)$:
  $t' = (\frac{\partial \mathcal{P}}{\partial x}(s,t), \frac{\partial \mathcal{P}}{\partial y}(s,t), \frac{\partial \mathcal{P}}{\partial z}(s',t))$ /* tangent vector */ (b)
  $\Delta\Omega_L = -K * t' \cdot U$ (c)
    produce $\&(\Delta\Omega_L)$ (d)

$+\Omega_U(s,t) > ?L(l_x,l_y,l_z)$:
  $t' = (\frac{\partial \mathcal{P}}{\partial x}(s,t), \frac{\partial \mathcal{P}}{\partial y}(s,t), \frac{\partial \mathcal{P}}{\partial z}(s,t))$ /* tangent vector */
  $\Delta\Omega_U = K * t' \cdot L$
    produce $+(\Delta\Omega_U)$
}

Figure 2.6: L-system table controlling the bending of skeleton in time.

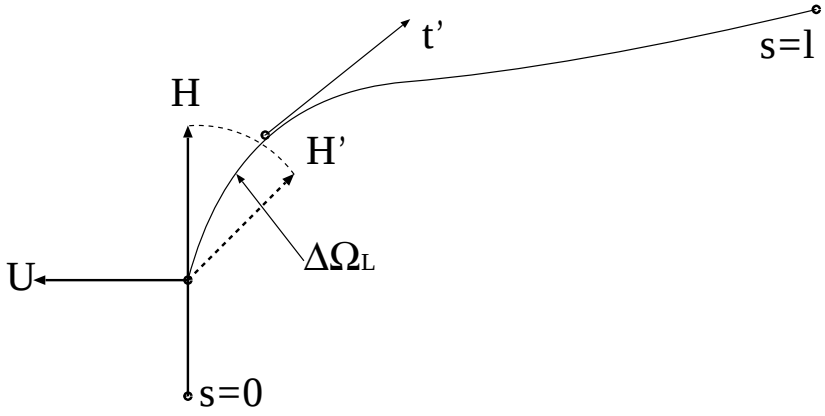

Figure 2.7: Framing the skeleton curves with $\Delta\Omega_U$ around axis U and angle $\Delta\Omega_L$ around axis L.

2.3.4 Growth Functions

Continuous processes such as the elongation of skeleton segments, and growth of cell clusters, over time can easily be described by the growth functions. Growth functions can be then included into algebraic L-systems as explicit functions or differential equations. Growth is often slow initially, accelerating near the maximum stage, slowing again and eventually terminating. A popular example of the growth function [15] is the *logistic* function which is a solution to the following differential equation

$$\frac{\partial r}{\partial t} = p\left(1 - \frac{r}{r_{\max}}\right) r \equiv g_{r_{\max},p}(r). \tag{2.1}$$

Logistic function monotonically increases from initial value r_0 to r_{max} with growth rates of zero at start and end of time interval $[T_0, T]$. It is an S shape function with a steep controlled by a parameter p.

It is well-known when the developmental processes begin and terminate in the case of normal development of organs, their growth follows specific time intervals for different individuals quite accurately, making the use of growth functions with explicit dependence on time reasonable for the biological modeling of organ growth. Therefore, the growth function is designed such that it approximates the collected statistical measurements from Table 2.1.

Length. We assume that skeleton elongation is given by the logistic function, in other words it is a solution of first order differential equation Eq. 2.1, see Fig. 2.5a.

Next, we need to find the initial condition and parameters r_{max}, p of function g such that the solution of Eq. 2.1 approximates the measured data with the smallest error. The length of a large intestine is found by numerical fitting as a function $l(t)$ which is a solution of the following differential equation

$$\frac{\partial l(t)}{\partial t} = \frac{1}{N} g_{28.62,0.07}(l), \tag{2.2}$$

where $g_{28.62,0.07}(l)$ is the logistic function and $l(28) = 2.76$ is the initial condition. The initial condition means that the length of large intestine at the time

of its creation is 2.76 mm. Left graph in Fig. 2.8 shows the growth function $l(t)$. Since, the maximum number of skeleton segments corresponding to the large intestine is N, the growth per segment is given after dividing by N.

Similarly, the growth functions are derived for each physiological part of our digestive system. All growth functions defined by differential equations are calculated by Euler explicit integration on fly during the recursive derivation of L-system string that models the skeleton growth.

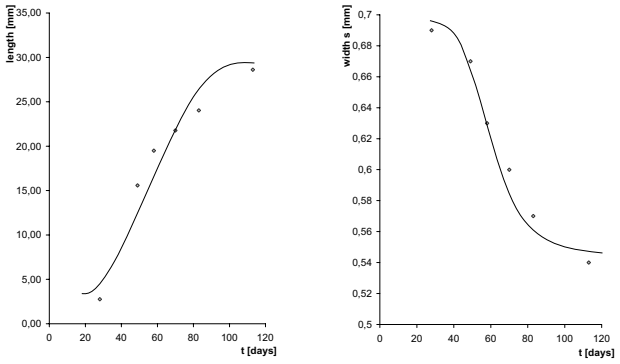

Figure 2.8: Graphs of growth functions. Left) Total length of large intestine in time. Right) Change of width parameter s in time, see Eq. 2.6.

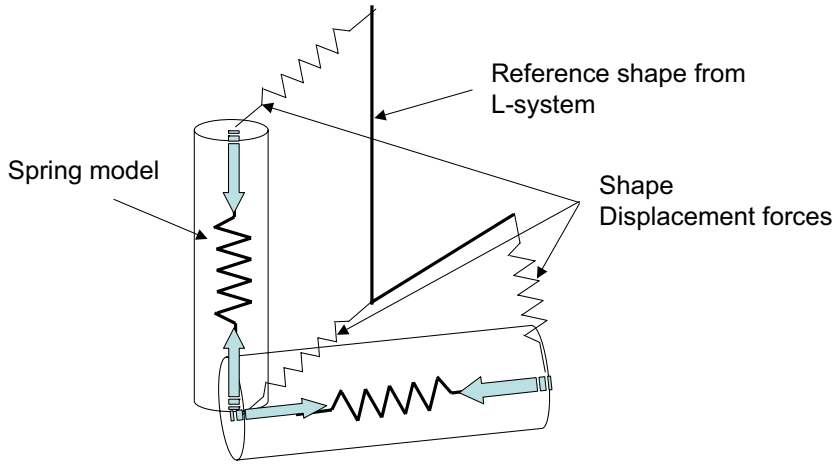

Figure 2.9: Two linear segments represented as spring system with thickness.

2.4 Skeleton Dynamics

The L-system will calculate the number of segments their length, orientation and thickness but it cannot solve the problem of self collision. The problem, discussed herein, is for a given number of segments with given initial position, length and thickness to find the position and orientation of skeleton segments in space due to constraints. The dynamics of the spring system in the external

force field can be used to find the global deformation of the skeleton due to the gravity. The dynamic system uses the damped spring forces with predefined rest length of a spring, spring forces minimizing the shape changes from the initial reference shape, bending forces, collision response forces, and external forces such as gravity force and user interaction force, see Fig. 2.9. The definitions of all the forces used in our system can be found in [69]. The proposed dynamic system tend to minimize its bending energy and the resulting shape will be close to the initial shape.

Space constraints applied on a spring system can handle the deformations due to the lack of space in abdominal cavity and self collision. The collision is handled by considering the thickness of each segment calculated by the L-system. The other known movements can be implemented by external forces applied on springs controlling the organ movement in space. For example, the looping during the herniation shown in first row in Fig. 2.12 can not be simulated just with space constraints it should be defined manually as external force defined in a script before the simulation process.

The size of the abdominal cavity grows during the organs' growth. The cavity growth is modelled by key frame animation where the key shapes are modelled for six developmental stages of the embryo.

The entire system consists of two steps: First, the actual length of skeleton segment is calculated from L-system with growth functions, second, the skeleton deformation in space is updated based on spring dynamics. From the above discussion we know that the L-system with growth functions will give us the string of skeleton segments with length parameter and the rotation modules representing the segment orientation in space. The skeleton dynamics is interconnected with growth functions as follows:

First, we make a turtle representation and create a spring corresponding to each line segment. Initial length and rest length of all springs is equal to the length of a segment calculated by L-system. This way we created the initial physical model with the spring equation of motion described in [69].

Second, the user predefined movements and space constrains are added into the spring system.

Third, the spring dynamics is recalculated using a simple explicit Euler integration method. After few steps of numerical integration we get the springs rested with the updated length and position due to the external forces.

Fourth, after an optimum has been found we must pass the new length and orientation of each segment back to the L-system string. After inserting new parameters in the L-system we can run the derivation step of the L-system again and the whole process repeats.

A single derivation step of the string by L-system is followed by 10 minimization steps of the spring model. The physical parameters of the springs such as stiffness and damping are fixed, as well as size of simulation step. The user normally needs to control the process by the growth functions while he can correct the process by the external force.

2.5 Shape Representation for Growth Animation

The polygonal models can not capture the development of such complex process as the growth of the digestive system. So far, we have created the skeletons of different physiological parts, we need to blend them together to get the smooth shapes. Even though, the convolution surfaces provide nice blending between several parts of organs, the control of the blend shape is very limited. The functional representation is a tool that generalizes the set theoretical operations and generates full range of shapes from simple object union to smooth blending. The animation of such surfaces follow the changes smoothly, even if the topology changes. Because of this advantage the *functional representation* is an excellent tool when the shapes to be modelled are from the natural world. We discuss herein the shape modeling based on skeleton calculated from dynamic simulation and L-system growth.

2.5.1 Function representation

The following material is a reiteration from [48, 3]. Let us consider closed subsets of 3-dimensional Euclidian space E^3 with the definition:

$$f(x_1, x_2, x_3) \geq 0, \tag{2.3}$$

where f is a real continuous function defined on E^3. The above inequality is called a function representation (F-rep) of a geometric object and function f is called the defining function. In three-dimensional case the boundary of such a geometric object is called implicit surface. The set of points $X_i(x_1, x_2, x_3) \in E^3$, $i = 0, ..., N$ associated with Eq.(2.3) can be classified as follows:

$$\begin{array}{lll} f(X_i) & > 0 & \text{if } X_i \text{ is inside the object,} \\ f(X_i) & = 0 & \text{if } X_i \text{ is on the boundary of the object,} \\ f(X_i) & < 0 & \text{if } X_i \text{ is outside the object.} \end{array} \tag{2.4}$$

Blending Union Operation

Intuitively the blending union operation between two initial objects from the set of function representations is a gluing operation. It allows us to control the gluing type in the wide range of shapes from pure set-theoretic union to convolution like summation of terms. After blending union operation between two subjects defined by functions f_1 and f_2 the resulting blended object has the following defining function:

$$\mathcal{F}(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + (\frac{f_1}{a_1})^2 + (\frac{f_2}{a_2})^2}, \tag{2.5}$$

where the absolute value a_0 defines the total displacement of the bending surfaces from two initial surfaces. The values $a_0 > 0$ and $a_1 = a_2 > 0$ are proportional to the distance between blending surface and the original surfaces defined by f_1 and f_2, respectively.

Skeleton based Defining Function

Convolution-based implicit modeling primitives developed in [36] were incorporated in Hyper-Fun project [45]. Let us consider from now on the defining function given by the convolution operator between a line skeleton segment and a kernel [36, 61], i.e. the *convolution cylinder* defined as

$$f_i(X) = \int_{V_i} \frac{1}{(1 + s^2 r^2(v))^2} dv - T, \tag{2.6}$$

where $r = d(X, \mathbf{r}_i(v))$ is the distance from the point $X \in E^3$ to the nearest point on the skeleton primitive $\mathbf{r}_i(v)$. The coefficient s controls the width of the kernel. The integration is performed over the volume, V_i of the skeleton primitive. The kernel function under the integral allows analytical calculation of field function $f_i(X)$ defined by skeleton line segment.

A convolution surface is implicitly defined by a potential function f obtained via convolution operator between a kernel and all the points of a skeleton. The convolution surface thus obtained defines a tubular shape.

Plus Operation between Convolution Cylinders

Convolution surfaces build from complex skeletons can be evaluated individually by adding the local defining functions for each primitive, because convolution operator is linear. With N skeleton primitives the above statement can be written as the following modeling equation in an implicit form:

$$\sum_{i=1}^{N} f_i(x_1, x_2, x_3) - T = 0, \tag{2.7}$$

where f_i is the defining function of convolution cylinder with i-th skeleton primitive and T is the iso-potential threshold value, see central image of Fig. 2.10.

Blending Union of Convolution Cylinders

The effect of blending union operation is demonstrated by two object primitives whose skeleton consists of two line segments one vertical and the other one diagonal, see Fig. 2.10 left. Note, that the left cylinder has a smaller radius than the one on right side. Considering four line segments as a single skeleton of geometric primitive results in the thick shape shown in center image. The right image shows a simple union operation between two convolution primitives.

The sequence of shapes shown on Fig. 2.11 are the blending union operations between two parallel geometric primitives. The geometric primitives and their skeletons do not change but the blending parameters used to blend them are different for each image. In orderer from left side the used parameters are $a_i = 0.01$, $a_i = 0.07$, $a_i = 0.3$, $a_i = 0.5$, and $a_i = 0.7$, respectively for $i = 0, 1, 2$.

Parameter a_i can be used to localize the blending operations therefore, in the case where the shape and size of geometric primitives must be preserved the blending union operation with different parameters a_0, a_1, and a_2 is a good choice. On the other hand when the blending shape is the main concern the convolution plus operation should be used, according to our experience. When both the shape of geometric primitives and that of blending are important the

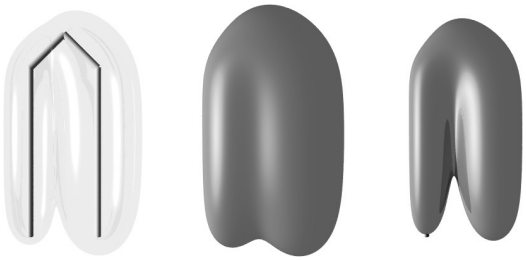

Figure 2.10: Union operations. Left: Two skeleton primitives, center: joining the skeletons into a single one it corresponds to plus operation between convolution cylinders, right: union operation of convolution surfaces.

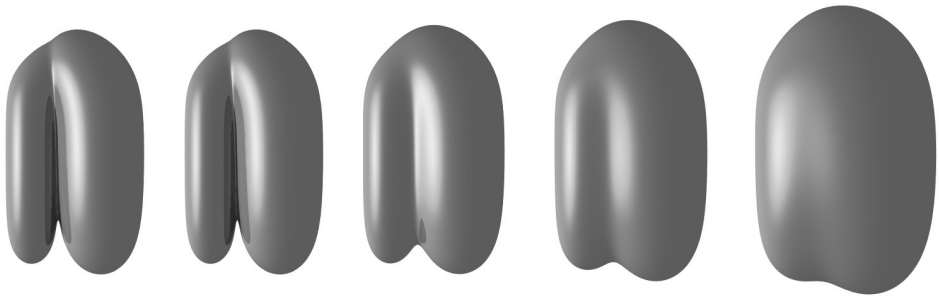

Figure 2.11: From left the blending union operation using $a_i = 0.01$, $a_i = 0.07$, $a_i = 0.3$, $a_i = 0.5$, and $a_i = 0.7$; $i = 0, 1, 2$.

small values of blending union parameters along with the localization of the blending area is a good choice. The F-rep blending union operation has similar advantages as simple convolution union with respect to minimizing unwanted bulges.

2.5.2 Shape from Skeleton

The measured organ models discussed in Sec. 2.2 were divided into physiological parts, at preprocessing stage, having different speed and direction of growth to suit the animation purposes. A single physiological part has the shape defined by the skeleton based F-rep. The skeleton of the physiological part is calculated from L-system and the dynamic system.

We represent the smooth shape of the digestive system in a compact way by piecewise linear skeleton and locally defined convolution cylinders along each linear segment of a skeleton. Thus, the resulting smooth tubular surface is represented by a real function as the blend union operation between many convolution cylinders. The shape of a convolution surface can be varied in several ways: by varying the skeleton, by varying the thickness of convolution cylinders with parameter s from Eq. 2.6, and by the iso-potential threshold value T. For example, the small and large intestines monotonically increase their thickness which can be modeled with the monotonically decreasing parameter s as seen

for the six developmental stages of intestine in Table 2.2.

Table 2.2: Convolution parameters for thickness of small and large intestine.

Embryo age (day)	Intestine			
	Small		Large	
	s	T	s	T
28	0.45	0.2	0.69	0.25
49	0.42	0.2	0.67	0.25
58	0.40	0.2	0.63	0.25
70	0.35	0.2	0.60	0.25
83	0.32	0.2	0.57	0.25
113	0.29	0.2	0.54	0.25

Thickness. As was already mentioned, the increasing thickness of convolution cylinders distributed along the skeleton segments is given by monotonically decreasing the width parameter s in time as shown in Table 2.2. We will transform the solution of Eq. 2.8, $\hat{s}$, that monotonically increases from 0.01 to 0.16 over the time interval $[28, 120]$ into a monotonically decreasing function s by Eq. 2.9, where $s_{max} = 0.7$:

$$\frac{\partial \hat{s}(t)}{\partial t} = g_{0.16,0.003}(\hat{s}), \quad \hat{s}(28) = 0.01 \tag{2.8}$$

$$s(t) = s_{max} - \hat{s}(t). \tag{2.9}$$

Function $s(t)$ is the growth function controlling the thickness of large intestine with a good approximation of data from Table 2.1. The graph of the growth function over the time is shown on right of Fig. 2.8.

2.6 Results

The largest (oldest) model of our digestive system has about 900 springs while running 10 iteration steps of the Euler integration method during the dynamic simulation, the simulation for this model on a Celeron 1.3GHz takes less then 3s. Please, note that most time consuming step is the parsing of long L-system strings, requiring 20 min for the digestive model of 120 days old embryo.

Shown in Fig. 2.12 are several frames from a generated animation simulating digestive growth based on the proposed L-system using the above growth functions. The environment forces and self collision were handled by the spring representation of results obtained from L-system. The shape of the digestive system shown in this figure undergoes global bending transformation and deformations resulting from gravity, animator intervention (looping process), and collision. Some of the intermediate shapes in Fig. 2.12 have disjoined elements due to aliasing in the implicit polygonizer that has difficulties to find a mesh for long thin structures.

2.7 Conclusions

We have presented a method for simulation of the growth of human embryo digestive system. The method uses the shape calculated based on F-rep using iso-surfaces generated by skeleton segments, which provides a smooth and compact representation of the surface usable for complex animations. We propose a method in which the organ growth and global bends are separate processes. The differential growth functions are introduced for an algebraic L-system which efficiently control the elongation of skeleton segments. The skeleton representation is transformed into the spring model within the physical environment which allows us to model the global bending of intestine colon, by user definition of external forces, and collision detection.

The user can guide the growth animation by the direct introduction of constraints or by growth parameters, thus benefiting from his initial knowledge of the growth motion. All F-rep models that are different on each animation frame have been rendered with a POV-Ray [44] ray tracing program and the F-rep polygonizer developed by international group under HyperFun project [3, 45].

Other organs can be modelled with this system as well, the simplest examples being the tree-like structure organs. Unfortunately, the development of the blood vessels and nervous system is not understood very well by embryologists. Nevertheless, we are using this current system to model the human embryo brain and its growth. In the future we plan to use this methods to model the growth of human embryo heart. This system is not limited to tree like structures. With the trivial extension of the L-system to parse the cycles in the skeleton we can model the loops, too. A possible extension, currently under development, is to use the skeletons consisting of triangular patches which should give us the opportunity to define the flat shapes like pillow.

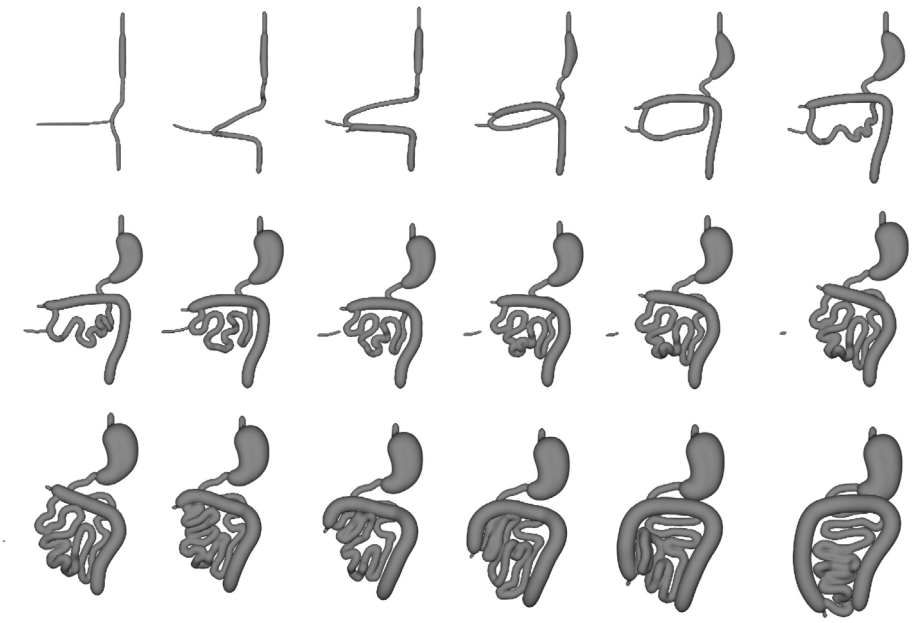

Figure 2.12: Development of a human embryo digestive system with proposed method taking into account the skeleton dynamics and growth functions in algebraic L-system. Function representation is used to define a smooth shape. Indicated stages from left to right represent 28, 34, 40, 49, 52, 58, 64, 70, 76, 79, 83, 94, 101, 107, 110, and 113 days of animation sequence.

Chapter 3

Computer Animation: Animation of Soap Bubble Dynamics, Cluster Formation and Collision

The deformable models are useful for variety of computer graphics applications. Because they are dynamic, our models are well suited to physics-based animation tasks. Such an application is dynamics of soap bubbles.

I will never end blowing the bubbles and observing the beautiful forms and colors of bubble clusters. This chapter describes the dynamics of thin films formed by soap liquids in the air. Plateau has made a lot of experiments and formulated three laws. His results can be attached to the deformable representation of bubble geometry as discussed in this chapter. We need to solve additional problems such as attraction of bubbles, collision with a wet plane, and enlargement of the bubble size during collision. Bubble creation and destruction are also presented. Big challenge is to calculate a common surface between two colliding bubbles.

Since bubble is two-dimensional surface we can restrict our model to 2D deformable surfaces. The surface models are used often in the current shape recovery algorithms. The goal of those algorithms is to reconstruct the original shape from given data sets such data can include range data, contour sets and cross-section images. McInerney and Terzopoulos in International Conference on Computer Vision proposed a deformable balloon model for shape estimation and tracking. Tanaka nad Kishini develop a geometric adaptive subdivision algorithm for surface reconstruction. Even though the algorithm supports local subdivision, it does not used 3D model and it does not guarantee a smooth solution. Ďurikovič in 1996 used the deformable meshes to reconstruct embryo organs from cross-sectional images by proposed elastic spring mesh that also changes the topology and automates the process of reconstruction. Contrary to many of the above techniques, the proposed deformable mesh provides a very efficient and intuitive way of representing shape at various levels of detail and can be applied to both sparse and range data.

This approach is closed to our soap bubble spring model. Each particle of the spring bubble is checked if it is in collision with either obstacle or the triangle patch of other bubble. During the complex collisions we should take care about possible self collision, too. Nevertheless, our elastic model can simulate the formation of common surfaces during the bubble collisions. Simple embedding of collision detection algorithm into our deformation surface allowed us to simulate common surfaces.

Advanced tuning of our model was necessary to simulate fine features of bubble collision such as enlargement of bubble due to the constant pressure. The pressure and the size of the bubble are related by Laplace-Young equation to the surface tension of a liquid mixture. Additional, new ides proposed in this chapter include the derivation of attractive forces between two bubbles and calculation of space optimization of bubble clusters. For the space optimization, the Plateau laws are used to determine the optimum arrangements.

This research was prosed by author on EUROGRAPHICS 2001 conference and it is the first work on dynamics of soap bubbles in 3D space. Several re-implementations were done for example by Prof. Nishita from Tokyo University, Japan.

The optical effects observed on bubbles are due to the light interference on the thin film surfaces with varying thickness. At this chapter I just use a fake colors that can approximate the color patterns very realistically. Physical models that can handle some optical effects are proposed in last two chapters.

This research is still under development and we hope to show the bath filed by a foam very soon.

What is happening when a soap bubble floats on the air? How do bubbles coalesce to form beautiful three-dimensional clusters? The physical-based model and animation described herein provide the answers. This chapter deals with a complete computer simulation of soap bubbles from a dynamic perspective, which should prove to be of great interest to physicists and mathematicians. We discuss the dynamic formation of irregular bubble clusters and how to animate bubbles. The resulting model takes into account surface tension, film elasticity, and shape variations due to gravity and external wind forces.

3.1 Introduction

Foams are of great interest to physicists and chemists studying macroscopic and molecular surface properties such as wetting, dyeing, foaming, coalescing and emulsification. Mathematicians have long been concerned with problems that require minimization of surface areas contained by a fixed boundary. Euler has investigated variational methods to prove the existence of geometric properties associated with minimum-area surfaces and to solve minimum-area problems. Finding the minimal surface of a boundary with specified constraints, usually having no surface singularities, is known as Plateau's problem in calculus of variations. Generally, there may be one, multiple, or no minimal surface spanning on a given closed curve in space. Despite substantial efforts made to obtain an analytic solution, only in the 1930s was the existence of a solution for the general case proven by Douglas [14] and Rado [55], although their analysis could not exclude the possibility of singularities. Plateau showed that the solution to some special cases could be produced by dipping wire frameworks into a bath

of soap solution.

Some experiments, however, are not possible without resorting to numerical calculation. Although the lifetime of pure soap bubbles can be prolonged by special bubble solutions containing glycerine, the bubbles are sensitive to the presence of impurities, dust particles, excess of caustic alkali or fat, making their study difficult. It is therefore necessary to create a plausible dynamic model capable of three-dimensional bubble clustering simulation.

There have been recent studies of interference produced by thin films [23] including effective summaries of thin-film geometries. More advanced methods take into account thickness variation of bubble films under gravity and derive equations describing the interaction between a plane monochromatic electromagnetic wave and the film [28]. The bubble clusters in such studies are considered to be the union of spherical shells and spherical caps calculated mostly with set theoretic operations usually implemented in ray-tracing algorithms.

The aim of this study is to suggest a dynamic, physically based model of a soap bubble that can simulate bubble formation, deformation, collision with a planar surface, and dynamic collision of two bubbles. Section 3.2 recalls the attainments on the geometrical arrangements of a collection of bubbles of different sizes, in particular double-bubble and triple-bubble configurations. Predominant surface tension involved in bubble dynamics is described in Section 3.3, and here we propose an interval range based on statistical deviations of surface tension for different soap solution concentrations. The measured values are then used in a proposed dynamic model of a single bubble in Section 3.4. In Section 3.5, we derive the dynamics of bubble collision with a plane and apply it to double-bubble coalescence. The final section, which is followed by conclusions, presents animated frames of simulated bubble formation, deformation and clustering.

3.2 Soap Film Configurations

A bubble is the minimal energy surface of a type formed by a soap film. In relation to its volume, it has the smallest surface.

3.2.1 Plateau's Laws

Analog solutions to the minimization problems can be produced by dipping wire frames into a soap-solution bath, as shown by experimentalist Plateau [50]. The minimum surface thus produced was found to have common geometrical properties for multiple bubbles, and can be stated as:

1. Three smooth surfaces of a soap film intersect along a line.
2. The angle between any two tangent planes to the intersecting surfaces, at any point along the line of intersection of three surfaces, is 120°.
3. Only four edges can join at a vertex, each formed by the intersection of three surfaces, forming together the tetrahedral angle $109°28'16''$ between any pair of adjacent lines.

Soap froth consists of a collection of bubbles of different sizes, with shape and arrangement governed by the first two of Plateau's rules. When a cluster is

formed from bubbles of different radii the internal surfaces are curved due to pressure differences, as stated in Section 3.2.2. The third rule applies only to minimal surfaces formed by a frame and not to soap froth.

3.2.2 Double Bubble

A double bubble is a pair of bubbles that intersect and are separated by a membrane bounded by the intersection. The double bubble illustrated in Figures 3.1 and 3.2 can have two possible configurations: an ordinary configuration in which two spherical shells are connected, and a nonstandard configuration in which one bubble is toroidal and the other dumbbell shaped. It had been conjectured that two equal partial spheres sharing a boundary of a flat disk separate two separate volumes of air using a total surface area that is less than any other boundary. This case of two bubbles having the same size was proved by J. Hass and R. Schlafy, who reduced the problem to a set of 200 260 integrals. Finally, M. Hutchings *et al.* proved the conjecture for arbitrary double bubbles in R^{n+1}, in March 2000 [27]. They showed that there is no nonstandard double bubble stable in R^{n+1}, which explains why we do not observe such configurations in nature.

The double bubble is thus area minimizing. It is not known, however, whether the triple bubble is also minimizing. Also unknown is whether empty chambers trapped inside the cluster can minimize area for the number of bubbles $n \geq 3$.

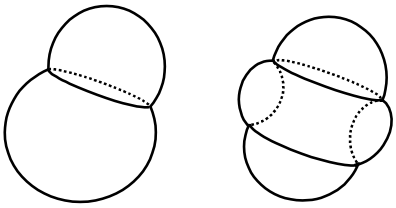

Figure 3.1: Double bubble configurations. left: standard and right: nonstandard.

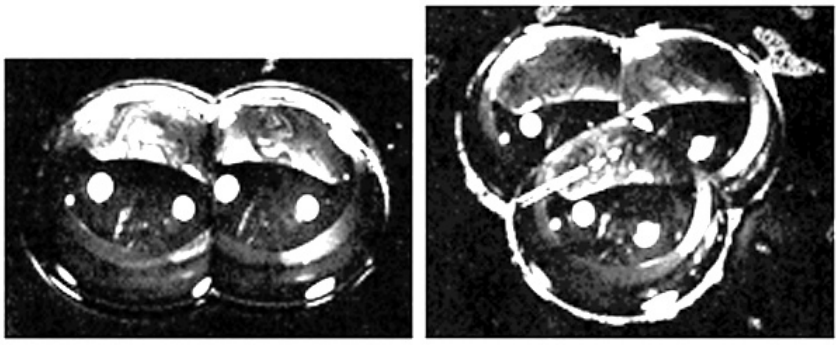

Figure 3.2: Photographs of real double bubble and triple bubble configurations.

Geometry

The major part of each bubble in a double-bubble configuration consists of spherical shells of soap film separated by a spherical cap of soap film. If the bubbles are of different size, the smaller bubble, which always has a higher internal pressure, will intrude into the larger bubble. Figure 3.3 shows the configuration of two bubbles with radii r_A and r_B and internal pressures P_A and P_B resulting from Plateau's laws. It is possible to show that the general property concerning the intersection of three surfaces of soap film at angles 120° provides a method for obtaining the following reciprocal relation [27, 23]

$$\frac{1}{r_B} = \frac{1}{r_A} + \frac{1}{r_C}, \tag{3.1}$$

where r_C is the radius of the common surface. Let M be an arbitrary point where the three surfaces of soap film meet. The cosine rule for the triangle AMB states that $AB^2 = r_A^2 + r_B^2 + 2r_A r_B \cos(\angle AMB)$, where the angle $\angle AMB = 60°$. The distance between bubble centers A to B, therefore, can be derived as

$$AB^2 = r_A^2 + r_B^2 - r_A r_B. \tag{3.2}$$

Considering the triangle AMC and angle $\angle AMC = 120°$, an equivalent equation can be derived for the distance AC, *e.g.*

$$AC^2 = r_A^2 + r_C^2 + r_A r_C.$$

The excess pressure across the surface common to regions A and B, p_{AB}, given by the Laplace-Young equation is constant and related to the radius of the partitioning surface by

$$p_{AB} = P_A - P_B = \frac{4\gamma}{r_C}, \tag{3.3}$$

where γ is the surface tension of a liquid mixture.

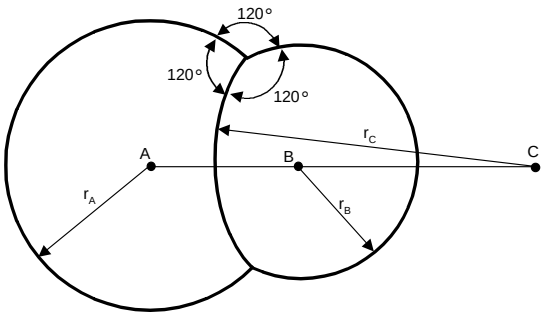

Figure 3.3: Curvatures of double bubble shells.

3.2.3 Triple Bubble

When three bubbles are in contact with one another, as shown in Figure 3.4, three interfaces meet, as well as three spherical shells all at angles of 120°. The centers of curvature of the three bubbles A, B, D and of the three interfaces

necessarily lie in a single plane. The centers of curvature of the three interfaces lie in a straight line, marked by C, E, and F.

Looking at a triple-bubble configuration as three double-bubble compositions, one realizes that the relationship in Eq. 3.1 holds for each pair of bubbles, and thus there are two additional reciprocal relationships

$$\frac{1}{r_D} = \frac{1}{r_A} + \frac{1}{r_E}, \qquad \frac{1}{r_D} = \frac{1}{r_B} + \frac{1}{r_F},$$

where the radius of the third bubble is r_D. Two separating surfaces are newly created with radiuses r_E and r_F between the bubbles with centers A, D, and between the bubbles with centers B, D, respectively. Analogically to Eq. 3.2, the distances between bubble centers A to D and B to D will be

$$AD^2 = r_A^2 + r_D^2 - r_A r_D, \qquad BD^2 = r_B^2 + r_D^2 - r_B r_D.$$

Finally, the excess pressure across the surfaces common to regions A, D, and B, D is given by following equations:

$$p_{AD} = P_A - P_D = \frac{4\gamma}{r_E}, \qquad p_{BD} = P_B - P_D = \frac{4\gamma}{r_F}.$$

The above statements, unfortunately, have not been proven to minimize the surface area. Nevertheless, they are generally used in calculations of thin film arrangements because their implications from Plateau's laws have been supported in numerous experiments.

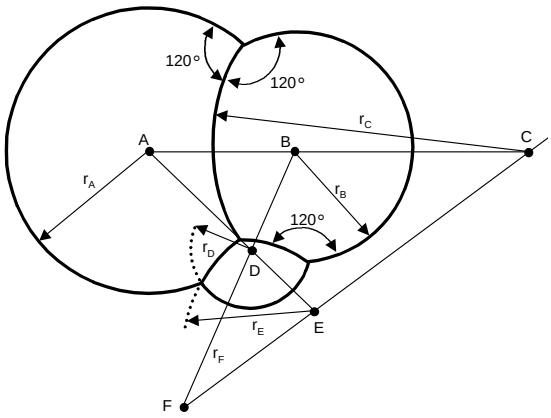

Figure 3.4: Curvatures of triple bubble shells.

3.2.4 N-bubble

One outstanding problem involving bubbles is the determination of the bubble arrangements with the smallest surface areas enclosing and separating n given volumes in space. The cluster of bubbles illustrates Plateau's three laws, fortunately, and thus the statements derived for a triple bubble also hold for each triple bubble in foam.

The Laplace-Young equation used in its simple form in Eqs. 3.1 and 3.3 resulted from zero excess pressure across any point on the surface of the soap film, which is not true for general bubble clusters. The Laplace-Young equation, however, can be applied in these general conditions and Equations 3.1, and 3.3 can also be generalized to foams.

Bubbles can produce hemispherical clusters when formed on a wet planar surface or on the surface of a soap-solution bath. The angles of intersecting surfaces are equal to 120°, while the angle between the bubble hemisphere and the planar surface is 90°.

3.3 Surface Tension

Within the water, at least a few molecules away from the surface, each molecule is connected with its neighbors on every side, so that any given molecule feels no net force. At the surface, however, things are different. There is no upward pull for every downward pull, since of course there is no liquid above the surface. Thus the surface molecules tend to be pulled back into the liquid. If the surface is stretched - as when you blow up a bubble - it becomes larger in area, and more molecules are dragged from within the liquid to become part of this increased area. This "stretchy skin" effect is called surface tension. Surface tension plays an important role in the way liquids behave.

3.3.1 Soap Solution

In a soap-and-water solution, the hydrophobic (greasy) ends of the soap molecule resist being in the liquid at all. Those that find their way to the surface squeeze between the surface water molecules, pushing their hydrophobic ends out of the water. This separates the water molecules from each other, decreasing the attractive force between them. Since the surface tension forces become smaller as the distance between water molecules increases, the intervening soap molecules decrease the surface tension.

3.3.2 Measurement

The rise of a liquid in a capillary tube is still the standard laboratory technique for the precise measurement and comparison of surface tensions. We have measured the viscosity of several bubble solutions discovered by bubble artists at a constant temperature of 20°C. At part of the surface, the gravity of liquid in the tube and the surface tension against air are equal (see Figure 3.5). Thus, the surface tension of the liquid with density ρ_2 against the surrounding gas ρ_1 can be derived from the Laplace-Young equation as follows:

$$\gamma = \frac{\Delta\rho r(h - \Delta h)}{2\cos\theta},$$

where $\Delta\rho = \rho_2 - \rho_1$ is the density difference, r is the tube radius, g is the gravitational acceleration, h is the height of the liquid column, and Δh is the height of the meniscus lens of liquid above h. In the case of pure water, θ is nearly equal to 0 at 20°C and it has the surface tension $\gamma_w = 0.072N/m$. The surface tension of a bubble solution against air was found to be about

one third that of water, and varies with the concentration of soap solution. Therefore, the interval range derived from statistical deviations is proposed to be $\gamma_s \in \langle 0.0195, 0.0265 \rangle$.

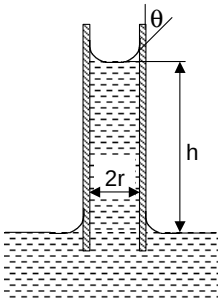

Figure 3.5: The rise of liquid in capillary tube.

3.4 Dynamics

Direct experiments summarized by Boys [9] have shown that a soap film, or bubble, is elastic. One can notice at least two things: large bubbles recover their shape slowly, while small bubbles become round much more quickly. Eventually, a large bubble oscillates much more slowly than a small one when it is knocked out of shape. Different kinds of energies are involved: potential energy, energies of interaction between film surfaces, and surface tension.

In this research we engaged in constraint-based modeling of deformable models. In actual practice, we will consider the elastic model developed in Terzopoulos and Fleischer [64]. The equation of motion governing the elastic deformable model is:

$$M\frac{\partial^2 x_i}{\partial t^2} = F_i(t) - \gamma\frac{\partial x_i}{\partial t} - \frac{\delta\varepsilon(x)}{\delta x_i}, \tag{3.4}$$

where x_i is the position of mass particle i of the object, M is the mass density, γ is the damping constant, $F_i(t)$ is the net externally applied force, and $\delta\varepsilon(x)/\delta x$ is the internal energy that resists deformation. After discretizing the equation, it can be rewritten as a set of first-order differential equations in the canonical form.

3.5 Applied Forces

The geometrical model of a bubble used in simulation is constructed of masses and springs governed by Lagrange's equation of motion, Eq. 3.4. The initial state of the bubble's geometrical model is a sphere subdivided into equilateral triangles, with the corners representing the particles and edges corresponding to damped springs. The force from the springs corresponds to the total internal force $\delta\varepsilon(x)/\delta x$ in the equation of motion. This section discusses all external forces comprising gravity, excess pressure, external fluid, repulsive, and interac-

tion forces

$$F_i(t) = F_{grav} + F_{excess} + F_{drag} + F_{rep} + F_{lennard} + F_{plane} + \dots$$

The total force acting on a particle, illustrated in Figure 3.6, determines the accelerations from which velocities and new positions can be extrapolated by integrating Eq. 3.4.

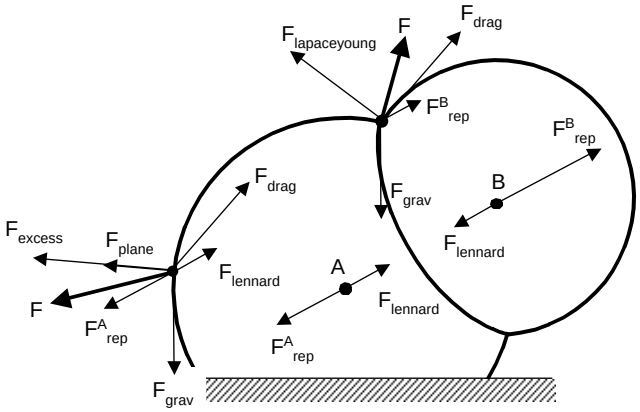

Figure 3.6: External forces applied on a particle. The excess pressure force, F_{excess}, is applied on particles belonging to spherical shells while Laplace-Young force, $F_{lapaceyoun}$, is used for other particles.

3.5.1 Gravity

The gravity is an external force acting on particle i, $F_{grav} = m_i g$. The mass of a single particle is derived from the density ρ and total volume V of soap liquid $m_i = \rho V/n$, where n is the number of particles.

3.5.2 Excess Pressure Force

The direction of the pressure force at any one point is heading in the direction of the surface normal vector at that point. Similar to Eq. 3.3, the excess pressure between the interior of spherical bubble P_A and that of outer space P_O is know to be $p_{AO} = P_A - P_O = \frac{4\gamma_s}{r_A}$, where γ_s is the surface tension for the liquid-air interface and r_A is the radius of bubble. Therefore, we can propose the force due to the excess pressure acting on a given particle i

$$F^i_{excess} = 4w_0 \frac{\gamma_s}{R_i} A_i N_i, \tag{3.5}$$

where R_i is a local radius of mean curvature at particle i, the total area of adjusted triangles is A_i, N_i is the unit normal vector, and w_0 is the unit conversion

constant. The local radius of curvature can be estimated as follows:

$$R_i = \frac{1}{K_i^{\frac{1}{2}}},$$

$$K_i = (3\Delta\theta)/A_i, \qquad \Delta\theta = 2\pi - \sum \theta_j,$$

$$\theta_j = \arccos[\frac{l_j^2 + l_{j+1}^2 - p_j^2}{2l_j l_{j+1}}], \qquad A_i = \sum a_j,$$

$$a_j = [s(s-l_j)(s-l_{j+1})(s-p_j)]^{\frac{1}{2}},$$

where $s = (l_j + l_{j+1} + p_j)/2$, see Figure 3.7. Note that all elements of excess pressure force are derived based on the geometric information only.

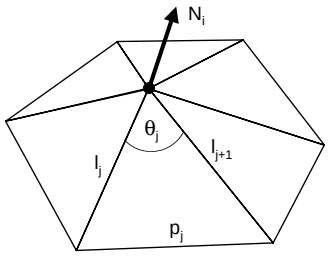

Figure 3.7: Curvature estimation.

3.5.3 Laplace-Young Excess Pressure Force Approximation

Another possibility for defining the force of pressure difference across a curved fluid surface, similar to Eq. 3.5, is the direct application of the Laplace-Young equation. Every point on the surface separating a liquid and gas has a maximum and a minimum radius of curvature, R_1 and R_2, respectively. These occur in planes that are perpendicular to each other, and that are both perpendicular to the tangent plane of the surface. The Laplace-Young equation relates the excess pressure across the surface at any point to the principal radii of curvature at the point by

$$p = \gamma_s \left(\frac{1}{R_1} + \frac{1}{R_2} \right).$$

Employing similar ideas as in previous sections we can derive the second form of the force due to the excess pressure at particle i as

$$F^i_{lapaceyoung} = 2w_0\gamma_s \left(\frac{1}{R_{1_i}} + \frac{1}{R_{2_i}} \right) A_i N_i. \tag{3.6}$$

For a spherical bubble the principal radii of curvature are equal to its radius, that is $R_{1_i} = R_{2_i} = r$ and mean curvature $H = 0.5(1/R_{1_i} + 1/R_{2_i}) = 1/r$. Therefore, the Equation 3.6 is the generalization of Equation 3.5. The first form of excess force, F_{excess}, is a good approximation for a spherical bubble, giving a smaller error than the second form, $F_{lapaceyoung}$, which uses two approximations of principal radii. Nevertheless, the last form is applicable to bubbles that have lost their spherical shapes, particularly when they are in collision or coalescence.

3.5.4 Drag Force

Bubbles immersed in flowing fluid experience forces from shear stress and pressure differences caused by fluid motion. In general, drag cannot be predicted analytically, and thus for most purposes it is calculated experimentally.

It can be shown [68] that the force of a uniform flow with speed $\mathbf{v}$ striking a flat surface of area A is resolved into normal and tangential components as

$$\begin{aligned} F^n_{drag} &= \alpha_n A|\mathbf{v}|\mathbf{v^n_r}, \\ F^t_{drag} &= \alpha_t A\mathbf{v^t_r}. \end{aligned}$$

Assuming a surface with velocity $\mathbf{v_s}$, the relative velocity with respect to a fluid velocity $\mathbf{v}$ is $\mathbf{v_r} = \mathbf{v} - \mathbf{v_s}$. The scaling constants α_n, and α_t depend on the viscosity of the surrounding field. The drag force is uniquely distributed among all particles forming the flat surface, usually formed by three particles of a triangular patch.

3.5.5 Bubble Coalescence

From the similar behavior observed in bubbles in a cluster and in atomic arrangements in a lattice, it is found that when the centers of two bubbles having the same size are separated by a distance that is large compared with the diameter of the bubbles, the force of attraction is very weak. As the separation between bubbles decreases, the force of attraction increases. This force reaches a maximum when the bubbles just come into contact. When the separation is decreased further the force of attraction rapidly decreases and becomes strongly repulsive. The repulsive nature of the force is due to the fixed amount of air in each bubble. The area, A_c, of the common surface separating the bubbles slowly increases and consequently the repulsive force increases. The repulsive force F^A_{rep} pointing from the first bubble center is

$$F^A_{rep} = p_A A_c,$$

where p_A is the internal pressure of the bubble.

3.5.6 Energies of Interaction between Film Surfaces

Interbubble interactions can be described by Lennard-Jones' potential energy, often used by physicists in molecular dynamics simulations and expressed as:

$$E_{lennard} = \left(\frac{d}{|r|}\right)^{2n} - 2\left(\frac{d}{|r|}\right)^{n},$$

where the threshold d represents the zone of minimum potential, n is a parameter and $|r| = |B - A|$ is the length of the vector connecting two bubble centers. The force derived from this potential is

$$F_{lennard} = -\frac{\partial E_{lennard}}{\partial B} = 2n\frac{d^n}{|r|^{n+2}}\left(\frac{d^n}{|r|^n} - 1 + l_d\frac{\dot{r}\cdot r}{|r|}\right)\frac{r}{|r|}. \qquad (3.7)$$

Figure 3.8 sketches the graph of energy potential, $E_{lennard}$, and the magnitude of corresponding repulsive force $|F_{lennard}|$.

Assuming two bubbles of radii r_A and r_B, Plateau's laws can be introduced to Eq. 3.7 from Eq. 3.2 by writing the resting distance

$$d = r_A^2 + r_B^2 - r_A r_B.$$

This force attracts the centers of the two bubbles until they reach the optimum distance d, thus creating a meeting angle of 120° between the bubbles.

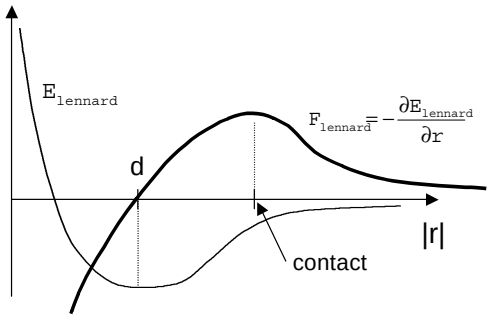

Figure 3.8: Coalescence of two bubbles: energy potential (thin line) and repulsive force (bold line).

3.5.7 Bubble Plane Interaction

The bubble with radius r_A will produce a hemisphere when it collides with a wet planar surface. The contact area of the bubble and the surface slowly increases and consequently the radius of the hemisphere, r_h, increases to a maximum, $r_h = \sqrt[3]{2} r_A$. The force from excess pressure, p_A, acting vertically upwards on the hemispherical section of bubble is known to be

$$p_A A_p,$$

where A_p is the contact area with the plane. Utilizing the above fact, the plane interaction force at each particle not in collision is proposed to be a fraction force in the direction of its normal vector N_i as follows

$$F^i_{plane} = w_i p_A A_p N_i, \qquad \sum w_i = 1, \tag{3.8}$$

where w_i is the weighting parameter. The contact area can be calculated as the area of all triangular patches, of subdivided bubble, belonging to the plane. The effect of plane interaction force, F_{plane}, can be seen in Figure 3.9. Please, note the increase of the bubble radius. With this understanding of the bubble plane collision for a horizontal plane orientation, the question arises whether a hemispherical surface appears for all orientations of the plane? As the gravitational force is negligible compared with the surface tension force, the hemispherical shape is solely due to surface tension for the liquid-air interface. Particles in collision with the plane are influenced by additional force from surface tension for the liquid-solid interface. This force determines whether the bubble absorbed in to the solid surface, pops or forms a hemisphere. For simplicity, the planar surface is assumed to be wet and thus the surface tension for the liquid-solid interface is zero. As a result, the hemispherical surface will slide down a non-horizontal planar surface.

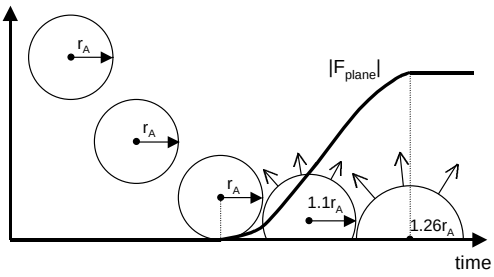

Figure 3.9: F_{plane} force is monotonically increasing in time from 0 to $p_A A_p$ during bubble collision with a plane.

3.6 Results

The proposed method has been implemented as our bubble simulator. Even though our models are fairly large, we are able to follow the progression of simulation interactively on our SGI O2 when the number of bubbles does not exceed six. A simulation preview can be visualized as a wire-frame or a simple Open GL model. When high-quality frames are required, the system can generate POV-Ray ray-tracing scene files automatically for animation purposes.

Before being able to create an adequate simulation, we had to establish spring parameters and scaling constants of external forces. Initially, the bubble position, its size and environment parameters such as wind flow forces are given as input. The system then determines the number of particles needed for surface approximation and the rest length of springs, and locates the bubble center. When a particle collides with the environment or the surface patch of the other bubble, only the tangent component of external force is used in calculating particle speed. We keep track of all bubbles in collision by storing them in a collision matrix. Once two bubbles are in collision they will remain forever in collision. Their speed will be calculated as the average of their respective centers, and coalescence forces are employed. In the current implementation the equation of motion is integrated by Euler's explicit method with time step $t = 0.02\ s$. Development of an implicit integration method is currently underway.

Several successful simulations have been completely calculated and visualized. The first example demonstrates the spring force that enables the surface to simulate natural movement of an elastic object, representing the tension force that tends to minimize the surface, and on the other hand, the excess pressure force that tends to maintain convexity of the surface and increase the surface area. The total influence of such forces creates spherical bubbles as observed in nature. Bubble motion and deformation by an external fluid are shown in the left of Figure 3.10.

The next simulation was performed to show the bubble colliding with a plane and to demonstrate the response to F^i_{plane} force shown in Figure 3.10 right.

Another interesting process successfully simulated is the creation of a bubble from a tube. Initially, the particles at the end of the tube form the disk with boundary particles nailed to the tube by constraints. Particles are connected by springs with short rest lengths. Next, constant air flow through the tube starts deforming the shape. During the creation process the rest length of spring is

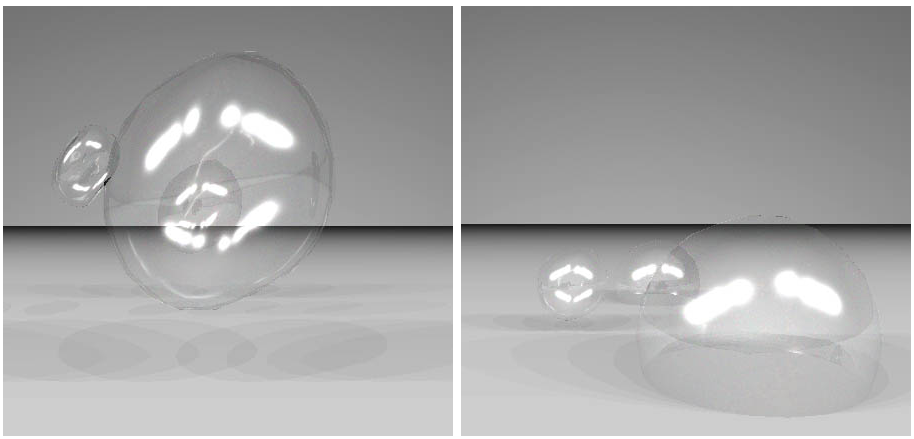

Figure 3.10: Deformation of bubble and bubble after collision with plane.

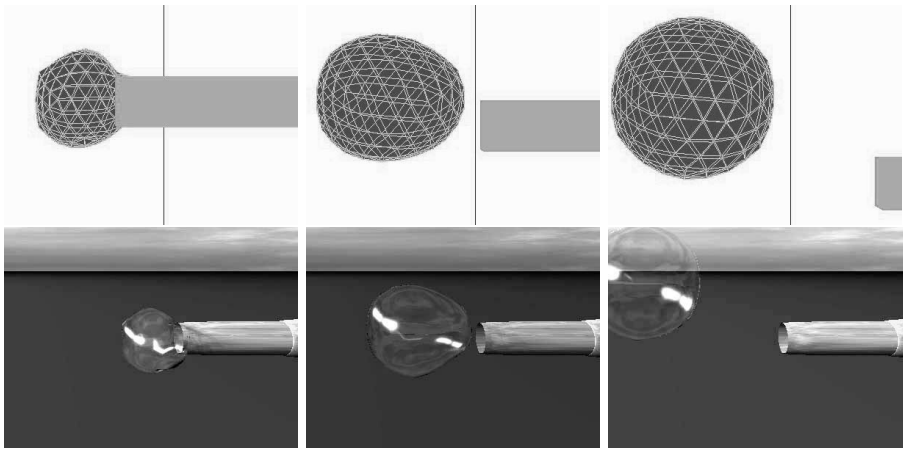

Figure 3.11: Bubble creation.

monotonically increased to a maximum limit. The limit is derived based on the predefined size of bubble just created. Finally, when the bubble grows to its maximum size, the hole at the end of tube is closed with additional triangular patches. For simplicity, the hole is formed by four particles arranged in a square. The screen shots of a bubble creation generated with the proposed simulator are shown in the top row of Figure 3.11, while the bottom row shows the composition of those simulated shapes with an animated scene.

The third example, shown in Figure 3.12, simulates bubble coalescence between two bubbles, employing both the repulsive force F^A_{rep} and surface-interaction force $F_{lennard}$. Those forces are acting on the bubble center that can be difficult to determine in bigger clusters.

Particles forming the interface surfaces, after the coalescence of the two bubbles, moved freely without any constraints in our simulations. One reason for this unnatural movement may be the large rest length of springs forming the interface surface, which is actually equal to that of the original bubbles. This

disturbing effect during the animation was eliminated by clipping out the interface surface and substituting it with a spherical cap that had proper curvature. Clipping was done directly in the ray-tracing program with set theoretic operations. It is not yet understood how to find the interface surface dynamically. However, Plateau's laws were successfully enforced between the outer bubble shells in our simulations.

Finally, Figure 3.13* shows a frame from an animation of a Japanese garden in which bubbles emerge from a bamboo fountain.

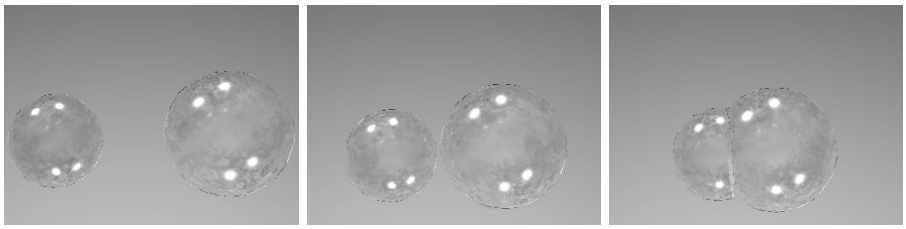

Figure 3.12: Forming double bubble.

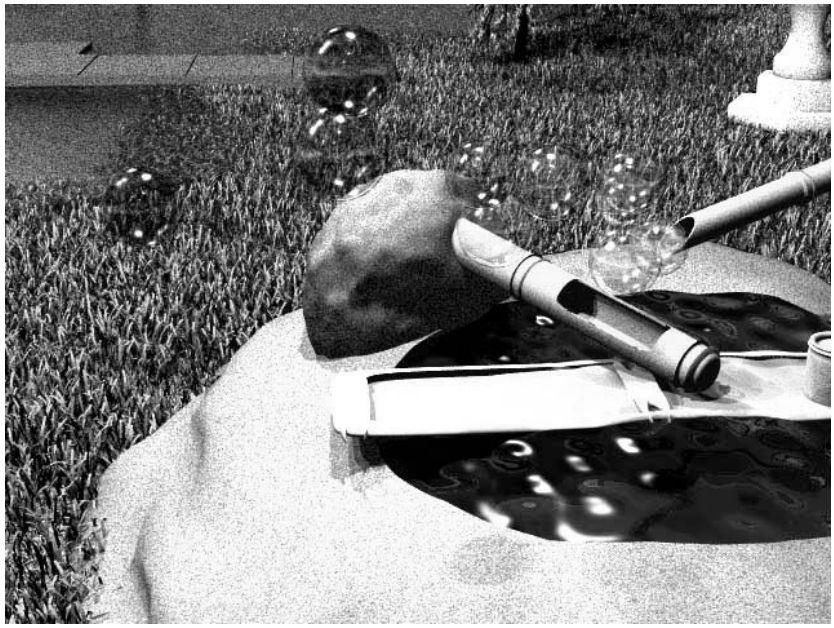

Figure 3.13: Bubble fountain in Japanese garden.

3.7 Conclusions

We have described a new model, which can be used for soap-bubble simulation and bubble-animation control, based on the Laplace-Young equation, Plateau's laws, and aerodynamics. In practice, application of the proposed methods is

quite simple. Yet the new model allows us to simulate complex phenomena such as bubble creation, collision with a plane, and collision of multiple bubbles.

Certain mechanisms remain open problems for future investigation and implementation, including convection, evaporation and suction produced by pressure gradients, which cause water to drain from bubble film and thus thin the film.

Chapter 4

Computer Animation: Accelerated Animation of Liquid Splash

So far presented PBM examples have dealt with modeling phenomena involving multiple elastic objects. In Chapter 4 we present a PBM method for modeling a different class of physical phenomena in particular liquids.

Some very nice animations have been generated in recent years by modeling the interaction between light and water. One characteristic of that work however, has been that the light effects were not done in interactive speeds, nevertheless, at present time it is quite common to implement such effects in the interactive speed with GPU. Another characteristic is that the motion of the water surface is approximated by function without physical background. Such methods included parametric functions and sinusoidal phase functions. Better models inlude papers by Kass and Miller from SIGGRAPH 1990 and Chen and Lobo from Graphical Models and Image Processing 1995. Kass and Miller use fast approximation to the two-dimensional shallow water equations to simulate surface waves in water of varying depth. Their model allows for the refraction and reflection of waves, and takes into account of mass transport, but it does not address the full range of three-dimensional motion found in liquid. Such motion includes rotational and pressure based effects responsible for the fluid's behavior. Chen and Lobo go further towards a physics-based fluid methodology by solving a simplified form of Navier-Stokes equations in two-dimensions. However, they assume that the fluid has zero depth, and calculate the elevation of the surface from instantaneous pressure only. This restricts the class of problems that can be solved using the method, notably, convective wave effects, mass transport, and submerged obstacles are not covered by their technique.

Comprehensive models of fluid motion do exist in the field of Computational Fluid Dynamics (CFD). Unfortunately, animator needs a clear understanding of the system of equations solved so that he can set initial and boundary conditions properly. As opposed to fluid simulator for graphics applications, correct conditions should be set automatically. CFD methods also restrict the external control, making it difficult to force a particular motion of a fluid.

Liquid dynamics has been studied for centuries. Navier-Stokes equation

can model different kinds of fluid motions, there are several numerical methods available to solve the equation. We present a solution through a finite difference approximation to the Navier-Stokes equations. This gives us the solution of a pressure and velocity over the simulated grid. This solution is then used to determine the behavior of free surfaces. Model can handle wave effects such as refraction, reflection, diffraction, together with rotational motion. The N-S equations are solved over a coarse, rectangular grid containing obstacles. Boundary conditions are set automatically. To obtain the detail motion from a coarse velocity field we focus attention on regions of fluid surfaces. The surface is represented as a net of massless marker particles carried around the fluid surfaces.

We contributed to the solution of a problem to find the accurate interface between the liquids or to find the liquid/air boundary. Recently, Fedkiw has presented the methodology of level sets to handle the continuous changes of liquid boundaries.

In this chapter we do not use the level set approach although it seems to be unavoidable in the future improvements of our methods. At the first stage of the development we focused mostly on the calculation and visualization speed approximately 1 frame per second. Fluid motion is calculated on MAC grid and particles are inserted into the grid for visualization purposes to find the liquid boundary.

Proposed idea consists of fast tracking of liquid surface by subdivision surface methods and hardware based rendering with many optical effects observed on liquids such as reflection, refraction and shadowing.

We present a method for simulation and visualization of a liquid splash motion. The proposed system is composed of a computational fluid dynamics and a polygon based renderer using multi texturing graphical acceleration. The system well supports the *Navier-Stokes* solver with adaptive time step and viscosity. The final liquid surface is obtained from the velocity field by rough estimation of the surface and then subdividing the surface. Rendering of the surface is performed by multi texturing method using the caustics, reflection, liquid and object textures.

4.1 Introduction

Physical based simulation, animation and visualization are playing an important role in the field of computer graphics (CG) in recent years. The improvements of physical based methods support CG-designers in producing realistic CG animation of natural phenomena; for example, rigid body collision [38], destruction of walls [43], explosions [70], fluid dynamics, steam motion and flames.

The time needed for simulating and rendering the physical based effects consumed several hours per frame using ordinary ray-tracing method. Many researchers apply computational fluid dynamics (CFD) for physical based animation to generate realistic fluid motions. *Foster* [20, 18, 21] developed a modeling and controlling method based on CFD, which reduces simulation time costs and implements a high controllability of fluid.

Kunimatsu [30] reduced the rendering time of fluid media to 60 seconds per frame. Their team utilized modern high performance Personal Computer (PC) and graphics acceleration hard-ware. However this method can not treat small

fluid objects such as splash and spray.

Our final goal is to develop a real-time system for realistic fluid animation. The system will employ the following specifications:

1. Dynamic fluid motion including mixture, splashing, self-collision, etc.
2. Realistic appearance including th refraction, reflection and caustics.
3. Real time simulation and animation.

For satisfying these terms, we planned to combine and improve the methods described in Foster [21] and Kunimatsu [30]. We add two new ideas to this hybrid method. The first, is the adaptive time step iteration and automatic control of system stability, which reduces simulation time costs. The second idea, is the introduction of reflection texture mapping to realistic water appearance.

The main application of the proposed method are in the motion picture industry. This system gives high quality previews for CG-designers therefore, results are easily evaluated and instantly modified. To render final high quality animation sequences, we focused on the appearance and dynamics of water using a non-real-time rendering algorithms.

The chapter has the following structure. Section 4.2 describes the simulation method. Section 4.3 describes the polygonization of water surface capable of following the splash changes and the hard-ware accelerated rendering method. Section 4.4 shows the rendering results and the time costs of simulation and rendering.

4.2 Simulation Method

The key point of this simulation method is to combine the application of Navier-Stokes equations on low resolution finite difference method and use maker particles for tracking water surface. This method requires only low calculation cost and it provides high controllability for water motions.

4.2.1 Physical Model

At human scale, the behavior of water is influenced by the following forces:

Momentum : Water moves under the laws of energy and momentum conservation.

Gravity : Gravity and pressure cause most water waves.

Viscous drag : This is the key producing the real behavior of water. This part leads directly to self-propagating rotation.

Pressure : Pressure is the important fact in the behavior of the surface of water. The motion of the water surface is influenced by changes in pressure within the volume.

Other motion factors, turbulence and surface tension, cause few effects relative to the above four forces at human scale.

The *Navier-Stokes* equation includes these four factors and completely describes fluid motion. The equation consists of two parts. The first part is a

non-linear equation, which shows relations among velocity $\mathbf{u}$, pressure p and forces:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \nabla \cdot (\nabla \mathbf{u}) + \mathbf{f} - \frac{1}{\rho}\nabla p, \tag{4.1}$$

where velocity field $\mathbf{u} = (u, v, w)$, ν is kinematic viscosity and $\mathbf{f}$ is the sum of external force vectors. Constant ρ is density of fluid, which is equal to 1 for water. Time is denoted as t and operator ∇ is

$$\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}).$$

The second part, of *Navier-Stokes* equation, models mass conservation of liquids,

$$\nabla \cdot \mathbf{u} = 0. \tag{4.2}$$

This equation states that the net inflow and outflow is zero. Together with suitable boundary conditions, the *Navier-Stokes* equation can be used for simulation.

Our simulation system runs on 3-dimensions and treats only the gravity force $g = 9.8ms^{-1}$ as an external force. Thus, Eqs. 4.1 and 4.2 can be written as [20]:

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} &= \\
-\frac{\partial p}{\partial x} + \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) & \\
\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} &= \\
-\frac{\partial p}{\partial y} + \nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}) & \\
\frac{\partial w}{\partial t} + \frac{\partial wu}{\partial x} + \frac{\partial wv}{\partial y} + \frac{\partial w^2}{\partial z} &= \\
-\frac{\partial p}{\partial z} + g + \nu(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}) &
\end{aligned} \tag{4.3}$$

and

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{4.4}$$

4.2.2 Simulation Grid

In order to solve the *Navier-Stokes* equations using central finite-difference method, all solid obstacles and the atmosphere in the scene are approximated using a series of fixed cubic grids. Velocity and pressure are defined on the grids.

Pressure is defined at the center of each cell. Each velocity component u, v and w in x, y and z directions, is defined at the centers of each face of a cell, shown in Figure 4.1. While solving equations in discrete form, velocities which do not lie on faces are required. In this case, unknown velocities are averaged among the nearest values, e.g, $v_{i,j,k} = \frac{1}{2}(v_{i,j-\frac{1}{2},k} + v_{i,j+\frac{1}{2},k})$.

Each cell has four content attributes which are *Full, Surface, Obstacle* and *Empty*. An *Empty* cell represents atmosphere, an *Obstacle* cell contains a solid obstacle, a *Surface* cell contains fluid, but faces at least one *Empty* cell, finally a *Full* cell is filled with fluid and does not face any *Empty* cells.

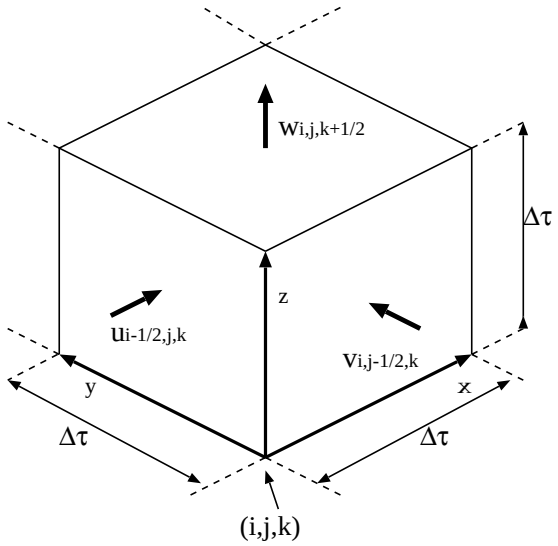

Figure 4.1: Location of staggered velocity components and pressure on a cell. $\Delta\tau$ is grid size.

4.2.3 Boundary Conditions

The *Navier-Stokes* equation can be applied automatically without testing surface or obstacle positions when the boundary conditions are set. Boundary conditions are determined from the attributes of a cell. A boundary is the interface between water and the atmosphere, or water and an obstacle. To simplify the explanation, we use a 2-dimensional model in (x, z) coordinate system.

Obstacle and water

Left image of Figure 4.2 shows an obstacle and a water surface. To avoid water passing into the obstacles, normal velocity on the obstacle face must be zero namely, $u_0 = 0$. If the surface of an obstacle is non-slip, tangential velocities are also zero. This condition is applied indirectly by setting tangential cell face velocity inside the obstacle equal to the opposite value of tangential velocity of the neighbor cell, $w_0 = -w_1$. If the surface of an obstacle is smooth and water can slip, inner tangential velocities are set equal to outer tangential velocity, $w_0 = w_1$.

For eliminating any acceleration across the obstacle boundary, the pressure of the obstacle cell is set the same as the pressure of the adjacent fluid cell.

Water and atmosphere

Equation 4.4 is used to set boundary conditions at *Surface* cell. If the cell is surrounded by three *Surface* or *Full* cells, the velocity of the open side is calculated from the divergence of the target *Surface* cell. For example, consider the situation on right of Figure 4.2, the velocity $w_{i,j,k+\frac{1}{2}}$ can be derived from

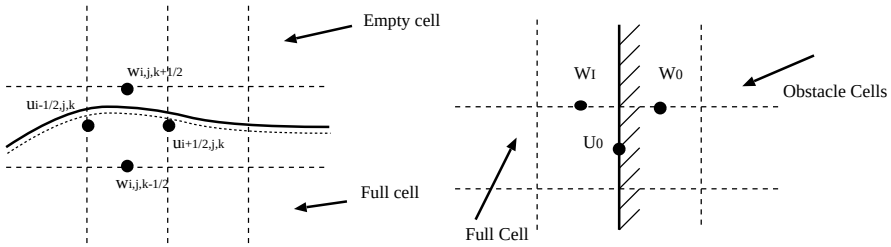

Figure 4.2: Boundary Condition. Left: The free surface cell. Right: The obstacle and the full cell.

explicit finite difference method of Eq. 4.4 as

$$\frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta z} = 0,$$

where Δx and Δz are grid sizes. Considering $\Delta x = \Delta z$, the above equation can be written as

$$w_{i,j,k+\frac{1}{2}} = w_{i,j,k-\frac{1}{2}} - (u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}).$$

In the case of cell surrounded by two fluid cells with attribute either *Surface* or *Full*, each open side velocity is set equal to the opposite side velocity. This modification satisfies Eq. 4.2. If the case when three sides of the cell are open, velocity of the side facing the fluid is carried to the opposite side and, the other two sides are not changed.

In the 3-dimensional system, there are 64 different combinations of *Empty* and *Full* cells. Please, note that the pressure of the *Surface* cell is set equal to the atmospheric pressure.

4.2.4 Numerical Method

The calculation step is divided into two parts. The first part calculates new velocities using Eq. 4.1 with the velocities and the pressures of the previous time step. The resulting velocities will not satisfy mass conservation, Eq. 4.2. Therefore, the second part, modifies the resulting velocities and pressures of former calculation using Eq. 4.2.

Calculation of velocity field

To calculate the velocity field, explicit finite difference approximations are applied to Eq. 4.3. The velocity $\tilde{u}_{i+\frac{1}{2},j,k}$ is a new velocity in next time step, Δt is time step length and Δx, Δy and Δz is grid size, $\Delta x = \Delta y = \Delta z = \Delta \tau$.

Velocity component of $u_{i+\frac{1}{2},j,k}$ is then updated by

$$\begin{aligned}\tilde{u}_{i+\frac{1}{2},j,k} = u_{i+\frac{1}{2},j,k} + \Delta t\{\frac{1}{\Delta\tau}[(u_{i,j,k})^2 - (u_{i+1,j,k})^2 \\ + (uv)_{i+\frac{1}{2},j-\frac{1}{2},k} - (uv)_{i+\frac{1}{2},j+\frac{1}{2},k} + (uw)_{i+\frac{1}{2},j,k-\frac{1}{2}} \\ - (uw)_{i+\frac{1}{2},j,k+\frac{1}{2}}] + \frac{1}{\Delta\tau}(p_{i,j,k} - p_{i+1,j,k}) \\ + \frac{\nu}{\Delta\tau^2}[u_{i+\frac{3}{2},j,k} + u_{i-\frac{1}{2},j,k} + u_{i+\frac{1}{2},j+1,k} \\ + u_{i+\frac{1}{2},j-1,k} + u_{i+\frac{1}{2},j,k+1} + u_{i+\frac{1}{2},j,k-1} - 6u_{i+\frac{1}{2},j,k}]\}.\end{aligned}$$

Similarly, we can write $v_{i,j+\frac{1}{2},k}$ and $w_{i,j,k+\frac{1}{2}}$, except the gravity component is added to $w_{i,j,k+\frac{1}{2}}$. In each iteration, new velocities are updated by calculating the above equations with previous velocities and pressures.

Mass conservation

The result of the calculation described in previous section does not satisfy Eq. 4.2. The efficient way of enforcing incompressibility is to modify pressures using the Laplacian operator [19],

$$\nabla^2 p = \frac{\rho\nabla\cdot\mathbf{u}}{\Delta t}.$$

Applying explicit finite difference approximations at the center of the simulation cell, the above equation can be discretized to

$$\begin{aligned}p_{i-1,j,k} + p_{i+1,j,k} + p_{i,j-1,k} + p_{i,j+1,k} + p_{i,j,k-1} \\ + p_{i,j,k+1} - 6p_{i,j,k} = \rho\frac{\Delta\tau}{\Delta t}(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k} \\ + v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k} + w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}).\end{aligned}$$

The above equation forms a linear system $\mathbf{AP} = \mathbf{b}$. $\mathbf{A}$ is a regular and symmetric matrix. The diagonal coefficient a_{ii} is equal to negative number of cells adjusted to cell i, and non-diagonal elements are set $a_{ij} = a_{ji} = 1$ if the cell at j is the neighbor of the cell at i, $\mathbf{P} = (p_{0,0,0}, p_{0,0,1}, \ldots, p_{i,j,k}, \ldots)$ is a vector of unknown pressures. The SOR [29] iterative method can solve this sparse linear system.

After calculating new pressures, the velocities of each cell are updated by

$$\tilde{\mathbf{u}} = \mathbf{u} - \Delta t\frac{1}{\rho}\nabla\cdot p.$$

4.2.5 Tracking Water Surface

To utilize massless marker particles is a simple and effective way for tracing free water surface. Marker particles are introduced into the simulation system at inflow points and follow the velocities of the cell which the particle is in. The particle does not have mass and does not influence calculations. Marker particles are just used for free water surface detection. The position and velocity of particles are calculated from velocities of adjacent cells using an area weighting interpolation scheme demonstrated in Figure 4.3. After new particle positions are found, attributes of each cell are changed as follows:

- A cell which does not contain any particles is an *Empty* cell.
- A cell which has at least one particle and faces at least one *Empty* cell is a *Surface* cell.
- A cell which has at least one particle and does not face an *Empty* cell is a *Full* cell.

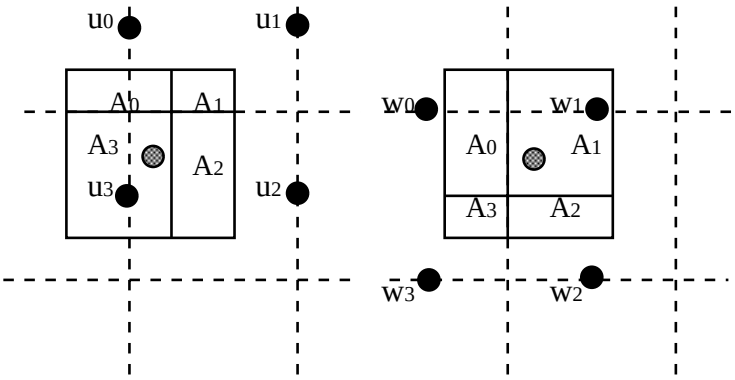

Figure 4.3: Gray point is position of particle p. New velocities of p are delivered from $u_p = A_0u_0 + A_1u_1 + A_2u_2 + A_3u_3$ and $w_p = A_0w_0 + A_1w_1 + A_2w_2 + A_3w_3$.

4.2.6 Stability

Long time step makes total simulation time shorter but can introduce the instability. We utilize an adaptive time step which guarantee the largest time step with stability. To keep this simulation system stable, viscosity ν and time step Δt must satisfy the following conditions [18],

$$\Delta t|\mathbf{u}| < \Delta\tau, \tag{4.5}$$

$$\nu\Delta t < \frac{\Delta\tau^2}{6} \tag{4.6}$$

and

$$\nu > \max\left[(\frac{\Delta t}{2}u^2 + \frac{\Delta\tau^2}{2}\frac{\partial u}{\partial x}), (\frac{\Delta t}{2}v^2 + \frac{\Delta\tau^2}{2}\frac{\partial v}{\partial y}), (\frac{\Delta t}{2}w^2 + \frac{\Delta\tau^2}{2}\frac{\partial w}{\partial z})\right]. \tag{4.7}$$

If the situation that viscosity does not satisfy Eq. 4.7 occurs, viscosity will be increased for satisfying this equation at an unstable cell. Each cell has This modification causes few visual effects to the simulation result. After modification of viscosity, time step Δt is checked by the Eqs. 4.5 and 4.6. If a time step does not satisfy them, it will be decreased to satisfy both conditions.

4.2.7 Implementation of Algorithm

The simulation system is composed of two parts. The first part initializes simulation grids, the initial water position and its velocities, obstacles, inflow and outflow positions. The second part iterates the calculation of the *Navier-Stokes* equation. The outline of the second part is as follows:

1. Determine the contents of each cell using the surface tracking method.
2. Set boundary conditions for the free surface and obstacle cells.
3. Check stability conditions and modify viscosity and time step.
4. Compute velocity for all cells filled with water using the *Navier-Stokes* Equations.
5. Compute new pressures and velocities which satisfy mass conservation.
6. Compute maker particle positions
7. Update the position of the surface.
8. Repeat

4.3 Rendering Method

The combination of texture mapping techniques and surface subdivision methods provides a good visualization of liquid surfaces. The rendering consists of following steps:

1. Polygonization of water free surface.
2. Generation of caustics textures.
3. Mapping refraction textures
4. Blending caustics, reflection and refraction textures.
5. Drawing water surface and scene.

Using this method, the refracted image is directly mapped to the water surface.

4.3.1 Polygonization of Water Surface

In this simulation system, the environment is divided into a low resolution gird. We divide the grid into smaller girds and determine more accurate surface positions, shown in Figure 4.4.

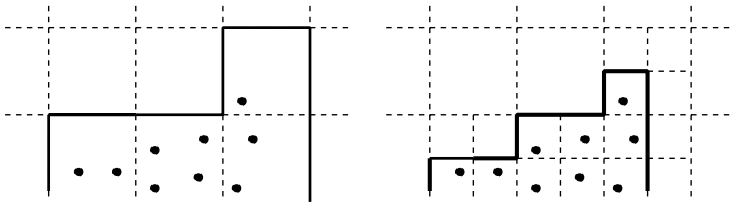

Figure 4.4: Subdivided grid and surface detection.

4.3.2 Subdivision Surface

The resulting polygon mesh is very rough for rendering realistic images. The mesh surface needs to be modified to be smoother by applying a surface subdivision algorithm. The *Catmull-Clark* or *Loop* subdivision surface [62] were used. Both subdivision algorithms are very simple to implement.

We use the Cutmull-Clark method for subdividing surface. The method can not handle non-manifold edges. To overcome the problem, non-manifold edges detected on the coarse-mesh are split into separate cells.

4.3.3 Caustics Texture

The effects of caustics make a more realistic appearance. At first, for each object two textures are created, one is the original texture which contains the objects surface image, and the other is the caustics texture initialized as zero. For each polygon of water surface mesh, an illumination volume is defined by sweeping in the light direction as shown in Figure 4.5. If an intersection occurs between the illumination volume and the object surface, intensity will be added to texels of caustics texture on the projected area of the illumination volume. The intensity is obtained from Equation [30]

$$I_{caustics} = (\mathbf{N} \cdot \mathbf{L}) \frac{S_1}{S_2} I_{light}, \tag{4.8}$$

where $I_{caustics}$ is an intensity of caustics and I_{light} is an intensity of light.

4.3.4 Reflection Texture

The reflection effect can be calculated and stored in to a independent texture. Consider Figure 4.6, the intensity of each texel is calculated as follows:

$$T_{reflection}(\theta) = \cos^n \theta, \tag{4.9}$$

where n is the user defined value. It determines the sharpness of reflection.

4.3.5 Refraction Mapping

To simulate refraction effects, the texture of an object is directly mapped to the water surface polygons. The texture coordinates, u and v, are calculated by tracing the ray's route from camera to target object by

$$n_{air} \sin \theta_1 = n_{water} \sin \theta_2, \tag{4.10}$$

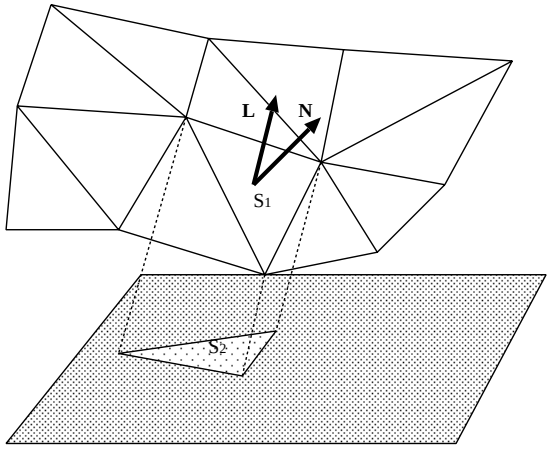

Figure 4.5: Generation of caustics texture. **N** is a normal vector of surface polygon, **L** is the light direction, S_1 is an area of surface polygon and S_2 is a projected area of illumination volume and the gray plane is the texture of an object.

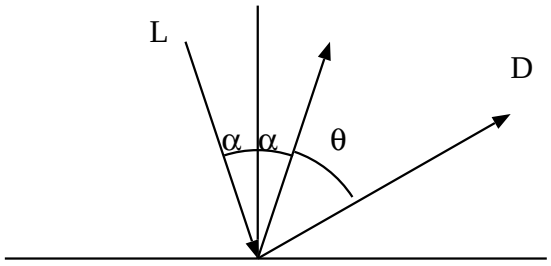

Figure 4.6: **L** is a light direction vector and **D** is a camera direction vector.

where n_{air} is a refractive index of air and n_{water} is a refractive index of water. In Figure 4.7, the texture of an object is directly mapped to the water surface and the texture coordinates, u' and v', are mapped to the vertex P of a triangle patch.

4.3.6 Blending of Textures

Reflection, caustics and object textures are blended by

$$\begin{aligned} T_{result}(u,v) = {} & T_{reflection}(u,v)T_{water}(u,v) \\ & + (1 - T_{reflection}(u,v)) \\ & \times \{T_{object}(u',v')(T_{caustics}(u',v') + I_{ambient})\}, \end{aligned}$$

where $T_{result}(u,v)$ is a final texture color at (u,v), $T_{reflection}$ is a reflection texture, $T_{water}(u,v)$ is a water original texture color $T_{object}(u',v')$ is an original texture color of the object at projected uv-coordinate of (u,v) and $T_{cautics}(u',v')$ is the intensity of the caustics texture. $I_{ambient}$ is an ambient intensity for all

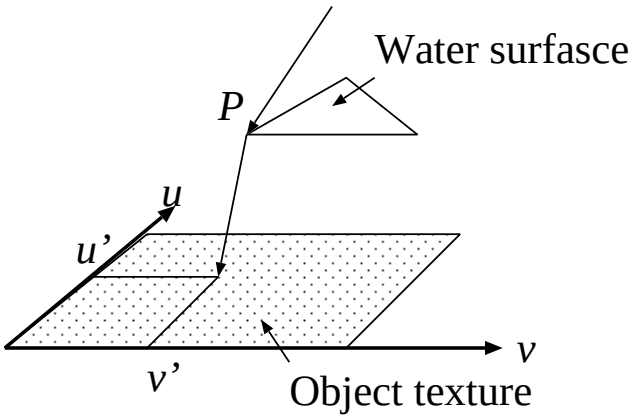

Figure 4.7: Projected mapping.

surfaces in a scene. The resulting texture is directly mapped onto water surface polygons and rendered directly, using graphic accelerated hard-ware.

4.4 Results

The system are executed on a PC which has *Athlon* 1.2G-Hz CPU, 512MB main-memory and *GeForce2 GTS* with 32MB video-memory. Figure 4.8 shows an example scene which water is falling into a small pool. The simulation grid resolution is 20x20x20 and the grid size $\Delta\tau$ is $10cm$. Initial water velocity is 1.4 m/s and the inflow shape is a circle with a radius of $40cm$. The time step Δt range is from $0.0003s$ to $0.01s$. The average rendering time is $27s/f$ and the simulation time is $11s/f$.

4.5 Conclusion and Future Work

We presented full solver of the *Navier-Stokes* equation with an adaptive time step method and fast rendering technique using graphical-hardware. The simulation and rendering method is examined and we can get even faster frame rates after tuning the techniques. The works under progress include the improving of caustics and refraction mapping.

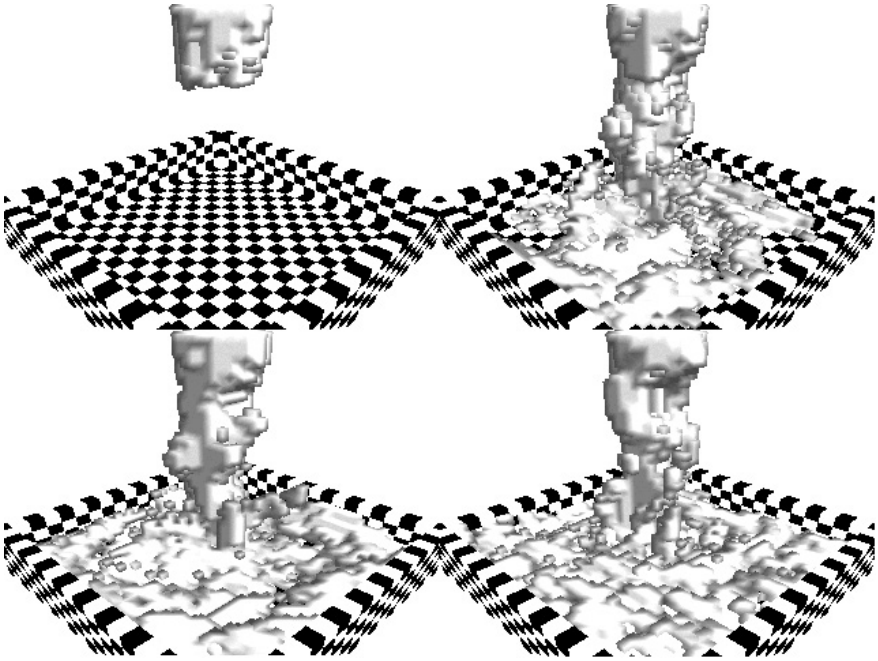

Figure 4.8: An example of resulting image

Chapter 5

Rendering: Physically-Based Model of Photographic Effects for Night and Day Scenes

So far presented PBM examples have dealt with modeling phenomena related to the shape of the object under consideration. Last two chapter will focus on PBM methods that that model the functions that are supposed to be calculated in the physical correct way. This chapter will define the PBM of a scattering and diffraction functions within the camera and human eye. Usually car drivers try not to look at the long lights of a car coming in opposite direction, however I had enjoyed the validation of our model on night drives.

During the past few years the computer graphics has focused mostly on photo-realistic rendering of the scenes. We have seen nice clean images but even with digital camera we can observe the light scattering within the camera system and the film emulsion or on the CCD chip. Similar effects can be observed with our human perception system due to the scattering in human eye.

In this chapter we propose the post-processing filter that is used to filter the high-dynamic range image with physically correct radiance values calculated with rendering software. All parameters in the filter have the physical meaning. The result is the scene with enhanced luminaries by corona and bloom effects.

5.1 Introduction

The limited range of CRTs prevents the display of luminaires at their actual luminance values. Taking into account the refraction, diffraction and specular distributions will create the blooming and coronas around the luminaires. The movie production often uses the special lenses to create effects around lights or explosions. First models used in digital image synthesis to add glare effects were proposed by Nakamae *et al.* [41] and improved by Rokita [56]. They have used the Eqs. 5.1,5.2 from next section as kernels in convolution image filtering. Vos [67] defined a point spread function that describes the redistribution of point

source energy onto the visual field of a human eye. The physical mechanisms and physiological causes of glare in human vision were studied by Spencer [63]. The above works focus mostly on human perception and do not account for visual masking effects of glare. Physical properties on camera image formation system was studied in astronomy for film calibration purposes. The first definition of analytical kernel and the effect of camera bloom on emulsion grain was investigated by astronomers [39].

Our approach has been to model the physical effects caused by the camera optical system and the film emulsion for computer animation purposes. We will derive the digital filters for bloom and corona camera effects using the known physical equations. Our focus is in digital simulation of optical filter or single camera stop effects for artists, architects and urban designers used to emphasize the light scenes of the virtual buildings. Since our rendered images consist of physically correct scalar luminance values we can reproduce the correct camera effects.

5.2 Camera physical effects

Let us consider a simplified optical formation system consisting of optical filter, lens and projection plane positioned along the z coordinate axis while projection plane is coincident with xy plane as shown in Fig. 5.1. Among the rays reaching the surface of the diaphragm only the rays passing through a diaphragm arrive at its focal plane or film plate.

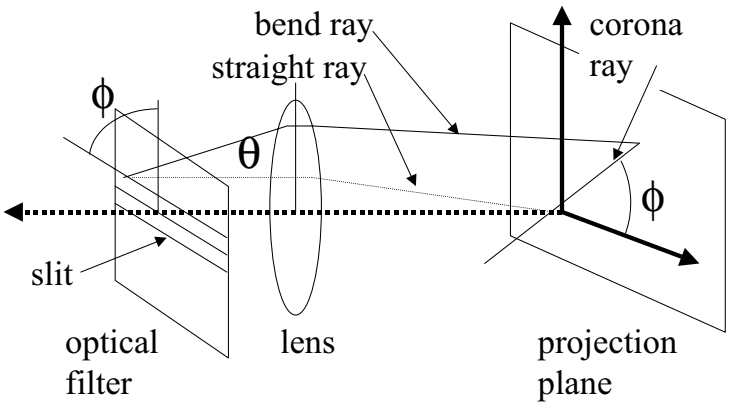

Figure 5.1: Camera image formation system.

5.2.1 Diffraction due to single diaphragm

All parallel beams passing through the lens are focused on a single point of the plane placed at the focal distance from the lens. A single small hole also called diaphragm placed before the lens creates the diffraction pattern at the projection plane which is the same pattern regardless of the location of the hole. If the diameter of hole is e, then it can be replaced by a slit with width e.

If rays with wavelength λ pass through the slit with width e , they deviate from the direct path about and angle θ. The intensity, I_s, in the diffraction pattern depends on angle θ as follows [8]:

$$I_s(\theta) = I_0 \frac{\sin^2 \alpha}{\alpha^2}, \tag{5.1}$$

where I_0 is the light intensity for $\theta = 0$, and

$$\alpha = \pi \frac{e}{\lambda} \sin \theta.$$

Note that the above equation describes the fact that the smaller is the slit width, the larger will be the diffraction pattern. The same is true for a hole.

5.2.2 Multiple holes of the same size

For many small holes distributed randomly, the diffraction pattern on the focal plane will be given by superposition of diffraction patterns coming from all particular holes. The problem of N regularly distributed holes with radius e and spacing d can be again replaced by a set of N parallel slits with width e and slit spacing d. The intensity, I_m, in the diffraction pattern given by multiple slits depends on deviation angle θ as follows:

$$I_m(\theta) = I_0 f \frac{\sin^2(e\alpha)}{(e\alpha)^2}, \tag{5.2}$$

where I_0 is the light intensity for $\theta = 0$, and

$$f = \frac{\sin^2(Nd\alpha)}{\sin^2(d\alpha)} \tag{5.3}$$

$$\alpha = \frac{\pi}{\lambda} \sin \theta. \tag{5.4}$$

Note, that if the slit spacing is irregular and the number of slits is big, then $f \approx N$, and the diffraction pattern is given by

$$I_m(\theta) = N I_s(\theta).$$

Therefore, the N holes of the same size and shape, will produce the diffraction pattern of a single hole amplified N times.

5.2.3 Diffraction on a slit network

The slit networks produce the corona diffraction pattern. Generally, the slit forming even sided polygon gives the star diffraction pattern with the same number of rays, unlike the slit forming odd sided polygon network which gives the star pattern with double number of rays as sides of polygon. Many camera filters available on the market produce corona patterns having the convergent rays with that same length. An example of a photo shot with rectangular slit network is shown in Figure 5.2 left.

5.2.4 Diffraction on camera stop

Camera stop is a hole with polygonal shape controlling the amount of light coming to the surface of the film. It is possible to produce the corona and bloom effect with a simple camera without any filter just by setting the stop and expose time. The real photograph of a scene shot during the day is shown in Figure 5.2 right. The Babinet's law claims that a hole of the same shape is always giving the same diffraction pattern. As a result we should see the same pattern for given camera settings. Number of rays in corona pattern produced by a hole obeys the similar rule as the slit network. For example, the triangular hole will always give a diffraction pattern the star with six rays, a square hole will give the star with four rays, ... [47]. The stop on camera objective used to shoot the photo had seven sided diaphragm, which explains why we see 14 rays in corona. Investigating the Fig. 5.2 right further one can see that the rays have the random length and are divergent.

Figure 5.2: Real photographs: left) shot with a filter having the triangular grid of scratches rightm) shot with an objective having the 7 sided stop.

5.2.5 Camera Bloom

This effect is attributed to the scattering of light in the optical system where the scatter contributions from lens and small particles within the film emulsion occur in roughly equal portions. Multiple scattering within the emulsion will occur from grain to grain until it is absorbed or leaves the emulsion. Figure 5.3 illustrates the situation when scattered light inside the camera is added to the light coming from object B. As result, the object B is blurred and its contrast decreases.

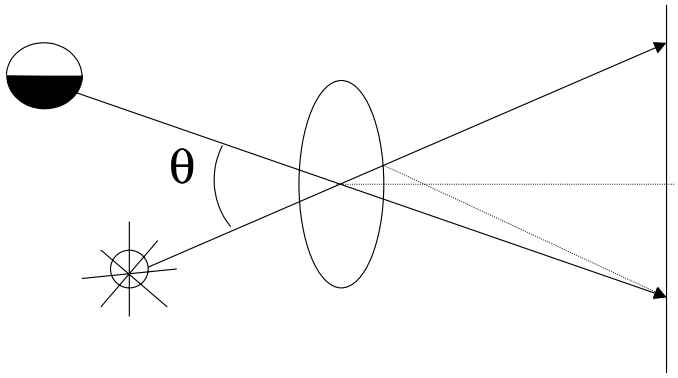

Figure 5.3: Camera bloom that results in reduction in contrast from scattered light.

5.3 Model of camera bloom and corona

This section will derive the equations that can be used to generate the digital image filters for bloom and corona camera effects. Derived filters based on known physical equations have parameters with intuitive physical meaning and can create wide range of camera effects.

Vos [67] defined a density function on the visual field that describes how a unit volume point source is "spread" onto other points of the visual field. The density function, $P(\theta)$, defined on the hemisphere of directions entering the camera is called the *point spread function* (PSF)and has the following form

$$P(\theta) = a\delta(\theta) + \frac{k}{f(\theta)},$$

where $\delta(\theta)$ is a delta function representing an ideal PSF with all energy in one point, a is the fraction of light that is not scattered, θ is the angle from the gaze direction, k is a constant between 3 and 50, and $f(\theta)$ is function of θ^n with $n \in (1.5, 3)$. Any PSF P is nonnegative and must satisfy the normalization condition on the hemisphere of directions entering the camera, where ϕ is the angle around the gaze direction and θ is the angle from he gaze direction measured in radians. Thus the function P conserves the energy in other words the energy is redistributed but not emitted or absorbed:

$$\int_0^{2\pi} \int_0^{\frac{\pi}{2}} P(\theta) \sin\theta d\theta d\phi = 1.$$

5.3.1 Alternative PSF definition

PSF is a nonnegative function defined in polar coordinates

$$P(r, \phi) = (1 - \varepsilon)\delta(r) + \varepsilon f(r, \phi), \tag{5.5}$$

satisfying the normalization condition of unique volume integral in polar coordinates:

$$\int_0^{2\pi}\int_0^{\infty} P(r,\phi)rdrd\phi = 1.$$

In above equations $\delta(r)$ is a delta function, r is a distance from the center of PSF located at the gaze direction, ϕ is the angle around the gaze direction in radians, ε defines a fraction of light that is spread to neighboring points of camera visual field, $f(r,\phi)$ is a function that determines the camera effect namely bloom or corona.

5.3.2 Adding the Bloom

It is now desirable to find a simple general analytical formula which quantitatively represents the observed bloom spread profile. Such a function $f(r,\phi) \equiv f_b(r,\phi)$ in PSF definition, Eq. 5.5, is

$$f_b(r,\phi) = \frac{\varepsilon}{(1+(r/R)^2)^{\beta}}, \tag{5.6}$$

where parameter $R \approx 30 - 120\mu m$ controls the filter width, and $\beta \approx 3-5$ is a constant. After substitution of Eq. 5.6 in to the PSF definition, Eq. 5.5 we derive the extension of Moffat kernel [39] which is obtained for $\varepsilon = 1$ by enhancing the energy for nearly orthogonal incident rays. For $\varepsilon = 0$ all light energy is concentrated in one single point and there is no bloom effect.

5.3.3 Adding the Corona

The corona effect that can simulate the randomness in the length, width and intensity fade out of corona rays is proposed to be $f(r,\phi) \equiv f_c(r,\phi)$ in Eq. 5.5 as follows:

$$f_c(r,\phi) = Cl(r)\cos(\frac{k\phi}{2})^{n(r)}, \tag{5.7}$$

where C is a normalization constant, k is the number of rays in visible corona. The fadeout effect of rays and their length in corona is suggested to be

$$l(r) = e^{-\sigma r/l},$$

where the mean ray length l is defined by the user and σ controls the intensity fadeout along the ray. To simulate more realistic effects namely random length of rays in corona, l can be considered as a statistical variable with a given mean and deviation. In our implementations the deviation is 25% of the mean, which produce the reasonable camera spot effects.

Let w be a ray width and s be a parameter defining the convergence and divergence angle of corona rays. Function $n(r)$ that determines whether the corona ray width converges or diverges is proposed as follows

$$n(r) = -\frac{1}{\ln(\cos\omega)},$$

where

$$\omega = (\frac{r}{w})^{s-1}.$$

The width of rays is constant for $s = 0$, alternatively it will converge and diverge for $s < 0$ and $s > 0$, respectively. Thin and long rays can be used to simulate the corona effects of optical filters while the long and divergent rays are good for simulation of corona effects produce by the camera stop.

5.3.4 Implementation

The above PSFs are applied as a postprocessing to the image $I(x, y)$ of luminance distributions in cd/m^2. The Cartesian coordinates (x, y) are the discrete image coordinates i.e. pixels. The modified luminance intensities $I'(x, y)$ of the image are then calculated by the standard discrete convolution method

$$I'(x, y) = \sum_{x_0, y_0} I(x_0, y_0) * K(x - x_0, y - y_0),$$

where $K(x, y)$ is the filter kernel derived from PDF by transforming it from polar coordinates to the Cartesian coordinate system:

$$K(x, y) = P(\sqrt{x^2 + y^2}, \tan^{-1}(\frac{y}{x})).$$

The normalization coefficient in the above PDFs can be calculated analytically or if it is not possible they can be approximated in discrete Cartesian space by using the normalization condition

$$\sum_{x,y} K(x, y) = 1.$$

The filters are independent of a particular image and their size is determined relative to the image width and height. Note that the filter is applied at each of bright points whose luminance is greater than the threshold value.

5.4 Results and Discussion

Since, it is possible to control the spreading fraction of energy independently from decay rate in Eq. 5.6, the PSF can have a uniform delta peak in the center with energy at large distance from center. The filter shape with parameters $\varepsilon = 0.005$, $\beta = 2$ and $R = 10.4$ pixels is demonstrated in the left of Figure 5.4. The bloom effect using the same parameters is shown in right of Figure 5.4. Central image in Figure 5.4 shows the scene with original luminaries.

Figure 5.4: Camera bloom: left) the filter profile center) the scene without any effects right) the scene luminaries enhanced by a bloom effect.

Figure 5.5 shows the simulation of camera effect on the stop with 7 sides resulting in 14 rays. On the left image of Figure 5.5 we show the corona filter profile using the parameter $\varepsilon = 0.005$, the ray length is set to $l = 15.6$, the number of rays is $k = 14$, and the divergence parameter is $s = 1$. Central figure shows the scene that was post-processed by the PSF using the same parameter set, the right figure shows the composition of both the corona and bloom effects.

Figure 5.5: Camera corona: left) the filter profile center) the scene with only corona effect applied on luminaries right) the scene luminaries are enhanced by both corona and bloom effects.

Different camera coronas can be produced by width and divergence parameters. On Figure 5.6 from top-left first image uses convergent rays with parameters $s = -1$, $w = 10$; next image simulates the effects produced by an optical filter using the constat width rays $s = 0$, $w = 3$; image in second row-left uses $s = 0.5$ $w = 3$ and divergent rays are shown on the last image for $s = 1$, $w = 3$. The length of ray used was approximately $l = 80$ pixels and the random factor was omitted in this figure.

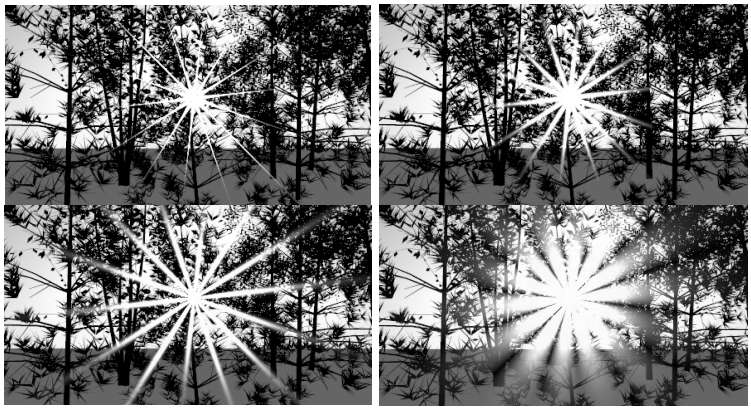

Figure 5.6: Camera corona parameters.

5.5 Conclusion

We have presented the mechanisms generally accepted by the film and camera community that are responsible for a corona and bloom effects in the camera image formation system. We have focused namely on scattering in the optical filter, lens, stop and film emulsion.

We have proposed the point spread function that can be converted into the digital filter to add the bloom effect to the image simulating the scattering within the film emulsion. Similarly the PSF simulating the both the divergent and convergent effects of corona was proposed. The performed experiments indicates that the camera effects improve the image perception and can be calculated in a reasonable time of up to 1 minute. The scalar luminance images were used in our examples.

Chapter 6

Rendering: Japanese Lacquer-Ware

Last chapter devoted to PBM in rendering will focus on PBM methods that model the BRDF functions particularly the full measured BRDF. Measured BRDF is a huge data table that can not be directly used for rendering in real time. Some compact way of storing the values is need. This chapter will define the PBM of a rendering equation for our material representation.

We choose the Japanese lacquer ware because it has rich visual effects. There are several techniques used to produce the lacquer ware, we chosen the most popular ones. The optical effects that can be rendered with the proposed approach include flip-flop effect, Fresnel reflection, sparkling, and the depth effect.

The first two effects can be captured by the bidirectional reflectance distribution function (BRDF) by using the BRDF measurement device. Therefore, the problem is simplified to the real-time visualization of BRDF. The proposed method calculates the sphere map and then projects it on the surface to be rendered. The techniques to calculate the sphere map in real time are novel. The flip-flop effect is modelled as mixture of several BRDFs according to the masking texture. The challenge solved in this chapter is to decompose the high dynamic range photograph into the masking channels according to the measured BRDFs for known pigments in the paint.

The last effects can be observed when the metallic pigments have larger size. We have successfully modelled those effects by modelling the real geometry of the sparkles and showed that the effects can also be sufficiently approximated by the random dot patterns.

The methodology we propose here gained a big success in Japan and is being used by a commercial company to render the human skin in real time. Our system has been demonstrated publicly on Japanese NHK TV and on several business meetings.

Japanese Lacquer Ware]Rendering of Japanese Artcraft

We present several methods for simulation of Japanese lacquer ware, a prominent Far East Asian handicraft art. We consider two most popular kinds of Japanese lacquer ware made by the makie and nashiji techniques. For rendering makie, we propose a method for preparing RGBA textures from digital photos of art items. The alpha channels of these textures control the weight with which

color channels are blended with the measured biderectional reflectance distribution function (BRDF) of a metallic finish. Both ray tracing and hardware based rendering are demonstrated. In the latter case, we show how the calculation of a sphere map texture used for BRDF visualization can be accelerated using a special coordinate system for tabulated BRDF. The *depth effect* manifested by nashiji lacquer is simulated by the explicit modeling of metal platelets immersed in absorptive material.

6.1 Introduction

From olden times lacquer work such as those shown in Fig. 6.1, commonly called *urushi*, has been used so widely for interior decoration, table ware and for other purposes that it is almost inseparable from the daily life of the Japanese. But urushi art of a highly artistic quality is inaccessible to the people in general, because the valuable materials, such as *genuine urushi* (lacquer juice), gold and silver, together with the skilled workmanship and the time required to make it, make its price almost prohibitive to them. However, in recent years cheap urushi art for utility purposes is being manufactured in large quantities for the use of the people who have a liking for anything new and novel. Because of this trend the merits of fine urushi art are gradually losing recognition.

Attracted by expressive beauty and richness of visual effects, which can be obtained using the old paints and techniques, we attempt to visualize the realistic appearance of urushi at interactive speeds. Realistic rendering of objects with complex optical properties, which change appearance with viewing and illumination directions, becomes of primary importance at early design stage and in electronic commerce. The specific properties of Makie technique include *flip-flop* visual color variation depending on viewing and illuminating directions while the properties of Nashiji technique include *depth* and *sparkling* effects. Very nice objects painted with very fine techniques consisting of mixtures of precious metals can be seen only in national museums. Since the production of such items took several years they can hardly be reproduced again. The other application of this research could be the computer-aided preservation of cultural heritage in digital form [2].

Figure 6.1: Digital photographs of a jewelry box. Left: painted with makie urushi technique and Right: painted with nashiji urushi technique.

The computer-aided preservation of cultural heritage is one of the recently

popular topics in computer graphics, geometric modeling, and virtual reality communities. Specifically, G. Pasko et al. [49, 46] made a great project called 'Virtual Shikki,' which is devoted to preservation of Japanese lacquer ware. Their work concentrates mostly on shape modeling using the implicit functions. Other related works to digital preservation of shape and texture of existing three-dimensional objects using the measurements and 3D scanning are the Digital Michelangelo project [17] and the Florentine Pieta project [1].

There have been many papers [12, 24, 60] related to the modeling of metallic and pearlescent paints in CG literature, which make it possible to simulate all the optical effects observed on urushi paints. Unfortunately, those methods require huge number of rays to obtain the good approximation of material radiance.

An approach for rendering the pearlescent and metallic appearance was proposed by S. Ershov et al. [16] where the BRDF is designed based on decomposing the paint layer into stack of sub-layers. Their method use the statistical approach for calculation of light scattering within the paint. However, their method does not focus on rendering itself.

The multi-image rendering algorithms such as light fields [32] and Lumigraphs [25] can capture the light distribution within the bounded region of 3-D space. The price paid for calculated light field at any point is the assumption of constant illumination and computationally expensive preliminary processing of input data, which cannot be computed on the fly.

Because we are concerned with applications related to electronic commerce and preservation of cultural heritage, such as web based trade and virtual museums, we focus on a hardware based rendering, in which visualization of metallic finishes in Japanese lacquer ware is done using a sphere map approach.

The sphere environment map was originally developed by Blinn and Newell [6] to interactively show specular reflection of distant environment. The idea was later elaborated by generalizing the BRDF [10, 26, 37, 51].

Debevec [13] combined captured environment maps with synthetic objects to produce the renderings with both synthetic objects and image based environments. Unfortunately, this technique does not work at interactive speeds.

This drawback was overcome with hardware acceleration and image based rendering described by B. Cabral et al. [11]. The authors used a sphere map which is view dependent representation. To avoid recalculation of the sphere map, the authors generate multiple sphere maps for different orthographic cameras. Afterwards, during the walk-through the sphere maps are interpolated using image based rendering (IBR) to ensure real time rendering. However, generation of reference sphere maps takes several tens minutes.

Our approach to real time rendering of objects with complex appearance, like Japanese lacquer ware items, includes online fast calculation of sphere maps. This widens the spectrum of possible applications to online ordering shape and design of lacquer ware items.

Although, the above mentioned rendering methods can handle many optical effects that occur in urushi paints, they will fail to correctly visualize the depth effects caused by small metallic flakes dipped in lacquer. We describe the explicit modeling method which simulates the sparkling appearance of nashiji.

The rest of the paper is organized as follows. Section 6.2 recalls the attainments on the urushi coating and decoration, in particular the makie and nashiji surface decoration techniques. Here we also summarize the most domi-

nant optical effects that can be observed on the most urushi items. The BRDF representation using the adaptive grid and the fast calculation of the sphere map applicable to real time rendering of virtual urushi art items is described in Section 6.3. In Section 6.4 we propose the fast rendering technique of makie art drawings which can clearly demonstrate the metallic color shifts (flip-flop) effects. Two rendering methods of nashiji art techniques which can handle the *depth* and *sparkilng* effects are derived in Section 6.5. Finally, we presents some results obtained using our approach and we conclude this paper.

6.2 The Urushi Coating and Decoration

The clear urushi is prepared from the sap of the lacquer tree by cutting the bark of tree. The cleared urushi from impurities is used for coating and decoration of the urushi art items mostly made of wood. The urushi art items are produced in three steps:

1. Preliminary coating where the rough wooden surface will be smoothed by applying of several layers of mostly dark or red urushi. At this step the substrate color is determined.
2. Finishing the surface to a smooth and glassy finish.
3. Decorative process, where the designs are drawn on a urushi ground by sprinkling the gold or silver powder over sticky urushi.

6.2.1 Makie: Sprinkling

Makie is the most famous surface decoration in urushi art technique. Patterns are painted with clear urushi or the red urushi and then the fine silver, gold and other metallic powders are sifted and adhered over the wet pattern to decorate the surface. The powders are sprinkled by means of bamboo or horn tubes covered with silk screen, as well as with a brush dusting.

There are many kinds of makie techniques differing mainly in what kind of urushi is used for pattern drawing and what kinds of powders are used.

6.2.2 Nashiji: Pear Skin Finish

A type of ground decoration, so termed because it recalls the speckled skin of the Japanese pear. It is created by lacquering the surface and while it is still wet the fine gold or silver powders or an alloy of gold and small amount of silver with a faint bluish green tinge are sprinkled over it. Subsequently after this dries, a coat of a thinner urushi is applied to fix the metal sparkles. The process is repeated to create multiple layers. Each layer has to be polished to a smooth glossy finish in such a way that the gold or silver sparkles remain still under the surface. Finally, the top varnish is made by repeatedly coated surface with transparent and yellowis lacquer.

6.2.3 Optical Effects of Makie

- **Flip-flop:** Makie urushi technique is a coating with complex optical behavior which includes *flip-flop* visual color variation depending on view-

ing and illuminating directions. The color variation is usually the smooth visual variation between two colors. Optical behavior of such paints is mostly described by a bi-directional reflectance distribution function.

Other effects observed include the Fresnel reflections on solid paint.

6.2.4 Optical Effects of Nashiji

- **Sparkling:** An impressive phenomenon that can be observed in Nashiji lacquer ware is the *sparkling* effect caused by large metallic flakes. Under direct illumination the flakes become visible as tiny shining mirrors [35]. When observed from far distance, the sparkles get blurred due to the finite resolution of human eye and one can observe a texture with irregular random fluctuations of brightness.

- **Depth:** As a result of sparkling the observer can get an impression of a very thick paint with the gold sparkles laying very deeply in it. The thickness impression can range from 3 $\sim$ 5mm but the actual paint thickness could be less then 1mm as in Fig. 6.2.

Figure 6.2: An example of nashiji urushi item in which the depth effect is perceived.

6.3 BRDF Visualization

The focus of this paper is on real time visualization of real items in virtual world showing the complex optical effects during the walk-around the artistic item. The real-time BRDF visualization is needed for this purpose. This problem is solved by proposed coordinate system for BRDF representation adjusted to a method of fast calculation of a sphere map as described bellow.

6.3.1 BRDF Representation

The BRDF of a metallic urushi surface is directional diffuse; that is, such a BRDF exhibits fairly sharp change near the specular direction. We have measured BRDFs using the setup described by Letunov et al. [31].

The tabular representation of BRDF using the spherical coordinate system (ψ, ξ) with polar axis along the surface normal and a uniform grid of points does not provide good accuracy in a specular peak area, unless the grid density is

very high, up to hundred thousand of tabular entries. The BRDF is represented in this system as $f(\psi_i, \xi_i, \psi_o, \xi_o)$, where

$$\begin{aligned} \overrightarrow{n} &= \text{Surface normal,} \\ \overrightarrow{\omega_i} = (\psi_i, \xi_i) &\quad \text{is incident direction relative to } \overrightarrow{n}, \\ \overrightarrow{\omega_o} = (\psi_o, \xi_o) &\quad \text{is outgoing direction relative to } \overrightarrow{n}. \end{aligned}$$

The similar problems related to BRDF representation were considered by Rusinkiewicz [57]. He uses the parameterizing the BRDF in terms of the halfway vector between the incoming and outgoing rays and a *difference* vector. We will consider the BRDF in terms of the specular direction and a local vector.

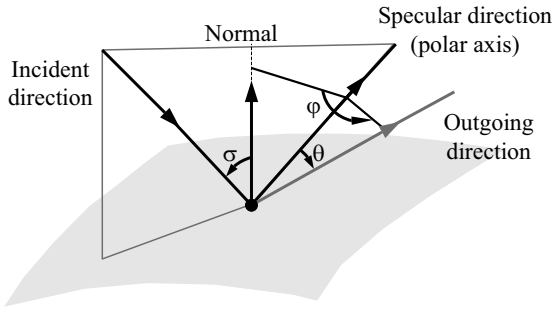

Figure 6.3: BRDF representation. A special coordinate system for BRDF representation.

We can decrease the size of the discretized BRDF stored in a table to several hundred entries. Each value in a table is associated with the discrete coordinates (θ, ϕ), of a rotated spherical coordinate system with the polar axis along the specular direction, as shown in Figure 6.3. The BRDF is represented in this system as $f(\theta_i, \phi_i, \theta_o, \phi_o)$, where

$$\begin{aligned} \overrightarrow{s} &= \text{Specular direction,} \\ \overrightarrow{\omega_i} = (\theta_i, \phi_i) &\quad \text{is incident direction relative to } \overrightarrow{s}, \\ \overrightarrow{\omega_o} = (\theta_o, \phi_o) &\quad \text{is outgoing direction relative to } \overrightarrow{s}, \\ \theta_o(\psi, \xi) &= \text{Angle between } \overrightarrow{\omega_o} \text{ and } \overrightarrow{s} \text{ directions,} \\ \phi_o(\psi, \xi) &= \text{Angle around } \overrightarrow{s}. \end{aligned}$$

As a result of this parameterization a sharp peak of diffuse BRDF will be in predefined area of BRDF domain around the polar axis in a new coordinate system. We can then discretize the BRDF and adjust the grid density making it more dense for small values of θ, in other words grid will be dense near the specular peak and sparse in the areas far from polar axis.

This method provides a high and almost uniform accuracy of approximation and does not need as many grid points as it would be necessary for an uniform grid.

6.3.2 Sphere Map Generation

The *sphere map*, shown in Fig. 6.4, is an image resulting from an orthographic projection of a sphere whose surface BRDF matches that of the target object.

The sphere is rendered for the same illumination and observation conditions as for target object.

Next we restrict ourselves to the case of several directional light sources and distant point light sources. Therefore, calculating the lighting equation for the sphere radiance map [11] amounts to the classic ray tracing of the sphere.

To render a sphere map, we select a pixel from sphere map, shoot a ray from the camera through this pixel, and find where it hits the sphere. At this point, we fire rays to all light sources and calculate the radiance from the sphere BRDF, the local normal, the viewing and illumination directions as

$$L(\overrightarrow{\omega_o}) = \sum_k f(\overrightarrow{\omega_{ik}}, \overrightarrow{\omega_o}) I_k(\overrightarrow{\omega_{ik}}),$$

where L is the radiance in the $\overrightarrow{\omega_o}$ outgoing direction, f is the BRDF, I_k is the incident radiance of the sphere at the point hit by the ray, coming from the k-th light source, $\overrightarrow{\omega_{ik}}$ is the direction to the k-th light source.

6.3.3 Fast Re-calculation

If we could create a good mesh on the sphere and shade only few vertices of the mesh and then interpolate all other points, the sphere map generation could be extremely fast. Furthermore, the color interpolation between mesh points can be done in hardware with Gouraud shading.

Below we will focus on the problem of construction of a good mesh for a single light source. To obtain the sphere map for several light sources, we just repeat the whole process for each light source and superimpose the sphere maps using the blending operation.

Looking at a "typical" sphere map image, shown in left of Fig. 6.4, for a single light source we see that a fixed polar or rectangular mesh will not be optimal, because, like for BRDF, there is a small area with large intensity gradients where the mesh must be fine and a large area with small gradients where we can use a coarse-grained mesh.

An optimal mesh should follow the BRDF changes. The BRDFs for metallic paints have a very strong dependence on angle θ in our BRDF representation, refer to Figure 6.3. Therefore, the best choice is the mesh with parametric lines along which the angle θ is constant, i.e. $\theta(\psi, \xi) = C_\theta$. The second family of parametric curves is naturally drawn from the highlight center along the constant angle ϕ, i.e. $\phi(\psi, \xi) = C_\phi$. The points (ψ, ξ) refer to the spherical coordinates relative to the $\overrightarrow{n}$ on the rendered sphere. Unfortunately, such mesh is not uniform along the ellipsoidal curves for constant θ. The mesh is improved by discrete arc-length reparameterization of parametric curves $\theta(\psi, \xi) = C_\theta$.

The resulting mesh derived for a measured BRDF of a gold metallic paint is shown in Figure 6.4 on right. If the number of light sources is no larger than ten, the time needed for calculation of a sphere map with our approach is a few tens of milliseconds for Pentium III machines with contemporary video cards.

6.4 Makie Simulation

This section describes preprocessing of the lacquer digital image to obtain information about distribution of colored urushi, metallic finish, and hardware

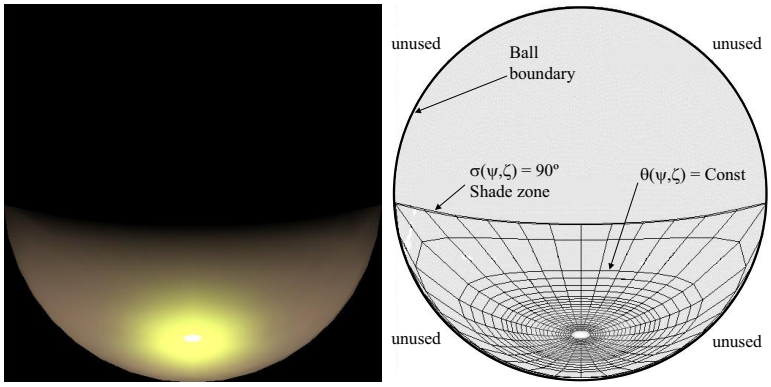

Figure 6.4: Sphere map. Left: Sphere map of a gold metallic paint. Right: Mesh for fast calculation of a sphere map.

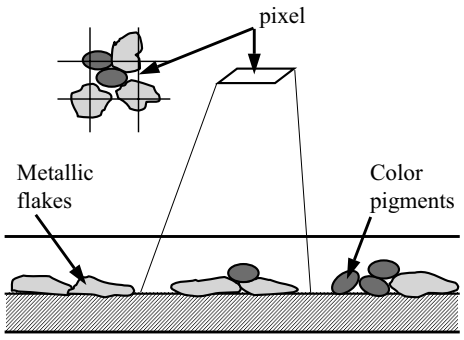

Figure 6.5: Cross section of the makie urushi showing pixel covering the sample.

based rendering of the makie lacquer ware.

6.4.1 Acquiring Optical Information

The surface of a makie item is usually painted with one, two or three types of colored urushi and one type of metallic finish. We prepare samples for each of the types of colored urushi used in the item under study. In particular, we make a metallic finish sample where platelets are sprinkled in such a density that no free space between them remains. We make photos of these samples under the same illumination conditions as that of the item. Because illumination should not vary significantly over the place where the art item or samples are put, it is best to use daylight illumination. We then extract radiometric information in the form of a high dynamic range (HDR) image from photos of the item and samples using Debevec's approach at http://www.debevec.org/Research/HDR/. If there is no too bright or too dark pixels on the image, then simple gamma transformation is enough. This is because CCD matrices are highly linear, so nonlinearity is added by the camera circuitry at the output, and this nonlinearity is only often gamma correction - at least, in some range of medium luminances. Further calculations are done either in XYZ or any RGB space, such as that of the camera after gamma transformation, obtained from CIE XYZ by a linear transformation.

Colored urushi paints are not intermixed on makie; different types of it are applied on different places. The color of each colored urushi varies over its patch only slightly because the application conditions such as, the layer thickness vary only slightly. On the other hand, the metal component of the finish consists of small thin (micrometers) platelets glued to the surface of colored urushi. In some types of urushi, pigment particles are also glued on the colored urushi after metal particles have been sprinkled on it. Figure 6.5, shows how sprinkled metal particles of makie are covered by a pixel of a CCD matrix in a digital camera. As a result, the colors of metallic and pigmented finishes mix additively rather than subtractively. This means that if metallic platelets occupy some fraction p of the colored urushi surface, then the BRDF f is a weighted sum of the BRDF of the metallic finish $f_{metallic}$ and those of colored urushi:

$$f = pf_{metallic} + \sum_l q_l f_l,$$

with p, q_l being positive scalar weights, f_l is the BRDF of the l-th colored urushi, and q_l is the fraction of the surface occupied by it. More general approach was considered by Lensch [58] were they used the image-based measuring method for BRDFs. We will consider $f_{metallic}$ as given from measurements and all other basis elements and weight must be estimated.

We assume that the BRDF of small patches of metallic finish, including individual platelets, is the same as that of a large patch consisting of platelets glued to the lacquer surface without gaps. This assumption is justified by the following facts:

1. Metallic platelets are glued practically in one layer.

2. They are well aligned with the laquer surface so light interreflections between neighboring platelets are negligible.

Similarly, we assume that the BRDF of a small patch covered with glued pigment particles used in a colored urushi is the same as that of a large patch. For colored urushi, this assumption is less justified, because two above facts are not true for them. But the BRDF of colored urushi is close to a Lambertian one plus the Fresnel reflection from the outer pigment-free layer, so differences between the BRDFs for small and large areas are not so important.

6.4.2 Selection the f_l Basis and Coefficients

Given the BRDFs, the radiance of a pixel in HDR image can be calculated as the BRDFs summed with lighting and integrated over the surface area covered by the pixel projection. Therefore, the radiance for three color channels is

$$\mathbf{T}(i,j) = p(i,j)\mathbf{m} + \sum_l q_l(i,j)\mathbf{c}_l, \tag{6.1}$$

where $\mathbf{m}$ and $\mathbf{c}_l$ are color triplets of a surface covered by only metallic finish and by only the l-th colored urushi, respectively. Therefore, positive weight are constrained by $p + \sum_l q_l = 1$ at pixels in drawing and $p + \sum_l q_l < 1$ at boundary pixels. Our goal is to find $p(i,j)$ and $q_l(i,j)$.

To this end, we change the basis of the three-dimensional color space used in such a way that $\mathbf{c}_l$ become basis vectors. At this point, we recall that most makie items have no more than three types of colored urushi. If we assume exactly three, then a new basis has three vectors collected in a matrix

$$A = (\mathbf{c}_1\mathbf{c}_2\mathbf{c}_3).$$

At this point, we assume that the HDR color of metallic finish $\mathbf{m}$ is contained in the convex hull of basis vectors $\mathbf{c}_l$, $l = 1, 2, 3$. Usually, this is so because $\mathbf{c}_l$ are either some red, green and blue colors or some red, green and white colors, while $\mathbf{m}$ is goldish.

Therefore, by multiplying the Eq. 6.1 from left side with inverse matrix A^{-1} we have the equation in new basis

$$\mathbf{T}'(i,j) = p(i,j)\mathbf{m}' + \sum_{l=1}^{3} q_l(i,j)\mathbf{e}_l, \tag{6.2}$$

where

$$\begin{aligned} \mathbf{T}' &= (T'_1, T'_2, T'_3) = A^{-1}\mathbf{T}, \\ \mathbf{m}' &= (m'_1, m'_2, m'_3) = A^{-1}\mathbf{m}, \end{aligned} \tag{6.3}$$

and $\mathbf{m}'$ is the triplet color of the metallic finish with respect to the new basis. The coordinate vectors $\mathbf{e}_l$ are the unit vectors $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$, and $\mathbf{e}_3 = (0,0,1)$.

Now we recall that colored urushi patches are not intermixed. Therefore, if we knew the fraction of metallic finish for each pixel p exactly, we would find that, in the limit of infinitesimally small pixels, only one of q_l should be nonzero for any pixel. In reality, because the color of colored urushi $\mathbf{c}_l$ is not absolutely

constant and two or three color patches may meet at a pixel, more than one q_l are nonzero. But, for most pixels, we have the largest q_l for the colored urushi that is actually present in the area covered by the pixel and other weights are small or zero.

Noting that q_l are positive, we can approximately find $p(i,j)$ as

$$p(i,j) = min\{T_1'(i,j)/m_1', T_2'(i,j)/m_2', T_3'(i,j)/m_3'\}. \tag{6.4}$$

The obtained $p(i,j)$ values are actually the alpha values of the generated texture shown on top-right of Figure 6.6. The second term of Eq. 6.2 after linear transformation to the monitor RGB is the RGB residual texture shown on bottom of Figure 6.6. The alpha component and the RGB image form the final RGBA texture.

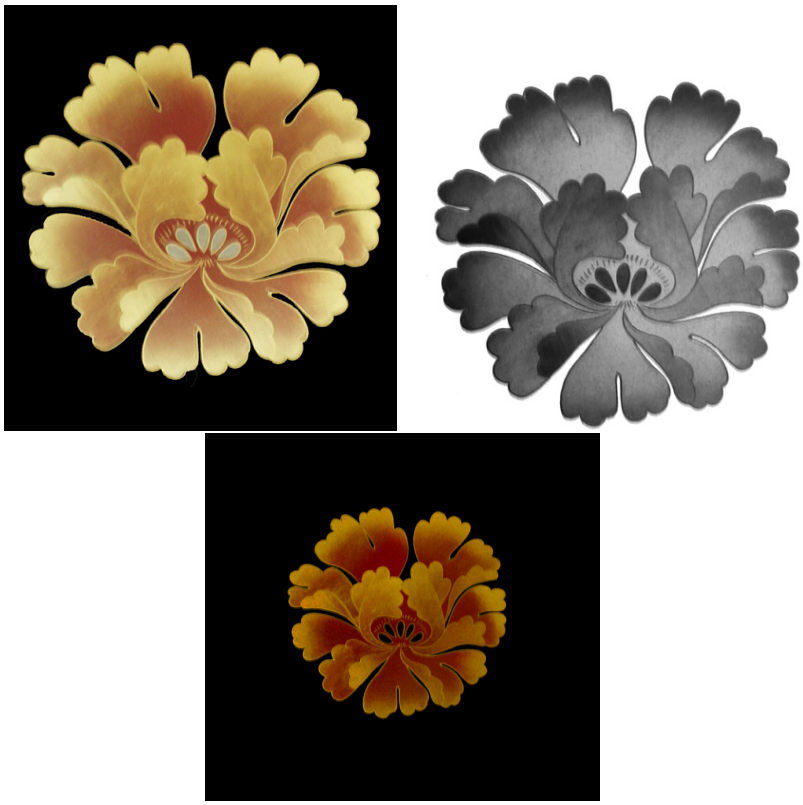

Figure 6.6: Texture decomposition. Left: Digital photo of a pattern. Right: Alpha channel used for weight of metallic BRDF. Bottom: Pattern without the metallic components.

6.4.3 Rendering Makie and Colored Urushi

Our approach employs a texture for rendering the Lambertian reflectance of colored urushi (see bottom of Fig. 6.6) and sphere map texture (see Section 6.3) for visualization of measured BRDF of metallic (gold or silver) finish. For controlling the ratio between two types of reflectances, we use the alpha channel

of the texture calculated from Eq. 6.4. Thus, in our implementation we render objects of arbitrary geometry using multitexturing method, one texture mapping uses the metallic sphere map and the other one is ordinary texture mapping with residual texture. The two texture images are then blended together according to the alpha channel.

The artistic drawing should be captured from flat areas for correct mapping to more complex geometries. The rendering runs in real time with different geometries.

6.5 Nashiji Depth Effect Simulation

There are many mechanisms responsible for perception of the depth effect produced by sparkles. One such mechanism called rivalry is the result of competition between images perceived by the left and right eyes. This mechanism can be demonstrated when one eye look at black and the other eye on white circle. The result of perception is gray and later it will continually change between white and black circles.

The similar mechanism can be present when looking at sparkles. The same flake can be seen bright by one eye and dark by the other one. As a result of rivalry perception mechanism we will perceive the depth effect.

We can simulate this effect by explicit modeling of the geometry of sparkles distributed along the geometry of the shape. The sparkles are defined as implicit superquadric functions [5]. The reason why we use the implicit functions is that they need less number of parameters to be defined thus saving the memory. For the simplicity, we assume that all sparkles have the same given shape. Their distribution, density and orientation is random according to the the user defined distribution functions. All flakes have the metallic finish and are immersed in the absorption media given by absorption coefficients for each RGB color channel.

Visualization of such complex scene is done by raytracing method yielding a stereo pair of images as shown in Fig. 6.7. The stereo visualization enables to simulate the rivalry perception mechanism. The explicit modeling of geometry sparkles has the advantage that we can freely zoom in and out while the sparkling patterns will change smoothly during the walk-around animations.

Figure 6.7: Stereo pair showing the sparkling depth effect.

6.6 Results

The proposed method for Makie visualization have been implemented as our real-time visualization system using Java3D. Users can observe the color shifts by rotating the object in real time as can be seen in Figure 6.8.

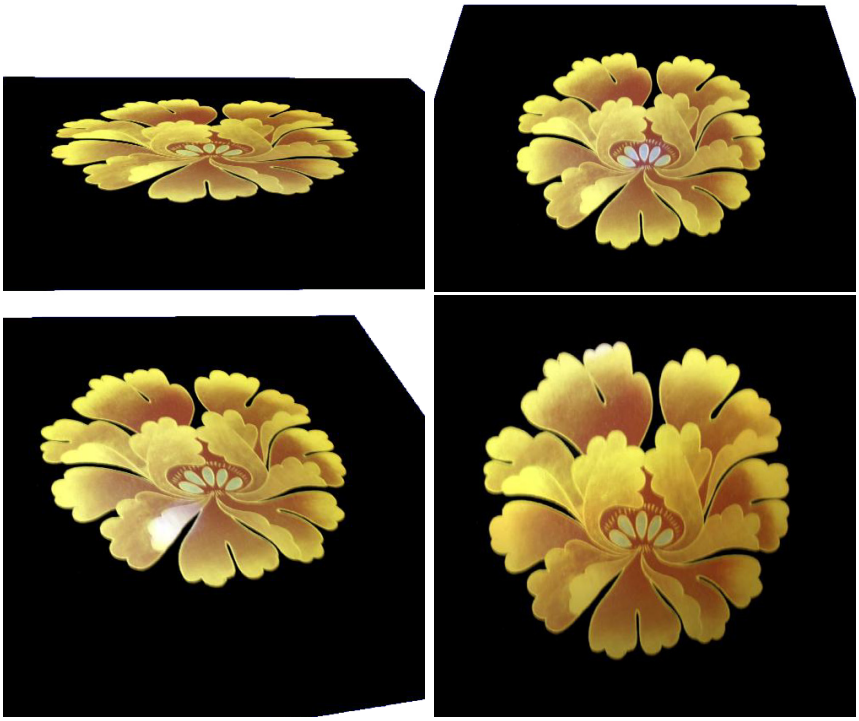

Figure 6.8: Makie rendering at different view angles.

Figure 6.9* shows a frame from an animation of a Japanese cup on tray showing the color changes of simulated makie technique the flip-flop effect.

For nashiji simulation, we have implemented the modeling tool for distribution of explicitly defined flakes over a parametric shape. Unfortunately, the raytracing of sparkles is computationally a heavy task, we leave the improvement of this method as the future research.

6.7 Conclusions

We have described new methods with focus on real time visualization of real artistic items in virtual world showing the complex optical effects. We consider two most popular kinds of Japanese lacquer ware made by the makie and nashiji techniques. We described interesting optical effects that can be observed in both techniques, which can be simulated by the proposed methods.

The real-time BRDF visualization method based on color interpolation between the vertices of an adaptive mesh on sphere map image was developed.

The depth and sparkling affects observed mostly on nashiji urushi technique were visualized by proposed explicit modeling of metallic flakes, which enable the user to smoothly zoom in and out the object surface and observe the depth effect in an image stereo pair. However, we left as future work the extension of our explicit sparkling technique to handle calculations in real time.

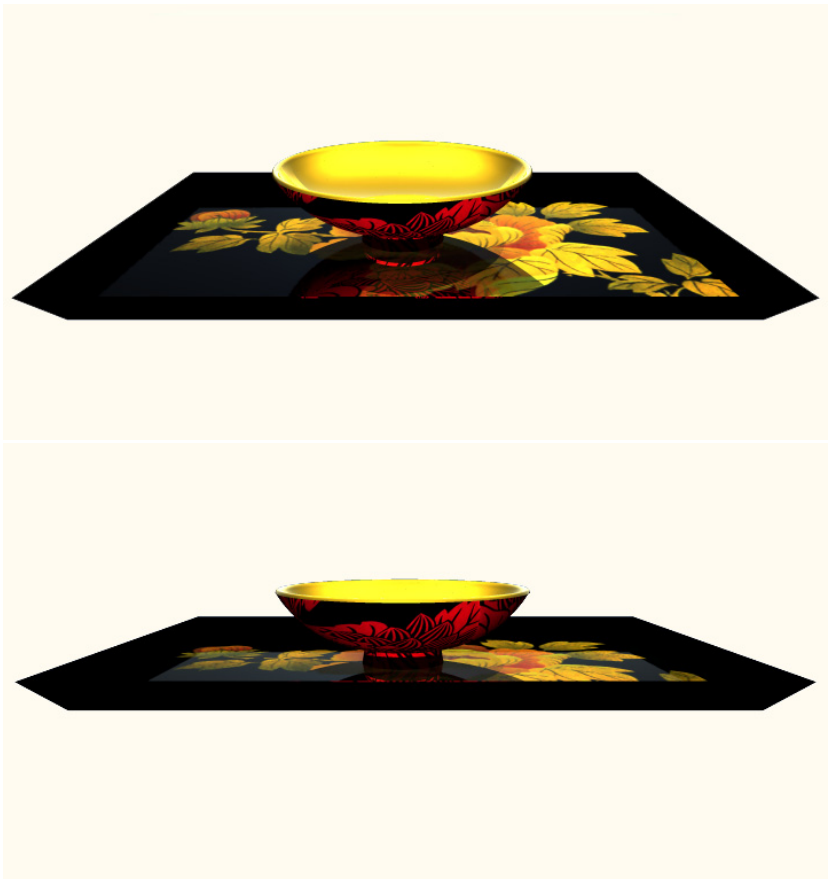

Figure 6.9: Animation sequence.

Chapter 7

Conclusions

In this habilitation we have presented our PBM framework and have demonstrated usefulness in shape modeling, motion modeling, and rendering. We have demonstrated the application of our framework to variety of problems in medical graphics, computer animation and rendering. The feature of our framework is that we have developed a theory upon which certain important aspects of the problems segmentation, 3D reconstruction, growth simulation, motion dynamics and representation of functions for rendering are treated in a unified way. Our research demonstrates success of PBM techniques in computer graphics community.

Bibliography

[1] Abouaf, J. The florentine pieta: Can visualization solve the 40year-old mystery? *IEEE Computer Graphics and Applications 19*, 1 (1999), 6–10.

[2] Addison, A., and Lous, Y. L. Emerging trends in virtual heritage. *IEEE Multimedia, Special Issue on Virtual Heritage 7*, 2 (2000), 22–25.

[3] Adzhiev, V., Cartwright, R., Fausett, E., Ossipov, A., Pasko, A., and Savchenko, V. Hyperfun project: A framework for collaborative multidimensional f-rep modeling. In *Implicit Surfaces '99, Eurographics/ACM SIGGRAPH Workshop, Bordeaux, France* (Sept. 1999), J. Hughes and C. Schlick, Eds., vol. 25, ACM SIGGRAPH.

[4] Aono, M., and Kunii, T. L. Botanical tree image generation. *IEEE Computer Graphics and Applications 4*, 5 (1984), 10–34.

[5] Barr, A. H. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications 1*, 1 (Jan. 1981), 11–23.

[6] Blinn, J., and Newell, M. Texture and reflection in computer generated images. *Communications of the ACM 19* (1976), 542–546.

[7] Bloomenthal, J. Modeling the mighty maple. *ACM Computer Graphics 19*, 3 (1985), 305–311.

[8] Born, M., and Wolf, E. *Principles of optics.* Pergamon Press, New York, 1953.

[9] Boys, C. *Soap bubbles: Their colors and forces which mold them.* Dover Publications, New York, 1959.

[10] Cabral, B., Max, N., and Springmeyer, R. Bidirectional reflection functions from surface bump maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., vol. 21, ACM SIGGRAPH, pp. 273–281.

[11] Cabralm, B., Olano, M., and Nemec, P. Reflection space image based rendering. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (1999), vol. 21, ACM SIGGRAPH, pp. 165–169.

[12] Dana, K., van Ginneken, B., Nayar, S., and Koenderink, J. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (1999), 1–34.

[13] Debevec, P. Rendering synthetic objects into real scenes: Bridging traditional and image based graphics with global illumination and high dynamic range photography. In *Computer Graphics (SIGGRAPH '98 Proceedings)* (1998), ACM SIGGRAPH, pp. 189–198.

[14] Douglas, J. Solution of the problem of platea. *Transactions on American Mathematic Society 33* (1931), 263–321.

[15] Edelstein-Keshet, L. *Mathematical models in biology.* Random House, New York, 1988.

[16] Ershov, S., Kolchin, K., and Myszkowski, K. Rendering pearlescent appearance based on paint-composition modelling. *Computer Graphics Forum 20*, 3 (Apr. 2001), C221–C238.

[17] et al., M. L. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of SIGGRAPH 2000* (Aug. 2000), ACM SIGGRAPH, pp. 131–144.

[18] Foster, N. *Modeling and animating fluid phenomena for computer graphics and special effects.* Ph.D. dissertation, University of Pennsylvania, Philadelphia, 1997.

[19] Foster, N., and Fedkiw, R. Practical animation of liquids. In *Proceedings of SIGGRAPH 2001, Computer Graphics, Annual Conference Series* (ACM SIGGRAPH, Aug. 2001), pp. 23–30.

[20] Foster, N., and Metaxas, D. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483.

[21] Foster, N., and Metaxas, D. Controlling fluid animation. In *In Proceedings of Computer Graphics International* (1997), pp. 178–188.

[22] Fracchia, F. D., Prusinkiewicz, P., and de Boer, M. J. M. Visualization of the development of multicellular structures. In *Proceedings of Graphics Interface '90* (May 1990), pp. 267–277.

[23] Glassner, A. Soap bubbles: Part 2. *IEEE Computer Graphics and Applications 20*, 6 (2000), 99–109.

[24] Gondek, J., Meyer, G., and Newman, J. Wavelength dependent reflectance funcion. *Computer Graphics (Proc. SIGGRAPH 1994)* (1994), 213–220.

[25] Gortler, S., Grzesczuk, R., Szeliski, R., and Cohen, M. The lumigraph. *Computer Graphics (Proc. SIGGRAPH 1996)* (1996), 43–54.

[26] Greene, N. Applications of world projections. In *In Proceedings of Graphics Interface '86* (1986), M. Green, Ed., pp. 108–1144.

[27] Hutchings, M., Morgan, F., Ritore, M., and Ros, A. Proof of he double bubble conjecture. *http://www.williams.edu/Mathematics/fmorgan/ann.html 23* (Mar. 2000).

[28] Icart, I., and Arques, D. An approach to geometrical and optical simulation of soap froth. *Computers & Graphics 23*, 3 (June 1999), 405–418. ISSN 0097-8493.

[29] Kincaid, D. R., Respass, J. R., and Young, D. M. Itpack 2c: A fortran package for solving large sparse linear systems by adaptive accelerated iterative methods. In *http://www.netlib.org/* (Netlib Repository, July 2000).

[30] Kunimatsu, A., Watanabe, Y., Fujii, H., Saito, T., Hiwada, K., Takahashi, T., and Ueki, H. Fast simulation and rendering techniques for fluid objects. *Computer Graphics Forum, (EUROGRAPHICS 2001) 20*, 3 (Sept. 2001), 57–66.

[31] Letunov, A., Barladian, B., Zueva, E., Vezhnevets, V., and Soldatov, S. Ccd-based device for brdf mesurements in computer graphics. In *IProceedings of Graphicon 99, August 26 - September 1* (Moscow, 1999), pp. 129–138.

[32] Levoy, M., and Hanrahan, P. Light field rendering. In *Proceedings of SIGGRAPH '96* (Aug. 1996), ACM SIGGRAPH, pp. 31–42.

[33] Liddell, C. M., and Hansen, D. Visualizing complex biological interactions in the soil ecosystem. *The Journal of Visualization and Computer Animation 4* (1993), 3–12.

[34] Lindenmayer, A. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology 18* (1968), 280–315.

[35] McCamy, S. Observation and measurement of the appearance of metallic materials. part i. macro appearance. *COLOR research and application 21* (1996), 292–304.

[36] McCormack, J., and Sherstyuk, A. Creating and rendering convolution surfaces. *Computer Graphics Forum 17*, 2 (1998), 113–120.

[37] Miller, G., and Hoffmans, C. *Illumination and reflection maps: Simulated objects in simulated and real environments*, vol. 26. ACm SIGGRAPHl, July 1984.

[38] Mirtich, B. Timewarp rigid body simulation. In *Proceedings of SIGGRAPH 2000, Computer Graphics, Annual Conference Series* (ACM SIGGRAPH, July 2000), pp. 193–200.

[39] Moffat, A. A theoretical investigation of focal stellar images in the photographic emulsion and applicaiton to photographic photometry. *Astronomy and Astrophysics 3* (1967), 455.

[40] Měch, R., and Prusinkiewicz, P. Visual models of plants interacting with their environment. In *Computer Graphics (SIGGRAPH '96 Proceedings), New Orleans, LA* (Aug. 1996), ACM SIGGRAPH, pp. 397–410.

[41] Nakamae, E., Kaneda, K., Okamoto, T., and Nishita, T. A lighting model aiming at the drive simulators. *Computer Graphics, Siggraph'90 24*, 3 (1990), 395–404.

[42] Noser, H., and Thalmann, D. The animation of autonomous actors based on production rules. In *Proceedings Computer Animation'96, Geneva, Switzerland* (Los Alamitos, California, June 1996), IEEE Computer Society Press, pp. 47–57.

[43] O'Brien, J., and Hodgins, J. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 1999, Computer Graphics, Annual Conference Series* (ACM SIGGRAPH, Aug. 1999), pp. 137–146.

[44] page, W. Pov-ray - the persistence of vision raytracer. In *http://www.povray.org/* (Jan. 2001).

[45] page, W. Hyperfun project. In *http://www.hyperfun.org/* (Jan. 2003).

[46] page, W. Virtual shikki web site. In *http://www.k.hosei.ac.jp/~pasko/Shikki/Shikki.html* (Jan. 2003).

[47] Parrent, G., and Thompson, B. *Physical optics notebook.* Society of Photo-optical Instrumentation Engineers, Billingham, MA, 1969.

[48] Pasko, A., Adzhiev, V., Sourin, A., and Savchenko, V. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8 (1995), 429–446.

[49] Pasko, G., Pasko, A., Vilbrandt, C., and Ikedo, T. Virtual shikki: Shape modeling in digital preservation of traditional japanese lacquer ware. In *IEEE Proceedings of the 17th Spring Conference on Computer Graphics - SCCG2001* (Budmerice, Slovakia, Apr. 2001), R. Durikovic and S. Czanner, Eds., pp. 147–154.

[50] Plateau, J. *Statique expèrimentale et thèoretique des liquides soumis aux seules forces moleculaires.* Gauthier Villars, Clemm, Belgium, 1839.

[51] Poulin, P., and Fournier, A. A model for anisotropic reflection. In *In Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '90 Proceedings)* (1990), F. Baskett, Ed., vol. 24, ACM SIGGRAPH, pp. 273–282.

[52] Prusinkiewicz, P., Hammel, M. S., and Mjolsness, E. Animation of plant development. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, ACM SIGGRAPH, pp. 351–360.

[53] Prusinkiewicz, P., James, M., and Měch, R. Synthetic topiary. In *Computer Graphics (SIGGRAPH '94 Proceedings)* (July 1994), A. S. Glassner, Ed., ACM SIGGRAPH, pp. 351–358.

[54] Prusinkiewicz, P., and Lindenmayer, A. *The algorithmic beauty of plants.* Springer-Verlag, New York, 1990.

[55] Rado, T. *On the problem of Plateau.* Chelsea Publications, 1951.

[56] Rokita, P. A model for rendering high intensity lights. *Computer and Graphics 17*, 4 (1993), 431–437.

[57] Rusinkiewicz, S. A new change of variables for efficient BRDF representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (1998), pp. 11–22.

[58] Rusinkiewicz, S. Image-based reconstruction of spatially varying materials. In *Proceedings of Eurographics Workshop on Rendering* (2001), pp. 95–103.

[59] Sadler, T. W. *Langman's medical embryology, 7th edition.* Williams & Wilkins, New York, 1995.

[60] Schramm, M., Gondek, J., and Meyer, G. Light scattering simulations using complex subsurface models. In *Proceedings of Graphics Interface 1997* (1997), pp. 56–67.

[61] Sherstyuk, A. Interactive shape design with convolution surface. In *Proceedings of Shape Modeling and Applications, Aizu-Wakamatsu, Japan* (Feb. 1999), IEEE CS Press, pp. 56–65.

[62] Shroder, P., and Zorin, D. Subdivison for modeling and animation. In *Cource Notes of SIGGRAPH 1998* (ACM SIGGRAPH, Aug. 1998), pp. 193–200.

[63] Spencer, G., Shirley, P., Zimmerman, K., and Greenberg, D. Physically-based glare effects for digital images. In *In Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '95 Proceedings)* (1995), ACM SIGGRAPH, pp. 325–334.

[64] Terzopoulos, D., and Fleischer, K. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (SIGGRAPH '88 Proceedings)* (held in Atlanta, Georgia; 1-5 August 1988, Aug. 1988), J. Dill, Ed., vol. 22, pp. 269–278.

[65] Tsingos, N., Bittar, E., and Gascuel, M. Implicit surfaces for semi-automatic medical organ reconstruction. In *Computer Graphics: Developments in Virtual Environments (Proceedings of Computer Graphics International '95), UK* (May 1995), Academic Press Ltd, pp. 3–15.

[66] Ďurikovič, R., Kaneda, K., and Yamashita, H. Animation of biological organ growth based on l-systems. *Computer Graphics Forum (EUROGRAPHICS'98) 17*, 3 (Aug. 1998), 1–13.

[67] Vos, J. Disability glare- a state of the art repor. *C.I.E. Jounal 3*, 2 (1984), 39–53.

[68] Wejchert, J., and Haumann, D. Animation aerodynamics. *Computer Graphics (Proc. SIGGRAPH) 4*, 25 (1991), 19–22.

[69] Witkin, A. M. An introduction to physically based modeling. In *Computer Graphics (SIGGRAPH'94 Tutorials)* (Orlando, Florida, 1994), ACM SIGGRAPH.

[70] Yngve, G., O'Brien, J., and Hodgins, J. Animating explosions. In *Proceedings of SIGGRAPH 2000, Computer Graphics, Annual Conference Series* (ACM SIGGRAPH, July 2000), pp. 29–36.