Computer Graphics Simulation of Organic and Inorganic Optical and Morphological Appearance Changes

Dhana Frerichs

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Engineering

Supervisor: Dr Christos Gatzidis (Bournemouth University),
Andrew Vidler (Ninja Theory Ltd.)

October 23, 2017

Abstract

Dhana Frerichs

Computer Graphics Simulation of Organic and Inorganic Optical and Morphological Appearance Changes

Organic bodies are subject to internal biological, chemical and physical processes as well as environmental interactions after death, which cause significant structural and optical changes. Simulating corpse decomposition and the environmental effects on its surface can help improve the realism of computer generated scenes and provide the impression of a living, dynamic environment. The aim of this doctorate thesis is to simulate post mortem processes of the human body and their visual effects on its appearance. The proposed method is divided into three processes; surface weathering due to environmental activities, livor mortis and natural mummification by desiccation. The decomposing body is modelled by a layered model consisting of a tetrahedral mesh representing the volume and a high resolution triangle surface mesh representing the skin.

A particle-based surface weathering approach is employed to add environmental effects. The particles transport substances that are deposited on the object's surface. A novel, biologically-inspired blood pooling simulation is used to recreate the physical processes of livor mortis and its visual effects on the corpse's appearance. For the mummification, a physically-based approach is used to simulate the moisture diffusion process inside the object and the resulting deformations of the volume and skin. In order to simulate the colouration changes associated with livor mortis and mummification, a chemically-based layered skin shader that considers time and spatially varying haemoglobin, oxygen and moisture contents is proposed.

The suggested approach is able to model changes in the internal structure and the surface appearance of the body that resemble the post mortem processes livor mortis, natural mummification by desiccation and surface weathering. The surface weathering approach is able to add blemishes, such as rust and moss, to an object's surface while avoiding inconsistencies in deposit sizes and discontinuities on texture seams. The livor mortis approach is able to model the pink colouration changes caused by blood pooling, pressure induced blanching effects, fixation of hypostasis and the purple discolouration due to oxygen loss in blood. The mummification method is able to reproduce volume shrinkage effects caused by moisture loss, skin wrinkling and skin darkening that are comparable to real mummies.

Contents

List of Figures

Acknowledgements

To Andrew Vidler and Christos Gatzidis,
For their supervision of my doctoral thesis.
Christos encouraged me to reach my potential,
His help and support was no less than essential.

Andrew forced me to face my fears,
And was a daily assistance for the last four years.
I know the whole business has been an ordeal,
Thanks for helping me out and for keeping it real.

Craig Powell needs thanking, of course.
He really was a constant resource,
Particularly for anything maths related,
Which without his help would have left me frustrated.

To Wil Driver and the Ninja team,
I hold you all in the highest esteem.
I'm grateful to all at the CDE,
And, for my funding, the EPSRC*.

Matthew Stoneham deserves a shout out,
His models and textures I've been using throughout.
And Robin Hansson, it needs to be said,
I'm terribly sorry for rotting your head.

To all of my family and those of my friends,
Who'll be awfully grateful when it finally ends.
I'd like to express my sincerest gratitude,
For putting up with my horrible attitude.

To Sam, of course, you know what you've done,
Even if sometimes I wasn't much fun.
For the last few years you've been there for me,
With second opinions and cups of tea.

*with grant code EP/G037736/1

Author's Declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes three original papers published in peer reviewed journals. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within Bournemouth University and Nina Theory Ltd. under the supervision of Dr Christos Gatzidis (Bournemouth University) and Andrew Vidler (Ninja Theory Ltd.).

The content of Chapter 2 was published in the form of a survey paper:
Frerichs, D., Vidler, A., and Gatzidis, C. (2015). A survey on object deformation and decomposition in computer graphics. *Computers & Graphics*, 52:18 – 32.

Parts of Chapter 4 were presented as a poster at SIGGRAPH Asia 2014:
Frerichs, D., Vidler, A., and Gatzidis, C. (2014). Object weathering simulation avoiding texture space stretching and discontinuities. In *SIGGRAPH Asia 2014 Posters*, SIGGRAPH ASIA '14, pages 37–37, New York, NY, USA. ACM.

The content of Chapter 5 was published as full length article:
Frerichs, D., Vidler, A., and Gatzidis, C. (2017). Biologically inspired simulation of livor mortis. *The Visual Computer*, 33(11):1453–1466.

Chapter 1

Introduction

The creation of realistic and believable virtual worlds with the help of computer generated models has become increasingly popular in the video game and film industry. In particular, in the games industry, one can observe an increasing trend towards realism, with more realistic graphics and physical behaviours. This is apparent in both the visual appearance of the content and the increased dynamics of the environment; such as changes in weather, day-night cycles and destructible environments. In a virtual world an object's realistic appearance is important. In order to make the world feel and appear alive however, the object should also change over time in accordance with its environment. Despite this, ageing effects such as rotting, have often been neglected.

This is particularly noticeable in the way human or other organic corpses are depicted in video game worlds. Bodies in the physical world are affected by a number of internal biological, chemical and physical processes after death, which considerably shape the corpse's appearance and internal structure of the body over time. Additional environmental activities, such as dirt deposits or moss growth, affect the body's surface appearance. However, in game environments, corpses show no signs of decay and tend to simply disappear from the world after a while. Corpses are meaningful elements in many games, both in terms of game-play and story line and as such realistic depictions of corpses can have a significant impact on perceived realism. Simulating the decomposition of a corpse adds realism to a scene by giving visual clues about the object's make-up, age and history. At the same time it helps to give the impression of a living, dynamic environment, which can be further enhanced by simulating the effects of environmental processes on the body's surface.

There is an ongoing debate about the psychological effects of violence in video

games, such as the potential to induce aggressive behaviour or to desensitise users to violence. It is an interesting question whether, and to what extent, the effects of violence in games on the user depend on the immersiveness of the virtual world and a realistic portrayal of death. For this it is essential to be able to model the realistic appearance of corpses.

Artists can create models and textures to replicate the different appearance changes of an object, but decomposition is a complex process. Reproducing decomposition by hand, especially when trying to preserve temporal coherence throughout the decomposition stages, is notoriously difficult. Furthermore, virtual worlds can have a vast number of different objects. Creating models for all the objects and all their possible appearances during the decay process would be very labour intensive and time consuming. Consequently, there is a need for automated generation of models of decomposing objects, which can be fine tuned by artists to obtain specific looks. This would help to reduce the work load for artists and allow them more time for creative work.

Object weathering and decomposition is an emerging area of research in computer graphics. These processes are often divided into chemical, mechanical and biological weathering, as seen in Dorsey et al. (2010). Chemical weathering includes corrosion, tarnishing, fading, combustion and phase changes. Mechanical weathering encompasses paint peeling, surface cracks, erosion and sediment deposits. Biological weathering describes the rotting and withering of organic objects, such as fruit, leaves and animal tissue. Although techniques to simulate changes in an object's appearance have become increasingly popular over the recent years, techniques to simulate organic decay are still mostly unexplored and research into this area is still ongoing. Corpse decomposition falls into the organic or biological category, although there are chemical and physical processes involved with this as well.

1.1 Research Objective

The aim of this research is to develop methods to recreate post-mortem appearance changes due to organic decomposition that are comparable to their real world equivalents. This involves an initial examination of the literature in the area of object deformation and decomposition, driven by external or internal processes, to establish the state of the art in organic decomposition and to identify unsolved problems. This will discern methods to recreate the appearance of corpses.

Human body decomposition is a complex degenerative process comprising of

a number of biological, chemical and physical activities. The occurrence of these activities depend greatly on the object's environment. Some can happen concomitantly while others are mutually exclusive. There is no single combination of processes that drive the decomposition and as such the appearance of corpses varies greatly. Considering all possible combinations and their effects on the body's appearance is too complex for the purpose of one doctoral thesis. Therefore this thesis will focus on a small selection of processes. These are livor mortis and natural mummification by dessication. Additional surface weathering appearances due to environmental activities, such as dirt and moss, are also considered. Other activities, such as putrefaction and autolisys, on the other hand, are not considered for this thesis.

In order to create realistic post-mortem appearances of organic bodies the internal biological and physical processes that drive the body decay need to be simulated. These are blood pooling, moisture diffusion and deformation dynamics. Additionally, the skin discolouration caused by chemical changes within the blood and skin composition, as well as the surface appearance changes caused by moss and dirt deposits, need to be generated. This requires a virtual model representing the physical constitution of the human body.

The human body is composed of different components and materials that react differently to the decay processes. To simulate livor mortis and mummification it is appropriate to differentiate between bones, flesh and skin. A layered object structure is used to represent the decaying body parts. A volumetric tetrahedral mesh is used to represent the internal body parts, such as flesh and bones. The differentiation between bone and flesh is achieved by adjusting nodal parameters to account for the physical differences between the two materials for the different processes. These are, for example, stiffness parameters, water content or blood composition. The skin is represented by a triangle surface mesh and high resolution texture maps.

The goal of the surface weathering is to add blemishes, such as dirt and moss, to the object's surface to provide the impression of the body being part of the environment, instead of an isolated object. In the case of corpse decay, dirt deposits and moss growths help to visually integrate the corpse into its environment. This is achieved using a method based on photon tracing, called *gamma-ton* tracing. *gamma-tons* are particles that carry materials, such as dirt, moss or humidity, and deposit these in the scene.

Livor mortis is one of the earliest signs of death and appears as pink and, later on, purple discolouration of the skin. The cause of these colouration changes is the blood pooling in the lower lying areas of the body due to gravity. Heavy red

blood cells sink down in the blood plasma and accumulate in the lower areas of the body. This causes pinkish discolouration in those areas and and makes paler the areas deprived of blood. Pinkish areas then turn purple and bluish as the oxygen leaves the blood. In order to achieve these discolourations the blood pooling dynamics and oxygen dissociation (oxygen loss) are simulated. The haemoglobin content and oxygen saturation values are then used when rendering the skin to achieve the changes in colouration.

Mummification by dessication is caused by water evaporation in dry environments. This leads to the shrinking of the body and the wrinkling of the skin. A physically-based approach is used to simulate the processes and appearance changes associated with natural mummification by desiccation. The Finite Element Method is employed to simulate the moisture diffusion and the resulting volume deformation. Position based dynamics are used to achieve the associated skin shrivelling and wrinkling effects.

The blood pooling simulation as well as the moisture diffusion are simulated on the tetrahedral volume mesh. The skin mesh acts as the outer skin layer, and is used for rendering. Deformations caused by desiccation are simulated on both the tetrahedral volume mesh and the skin mesh. The skin mesh is constrained to the volume mesh and, as such, the deformations of the volume mesh affect the geometry of the skin mesh. The constraints soften and may break with ongoing desiccation to simulate skin slippage.

The skin colouration changes are due to changes in the skin composition. In order to reproduce these the chemical changes that cause colouration changes, namely haemoglobin decay and oxygen decrease, need to be simulated. Only the triangle surface mesh is used for rendering. However, the skin is represented by an epidermis (outer) and dermis (inner) layer. The epidermis reflection is modelled using texture maps, with any influences of blood on skin colouration removed. For the dermis contribution to the skin colour, the boundary faces of a tetrahedralisation are used. The values on the tetrahedral boundary nodes are projected onto the surface triangle mesh and used as coordinates to look up the colouration based on blood components and water content. The two skin layers are then combined to get the final skin colour. A separable screen-space rendering technique is used to account for subsurface scattering within both the epidermis and dermis.

The surface weathering approach is an extension of the *gamma-ton* simulation described by Chen et al. (2005) and Günther et al. (2012), that aims to reduce the visual artefacts caused by texture seams, inconsistent depositing sizes and to give the user more control over depositing shapes and dimensions with texture

splatting. The weathering simulation is demonstrated by example of rusting and moss growth on a model of a water hydrant and the Stanford bunny. The texture spaces of the models have been constructed to be particularly affected by the Günther et al. (2012) approach to better illustrate the problems associated with it. This includes texture seams in visible areas and greatly varying texel sizes in world space. The Günther et al. (2012) material deposit approach is applied to these models to show the visual artefact. These results are directly compared to the result obtained using the new method described in Chapter 4, with respect to consistencies in depositing size, stretching artefacts and deposit discontinuities. Additionally, the texture splatting technique and control over depositing dimensions are demonstrated on a number of examples.

These decomposition methods are applied to models of a human arm and human head with internal flesh and bones layers and an outer skin layer. Images and video clips are generated of the resulting post mortem appearance. These will be compared to photographs of dead bodies that have been subject to the analogous real world decay processes and descriptions of these processes in literature. A qualitative evaluation by visual comparison is employed for two principal reasons. Firstly, the goal of the human body decomposition simulation is to realistically reproduce the visual appearance of corpse decomposition, rather than to correctly model physical behaviours. Secondly, statistical data on livor mortis and mummification particularly is difficult to acquire for use in a quantitative evaluation. Most data available is based on photographs, descriptions and total weight loss when fully mummified.

The livor mortis method needs to handle blood pooling (haemoglobin transport), fixation of hypostasis, pressure induced blanching, oxygen dissociation, and the resulting skin colouration changes caused by these processes. The pooling effects are tested by examining the areas of discolouration in the visual results. For this the models are rotated before and during the livor mortis simulation to demonstrate that the blood dynamics move under the force of gravity. The colouration of the skin changes throughout the livor mortis process. To evaluate the accuracy of the skin shader, simulation results are recorded at different stages and compared to photographs of livor mortis at the corresponding degree of progression. Pressure induced blanching effects are evaluated by applying pressure to an area on the object's surface and record the retreating pink colouration. Fixation by hypostasis is demonstrated by rotating the object during and after fixation as well as by means of applying pressure to the object's surface to show the reduced effects of the blood dynamics expected during fixation.

The mummification process starts some time after the fixation of hypostasis has completed. The mummification simulation consists of four parts; humidity

diffusion, volume deformation, skin deformation and skin shading. The humidity diffusion is evaluated by examining whether its progression follows the same patterns expected from the description in dehydration literature. Visual comparison of the simulation results with photographs of dessicated mummies are used to evaluate the realism of the volume shrinking, skin deformation and skin shading. Additionally, the progression of the simulated mummification is compared to descriptions and photographs at different stages during an experimental mummification of a human lower leg. A comparison with the lower leg was chosen as the experiment provides descriptions and colour photographs of the mummification process throughout. Furthermore, the experiment was performed using a detached limb, which corresponded with the detached head and arm model that were available for this project.

1.2 Contributions

In this section the main contributions of this thesis are summarised. This includes a survey of literature on object decomposition. The human body decomposition simulations are divided into three processes, surface weathering, livor mortis and mummification by dessication. The contributions for each of these are outlined below.

Survey of Related literature

Firstly, a comprehensive survey of literature in the area of object morphology changes and decomposition in computer graphics is presented in Chapter 2. The survey focuses on techniques used to simulate morphology changes due to natural influences such as cracks, fractures, patina, corrosion, erosion, burning, melting, decay, rotting and withering. This includes a critical summary of current approaches and identifying limitations of current methods and gaps in the literature. Particular attention is paid to the different object representation and deformation techniques used.

Surface Weathering

The surface weathering technique proposed in this thesis is based on the work by Chen et al. (2005) and Günther et al. (2012). This thesis intends to improve on their work to avoid some of the visual artefacts that can occur using the

Günther et al. (2012) method and provide more user control over the appearance of deposits. These can be summarised as:

- An approach to deposit materials on an object's surface by projecting a square (the depositing square) onto the object's surface. The square is then translated into texture space and filled with the material to be deposited. This differs from the Günther et al. (2012) method, who use a fixed number of texels per deposit and the method by Chen et al. (2005) who use surfels. The square projection method is able to reduce depositing distortions, such as varying deposit dimensions and stretching. Furthermore, it allows for user control over the sizes of deposits.
- A world space subdivision of the depositing square into non-overlapping polygons, each residing on a single triangle face. Each polygon is then translated into the object's texture space and filled with the material deposits. This helps with avoiding discontinuity errors in material deposits along texture seams.
- A texture splatting approach in which a template texture is mapped onto the depositing square. The template texture is a grey scale image that acts as a stencil. The splatting textures can be used to control the shape of the deposits, for example to create sharp or soft boundaries on the deposits or to avoid unnatural polygonal shaped deposits.

Livor Mortis

A novel livor mortis simulation is presented that is able to model the post mortem appearance changes due to blood pooling. This involves:

- A simulation of post mortem blood dynamics on an irregular tetrahedral volume mesh by transferring haemoglobin along tetrahedral edges in a force direction. This is used to simulate blood pooling and blanching:
  - For blood pooling gravity is used as the force and haemoglobin is moved along edges in the gravity direction. This allows the accumulation of haemoglobin in lower lying areas of the body, even if the model is rotated during the simulation. Furthermore, the irregularity of the tetrahedral mesh mimics the patchy appearance of early livor mortis.
  - To simulate pressure induced blanching the pressure direction is used as the force and haemoglobin is moved away from the pressure point.

This results in haemoglobin being pushed out of the pressure zones which causes the blanching effects.

- A method to fix the haemoglobin on the tetrahedral nodes to simulate the fixation of hypostatis caused by the breaking of the blood vessels and the subsequent staining of the surrounding tissue.
- A method to simulate the oxygen dissociation over time which affects the colouration of the blood, turning it from bright red to a dark red.
- A layered skin shader approach that is able to reproduce the colouration changes to the skin caused by changes in haemoglobin and oxygen saturation.

Mummification

The mummification simulation aims to reproduce the visual affects of natural mummification by dessication. The method for the dessication and volume deformation is based on previous work on fruit rotting by Liu et al. (2012b), which use the Finite Element Method to solve the humidity diffusion in the flesh volume of the object and the resulting volume shrinkage. This method is applied to the more complex model of human body parts and combined with the livor mortis simulation.

- As opposed to Liu et al. (2012b), the skin deformation is achieved using position based dynamics. Tracking constraints connect the skin to the flesh layer. This results in the skin being pulled along if the internal volume shrinks. Further stretching and bending constraints control the skin deformation, causing the formation of wrinkles. This allows the simulation of fine wrinkles on thin areas, such as fingers, and larger wrinkles and folds on the arm, rather than the skin buckling effects observed in fruit rotting.
- The effects of water loss on the connection of the epidermis and dermis layers are simulated by loosening the tracking constraints with water loss to simulate skin slippage. This separation of the epidermis from the dermis is often observed in mummification by dessication. Furthermore, the stretching constraints are stiffened with water loss to achieve the stiffer dynamics as the skin turns leathery.
- A skin shading approach to visualise the skin colour changes caused by water loss. The skin shading approach is an extension of the one proposed

by livor mortis; it considers moisture content, as well as haemoglobin content and oxygen saturation to determine the colour of the dermis layer.

1.3 Background on Ninja Theory

The Engineering Doctorate (EngD) is an EPSRC funded doctoral training course with a three year industrial placement. The sponsor company for this project is the video game developer Ninja Theory Ltd. Cambridge-based Ninja Theory is continuously exploring new gaming technologies that may help in creating new and exciting game experiences, strong characters and stunning art and design.

With their first game, the combat-based, action adventure game Heavenly Sword (2007), Ninja Theory pioneered the use of performance capture in video games. Performance capture has also been used in their subsequent games, Enslaved: Odyssey to the West (2010) and DmC: Devil May Cry (2013), a reboot of Capcom's Devil May Cry franchise. A re-mastered version of the game, Devil May Cry: Definitive Edition, was released in 2015. Currently, Ninja Theory is working on an action, hack and slash title called Hellblade: Senua's Sacrifice. With Hellblade, Ninja Theory continue to promote the use performance capture for video games. The studio presented a new technology, called real-time cinematography, at the 2016 SIGGRAPH Real Time live, see Antoniades (2016), which won the SIGGRAPH Award for Best Real-Time Graphics and Interactivity. Real-time cinematography allows for live capturing and realistic rendering of digital characters.

With Hellblade, Ninja Theory proposes a new independent AAA game development business approach; creating, funding and owning the IP of a game, while aiming for the high quality of AAA games, see Ninja Theory Ltd. (2014b). Ninja Theory takes an open development approach with their new independent AAA title, which allows those who are interested to follow the game development process on a dedicated website Ninja Theory Ltd. (2014a).

Ninja Theory also worked with Disney Interactive Studios on Disney Infinity 2.0: Marvel Super Heroes (2014) and created the Disney Infinity 3.0 Twilight of the Republic Play Set released in 2015. The studio developed a free-to-play mobile game, called Fightback in 2013, and more recently released the VR game DEXED (2016), the studio's first self published title.

The work in this doctorate is of interest to Ninja Theory and the game industry in general, as they continue to push technological barriers in games to create

distinctive and realistic worlds.

1.4 Thesis Structure

In this chapter the research objective and its real world context was outlined. The following chapter (2) is a summary of the related literature in the area of object morphology changes due to environmental influences, in which the current state of the art is critically evaluated and gaps in the literature identified. An object representation constructed using concepts from the literature review is defined in Chapter 3. Different object representations are chosen for the simulation, visualisation and collision detection, as is the current practice in simulation systems.

The main body of this thesis is concerned with the organic decomposition of human remains. This is a complex process influenced by various connected and mutually exclusive activities. For this reason, three subjects were chosen that address three processes of varying effects on the decomposed body.

Surface appearances changes due to external influences are described in Chapter 4. This consists of colouration and texture changes on the object's surface caused by material deposits. The surface weathering approach is based on a particle simulation called *gamma-ton* tracing, which will be explained in Section 4.2. Section 4.3 proposes a material deposit method that is able to avoid visual depositing anomalies of previous methods. The results and the method's limitations are demonstrated in Section 4.5.

A biologically-inspired Livor mortis approach is presented in Chapter 5. Livor mortis refers to the post mortem colouration of the skin caused by blood pooling. Although livor mortis mainly shows itself as skin colouration changes, the changes are driven by processes internal to the body. Section 5.2 provides biological background information required for the livor mortis simulation. Livor mortis is simulated by modelling the internal blood dynamics described in Section 5.3. These include haemoglobin transport due to gravity (Section 5.3.2), oxygen dissociation (Section 5.3.3), pressure induced blanching (Section 5.3.4), and, finally, fixation of hypostasis (Section 5.3.5). The effects are visible as pinkish and later purple skin colouration which are visualised using a layered skin shader described in Section 5.4. The result of each process of the livor mortis simulation is discussed in Section 5.5.

The last process modelled as part of this doctorate is natural mummification by dessication, shown in Chapter 6. In contrast to surface weathering and

livor mortis, mummification affects the morphology of the objects and leads to major geometrical changes of the body. Section 6.2 provides a short introduction to mummification and in particular mummification by dessication. This is followed by Section 6.3, explaining the Finite Element Method which will be used as the main approach to model the humidity diffusion (Section 6.4) and resulting volume deformation (Section 6.5). For the skin deformation, position based dynamics are used, as explained in Section 6.6. Apart from deformation, mummified tissue also undergoes colouration changes. These are produced by a modified version of the livor mortis skin shader. The modified skin shader is explained in Section 6.7. The mummification results are examined in Section 6.8.

Finally, a conclusion on the realisation of the research objective is drawn in Chapter 7. The limitations of the overall approach are outlined in Section 7.1 and ideas for future work in the area of organic body decomposition are suggested Section 7.2.

Chapter 2

Review of Morphology Changes Literature

2.1 Introduction

Real world objects are not immutable and free of blemishes, but tend to change in appearance over time when exposed to their environment. As such, simulating these phenomena can have a significant impact on the perceived realism of a computer generated scene. Artists can manually compose and manipulate different textures and models which illustrate the different stages of objects under alteration. However, this can be a very labour-intensive and time-consuming process. An artist may have to create textures and different models for every object under different environmental settings. An alternative approach is to automate the generation and modification of textures and models in order to achieve the desired effects. Currently researchers in the area have developed a number of different approaches to model a variety of natural influences on virtual objects.

While methods to simulate erosion have been present for some time now, as seen in Kelley et al. (1988); Musgrave et al. (1989), other processes such as decay of organic objects are still emerging and evolving. Mérillou and Ghazanfarpour (2008) offer an overview of weathering and ageing phenomena in computer graphics, however, the methods described here mainly focus on surface changes, such as texture generation, BRDFs (Bidirectional Reflectance Distribution Functions) and colouration changes and do not consider rotting and withering of organic objects. There are several papers that focus on general

deformable models in computer graphics such as Gibson and Mirtich (1997); Nealen et al. (2006), but they do not consider the phenomena that cause the deformation.

Our aim is to give an overview of the state of the art in the area of object morphology changes due to the influence of the environment. In contrast to Mérillou and Ghazanfarpour (2008); Gibson and Mirtich (1997); Nealen et al. (2006), the focus of this chapter is the natural phenomena which lead to structural and morphological changes. Only the environmental processes that deteriorate objects over time are considered. Phenomena that do not lead to the decline of an object, such as wetting, are not considered in this survey. Furthermore, the focus is on methods that consider geometrical and structural changes, such as fracturing, deformations and holes and pay less attention to those that only affect the object's shading and surface (such as staining, dust, tarnishing etc.). Natural phenomena that satisfy these criteria include weathering and ageing effects, such as corrosion, rotting, erosion and other phenomena such as burning and melting.

The changes an object undergoes depend on the natural phenomena involved and, as such, the different methods are categorised according to the physical process behind the deformation. Aside from the methods used to simulate the phenomena themselves, this chapter will look at the techniques used to model and deform the affected object. Objects such as fruit and rocks tend to consist of heterogeneous material or are made of different layers which affect their appearance and degeneration. This chapter also contains a discussion of how representative a chosen model is of its real world counterpart and to what degree this effects the simulation results. The deformation techniques will be evaluated according to their level of effect on the object, i.e. whether they affect the surface geometry, the thin shell geometry or the whole volume. This chapter of the thesis has been published in the Elsevier's Computers & Graphics journal, in the form of a survey, see Frerichs et al. (2015), which concentrates on the different natural phenomena that lead to major geometric deformations of an object.

2.2 Overview

The different object weathering and decomposition processes have been classified as chemical, mechanical and biological weathering by Dorsey et al. (2010) and Mérillou and Ghazanfarpour (2008). Chemical weathering includes corrosion, tarnishing, fading, combustion and phase changes. Mechanical weathering

refers to peeling, cracking, erosion and sediment decompositions. Biological weathering, on the other hand, often refers to the rotting and withering of organic objects. However, an object's weathering and ageing is often the result of different processes. For example, erosion has been described as a mechanical process by Dorsey et al. (2010) and Mérillou and Ghazanfarpour (2008), who consider hydraulics to be the cause of erosion. There is another type of erosion called spherical weathering, which is described as chemical erosion, see Beardall et al. (2007); Jones et al. (2010). For these reason it seems more appropriate to classify the weathering processes by natural phenomena.

The subsequent sections focus on simulations of the different natural phenomena that effect an object's appearance and geometry. The different methods are classified in terms of the phenomena that drive the deformation.

- Cracks and Fractures (Section 2.3) appear on objects over time due to fatigue or impact with other objects.
- Patina and Corrosion (Section 2.4). Patina is material film produced by oxidation and can be found on copper and bronze. Corrosion, such as rusting, is a chemical process that leads to the decomposition of metallic objects. It can have a significant impact on an object's appearance, as it causes the loss of material which can then lead to holes and major deformations.
- Erosion (Section 2.5) of terrain and rock formations results in the displacement of material and can shape whole landscapes.
- Burning (Section 2.6), objects decompose over time as their material is consumed by fire. Additionally, thin sheet objects crumple and bend.
- Melting and other phase transitions (Section 2.7) deform an object gradually by changing it, for example, from a solid to a liquid state (or vice versa). The liquid mass is then free to move, which results in material being removed from the solid object.
- Decay, Rotting and Withering (Section 2.8) phenomena, such as fruit rotting and leaf withering, greatly deform the object over time.

A short overview of Vascular Modelling and Graph Algorithms is provided in Section 2.9. Finally, Section 2.10 will critically summarise the current methods and identify gaps in the literature.

2.3 Cracks and Fractures

Cracks and fractures can indicate an object's composition, age and environment. This section will only consider three-dimensional cracks and fractures that affect the object's geometry. For an overview which also includes surface cracks see Muguercia et al. (2014) for a publication on fracture modelling. A popular approach is to pre-compute a crack pattern and later apply it to the three-dimensional object, which avoids expensive stress computation at runtime, see Desbenoit et al. (2005); Valette et al. (2006); Su et al. (2009); Müller et al. (2013). This is often referred to as pre-scoring. The first section will consider work that is concerned with on-the-fly crack and fracture propagation, followed by a section on methods using pre-scoring.

2.3.1 Crack and Fracture Propagation

Cracks are formed due to stress from internal and external forces which need to be modelled in order to simulate crack propagation. There are a number of approaches concerned with simulating cracking and fracturing of volumetric objects on the fly. Tetrahedral meshes are commonly used to represent volumetric objects in physically-based simulations. They can be used as the basis for physical-based deformation techniques, such as Mass-spring systems and the Finite Element Method (FEM), which can be used for stress and strain computations to determine the cracking path.

Mass-spring systems are a simple way of modelling deformable objects, are easy to construct, and generally require less computational time than the FEM. The object is represented by a finite set of nodes, called point masses, where each point has mass and position. The point masses are connected by elastic springs that act under the physical laws of spring dynamics (see Nealen et al. (2006) for a more detailed explanation on Mass-spring systems). The vertices of a tetrahedral mesh can act as the point masses in a mass spring system, where the tetrahedral edges represent the springs. Cracks can be formed by cutting springs that suffer under mechanical fatigue. Hirota et al. (2000) and Aoki et al. (2004) use this model to simulate crack formation on drying objects. In both approaches measured parameters are used for the strain computations. In addition to this, Aoki et al. (2004) introduce a moisture model to affect the strain parameters according to moisture loss. The creation of cracks by spring cutting restricts the cracking path to the tetrahedral edges which result in a 'jagged' crack appearance. Furthermore, the visual quality of the cracks depends heavily on the tetrahedral size.

Mass-spring systems are restricted in realistic mechanical behaviour and visual quality. For this reason, a physically more accurate approach, like the Finite Element Method (FEM), is desirable. Instead of the discrete model of a mass-spring system, the object is a continuum with mass throughout. Dynamics are governed by equations from continuum mechanics. In order to solve these the object is broken into a finite number of elements. The stress and strain can then be solved for each element. O'Brien and Hodgins (1999) use the FEM to simulate brittle fracturing based on elastic fracture dynamics. The object is again represented by a tetrahedral mesh, whose vertices serve as finite element discretisation for solving the strain or separation tensor. This provides a per-vertex maximum stress criterion for the crack propagation. A crack is created by computing a fracture plane that cuts the affected tetrahedra. This allows for arbitrary cracks that are not restricted to tetrahedral edges. However, this approach requires local remeshing at the fracture path which can be computationally expensive. Their brittle fracture simulation is later extended by O'Brien et al. (2002) to simulate ductile fracture by accounting for plastic strain during strain computation. This is done by separating the strain into plastic and elastic components. Hegemann et al. (2013) also consider ductile fracture but instead evolve fractures on a level set using an embedded geometry approach. In their approach tetrahedral elements reside in a background lattice and a level set is defined on the mesh nodes of the undeformed object to act as the material boundary. Griffith's energy minimization is used as fracture evolution criteria in order to evolve the level set. This means that changes in the object's geometry are handled by the level set method and thereby no remeshing is required. Level set methods do come with some limitations, as they can be memory intensive and are restricted in representing thin geometry. As in previous methods, Hegemann et al. use the FEM to solve the elasto-plastic dynamics.

A drawback of the FEM can be its computational complexity. This may lead to problems in real-time applications, such as the in-game fracture simulation by Parker and O'Brien (2009). In order to simplify the deformation process Parker and O'Brien take a free form deformation (FFD) approach. The idea behind FFD is to enclose an object in a simpler structure, called the proxy object. The proxy object is constrained to the simulation object in such a way that deforming the proxy object deforms the simulation object. The fracture object (graphical representation) is enclosed in a tetrahedral mesh (simulation mesh). Then, every vertex of the fracture object is assigned to its surrounding tetrahedron by expressing its position relative to that tetrahedron using barycentric coordinates. Using a different representation for the simulation object and the

graphical object has become common practice. The graphical mesh can be very complex and consist of a very large number of triangles. By embedding it in a less complex tetrahedral mesh the simulation process can be simplified without the loss of detail on the graphical mesh. The graphical mesh is deformed by moving its vertices in response to the deformation of the enclosing tetrahedron. The tetrahedral mesh itself deforms in response to the stress field, which is solved using the FEM. The simulation mesh is stored in texture maps in GPU memory, which allows for the deformation to be performed on the GPU. In contrast to previous methods, O'Brien and Hodgins (1999); O'Brien et al. (2002), Parker and O'Brien do not split individual tetrahedra as this would result in growing numbers of tetrahedra, which is undesirable when aiming to maintain a constant frame-rate. As observed in the approaches of Hirota et al. (2000) and Aoki et al. (2004), introducing cracks only along tetrahedral edges can lead to a 'jagged' crack appearance. However, due to their graphical model having higher resolution than the simulation model, they can avoid this. Artists can specify how the fracture surface should appear when generating the graphical model.

The method described by O'Brien and Hodgins (1999) uses a per-vertex stress criterion for their stress computation approach which relies heavily on the size of affected triangles. To avoid this, Molino et al. (2005) propose a virtual node algorithm to simulate breaking and fracturing of objects. Their approach builds on duplicating the simulation mesh, where each copy represents a portion of the material of the original mesh. Virtual nodes are used to keep track of the number of copies and the material contained in each. In addition to this, level set values are stored at nodes on the mesh and can be used as a prescoring control structure to direct fracturing. This allows the user to specify how a particular element cracks and thereby produces some degree of heterogeneous cracking behaviour. A diagonalized finite elements method is used to model the deformations. As in previous methods, the FEM is combined with an embedded geometry approach which helps to reduce the computational complexity while preserving sharp corners in the graphical representation. Bao et al. (2007) use this virtual node approach to simulate fracturing of rigid material on triangle meshes. The virtual node approach works with both tetrahedral volume and thin shell objects. However, performance can suffer under the increase in geometry due to duplication of the simulation mesh.

Aside from volumetric objects, there have also been approaches concerned with simulating cracking, ripping and fracturing of thin shells. In the case of thin shells, a polygonal mesh can act as finite element discretization of the object. Elastic deformations of thin shell objects can be simulated on a triangle mesh by

using the triangle vertices as finite element discretization and solving the elasto-plastic equation using the FEM, see Pfaff et al. (2014). Gingold et al. (2004) consider inelastic deformation. They define bending strains on the triangle faces to compute the fracture path. Both approaches consider exclusively thin shell objects though their fracture path is not restricted to triangle edges but can also fracture individual triangle faces. They achieve this by introducing new internal vertices along the fracture path and applying global re-meshing to increase resolution at the crack tip.

The cracking and fracturing behaviour differs with different materials, and, as such, cracking and fracturing of heterogeneous objects would result in different effects than it would on homogeneous objects. Despite this, the majority of proposed methods do not consider heterogeneous materials. Busaryev et al. (2013) focus on simulating fracturing of objects made of multiple layers. They simulate thin-plate fracturing of paper-like objects represented by a triangle mesh. The same triangulation is used for all layers in material space which allows layers to be connected by springs. Large deformations result in the spring breaking and world space positions are defined per layer to account for these layer-specific deformations. The triangle mesh is used as a finite element discretisation in order to solve the thin plate dynamics. The crack paths are determined by computing the separation tensor as in the O'Brien and Hodgins (1999) method, but a relaxation method is proposed to prevent the object fracturing into small pieces. Additionally, fracture-aware Delaunay remeshing (Cheng et al., 2012, chap. 6) is applied for better mesh quality around the fracture path. The Busaryev et al. approach only works on thin shell objects. Molino et al. (2005) allow for some degree of heterogeneous cracking behaviour by means of level set prescoring. Apart from those, simulating the cracking and fracturing behaviour of heterogeneous volumetric objects has not been considered.

As in the Busaryev et al. (2013) method, mesh based models might suffer under the need for remeshing. There are other ways of representing deformable objects however that are not based on meshes. Such meshless methods lack connectivity and therefore do not require remeshing. With that in mind, Wicke et al. (2005) choose point samples to model the surface of thin shell objects and embed parametric curves in this structure that represent the object's fibre. The mesh is deformed by moving the points according to stress forces which are computed directly from the surface parameters. Cracks and fractures are formed by cutting affected fibre. Their method suffers in quality with large deformations as they do not resample the simulation nodes and it can not handle very stiff materials. Connectivity is present in their representation in the form of connecting fibres and, as such, is not a truly meshless approach. Pauly et al.

(2005) use a meshless approach without connectivity but make use of an embedded surface representation inspired by Müller et al. (2004) (see Section 2.7.2). Here the surface is represented by surfels which are embedded in a volumetric domain. The crack propagation is controlled by a stress tensor solved using the FEM on the volumetric domain. To avoid quality suffering under large deformations they make use of resampling. During crack propagation more surfels are continuously added along the fracture path.

2.3.2 Prescoring

In order to reduce computation time at run-time a fracture pattern can be created offline and applied to the object at run-time. This is referred to as prescoring. Desbenoit et al. (2005) and Valette et al. (2006) both use prescoring and apply a crack or fracture path to a triangle surface mesh. Desbenoit et al. (2005) choose a non-physical approach and generate a fracture pattern atlas from real world photos. In contrast, Valette et al. (2006) create a two-dimensional mesh of cracks with depth values which can be mapped onto the object. They use a shrinkage volume propagation method to compute the crack path and enlarge cracks. The object needs to be parametrized when applying the crack pattern and, as such, must be remeshed. Due to choosing a surface representation, their method can only generate cracks starting from the surface and running along the surface of the object. Internal cracks are therefore not possible with their representation.

A volumetric fracture pattern can be used to allow for internal cracking. Su et al. (2009) create a whole volumetric fracture pattern using the object's bounding box. The fracture volume emanates from a point of impact. This fracture volume is then mapped to the simulation mesh by rotating and translating it until their impact points match, at which point they are merged (see Figure 1). Using a volumetric prescoring allows for internal crack and fractures. However, this method requires the construction of a level set representation of the object in order to merge the simulation mesh with the volumetric fracture pattern.

Müller et al. (2013), on the other hand, avoid the use of level sets by introducing volumetric approximate convex decomposition (VACD), in which the mesh is fragmented into convex pieces using Voronoi decomposition. This structure enables the authors to apply a prescored fracture pattern directly onto the convex shape. Affected convex shapes are clipped with the fracture pattern to produce the fractured object. This method is able to represent sharp edges and thin features as it does not need to use a level set when applying the fracture pattern. Furthermore, the fracture simulation runs in real-time for

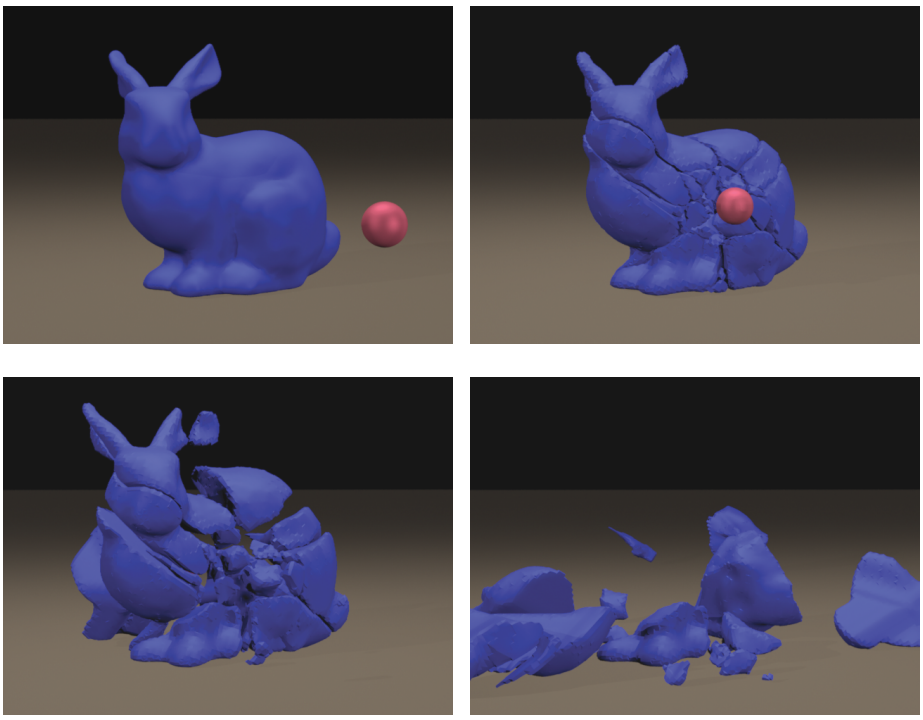


Figure 1: This Figure is taken from the Su et al. (2009) fracture simulation paper and is demonstrating their prescoring approach used to shattered objects

large objects so it can be used in real-time applications. Müller et al. do not apply a physically-based fracture simulation. Instead, their fracture patterns were produced by artists. Another example of a non-physically-based approach is described by Glondu et al. (2012). They choose an example-based fracturing approach in which fracture statistics are extracted from photographs and then applied to the object.

2.4 Corrosion, Rusting and Patina

Patinas are films or incrustations on a metallic surface and are produced when material is removed or added to a surface, or through chemical reactions like oxidation. The occurrence of patinas is dependent on the material's characteristics and environmental factors. Taking this into consideration, Dorsey and Hanrahan (1996) introduce a layer representation analogous to its real physical counterpart. Each layer is made of a homogeneous material with space varying thickness. The adding or removing of material is achieved by simply increasing or decreasing the layer thickness value using coating, eroding, filling and

polishing operators. Dorsey and Hanrahan do not simulate the physical phenomena behind patinas but instead control the depositing and growth pattern with fractal surface growth models. This allows for a simple and efficient patina simulation, which however is not influenced by environmental properties such as rain and pollution. The combined thickness of all layers is used in the rendering process and could be extended to alter surface normals or create displacement maps. Still, they do not model any geometric deformations caused by removal or addition of material. In contrast to this, the Dorsey et al. (1999) approach for simulating stone weathering does model geometric changes. They introduce a slab data structure, which is a thick band of volumetric entities aligned with the surface that restricts the simulation and deformation processes to an area around the object's surface. The model is voxelised and a signed distance is computed at each voxel and stored in the slabs. The signed distance values are used to generate a density function representing the stone volume. Erosion of stone is achieved by reducing the density value at the affected area. While this slab data approach reduces computation time, it is only able to affect an area around the surface. This is arguably sufficient for simulating weathering of stone, as the majority of stone ageing effects only occur in a thin layer on the surface. It is not, however, adequate for simulating the ageing of other organic objects, such as fruit, where the deformation often starts from the internal flesh layer. It is also limited in simulating weathering of softer material, such as metal, where rusting can lead to holes.

Mérillou et al. (2001) focus on the corrosion of metals due to rusting. A corrosion map is constructed, which includes the colour and height values of the object. A height value under a certain limit represents a hole. The corrosion map is constructed by predicting a starting point for corrosion and then spreading the rust using random walk. The corrosion map is then applied to the mesh, affecting its surface parameters and geometry. Their method is able to model geometric rusting effects, such as holes, with reasonable results. It is however difficult to control and does not model fractures of objects weakened by rust. Similarly, Chang and Shih (2003) choose a random rusting seed and spread the rust using an L-system. They concentrate on the specific phenomena of rusting in seawater and introduce a simple current model that effects the tendency of a surface to rust and the rust distribution. As in the Dorsey and Hanrahan (1996) method, the metallic object is represented in layers (see Figure 2), which allows them to model a variety of different corrosion effects, such as pitting, cracks, blisters and uniform corrosion. Pits and cracks are achieved by subdividing the effected triangle surface. Blisters are created by moving the affected vertices along its surface normal. In a similar way to the stone weathering approach

by Dorsey et al. (1999), the rusting effects of the Chang and Shih (2003) method are limited to the surface and do not allow for holes such as in Mérillou et al. (2001), On the other hand, there is consideration of other rusting effects beyond holes such as blisters, cracks and pitches. As in previous methods, no fracturing of the corroded object is considered.

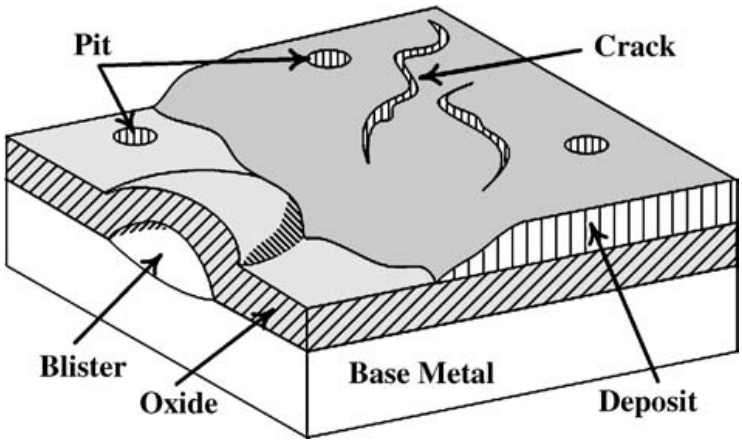


Figure 2: Layered structure of rusting object Chang and Shih (2003).

Wojtan et al. (2007) consider acidic corrosion, as well as erosion and sedimentation. Their approach is based on Melek and Keyser (2003); Losasso et al. (2006) (see Section 2.6.1, 2.6.2 and 2.7.1). The object's surface is represented by a level set stored in a three-dimensional grid. The level set is advected according to a velocity field. The velocity field is created at the solid-liquid interface and is affected by the dynamics of the surrounding fluid. The fluid is represented by a particle level set and simulated by solving the incompressible Navier-Stokes equations using finite difference. Their representation can handle separation of the solid object by detecting broken pieces and creating a new object for each piece. Furthermore, they are able to simulate different corrosion, erosion and sedimentation scenarios. They do not handle the crumbling and collapsing of structures weakened by corrosion and their method is restricted in modelling thin geometry due to the level set representation of the object.

There have been efforts to construct a general weathering simulation that can also handle rusting by Chen et al. (2005) and Günther et al. (2012). Particles, called γ-*tons*, which carry wear-inducing materials are tracked through a scene and deposit or pick materials up from the surface of the intersected object. Material deposits are stored in surfels (Chen et al. (2005)) or texture maps (Günther et al. (2012)), which are then used to generate the weathering effects. In the Chen et al. (2005) approach, displacement maps can be used

to achieve small surface changes, for example due to rusting. On the other hand, Günther et al. (2012) do not consider surface deformation, though their approach runs entirely on the GPU. In contrast to other rusting methods, Chen et al. and Günther et al. can simulate rust-bleeding effects. There is evidence of non-physically-based approaches that can be used to achieve rusting and other weathering effects (Gu et al. (2006); Wang et al. (2006); Becket and Badler (1990)) though they account for shading effects only.

The majority of methods described above model deformations due to rusting and other corrosion, however, they constrain the deformation effects to a relative thin surface layer and, as a result of this, do not allow for major deformations such as holes. Mérillou et al. (2001) do consider the occurrence of holes through rusting but do not account for the effects of rusting on the structure of objects, which could lead to fractures. The Wojtan et al. (2007) more general corrosion approach models major geometric deformation and object splitting by making use of the level set method.

2.5 Erosion

Fractal-based methods, such as the Brownian motion, are popular when creating landscapes, as they allow for varying levels of detail at different scales Mandelbrot (1982); Miller (1986). However, fractal-based methods do not consider the weathering and erosion processes that shape landscapes and stone formations. Water flows can shape landscapes in a significant way. They create river beds, canyons, river stream networks, valleys and so forth. Thermal erosion can smooth steep slopes and can lead to talus slope creation. This section will discuss methods to simulate hydraulic and spherical erosions, that deform terrain and stone formations.

2.5.1 Hydraulic and Thermal Erosion

Early methods that model hydraulic erosion represent the terrain with height maps or elevation models, since they are sufficient to store and easy to manipulate. The loss and gain of matter due to erosion can be achieved by decreasing or increasing the height value at affected areas. Kelley et al. (1988) and Musgrave et al. (1989) create a stream network that is then applied to the terrain by adjusting the height values along the stream network. Both methods share the same advantage with fractal-based terrains, insofar that the landscape can be modelled at varying degrees of detail. However, the Kelley et al. method

does not consider the topology of the underlying terrain, which plays an important role in the hydraulic erosion process. Musgrave et al. simulate water flow by transferring water to lower lying neighbouring cells, which only accounts for local topology. Due to only considering local topology the inertia of flowing water is ignored, which makes it difficult to model large scale valley deformations due to water flow. For this reason Chiba et al. (1998) create a velocity field of water flow to drive the erosion process. The water flow is simulated by particles that are placed at the grid points of the terrain and their collision with the ground leads to erosion. In contrast to the previous two methods, the velocity fields created from the particle flow reflect the global topology of the underlying landscape. Krištof et al. (2009) placed additional particles on the boundary of the terrain, to enable sediment exchange between the ground and water. The information in the boundary particles is then used to update the height values of the underlying terrain. This is achieved by coupling the Eulerian approach for the erosion simulation with a Lagrangian approach, called Smoothed Particle Hydrodynamics (SPH), for the fluid dynamics.

A more recent method combines hydraulic erosion with tectonic-driven uplift to generate large scale mountain terrains. Tectonic uplift causes mountain growing due to the collision of the earth plates. User generated uplift maps are combined with stream networks. Uplift is controlled by a user defined uplift map, which acts as an uplift gradient mask. Erosion is simulated using the stream power equation as described by Whipple and Tucker (1999). Erosion is applied to a randomly initialised planar graph, called the stream graph. The stream graph is initialised randomly and holds elevation and flow information. The stream power equation is applied to this graph, driving the erosion process. Then the graph is converted into an elevation model which is used to represent the terrain. Their method considers flow dynamics in lakes and lake overflows.

A height map representation of the terrain also allows to simulate deformations of mud, sand and snow due to impact of other objects, as per Sumner et al. (1999). The objects deforming the terrain are represented using a contour map, which works similarly to a distance field but has zero values outside the object. Deformation of the underlying terrain is achieved by decreasing the height of the area that intersects with the object and distributing the lost height onto the closest non-intersecting cells. In consecutive steps, an erosion algorithm is used, which detects steep slopes and moves material down the slope for a more realistic look. A contour map representation of the object allows for simple terrain deformation caused by intersection, but it does not account for the effect of the object's velocity on the terrain deformation. This causes the material to be distributed evenly around the object, when in fact material displacement

should be larger in the direction of travel. Onoue and Nishita (2003) improve on this by considering the object's direction of movement during granular material displacement. In addition, they introduce a height span map, which is a two-dimensional array storing the span of an object at that grid point together with a height value of a granular material on top of that span. This allows them to model granular material on surfaces, which is not possible with a simple height map. Their simulation runs in interactive time, which allows for some user control during the simulation though remaining unusable in a real-time application.

Height maps are a simple way to represent and deform terrain but they are restricted in landscape representation, as they cannot model concave structures, such as tunnels and overhangs. They are also unable to represent different geological layers of varying erosion resistance. Voxel representations on the other hand allow for all this and have been used to simulate different erosion effects. Beneš et al. (2006) describe an approach in which a voxel model of the environment is used, where a voxel can hold liquid, air or material. This allows them to solve the fluid dynamics using the Navier-Stokes equations, solving for the velocity and pressure fields of each voxel. Erosion and depositing of material is achieved by changing the state of boundary voxels from material to water/air or vice versa. Using a voxel structure allows Beneš et al. to model a variety of effects, such as springs, receding waterfalls, concave and overhang formations. On the other hand, voxel structures require more memory than height maps and surface meshes. Objects modelled by voxels can suffer a visual artefact called step effect.

Nagashima (1998) and Beneš and Forsbach (2001, 2002) introduce a new terrain representation to capture the advantages of multiple material layers of voxels whilst keeping the efficiency of height maps. The terrain is modelled as a two dimensional grid, as with a height map. Instead of storing height values, each cell consists of an array of height and other material-specific values, such as stiffness. Erosion and deposition of material is achieved by simply decreasing or increasing the height values at the topmost layer, which can reveal underlying layers. Nagashima (1998) sets an initial river course using mid-point displacement to guide erosion. Beneš and Forsbach (2001) on the other hand simulate hydraulic erosion by adding a water layer. The water layer is used to distribute soil and water to neighbouring cells. The layered representation enables the erosion of different materials at different rates, according to a durability value. They do not demonstrate concave or overhang formation. However, Peytavie et al. (2009) and Löffler et al. (2011) later extend the layered structure to a stack-based terrain, allowing for real-time rendering and the construction of

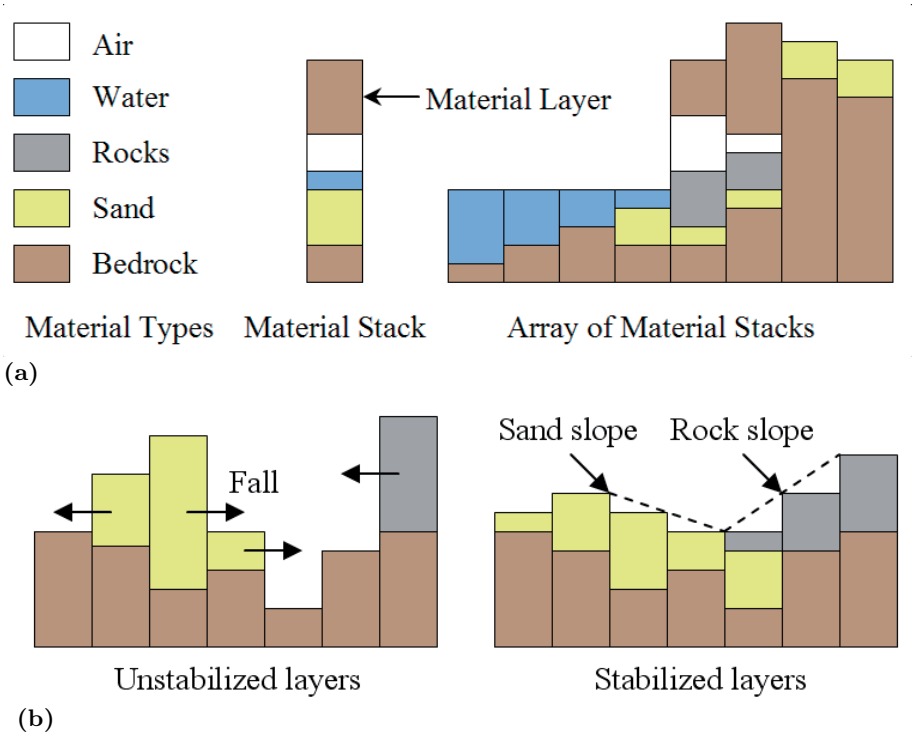


Figure 3: a) shows a stack-based terrain structure which allows for different material layers and overhang formations by Peytavie et al. (2009). b) shows the Peytavie et al. (2009) stabilisation simulation.

level of detail hierarchies (see Figure 3a). They show that it can be used to model concave structures and overhangs. Furthermore, Peytavie et al. (2009) describe a stabilisation algorithm (see Figure 3b) by moving material from one stack onto neighbouring stacks according to their repose angle and material type.

The layered height field structure can also be used to represent the water, sedimentation, velocity and other simulation parameters aside from the different material types, as described by Neidhold et al. (2005); Mei et al. (2007); Beneš (2007). This means that the fluid dynamics for the hydraulic erosion can be implemented on a two-dimensional grid to accelerate the process. Shallow water models are an efficient way to simulate water flow on terrain where both the terrain and water are represented by height fields. To accomplish this Mei et al. (2007) use a hydrostatic-pipe model, in which water is exchanged between neighbouring cells due to pressure difference. The results are then used to calculate the velocity field required for the erosion. The pipe model is efficient and runs entirely on the GPU, making use of its parallelism. Another approach based on a shallow water model is demonstrated by Beneš (2007). A shallow water

version of the Navier-Stokes equation is used to compute fluid dynamics on the two-dimensional grid, treating the soil as fluid of high viscosity. These methods however do not make use of the layered structure's advantage of being able to represent different material layers which allow formation of concave structures and overhangs. For this reason, Št'ava et al. (2008) extend the layered structure to also represent different material layers. Materials can then be exchanged between the topmost layers to account for erosion and deposition. As in previous methods, they do not demonstrate concave or overhang formations.

Polygonal surface meshes are also able to represent complex terrains with overhang and concave formations and are commonly used in real-time applications. Deformations of objects represented by a polygonal surface mesh are usually handled by vertex displacement. Bezin et al. (2010) present a hydraulic erosion method that operates directly on a triangle mesh representation of a terrain. A particle simulation is used to simulate the water flow which drives the erosion and sedimentation. A particle's collision with triangles leads to erosion of the underlying triangles and sediment deposition. Erosion is modelled by displacing vertices downwards and sediment deposition by displacing vertices outwards. The choice of representing the terrain using a triangle mesh allows the simulation to run on the GPU at interactive rates. Polygonal surface models are not always ideal for large deformations, as self collision and topological inconsistency are likely to occur and can be complex to resolve. The Bezin et al. method cannot handle complex topological changes such as the splitting and merging of objects. Furthermore, this method is not able to represent objects with different material layers. This has been extended by Bézin et al. (2014) to model objects with different topological layers and handle topological inconsistencies. A three-dimensional generalized map (3-G-map) is introduced that is able to represent different geological layers. Self collisions are detected and resolved with a corrective operation to avoid topological inconsistencies. This might involve the creation of new vertices. As in the previous method, a particle system is used to simulate the water dynamics, but additional boundary particles are used at the object boundary. Similar to the Krištof et al. (2009) approach described above, the boundary particles interact with the water particles for sediment interchange. Their method uses varying residence for different rock types in order to simulate different erosion effects on different layers. In addition to this, overhang, cave and hole formations can be modelled.

2.5.2 Spheroidal, Cavernous and other weathering

Spheroidal weathering is a chemical erosion process that results in rounded rock formations. Cavernous weathering on the other hand leads to holes and cave-like formations. Both spheroidal and cavernous weathering require a representation that allows for concave and overhang formation and, as such, height maps are not sufficient. Voxels are able to model those formations and material removal is straightforward. Ito et al. (2003) choose a voxel representation for their rock model which allows them to simulate crack formations in rocks called joints. Each voxel is bounded to its six adjacent ones and joints are formed by removing these links. Disconnected voxels are subject to external forces determining their movement, allowing them to fall or slide into a new position. Beardall et al. (2007) on the other hand focus on modelling specific stone formations, called goblins, which are formed due to spherical weathering. The object is represented in an axis-aligned voxel structure. Spherical weathering is achieved by gradually decreasing the decimation level of all voxels within a sphere and eventually removing it. The radius of the sphere depends on the weathering resistance of the stone and its air-to-stone ratio. The air-to-stone ratio is introduced so that areas with a high air value, like edges and corners, weather more quickly. This results in corners and edges being smoothed and the object becoming more rounded. This method is extended by Jones et al. (2010) who approximate the signed mean curvature on the voxel grid in order to direct weathering. This directability allows them to simulate both spherical and cavernous weathering. In addition to this, the Ito et al. (2003) method is used to account for the rockfall caused by weathering. Beardall et al. (2007) and Jones et al. (2010) consider varying rock durability within the rock formation, allowing them to achieve different erosion effects.

Voxel structures are able to model scenes containing concave formations, overhangs and different material types but they do not scale well and are memory-demanding. The Delaunay Deformable Model (DDM) can be used to represent the eroding objects. DDMs are a Lagragian approach to modelling deformable surfaces, that allow for different resolutions and are stable under surface modification. The object's interface is represented by a triangular mesh. The interface is then embedded in a tetrahedralization of the interface points. The mesh is updated by triangulating the object using Delaunay triangulation at every time step. For a more detailed explanation on DDM see Pons and Boissonnat (2007). Tychonievich and Jones consider both spherical weathering and hydraulic erosion on objects represented by a DDM. Their spherical weathering is based on directable weathering Jones et al. (2010). The hydraulic weathering on the other

hand is driven by an Eulerian fluid simulation, where the grid keeps track of water, air and sediment content. The DDM and fluid communicate to determine the erosion and deposits on the DDM and the sediment levels in the water. Deformation due to erosion and deposition is achieved by moving the vertices to a new location and re-meshing the object using Delaunay triangulation whilst also advecting the material properties. This needs to be done at each time step, which can be computationally expensive. Tychonievich and Jones only place vertices on the surface which creates degenerated tetrahedra. This means that their method is not suitable for numerical simulations of the interior and the simulation only works well for changes that propagate from surface inwards. This is arguably sufficient to model erosion but it can not be used to model phenomena that are affecting the internal parts, such as rotting or deformations caused by dehydration.

Other erosion approaches on triangle meshes have been proposed by Skorkovská and Kolingerová (2015, 2016). These methods are able to model multiple materials within the same object using Binary Space Partitioning (BSP). Here the scene in subdivided into cells which are stored in a binary tree, where each leaf contains the subspace's material information. This allows the use of triangle meshes to represent the object's surface, while its volume is represented by the material assigned in the BSP tree cells. For the erosion simulation an approach based on SPH, as in Krištof et al. (2009), is used. Skorkovská and Kolingerová (2016) extend this by proposing a method to automatically generate the BSP tree. Given a volumetric representation of the object, an isosurface extraction method is used to generate the triangle surfaces. Furthermore, implicit surfaces can be used as a splitting function in the BSP tree.

2.6 Burning

Burning scenes involve simulating fire and smoke dynamics. However, in order to appear realistic, decomposition, crumpling and bending of burning objects need also be considered. The modelling techniques can be divided into volume models and surface meshes, which focus on volumetric object decomposition and thin-shell object deformation (respectively).

2.6.1 Volume Models

During burning, material is consumed which results in underlying material being revealed. Therefore, a volumetric representation that is able to model material

removal and different material types is desirable. A combination of different three-dimensional grid structures has been most commonly used to represent objects with those specifications. A simple three-dimensional grid (voxel structure) is used in the Zhao et al. (2003) approach on burning volumes. The volumetric object is represented by a voxel structure where fuel properties are stored at each voxel. These properties are modified when the corresponding voxel is burning. Additionally, an enhanced distance field is defined on the grid structure. This enhanced distance field contains links to the closest boundary point for each cell. It is used for the fire front propagation on the object's surface, which is done by moving the fire front along a virtual iso-surface. This iso-surface only needs to be specified for the fire front propagation but does not need to be generated for rendering purposes. Instead, the consumption of material by fire is visualised by removing burned voxels from rendering, revealing underlying layers. This work does not, however, model any geometrical deformation of the object. The distance field representation has the advantage that the computational cost is more predictable and stable, as it is related to the volume data size rather than the topological complexity of the object. Melek and Keyser (2003, 2004) also make use of distance fields on volumetric grids, but use multiple grids to represent an object. A signed distance field on a regular 3D grid represents the object's boundary and is used to model the actual decomposition and resulting deformation on the object. Another grid holds information on the amount of solid fuel at each location. The fuel amount in the second grid decreases when burning but leaves behind some residue which is accounted for in the first grid. The fluid dynamics for fuel, air and exhaust gases are computed on a third grid. The grids representing the solid object do not need to be the same as the fluid simulation grid. Instead, a coarser grid could be used to represent the solid for a more detailed model. The multiple grid structure allows different simulators to be used for the different stages in the heat transfer. This means that a faster but less accurate (or slower, more accurate) simulator can be used at certain simulation stages while keeping a more accurate (or less accurate) simulator at others. The moving object boundary is tracked using the level set method on the signed distant field in the first grid, which provides a simple way to handle complex topology changes.

A multi-representation of an object, such as the one described above, allows for different representations of the same object to be used for the different simulation and rendering stages. This means that the most efficient or appropriate representation can be used for different stages. This is considered by Melek and Keyser (2005), who propose a general multi-representation framework for physically-based based modelling, which is based on the multi grid structure by

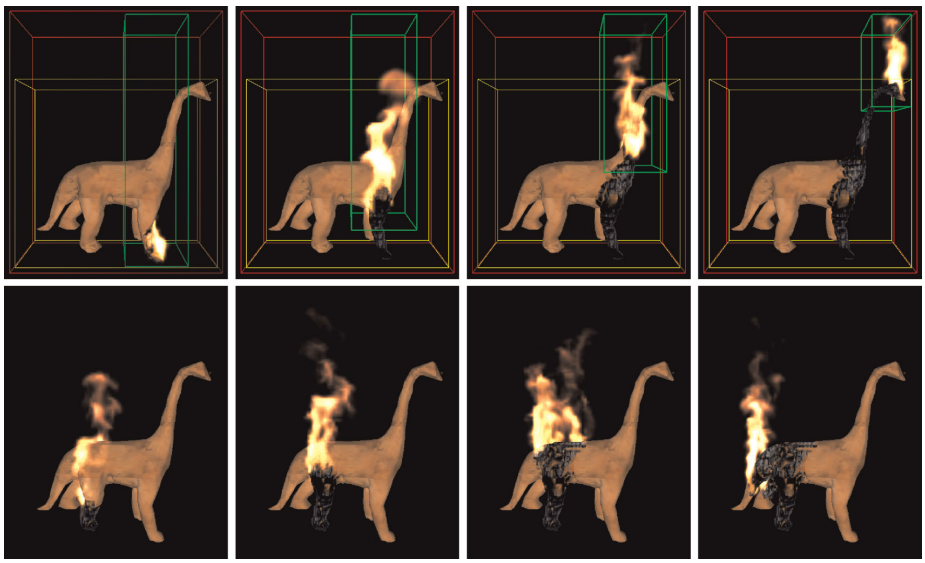


Figure 4: The top row shows the grid structure in the Zhu et al. (2011) burning simulation. The bottom row shows a burning simulation starting from a different ignition point. The images are taken from the Zhu et al. (2011) paper on solid combustion.

Melek and Keyser (2003, 2004). A volumetric grid structure allows for fluid simulations and heat distribution in the air and within the solid. A signed distance field defined on a volumetric grid can easily handle complex boundary deformations caused by material consumption. An additional distance field is introduced to create a temporary polygonal representation that allows for interactive rendering. Another multi-representation is introduced by Zhu et al. (2011). Their hybrid grid structure consists of a global grid, a local grid and a moving grid (see Figure 4). The global grid encompasses the whole simulation space and is used for the fire visualisation. The local grid is a volumetric data set representing the object and is used to reconstruct the surface at each frame. The moving grid is used for the fire front propagation. Both the fire front and object boundaries are tracked using the level set method, as in previous approaches. Liu et al. (2012a) subdivide the simulation space into a voxel structure and compute the heat distribution on the resulting vertices. Again, the level set method is used to track the interface represented as a signed distance field.

The methods described in this section, excluding the method proposed by Zhao et al. (2003), simulate the deformation of an object due to material consumption. In the case of burning thin shell objects the bending and crumbling of the object due to heat need also be considered.

2.6.2 Surface Mesh

Thin objects, such as sticks and paper, crumple and bend in addition to losing material and can be modelled using polygonal meshes. The burning of thin plate objects is considered by Losasso et al. (2006). The thin shell object is represented by a triangle surface mesh that is embedded in a parent simulation mesh as in Molino et al. (2005) (See section 2.3.1). This FFD-based approach has the advantage of keeping a detailed graphical representation while simplifying the computational complexity. The object's boundary is defined by the zero-isocontour of a level set function defined on the nodes of the simulation mesh. In order to model the material loss due to fire the level set is evolved on a background quadtree data structure. The results are then interpolated back to the simulations mesh. Adaptive remeshing is applied to account for the deformations caused by fire and to refine the boundary elements. Refining the boundary allows for more accuracy in these areas, while keeping larger elements for efficiency in others. Smoke and fire are simulated using an Eulerian grid-based approach. For this the authors introduce a two way solid-fluid coupling method that is able to combine their Eulerian fluid approach with their Lagrangian mesh-based approaches for solids, without compromising quality in either. This is particularly beneficial in a simulation that considers phase changes, such as solids turning into liquids and gases. Losasso et al. can only simulate the burning phenomena on thin-shell objects but they use a similar approach to simulate melting of volumetric objects which is explained in Section 2.7. They do not consider the crumpling and bending associated with thin shell objects burning.

In order to model these effects Melek and Keyser (2006) plus Liu et al. (2009) propose to use FFD. A lattice is placed around the burning object and the object's properties are mapped onto the lattice faces. A difference between two faces leads to deformation of the lattice, which in turn deforms the burning object. Liu et al. (2009) assign one of three states (burning, burned and unburned) to each of the lattice cells so that a difference in heat between two connected faces causes the imbalance that leads to the deformation of the proxy object. In this approach the bending dynamics depend merely on the heat distribution on the object's surface ignoring the physical composition of the object which effects both the burning and deformation processes.

Two examples are cloth and paper, which are both made of fibre. This physical structure needs to be modelled in order to accurately reproduce the deformation dynamics. Mass spring systems can be used to represent an object's fibrous structure. Larboulette et al. (2013) demonstrate this in their approach to sim-

ulate burning paper. The paper is modelled as a thin shell object which acts as the mass spring system. In their approach, the vertices of the surface mesh are the mass points and the connecting edges are the springs, which represent the paper's fibre. The mass spring system is coupled to a heat propagation solver, which consists of particles transferring energy onto the mass nodes. A change in heat on those nodes affects the length of the connected springs, which are cut or shortened in response. Jeong et al. (2011) and Melek and Keyser (2007) also used a mass spring system but combine it with FFD. The object's properties are mapped onto the edges of a proxy object. The proxy object itself acts as a mass spring system and deforming it warps the coordinate system of the simulation object. The idea is to use a simpler representation for the proxy than for the deformable object in order to simplify the deformation process induced by the mass spring system. Jeong et al. (2011) embed the object in a body-centred square (BCS), which acts as the mass-spring system. The nodes have mass and carry heat information. If the heat reaches the ignition point, the node's mass is decreased. A burned-out node is removed from the system and the mass-spring system needs to be remeshed. In addition to this, Jeong et al. maintain level set values at each lattice point in order to track the object's boundary.

The methods used to simulate burning objects represented by surface meshes focus mainly on thin objects such as matches and thin plates. Amarasinghe and Parberry (2011, 2013a) aim for real-time simulating of burning 3D objects for use in video games. As such, their method is designed for objects represented by polygonal surface meshes and the deformation due to melting is achieved by vertex displacement. The object is divided into uniform blocks, which are weighted according to the number of vertices they contain. If the weight of a block changes, it will rotate and thereby change the location of the vertices contained. The Amarasinghe and Parberry (2013a) approach does not only simulate material consumption, but is also able to simulate bending due to burning on volumetric objects. However, since they use surface models, they cannot reveal underlying layers and only model the burning of objects made out of homogeneous materials.

2.7 Melting and other phase transitions

Simulations concerned with phase transitions need to consider heat conduction in the scene and within the object, the changes in the object state from solid to liquid or vice versa, and the resulting object deformation and fluid dynamics. This section will consider methods that simulate the melting of ice, wax and

other viscous flows as well as the mixing of granular material with water. The section is divided into the different modelling techniques used to represent the deformable object.

2.7.1 Grid-based

Melting is the transition from solid state to a liquid state. As solid material turns into liquid, the shape of the solid object changes. Grid-based approaches permit the use of Eulerian approaches for the fluid dynamics and level set values can be stored at the cells to track the object's boundary. To model the deformations due to material loss, early methods chose a voxel representation for the ice object as it allows for a volumetric representation of the object and simple erosion of ice during melting. In order to simulate the transition from solid to liquid the heat conduction on the object needs to be modelled. The voxel structure can be used to solve the heat conduction by transferring heat between neighbouring voxels, as Fujishiro and Aoki (2001) demonstrate. To account for regelation (re-icing) Fujishiro and Aoki use cellular automata and map each cell to a voxel. A cell can hold amounts of ice and water and, if a cell holds both, then regelation occurs. Their method is able to simulate erosion of ice due to melting but does not consider simulating the meltwater produced by the melting effect. Similarly, Wei et al. (2003) also connect cellular automata to the voxel structure but use this to handle the heat conduction within the object. Cellular automata transition rules are used to transfer amounts of liquid between cells. When the temperature of a cell reaches a certain threshold, the state of the cell changes from solid to liquid. Liquid cells are affected by gravity and other spreading effects, which are also controlled by the transition rules. This allows them to simulate the melting and flowing of a variety of melting material with different viscosities. However, Wei et al. only consider viscous flow and are not able to simulate fluids of low viscosity, such as ice melting into water.

As grid-based approaches are constricted by the grid width, they are not ideal to model solid with high details and sharp corners. Therefore, Losasso et al. (2006) choose a Lagrangian mesh-based approach to represent the solids but combine it with an Eulerian grid-based approach for the fluid. This is done using dynamic mesh adaptation. Grid based approaches are an efficient way for simulating fluid dynamics whereas mesh-based approaches allow for more detailed representation of solids. Losasso et al. show that the two can be combined to simulate phase transitions from solid to liquid while keeping a high quality simulation for both processes. A body-centred tetrahedral lattice is constructed around the simulation mesh using red-green refinement. Then,

a level set function is defined on the nodes of the red-green simulation mesh. This allows them to leave the simulation mesh unchanged and instead evolve the level set. The results are then interpolated back to the simulation mesh. The meltwater is represented by a level set function with particles generated on the object's surface to avoid volume loss.

Another grid-based method to representing fluids is the volume-of-fluid (VOF), which is a colour function whose value denotes the volume fraction at each grid cell of a liquid. The Fujisawa and Miura (2007) simulation of ice melting is based on thermodynamics with thermal radiation using photon mapping. They represent the ice object as a VOF, which allows them to simulate phase changes by decreasing or increasing the function value. VOFs have the disadvantage that their values vary abruptly at the liquid boundary, which causes advection problems. Fujisawa and Miura solve this by using a constrained interpolation profile. Both the solid object and fluid body are represented by a VOF, which allows the simulation of phase changes by subtracting volume from one VOF and adding it to the other. They consider the flowing of melt water but are not able to simulate the thin film of water on a melting object, as they are constricted by the grid width.

Wang et al. (2012) demonstrated how a hierarchical lattice structure can be used to simulate the mixing of multiple liquids and melting solids. The simulation space is discretised into a lattice, called the global grid, where each cell is either gas, liquid or boundary and its state changes with the fluid motion. For each solid object a grid in local coordinates is constructed and placed into the global grid. The solid grid is used for the heat distribution and to hold the fraction of solid. A lattice-based representation of the solid simplifies the material erosion process and potential splitting of the object into pieces. Again, a level set method is used to track the object's boundary. Observed disadvantages of this grid-based approach is the fact that the object's surface needs to be extracted and that grids are not ideally suited to representing detailed, sharp edged surfaces. For the fluid simulation the lattice Boltzmann method (LBM) is chosen as it is easy to parallelise which enables faster processing using the GPU. Grid-based methods in combination with level sets can also be used to model the freezing phenomena. This is demonstrated by Kim et al. (2006) who introduce a physically-based method for icicle formation. A thin film version of the Stefan problem is solved using the level set method. As the level set method does not provide the resolution needed to track small scale features, such as ripples on icicles, the tracking of those is handled separately.

2.7.2 Particle and Point Based

Particles are a popular way of representing objects and fluids that undergo large deformations, because they do not need to maintain connectivity information. They can be used to represent both solids and fluids, which makes them a good choice when simulating phase changes. There have been a number of approaches concerned with simulating phase changes using particles. One way is to represent the fluid by a volumetric grid filled with particles that mark which cells are filled, called the marker-and-cell method (MAC). A solid object can be represented as a fluid with high viscosity using the MAC methodm see Carlson et al. (2002). The phase transition is then achieved by varying the viscosity over space and time according to temperature changes and applying the Navier Stokes equations to solve the fluid dynamics. However, it is not ideal for modelling ice melting, as the viscosity of water is low and the Carlson et al. (2002) method is intended for simulating liquids with high viscosity. Matsumura and Tsuruno (2005) consider natural convection that occurs in the air surrounding the ice. Two grids are used for the heat conduction in the air and the thermal energy transfer within the ice. If a particle's temperature reaches a certain threshold it then turns into water. Another application of the grid structure, with particles to simulate phase changes, is ice formation. In this case, the particle grid method can be used to achieve needle-shaped and egg-shaped air bubbles within the frozen object, as described by Im et al. (2013). The grid is able to handle water temperature, diffusion and exchange of dissolved air between the ice and water particles. A particle's temperature is interpolated from neighbouring grid cells and the grid cell is then updated using the particle's data. A signed distance function is used to generate the ice volume.

Grid-based methods used for simulating ice melting neglect the flow of thin water film on the melting ice, as the grid tends to be too coarse to handle such small features. Iwasaki et al. (2010) present an ice melting simulation that can handle the flow of melt water and water droplets on the object's surface and also the breaking up of ice pieces by modelling the melting object and water using particles. In addition to this, their particle simulation runs on the GPU, making use of efficient parallel algorithms to speed up the particle dynamics. Similarly, Lii and Wong (2014) concentrate on simulating the thin layer of water flow on the melting object. The exterior particles in their melting simulation carry an attribute called virtual water that accounts for the amount of water surrounding it. The virtual water is transferred between the exterior particles to simulate the water flow. The melting effect is achieved by gradually covering a fraction of the ice particle in virtual water, and when the amount of virtual water exceeds

a certain threshold, a water particle is created. A density function, similar to a signed distance function, is used to track the iso-surface of the fluid. Other particle methods concentrate on simulating melting metal, plastic, wax and lava. Paiva et al. (2006) use particles that represent small bodies of fluid, whose viscosity varies with temperature. Mass and volume of particles is interpolated between particles using smooth-particle hydrodynamics (SPH), which can also be used to simulate the mixing of granular materials with fluids, as per Lenaerts and Dutr (2009). Alternatively, particles can be subject to restoring forces which constrain them into a lattice structure as Wicke et al. (2006) shows. The particle lattice structure is used to model the solid objects. The liquid particles on the other hand are not influenced by elastic restoring forces and are therefore free to move. Phase changes can be simulated by weakening or removing the restoring forces. A strain computation is integrated in the fluid simulation to allow for phase transitions. Particle models, similar to grid-based approaches, are not ideal for a detailed surface appearance as a high particle density would be required, which in turn is computationally expensive. A low particle density could also prove problematic in terms of simulation accuracy.

Point-based approaches are an alternative surface representation to polygonal meshes and have the advantage that they do not need to maintain connectivity and topology information. This representation can be used to animate elastic, plastic and melting objects. Müller et al. (2004) represent the object boundaries by surface elements (surfels). Additional physical volume elements (phyxels) are used to represent the volume. Phyxels carry physical quantities such as location, density, velocity, strain, stress etc. The surfels are carried along with phyxels, such that the surface dynamically adapts to the deformation of the underlying volumetric model. The displacement vector of a surfel is approximated by its neighbouring phyxels using the moving least square (MLS) approximation, which is also used to compute strain and stress. Müller et al. choose this multi-representation to combine the ease of handling topology changes of implicit representation with the advantage of detailed surfaces that explicit representations possess.

2.7.3 Surface Mesh

For real-time simulation, an object, represented by a surface mesh, can be melted using a polygon folding technique. Amarasinghe and Parberry (2013b) concentrate on the real-time simulation of melting objects for use in video games, alongside their previous work on burning. As such, their method works with polygonal surface meshes using a polygonal folding method. The vertices of

the simulation mesh are categorized into “flow true” vertices (allowed to move), “fixed” vertices (not allowed to move), and, finally, “base point” vertices, which guide the deformation. The object is decomposed into a grid of body-aligned blocks. The weight of each block is determined by the number of particles it contains and empty boxes are discarded. Their simulation runs in real-time though at the cost of realism. Furthermore, they only consider a single heat source.

2.8 Decay, Rotting and Withering

Biological weathering is still relatively unexplored and little work has been done in the area of rotting and withering of organic objects. Fruit and other organic objects tend to consist of a number of different layers which decay in several different ways. Fruit, for example, consists of a flesh and skin layer. As the flesh layer shrinks due to water loss, the skin layer wrinkles or rots away, revealing the underlying flesh. Leaves on the other hand consist of a lamina, which makes up the body and a vein system. As the water in the leaf dehydrates, the leaf crumples and discolours. Hong et al. (2005) and Jeong et al. (2013) focus on the modelling and deformation of leaves. Hong et al. introduce a leaf modelling framework consisting of two parts. First, a three dimensional skeleton is created that represents the larger veins of the leaf. The second step involves generating a triangle surface mesh to model the leaf lamina. The deformation of the leaf is based on free form deformation. The skeleton acts as the proxy object which is bounded to the surface mesh in such a way that deforming the skeleton will deform the leaf surface itself. However, the authors do not consider simulating the actual weathering process that drives the deformation. Instead, the user is required to manually deform the skeletal structure in order to achieve crumpling and buckling effects. In contrast, Jeong et al. (2013) focus on the withering of leaves by modelling the osmotic water flow within the leaf. This allows them to model the bending and shrinking effects due to dehydration. They choose a doubled-layered structure model for the leaf representation; a Delaunay triangulation of the leaf surface and its underlying Voronoi diagram. The vertices of both layers represent the particles in a mass spring system, where the edges are the springs. Additional springs connect the triangle vertices with the neighbouring Voronoi vertices. The length of these connective springs represents the thickness of the leaf which affects the deformation. The mass of a particle is controlled by its water content. The water flow within the leaf is simulated by distributing water towards dehydrated particles until a water equilibrium is reached. The changes in water content in

particles lead to the shrinking or expanding of connecting springs. The double layered structure enables the complex wrinkling and curling effects of withering leaves due to differences in water content. A drawback of this method is that its self intersection method limits the time step and as such can not be used in real time applications.

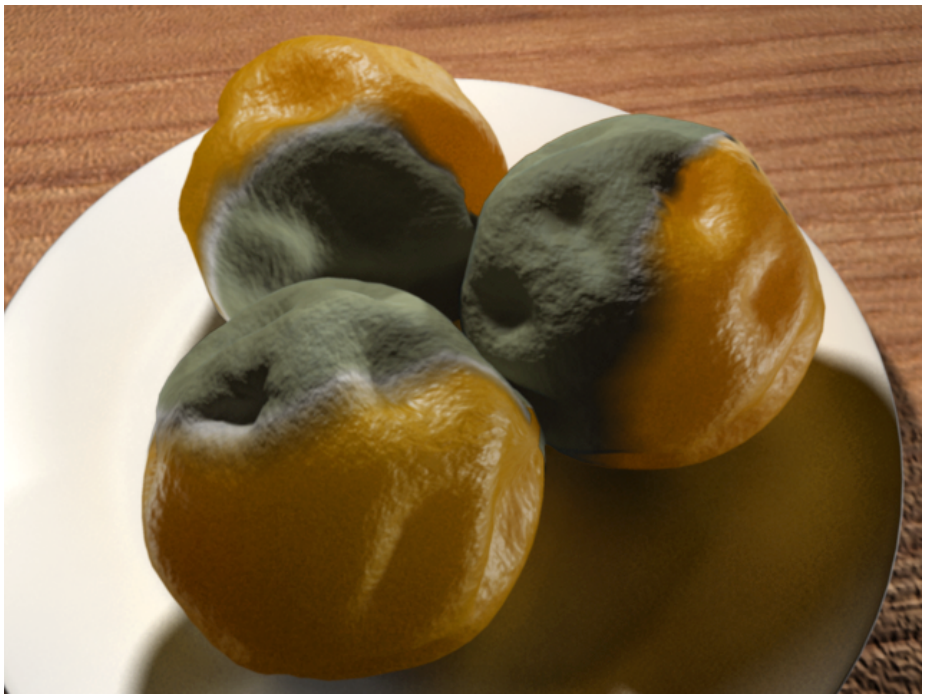
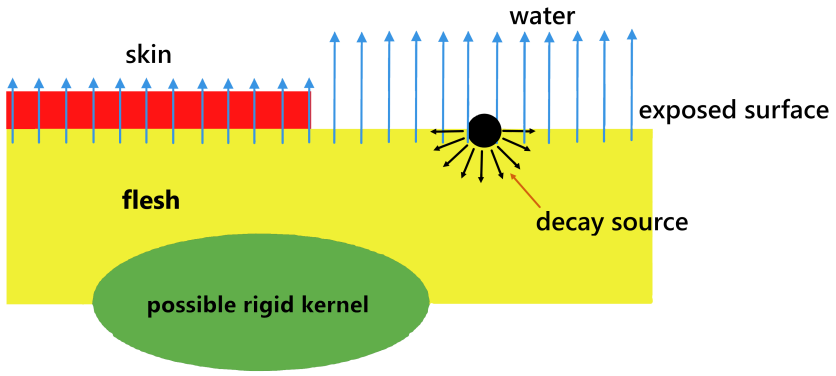
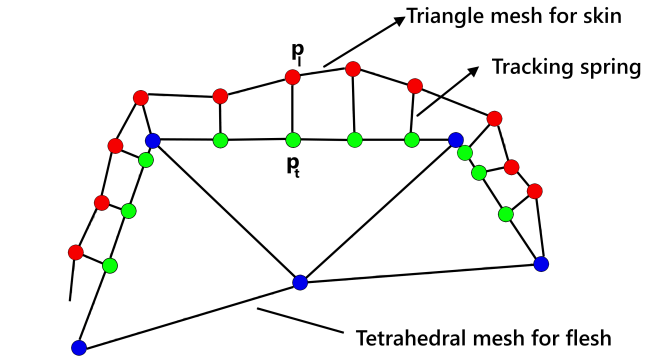


Figure 5: Simulation of decaying fruit Kider et al. (2011).

Fruit and other heterogeneous objects are made of different materials that are affected by their environment in different ways. Kider et al. (2011) and Liu et al. (2012b) simulate rotting of heterogeneous objects such as fruit. To accurately model the effects of ageing on different materials in the same object, a different representation is used for the skin versus the flesh of the fruit. Kider et al. (2011) model the flesh layer as a polygonal mesh representing a point mass system. The fruit skin is represented by a cloth like simulation and is constrained to the flesh layer. Texture maps are used to model the soft rot distribution on the object. Springs that are located at spots affected by soft rot are decreased in length, which leads to deformation of the internal volume. Water loss does also lead to a decrease in the springs' resting length. As the flesh layer deforms due to changes in the spring length, the skin is pulled along, which leads to its wrinkling behaviour. See Figure 5 for the results of this. Similarly, Liu and Fan (2015) use a mass spring approach to model morphology changes in fruit due to the



(a)



(b)

Figure 6: a-b) show the layered representation of a heterogeneous object used by the Liu et al. object withering simulation. The images were reproduced by the author. For the original see Liu et al. (2012b)

sunscald disease. The pressure changes caused by water loss are computed and used to update the mass displacement, which drive the deformation.

Liu et al. (2012b) use a similar object representation as Kider et al. (2011). Here the skin layer is also constrained to the flesh layer by elastic springs. However, they choose a tetrahedral representation for the internal flesh volume which comes with a few advantages over the parametrised method by Kider et al.. One advantage is that a tetrahedral representation can model the internal flesh which allows them to simulate the rotting process on cut fruit as well as full fruit. The dynamics of the internal flesh are dependent on the object's water content and can be computed using the finite element method on the vertices of the tetrahedral mesh, which is more physically accurate than the spring model used in the previous method. They also extend the simulation of rot spreading to a 3-dimensional model that allows the rot to spread into the flesh layer. A

limitation of their method is that the FEM is not able to generate the large scale curling one would observe in some cut fruit.

Similarly, Liu et al. (2015) are concerned with fruit dehydration, in particularly the morphological changes in core/shell fruits due to dehydration. Core fruits considered in their approach are jujubes, raisins and plums. The fruit itself is also represented in three layers, namely skin, fruit flesh and an internal hard stone/kernel. Their focus is on the buckling and wrinkling effects of the fruit skin. As with the Liu et al. (2012b) methods, a physically-based approach is used to solve the dehydration and flesh volume deformation using the FEM, where the deformation gradient is broken down into an elastic and dehydration induced part. A critical dehydration deformation factor is introduced, which determines when the fruit skin starts to wrinkle and buckle. This is computed by solving an eigenvalue problem involving the initial material and geometric stiffness. When the critical dehydration deformation factor reaches a specified threshold random buckles with different orientations are generated. The generation of the buckles are controlled by geometric and material parameters, such as skin thickness and stone/kernel size. Ring rot has also been considered in literature (Fan et al. (2013)), though only affecting the object's shading. Wood ageing has been considered by Yin et al. (2004), who model the erosion and cracking in ageing wood using a tetrahedral mesh. Each vertex has a value according to whether it lies inside or outside the object. Erosion is achieved by changing an inside vertex to an outside vertex, which reveals underlying vertices.

2.9 Vein Modelling and Graph Algorithms

Some decay processes, such as leaf drying (See Jeong et al. (2013) and Hong et al. (2005) described in the previous section) and livor mortis, involve simulating fluid motions in a vein network. In order to believably recreate blood and water flow in vessels, the vascular network need to be modelled appropriately. Runions et al. (2005) describe a process to model leaf venation patterns by iteratively applying geometry iterations to a set of points embedded in the leaf blade. The points represent vein nodes and hormone sources. Additionally, a vein connection graph is used to keep track of the vein formation and width. Vein development is driven by leaf growth and hormone sources. Leaf growth causes modification of hormone source placement and leaf development. Veins grow towards their nearest hormones source. For this, a Voronoi diagram is kept to allow for fast proximity testing.

In mammals, blood flows through a network of blood vessels. The vascular

network consist of veins, arteries and capillaries. The arteries and veins transport blood thorough the body into the capillaries which reach into the body tissues. The large main arteries and veins further into the interior of the body branch into vessels of decreasing size, until turning into the small capillaries reaching into the body's tissues. Blood flow in the vascular network is a complex process and still not fully understood. Modelling blood dynamics in the vascular system often involves numerical approximation of partial differential equations. Formaggia et al. (2010) describe how the blood vessel networks can be modelled from a set of images that were acquired using clinically available imaging techniques, such as X-ray or Magnetic Resonance Imaging (MRI). From these images a computational mesh is built. This requires a segmentation of the input images to identify the vascular structures from which the vessel walls are extracted. A surface mesh representing the vessel walls is generated using surface elements, such as triangles. The volume enclosed by the surface mesh is then divided into volume elements, which are usually tetrahedra or hexahedra. The blood flow can be simulated on the tetrahedral mesh by solving partial differential equations with the help of the FEM, as shown by Formaggia et al. (2010). See Antiga and Steinman (2017) for a tool that is able to model blood vessels from images using the segmentation method described, and Kirbas and Quek (2003) for a review of different vessel extraction techniques. Müller and Toro (2014), on the other hand, describe a method to model the human circular system (including the vascular network) as a one-dimensional computational model. A one-dimensional hyperbolic system is used to model the larger blood vessels, which is combined with a zero-dimensional model. See Boileau et al. (2015) for a discussion of different one-dimensional computational models used for haemodynamical simulations.

Computational simulations involving vascular modelling have helped research into the mechanisms of arrhythmia. Current research moves towards creating patient specific models for disease risk detection and drug testing as described by Quarteroni (2015). Similar to the methods described above, a model of a human body is produced from CT/MRI data. The image data is used to construct a tetrahedral volume mesh that acts as a reference model. The references model is then transferred to a patient's individual model on which the blood dynamics are simulated. The blood flow is simulated on a vascular graph that consists of nodes connected by edges that represent elastic tubes. The flow itself is expressed by viscous incompressible fluid flow on the graph's edges, solved using the FEM on the tetrahedral mesh. However, this method is reported to be too expensive to run in real-time. Gödde and Kurz (2001) describe a method for blood vessel growth and remodelling on a two-dimensional isometric grid that

considers the effects of pressure, flow, and velocity distribution in the network. Haemodynamic quantities are computed at the vascular network branches using methods from electrical engineering and graph theory. The nodal admittance matrix method and mass conservation are used to calculate blood flow in all nodes at the same time. Blood vessel growth and remodelling is regulated by shear stresses, and shear stresses are affected by blood viscosity.

Similarly, Reichold et al. (2009) simulate blood flow in the brain on a vascular graph inspired by the problem of electrical circuits. The vascular network is represented by vertices, and edges connecting pairs of vertices. Each vertex holds a blood pressure value. Differences in blood pressure between two connected vertices lead to blood flow through the connecting edge. The amount of flow is controlled by the size of the pressure difference. This results in a system of equations that are solved to obtain the pressure and flow values in the graph. In order to simulate oxygen extraction from the vessels into the brain tissue the vascular system is connected to a computational grid that models the brain tissue.

Aside from blood flow, graph algorithms have also been used to model water flow Price and Ostfeld (2014). For water flow problems, the network is represented by a directed graph. Bang-Jensen and Gutin (2008) describe a directed graph as a graph where all edges have an assigned direction, and flow on that edge can only occur in that direction. In a flow network, edges generally have a lower bound, flow capacity and cost assigned, which influence the flow behaviour. Alternatively, these values can also be assigned to the graph nodes instead. The flow is controlled by a flow function. A special case of a directional graph is the Directed Acyclic Graph (DAG). A Directed Acyclic Graph has no cycles, i.e. there is no valid path back to a node already visited. Boulos and Altman (1993) use a DAG to determine water quality parameters in a water distribution network. The acyclic nature of the graph allows for topological sorting of the graph nodes required for this problem. Flow in a graph can be described as a minimum cost flow problem, which is concerned with finding the cheapest way of transporting an amount of flow through the graph. In the Bang-Jensen and Gutin (2008) water flow method, the graph nodes represent water sources, junctions, water storage units and consumers, whereas the connecting edges represent pipes and pumps. If the max flow limitation between a source and target node is fulfilled, the successive shortest path algorithm is applied to solve a water balance problem. This will return a minimal operating cost for water flow. Other algorithms for solving minimum cost flow problems include, among others, cycle cancelling methods (see Klein (1967) and Goldberg and Tarjan (1989)) and the Out-of-kilter algorithm developed by Fulkerson (1961).

The vessel modelling methods described are concerned with a realistic modelling of vascular system, such that blood flow can be simulated for medical purposes. In these cases an exact modelling of the vascular system, and the blood flow within, is important. However, if the objective is visual appearance rather than physical correctness, simpler, less expensive methods can suffice. In the case of livor mortis, the behaviour of blood after death follows different dynamics than the vascular flow simulated in the methods described above. The main force of the fluid motion in a living body is the heart. After death, the heart no longer functions, and as such blood dynamics are controlled by external forces such as gravity. Nonetheless, considering graph algorithms to model blood flow after death might be a productive area of research.

2.10 Conclusion

This review of literature addresses the problem of simulating object morphology changes due to natural influences, such as erosion, corrosion, melting, burning, rotting and other processes. The changes an object undergoes depend on the natural phenomena involved, thus the different methods were categorised based on the natural phenomena that are behind the deformation, whilst at the same time paying particular attention to the modelling and deformation techniques used. Table 2.1 shows an overview of the methods used to model these phenomena and their effects on the objects. The different methods are categorized according to the modelling techniques used to represent the deformable object. The deformation technique and its level of change on the object is considered. Additionally, this section examines whether the described methods consider objects of a heterogeneous/layered make up.

2.10.1 Modelling and Deformation Conlucions

Looking at the different modelling and deformation techniques used for simulating object degeneration, I observed that the different deformation effects can be categorised into three different types. These are:

1. *Material loss:* Material is removed from the surface. This is the case in erosion, corrosion, melting or due to consumption by fire.
2. *Breaking:* This includes the ripping, fracturing, cracking and tearing of the object without loss of material.
3. *Whole body deformation:* This includes thin shell deformations, such as

Paper	Representation	Deformation	LoC	Phenomena	Hetero
Surface					
Günther et al. (2012)	Mesh	n/a	0	Corrosion	No
Chen et al. (2005)	Mesh	Vertex displacement	1	Corrosion	No
Mérillou et al. (2001)	Mesh	Vertex displacement	3	Corrosion	No
Bezin et al. (2010)	Mesh	Vertex displacement	1	Erosion	No
Amarasinghe and Parberry (2011, 2013a)	Mesh	Vertex displacement	3	Burning	No
Melek and Keyser (2006)	Mesh	FFD	3	Burning	No
Desbenoit et al. (2005); Müller et al. (2013)	Mesh	Geometry-based (prescoring)	3	Fracture	No
Amarasinghe and Parberry (2013b)	Mesh	Polygonal folding	3	Melting	No
Hong et al. (2005)	Mesh + skeleton (piecewise linear curve)	FFD	0	Witherin leaf	No
Jeong et al. (2013)	Mesh + voronoi diagram	Mass-spring	2	Witherin leaf	No
Chang and Shih (2003)	Mesh + layered thickness values	De-/increase thickness, moving 2 vertices	1	Corrosion	Yes
Valette et al. (2006)	Mesh + height map	Geometry-based (prescoring)	3	Cracks	No
Su et al. (2009)	Mesh + level set	Geometry-based (prescoring)	1	Cracks	No
Kider et al. (2011)	Mesh layers	Mass-spring	3	Fruit decay	Yes
Liu et al. (2009)	Thin shell	FFD	2	Burning	No
Pfaff et al. (2014)	Thin shell	FEM	2	Fracture/ tearing	No
Busaryev et al. (2013)	Thin shell layers	FEM	2	Fracture/ tearing	Yes
Larboulette et al. (2013)	Thin shell	Mass-spring	2	Burning	No
Gingold et al. (2004)	Thin shell	Geometry-based	2	Fracture	No
Losasso et al. (2006)	Thin shell + quadtree	FFD + level set	2	Burning	No
Molino et al. (2005)	Thin shell/tetrahedra + level set	FEM	3	Fracture	Yes
Bao et al. (2007)	Thin shell/tetrahedra + level set	FEM	3	Fracture	No
Jeong et al. (2011)	Thin shell+ BSC + level-set	Mass-spring + level set method	2	Burning	No
Bézin et al. (2014)	3-G-maps	Vertex displacement	3	Erosion	Yes
Height Maps					
Kelley et al. (1988); Musgrave et al. (1989); Chiba et al. (1998); Krištof et al. (2009); Sumner et al. (1999)	Height values	De-/increase height	1	Erosion	No
Onoue and Nishita (2003)	Height span map	De-/increase height	3	Erosion	No
Beneš and Forsbach (2001); Neidhold et al. (2005); Mei et al. (2007); Beneš (2007)	Layred height values	De-/increase height	1	Erosion	No
Nagashima (1998); Beneš and Forsbach (2002); Št'ava et al. (2008)	Layred height values	De-/increase height	1	Erosion	Yes
Dorsey and Hanrahan (1996)	Layred thickness values	De-/increase layer thickness	0	Patina	Yes
Volume					
Hirota et al. (2000)	Lattice (cubic)	Mass-spring	3	Cracks	No
Fujishiro and Aoki (2001)	Voxels	Mathematical Morphology	3	Melting	No
Wei et al. (2003)	Voxels	Cellular automata	3	Melting	No
Zhao et al. (2003)	Voxel grid	Chang voxel state	0	Burning	Yes
Beneš et al. (2006)	Voxel grid	Chang voxel state	3	Erosion	No
Beardall et al. (2007); Jones et al. (2010)	Voxel grid	Chang voxel state	3	Erosion	Yes
Fujisawa and Miura (2007)	VOF	Other	3	Melting	No
Zhu et al. (2011)	Multiple grids	Level set method	3	Burning	Yes
Melek and Keyser (2007)	Multiple grids	FFD + Mass-spring	2	Burning	No
Wang et al. (2012)	Multiple grids	Lattice Boltzmann Method (LBM)	3	Melting	No
Wojtan et al. (2007)	Multiple grids + Signed distance	Level set mehod (and FEM)	3	Corrosion	No
Melek and Keyser (2003, 2004, 2005)	Multiple grids + signed distance	Level set method	3	Burning	No
Dorsey et al. (1999)	Voxel layers + density function	FEM	1	Weathered stone	No
Liu et al. (2012a)	Signed distance	Level set method	3	Burning	No
Kim et al. (2006)	Signed distance	Level set method	3	Freezing	No
Aoki et al. (2004)	Tetrahedra	Mass-spring	3	Cracks	No
Liu et al. (2012b)	Tetrahedra + Mesh	FEM+Mass-spring	3	Fruit decay	Yes
Liu et al. (2015)	Tetrahedra + Mesh	FEM	3	Fruit decay	Yes
O'Brien and Hodgins (1999); O'Brien et al. (2002); Parker and O'Brien (2009)	Tetrahedra	FEM	3	Fracture	No
Hegemann et al. (2013)	Tetrahedra + level set	FEM	3	Fracture	No
Beneš et al. (2006)	Tetrahedra + octree	FFD + level set	3	Melting	No
Tychonievich and Jones (2010)	DDM	Vertex displacement	3	Erosion	Yes
Meshless					
Wicke et al. (2005)	Point-based	Mass-spring	2	Fracture	No
Pauly et al. (2005)	Point-based + surfels	Meshless	2	Fracture	No
Müller et al. (2004)	Point-based + surfels	Moving Least Square	3	Melting	No
Paiva et al. (2006); Iwasaki et al. (2010)	Particle	Meshless	3	Melting	No
Carlson et al. (2002); Matsumura and Tsuruno (2005)	Particles in grid	Meshless	3	Melting	No
Wicke et al. (2006)	Particles in closet packed structure	Meshless	3	Melting	No
Lii and Wong (2014)	Particles + density functions	Meshless	3	Melting	No
Im et al. (2013)	Particles + signed distance function	Meshless	3	Freezing	No

Table 2.1 Comparison of different methods to simulate morphology changes due to natural influences. LoC stands for Level of Changes, where 0 : texture/shading, 1 : surface, 2 : shell geometry and 3 : volume. 'Hetero' stands for heterogeneous.

wrinkling and volumetric deformations, such as the shrinking effects due to dehydration or rotting fruit. In comparison to 1, no material is removed and deformation can be anywhere in the object, i.e. it is not restricted to the surface.

Material loss can be caused by erosion, corrosion, melting and burning of volumetric and thin shell objects. The deformation technique used to account for material loss depends on the chosen object representation. A polygonal surface mesh can model thin shell objects, such as paper, or act as a boundary representation for a volumetric object. Inward vertex displacement has been used to implement material removal due to corrosion, erosion and burning of volumetric objects represented by polygonal surface meshes or Delaunay Deformable Models (DDM). In order to model material loss on thin shell objects due to consumption by fire, level set values are kept at lattice points to track the object's boundary. In terrain modelling, height fields, height span maps and stack-based terrains have been used as they are simple and efficient ways to model a terrain. Deformations due to erosion can be achieved by decreasing or increasing the height values at affected areas. Volumetric representations include voxels, grid-based and tetrahedral meshes. Voxels have been used to represent melting, burning and eroding objects. Material loss is achieved by removing voxels or changing their state/material. A (multiple) grid structure or tetrahedral mesh can hold signed distance values and other parameters that represent an object. This allows the use of the level set method to account for material loss due to burning, melting and corrosion. Meshless methods such as point-based methods and particle systems have been used to simulate the fluid flow of a melting object, which results in material removal from the melting solid.

Fracturing, cracking and tearing have been simulated on thin shell and volumetric objects. Volumetric or thin shell objects represented by a surface mesh can be fractured or cracked using prescoring. For physically-based fracturing and cracking simulations the object can be represented by a cubic lattice, tetrahedral mesh or polygonal surface mesh. Any of these representations can act as a mass-spring system or finite element discretisation. Fracturing and tearing effects of thin sheet objects have been achieved using the FEM to compute the fracture path. This has been extended to the fracturing of volumetric objects modelled by a tetrahedral mesh. Alternatively, mass-spring systems can form cracks by cutting connective springs that suffer under mechanical fatigue. Point-based methods have also been used for fracture simulation.

Whole body deformations include thin shell deformations and volumetric deformations. Thin shell deformations include crumbling and bending of burning

objects and withering leaves. These can be achieved by representing the thin-shell object as a mass-spring system. Alternatively, a proxy object can encompass an object and deform it using free form deformation (FFD). The proxy object can either act as a mass-spring system or its deformation is controlled by differences in parameters of neighbouring cells. Whole body deformations of volumetric objects, such as the shrinking of rotting fruit, can be achieved by using a mass-spring system or FEM. A layered surface mesh, tetrahedral mesh or grid structure can act as mass-spring system or finite element discretisation that allows for deformation anywhere within the object, not just its surface. As in surface-based approaches, a grid acting as a mass-spring system can be used as a proxy object of a burning simulation to account for the bending effects.

2.10.2 Stack-based Terrain

In the case of corrosion and erosion simulations, layered thickness or height map values (also called stack-based terrain) have been used to account for the different material layers. Stack-based terrains are able to model heterogeneous terrains and are able to model overhang and concave formations while at the same time retaining the benefits of height maps, i.e. those of being efficient and easy to manipulate. Despite this, erosion simulations using this data structure have not exploited it to its full potential. It would be interesting to see how the different erosion phenomena can be implemented on a stack-based terrain. I believe there is room for improvement on the approach of an eroding stack-based terrain, which could result in more visually appealing yet still efficient terrains.

2.10.3 General Weathering and Ageing

The Chen et al. (2005) and Günther et al. (2012) approaches are able to simulate a number weathering effects. However, these only effect an object's shading or small surface geometry and do not allow for major deformations. A general weathering and ageing simulation for volumetric objects is an interesting area of research that has not been exploited. Global weathering and ageing would be able to model interactions of different objects and phenomena. This would help the objects being perceived to reside in the same world rather than being isolated from each other.

2.10.4 Modelling Heterogeneous Objects that are subject to Degeneration

Throughout this chapter I have noted modelling techniques that can represent deformable objects and how representative they are of their real world counterparts. Objects tend to be of a heterogeneous make up which affects their appearance and deformation behaviour. Despite this, the majority of approaches assume objects that are homogeneous.

Heterogeneous thin shell objects have been modelled by multiple layers of triangle meshes in a thin sheet fracture and tearing simulation. For volumetric objects, multiple boundary representations have been used to represent areas with different material properties. One such model is the 3-G-map used to simulate erosion on heterogeneous objects. However, surface representations are not ideal to model heterogeneous volumetric objects as they lack information in the internal parts that are required for some physically-based simulations in order to obtain accurate results. This is especially the case for non-erosion based deformations, i.e. deformation types 2 and 3 above. Volumetric representations, such as grid structures and tetrahedral meshes, are good models on which to simulate all three types of deformations. Furthermore, they are able to represent heterogeneous materials. Yet, aside from erosion simulations, they have barely been applied to model heterogeneous objects deformed by natural influences (see Table 2.1). The current methods for simulating fracturing and cracking consider the volumetric object to be homogeneous but many objects are not homogeneous. A pencil for example is made of a solid pigment core surrounded by a layer of wood. A representation that accounts for a heterogeneous make up is important, as different materials in an object can greatly affect the fracture path and appearance. In the case of the pencil, the fracture path along the wood would be splintered, whereas the solid pigment core would look more smooth. Another effect is that the more fracture-resistant internal core might stop the fracture path from propagating or force it to change direction. Simulating geometric deformations of heterogeneous volumetric objects due to cracking, fracturing, burning and melting have been neglected despite their importance to realistically simulating object degeneration. The simulation of those phenomena on heterogeneous objects is an interesting subject for future research which has not been exploited to its fullest.

Some heterogeneous objects are made of layers which behave very differently to each other. Fruit usually consists of a skin and flesh layer. The skin layer is very thin in contrast to the volumetric flesh and as such a single representation might not be ideal to represent both layers. Kider et al. (2011); Liu et al. (2012b)

introduce a multi-representation to model fruit and similar objects that are subject to withering. In the Liu et al. approach a different representation was chosen for the skin (triangle mesh) and flesh (tetrahedral mesh). This allowed to treat the skin as cloth-like material for wrinkling and the internal part as a volumetric object that can shrink due to dehydration. The multi-representation allows the treatment of each layer in a different way but does not consider that rotting can lead to holes in the skin layer which would reveal internal rotting flesh. Beyond the Kider et al. (2011); Liu et al. (2012b) approaches on fruit rotting, the modelling of organic heterogeneous objects in such a way where they would have properties inspired by their real world representatives, leading to realistic and accurate morphological changes over time, has been somewhat neglected. Furthermore, the methods do not handle ripping or consumption of the skin due to rotting, which would reveal the internal flesh, although Liu et al. (2012b) did model fruit rotting on precut fruit. This might be due to the complex physical processes behind the decay phenomena, the complex make-up of the rotting body and the resulting processing power needed for the simulation, though the solution of such a problem could be of significant interest nonetheless, as it appears to form an intriguing area of research.

This review in literature has revealed a lack of work in the area of organic decomposition and in particular in simulating human body decomposition. Most work on ageing and weathering focuses on surface appearance changes which on their own are not sufficient to model the strong morphological changes a body undergoes after death. Geometrical changes due to fire consumption, melting, erosion etc. have been carried out, but the methods used are not easily transferable to organic decomposition. Works on fruit rotting and leaf withering are the closest work to human body decomposition that could be observed. Some of the methods and ideas can be utilised for simulating human body decomposition. The leaf withering simulation models the vein network in the leaves to simulate osmotic water flow. Human bodies have a vein structure to transport blood through the body, but in contrast to the water dynamics in leaves, blood flow is driven by the pumping of the heart. After death, when the heart stops beating, the blood dynamics are controlled mainly by gravity and resemble the osmotic water flow in the leaves to some degree. Human body decay can be caused by many different processes. Mummification by dessication is due to water evaporation from the body, which is similar to the fruit drying simulated by Liu et al. (2012b). Other human decay processes include putrefaction and autolysis which resemble the rotting of fruit. But in contrast to the fruit rotting described by Liu et al. (2012b) and Kider et al. (2011), putrefaction and autolysis happen inside the body and spread to the

outside as opposed to spreading from the surface to the inside, which is the case with fruit rotting. Therefore, fruit rotting approaches are not applicable to model putrefaction and autolysis. Human, and other vertebrate bodies, have a more complex make up than fruit or leaf and the processes involved in their decomposition tend to be more complex too. To summarise: although some of the ideas presented in this chapter can be re-purposed in simulating human body decomposition, they need also to be supplemented with new methods.

2.10.5 Applications

Morphology changes can be important indicators of an object's make up and the environment in which it exists. Simulating these effects is important in order to create realistic virtual worlds. Realism has become increasingly popular in the real-time entertainment industry. Most of the current methods for the simulation of object degeneration due to natural phenomena are too slow to be used in video games. Research into faster and more reliable methods, such that they can be used at run-time, is an ongoing subject of research. Methods that are developed with real-time performance in mind tend to achieve this at the cost of realism. An intriguing area of research is finding ways of storing the deformation effects efficiently and using those to reproduce the results in real-time. Similar methods have been applied to cracking and fracturing simulation (prescoring) and it will be interesting to see to what degree this can be extended to other phenomena.

Cultural heritage and archaeology are other additional areas in which object degeneration plays a major role. Ageing and weathering simulations can help us understand how objects age under different environmental conditions. Alternatively, the field of cultural heritage and archaeology might be interested in how an aged object used to look before it started degenerating. Inverse weathering is a fascinating topic and a promising area for future research. Simulating object degeneration for cultural heritage and archaeology would require physically plausible object representations that allow for accurate physical simulations. However, physically-based simulations tend to require a great deal of parameter tweaking, which makes them difficult to control. Therefore, more emphasis needs to be put on finding the right balance between usability and physically plausible simulations.

Chapter 3

Object Representation

Although there has been published work considering fruit rotting and leaf withering, there is currently no published output that aims to simulate human body decomposition involving livor mortis, mummification or putrefaction. As outlined in the previous chapter, most approaches to weathering and decomposition still assume objects that are homogeneous. There has been some work on the decomposition of objects that consider the object to be of a more organic make up. These are the works on fruit rotting by Kider et al. (2011) Liu et al. (2012b), and leaf withering by Jeong et al. (2013).

As fruit is made of different layers, similar to a human body, a layered model is used to represent the object. The fruit's skin can be represented by a surface mesh that acts as a mass-spring system which can be used to model the wrinkling deformations. For the internal flesh Kider et al. (2011) choose a similar polygonal representation that is deformed by a mass-spring system whereas Liu et al. (2012b) choose a tetrahedral mesh whose vertices function as a finite element discretisation. Texture maps that hold nutrient and soft rot information on the object's surface can be used for a reaction diffuse model to guide fungal growths Kider et al. (2011). Jeong et al. (2013) focuses on modelling withering leaves caused by dehydration of the leaf. Water in the leaf flows through the leaf's veins and drives the deformation and discolouration of the leaf by osmotic water flow. A layered model of a triangle mesh and underlying Voronoi diagram is used to represent the leaf, where the edges are the leaf's veins and the vertices hold information on water content and solute concentration required for the osmotic flow simulation. The water flow in the leaf's nodes changes both the morphology and shading of the leaf. Although the blood dynamics in the human body are driven by different processes, the network of blood veins

somewhat resembles the network of water veins in the leaf. The aforementioned method only considers thin shell objects. For the human body model a volumetric representation would be required. The approach of using the edges of the mesh to mimic a vein network can be re-purposed to mimic the blood veins in the human body model.

In computer graphics and simulations it is common to use different representations of the same object for different processes within the framework. For physically based simulation there are advantages to using different representations for the simulation and visualisation parts of the application. Often a third representation for collision detection and handling is used. This allows for the most advantageous (e.g. in terms of speed or visual quality) representation to be used for each process. For physically-based simulations a volumetric representation might be required to achieve good simulation results. However, volumetric representations are not ideal for rendering, for which a triangle mesh is much more efficient. Both triangles and volumetric meshes are inefficient for collision detection. Making use of spatial data structures to represent the object, such as octrees or spatial hashing (See Section 3.3), allows for faster collision detection.

The human body is made of different components that are affected by decomposition in a variety of ways. Flesh decomposes at a higher rate than bones. Skin tends to wrinkle as the internal flesh decays and turns leathery, see Dettmeyer et al. (2013). As such, a representation is needed that is able to model the internal components, such as flesh and bones as well as the skin layer. In Section 2.10.1 the different object representations for three different deformations (material loss, breaking, whole body deformation) were discussed. The human body decomposition fits in the *Whole body deformation* category. A representation for a layered object under this deformation category was described by Liu et al. (2012b). A tetrahedral volume mesh was used to represent the fruit flesh volume and a triangle surface mesh to represent the skin. A similar object representation can be used to represent human body parts.

For this project different representations for the objects are used for the simulation, rendering and collision parts of the simulations. These are described in the following sections.

3.1 Simulation Representation

The human body is a heterogeneous mass where each component responds differently to the decay process. In the case of mummification, the flesh parts (i.e. muscle and fat layers) shrink considerably, whereas the bones do not deform at

all. The body's skin also undergoes some structural changes, such as shrinking and wrinkling. This also holds true in the case of livor mortis, which is caused by post mortem blood dynamics. Blood is transported through the vascular system, which does not flow through the bones nor does it reach the outer layer of the skin (epidermis). Differentiating between the different materials (or layers) is therefore essential when simulating the effects of livor mortis and mummification by dessication. Furthermore, livor mortis and mummification are body internal processes that affect the body's appearance from the inside. In order to simulate these processes a model that is able to represent the volumetric interior of the body's flesh and bone parts is required. In contrast to this, the object's skin is a thin structure that shows different deformation behaviours to the volume, such as wrinkling and buckling. It is therefore appropriate to use a thin shell representation for the skin that is able to model these deformations.

The related problem of fruit rotting was discussed in the literature review, found in the previous section. Like human bodies, fruit consists of distinct layers that act differently when dehydrated. These layers include the fruit flesh, the fruit skin and a hard solid kernel. This requires similar consideration of the object representation as for the human body mentioned above. Liu et al. (2012b) chose to model the fruit's internal volume layers (fruit flesh and kernel) with a tetrahedral volume mesh and the skin layer with a triangle surface mesh. Both layers are connected by tracking springs that connect a skin node to the underlying tetrahedral boundary face. The object representation used to represent the human body in the human body decomposition simulations is based on this representation of the Liu et al. (2012b) model for fruit. The skin is represented by a triangle surface mesh and the internal components, such as flesh and bones, are represented by a surface aligned tetrahedral mesh. The two layers are connected by tracking constraints, similar to the springs in the Liu et al. (2012b) model. For livor mortis and mummification the body is considered to be made of bones, flesh and skin, though a more complicated body with organs is also possible with this model.

Tetrahedral meshes have proven successful in modelling human tissue for interactive tissue surgery simulators, such a the ones described by Bro-Nielsen and Cotin (1996); Cotin et al. (1999); Zhang et al. (2010); Courtecuisse et al. (2014). They allow the definition of the interior of a volume, which is not possible with surface meshes. Furthermore, they can act as finite element descretisation for physically based simulation based on continuum mechanics. This will be shown to be convenient for simulating the humidity diffusion and volume deformation for the mummification simulation described in Chapter 6. Similar as the triangle edges in the drying leaves simulation by Jeong et al. (2013), the edges of the

tetrahedralisation are used to mimic the vascular system in the human body on which the blood dynamics for the livor mortis simulation are modelled.

Another way of representing the interior of a volume is with voxel structures. As discussed in the literature review, voxel and grid structures have been used to model objects that are subject to burning, melting and erosion. A summary of this was provided in Section 2.10. An advantage of voxel and grid structures is that that they can model the object and its surrounding in a single representation, as has been done with terrain modelling (see Beneš et al. (2006)) and burning objects (see Liu et al. (2012a)). This allows for water flow, sediment transportation and heat transfer to be modelled around and inside the object. The loss of material driven by these phenomena can be achieved by removing voxels or changing their state from material to air. Material removal is not required for surface weathering, livor mortis or mummification, which makes this advantage of voxel structures less important. Material removal could be an additional feature for surface weathering, but small changes to the object's surface caused by material removal can be achieved by the displacement of surface vertices instead.

Tetrahedral representations have three major advantages to voxel representation for this specific application. Firstly, in contrast to a voxel representation, a tetrahedral mesh can be simply aligned with a triangle surface mesh of the same overall geometry. As the human skin is modelled by a triangle mesh, this is an important advantage. This also simplifies the construction of tracking constraints between the skin and tetrahedral layers, because the tetrahedral boundary faces and the triangle mesh have nearly matching geometry. Secondly, tetrahedral meshes can consist of tetrahedra of varying volume. This often leads to tetrahedra of small volumes close to the boundary and larger elements in the interior, allowing for more detailed representation close to the object's surface and a more efficient representation on the inside. Furthermore, the network of edges from such a mesh resembles the layout of the vascular system, with fine blood vessel close to the skin and larger ones inside the body. This proofs particularly useful in the livor mortis simulation, where the blood dynamics are simulated on the tetrahedral edges. Thirdly, the irregularity of a tetrahedral mesh has a beneficial side effect in the livor mortis simulation. It allows the recreation of the patchy appearance of early livor mortis described further in Chapter 5. Using the tetrahedral edges to model the blood dynamics after death can have the disadvantage of leading to sharp edges along blood accumulation boundaries, particularly with a coarse tetrahedralisation at the object's surface. This can be mitigated with a more detailed tetrahedralisation at the object's surface or by using a texture map to model finer blood vessels on the object's

surface (see blood vessel texture map in Section 3.2).

The advantage of voxel and grid structures in modelling the area around the object, as well as the object itself, could benefit the livor mortis and mummification simulations. The surrounding grid could be used to simulate air circulation and heat transfer around the object. Heat transfer and air circulation would influence dehydration and livor mortis. Lower temperature can slow livor mortis, high temperature and good air circulation benefit mummification. However, this does not require the object itself to be represented as a voxel structure. Instead the environment around the object can be subdivided into a grid or voxel representation that is able to interact with the boundary of the object. For this project temperature and air circulation are not considered. However, implementation of this with a grid or voxel structure might be worth considering for future work.

The skin is represented by a triangle mesh acting as a thin shell object. Representing the skin as a surface mesh has a number of advantages. Firstly, it can be used as cloth-like simulation for wrinkling dynamics as shown in Kider et al. (2011); Liu et al. (2012b). Furthermore, modelling the internal volume with a tetrahedral mesh allows the tetrahedral boundary to be aligned with the skin layers. This simplifies the construction of the tracking constraints connecting the two layers. The tracking constraints drive the skin deformations caused by the shrinkage of the underlying volume mesh. The deformation of the skin mesh requires information on the moisture content at each vertex. The surface mesh receives this information from the underlying tetrahedral mesh through tracking constraints. There are a variety of methods for simulating cloth dynamics, such as mass spring systems, FEM and position based dynamics that all work well on triangle meshes. For this project, position based dynamics are used to solve the skin deformations, as they allow for direct control over the vertices' positions and wrinkle formation. Polygonal representations are chosen for both the volume and skin representation, as there are fast and stable algorithms for rigid and deformable body dynamics on polygon meshes. Furthermore, polygonal meshes (in particular triangle meshes) are the most popular object representation in video games.

The object model is constructed as follows. As input, the simulation takes a triangle mesh representing the skin and further triangle meshes representing additional internal layers (i.e. flesh and bones), as shown in Figure 7a. The layers representing the boundaries of the volume components are then used to construct the tetrahedral mesh representing the internal structure. The tetrahedral meshes used in this doctoral thesis were constructed using the free tetrahedralisation tool TEN24-Digital-Capture (2016). The triangle mesh representing

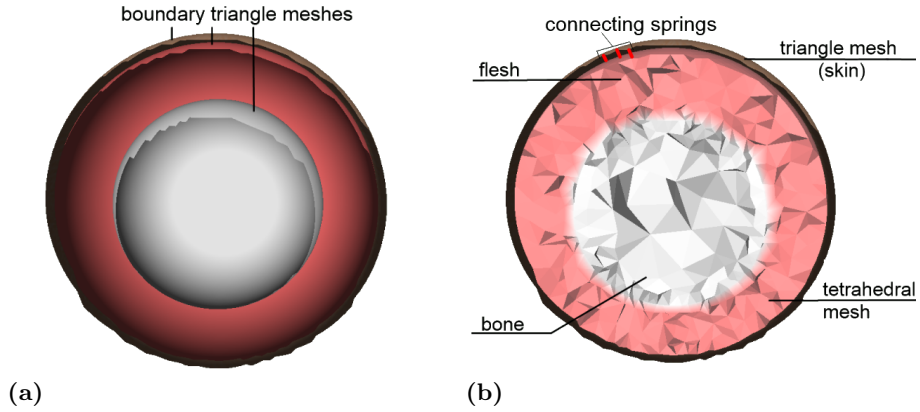


Figure 7: a) input: A set of triangle meshes representing the boundaries of the object's layers. b) simulation mesh: The model consists of a skin layer (triangle mesh) and internal bone plus flesh layers that are represented by a tetrahedral mesh. The two layers are connected by springs.

the object's outer boundary is used as a thin shell representation of the skin. This can be a higher resolution mesh than the surface mesh representing the boundary of the flesh layer, but needs to have the same overall shape. Figure 7b shows an object model representing skin, flesh and bone layers.

The surface weathering approach only affects the surface of the object. As such, only the surface triangle mesh is used for the surface weathering. The other two simulations, on the other hand, are built on the tetrahedral volume mesh. For the livor mortis simulation the tetrahedral nodes hold simulation parameters such as blood capacity, haemoglobin content and oxygen saturation. This permits different materials to be represented by varying these simulation parameters. Bones are differentiated from flesh by assigning zero blood capacity to all bone nodes, thus preventing blood from moving through the bone volume. The edges of the tetrahedralisation act as the veins of the human body on which to simulate the post mortem blood dynamics. For the mummification simulation a finite element discretisation is used to model both moisture diffusion and resulting deformation of the flesh volume. To achieve this, moisture content information is assigned to the tetrahedral vertices. Similar to the livor mortis simulation, flesh and bone materials are differentiated by the moisture content assigned to their vertices. Bone vertices hold no moisture whereas flesh nodes hold the percentage of moisture saturation. For the deformation dynamics the bones vertices are fixed, whereas the flesh vertices are free to move.

Tracking constraints connect each node of the triangle mesh to a point on the boundary of the tetrahedral mesh. These points are called tracking points. Each point is found by identifying the closest point of a vertex of the triangle

mesh to the boundary of the volume mesh. Each tracking point is expressed in term of the vertices of the boundary face on which it resides using barycentric coordinates. The tracking forces connecting the skin layer to the tetrahedral volume mesh will exert pulling forces on the skin mesh as the volume shrinks. The strength of the tracking constraints are affected by the moisture content at the tracking point. Position based dynamics can than be used to correct the skin mesh itself. The edges of the triangle mesh build a set of stretching and bending constraints. The tracking constraints connecting the two layers are another set of constraints. Projecting these constraints onto the vertices of the triangle mesh allows simple position based dynamics for skin wrinkling effects.

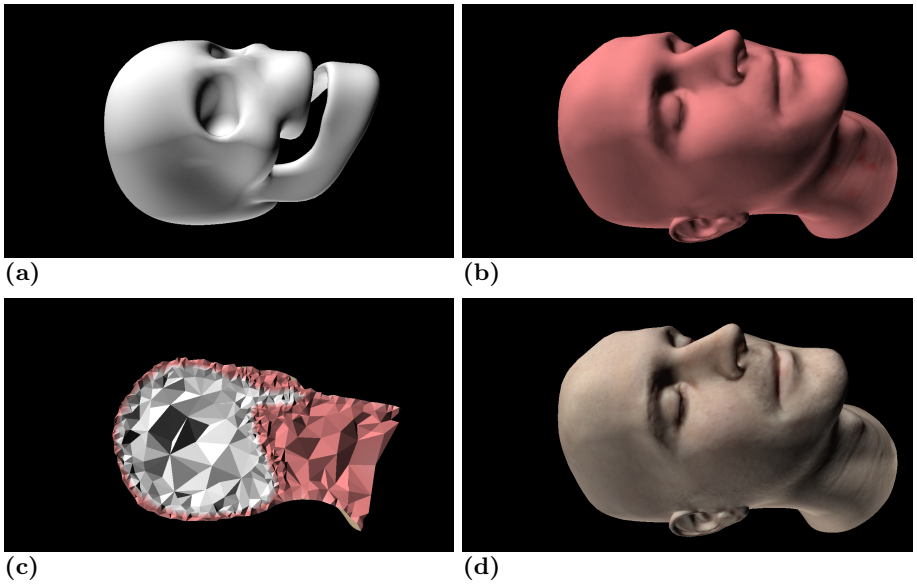


Figure 8: The first row shows the input triangle meshes. a) is the internal bone structure and b) the flesh layer. The resulting tetrahedral mesh is shown in c). d) shows the texture mapped triangle mesh representing the skin layer.

3.2 Visualisation and Rendering Representation

In many applications, and, in particular, in computer games, the three-dimensional object tend to be represented by polygon meshes (e.g. triangle or quadrilateral meshes), which represent the object's surface. These have many advantages, particularly in real-time applications as they are faster to render than other representation due to their suitability for scanline rendering and support of texture maps. Of those, triangles are the simplest polygon, and are also the main rendering primitive supported by graphics cards. Furthermore, triangle

meshes are able to represent both thin shapes and solid objects and are necessarily planar. Diffuse texture maps and normal/bump maps can be used to add additional detail on the triangles surface.

A high detail triangle mesh is chosen as the representation of the body's outer layer. For the human body decomposition simulation the triangle mesh represents the outer skin layer, called the epidermis. For the surface weathering simulation the triangle mesh represents the outer layer of the object, which can be metal, wood or any material defined be the user. The support of texture maps has significant rendering advantages as it allows the use of texture maps to specify skin reflectance and transmittance properties, as well as high surface details.

A number of texture maps are used to represent the skin's make up and reflectance properties for each object. These include;

- normal map
- epidermis reflectance/diffuse map
- melanin distribution map
- (optional) blood vessel texture map
- material maps/atlas

The normal or bump map specifies varying normals on a triangle surface. This allows for more detailed light reflection, such as pores and fine wrinkles on the otherwise flat triangle surface. The epidermis reflectance map specifies the skin colour without haemoglobin contribution over the object's surface. In the case that the object's surface is not skin, a diffuse map specifying the object's colour is used. The melanin distribution map specifies the amount of the chromophore melanin in the body's epidermis layer. This is used to compute the light absorption of light by melanin when computing the haemoglobin contribution to the skin colouration. An optional blood vessel texture can be used to model the effects of small blood vessels in the dermis layer. It could also be used to model the superficial veins that can sometimes be observed on the inside of the wrist or back of the hand. An additional texture, called the blood colour look-up texture, is globally defined. This texture acts as an atlas for the dermis colouration. The skin layer receives per vertex haematocrit, oxygen saturation and moisture content information from the tetrahedral mesh using the tracking constraints. These values are then used to look-up the dermis colour in the blood colour look up texture. The material maps are used for the surface weathering simulation. Each of the four RGBA channels of the texture map specifies one material, such as metal, moisture or rust. The amount of each material present at each point

on the object's surface is recorded in the dedicated colour channel. If more than four materials are needed, additional texture maps are used.

This set-up of texture maps allows for a variety of different render effects, either by simple alpha mapping, as used by the surface weathering approach in Chapter 4, or more complex skin shading approaches with subsurface scattering, as used for the livor mortis (Chapter 5) and mummification simulations (Chapter 6). The rendering approach used for the livor mortis and mummification simulation can be described as layered rendering. The skin is represented in two layers, called the epidermis (outer layer) and dermis (inner layer). Each is rendered in an independent render pass before being combined. The epidermis render pass uses the epidermis diffuse map whereas the dermis uses the per vertex blood and moisture information to look up the colour from the global blood colour look up texture. In the combination of the epidermis and dermis render results the melanin map acts an absorption layer.

3.3 Collision Representation

For the surface weathering simulation the octree data structure was shown to be a good choice for collision detection, due to its fast ray-triangle intersection abilities. For this, the space the objects occupy is partitioned into an octree structure. However, the creation of an octree was very slow for high detail surface meshes. This is not problematic in the case of the surface weathering approach nor the livor mortis simulation, as the geometry of the object is not changed during these, which allows the octree to be constructed off-line. However, the surface mesh does undergo deformation during the mummification simulation. The octree partitioning structure was replaced with a hash table representation of the object, called spatial hashing Müller et al. (2003). Creating a spatial hash table for the object is quick and simple. The object's space is described by a three dimensional grid. Collisions can then be detected by simply finding the starting and, if it is known, the ending position of a ray within the object's grid and then walking along the ray, testing each cell along its path that contains triangles. Spatial hashing can be used for collision and self-collision. It works well with scenes where primitives have similar sizes. The collision object is specified in the following Table 3.1

(min_x, min_y, min_z)	minimum values of object bounding box
(d_x, d_y, d_z)	dimension of object bounding box
h	grid spacing
(n_x, n_y, n_z)	dimension of grid
$n = n_x \times n_y \times n_z$	number buckets
$\{\boldsymbol{B}_1, ..., \boldsymbol{B}_n\}$	list of buckets making up the grid
$H(i, j, k) : \mathbb{N}^3 \rightarrow \mathbb{N}$,	hash function, where
	$i \in [0...n_x]$, $j \in [0...n_y]$, $k \in [0...n_z]$

Table 3.1 The attributes specifying a collision object

3.3.1 Construction of the Spatial Hash Table

The spatial hash table is created by first finding a bounding box for the object. For this the minimum and maximum coordinates min_x, min_y and min_z and max_x, max_y and max_z are established. From these the dimension of the bounding box can be computed $(d_x, d_y, d_z) = (max_x - min_x, max_y - min_y, max_z - min_z)$. Then a grid spacing h is chosen within the bounding box. In this project the grid spacing was chosen to equal the average size of the primitives making up the object. The grid spacing is then used to subdivide the bounding box into a regular three-dimensional grid, where each cell is of dimension $h \times h \times h$. The grid then has dimension $n_x \times n_y \times n_z$, where $(n_x, n_y, n_z) = (\lfloor \frac{d_x}{h} \rfloor + 1, \lfloor \frac{d_y}{h} \rfloor + 1, \lfloor \frac{d_z}{h} \rfloor + 1)$. Each grid cell is represented by a bucket $\boldsymbol{B}_h$, which holds the vertices of all primitives whose bounding box intersects the grid cell. A bucket is accessed using the hash function:

$$H(i_x, i_y, i_z) = (i_x * n_y + i_y) * n_z + i_z \tag{3.1}$$

where i_x, i_y, and i_z are the grid indices. Then the bucket at grid location (i_x, i_y, i_z) is $\boldsymbol{B}_{H(i_x, i_y, i_z)}$.

After constructing the grid structure, the index to each triangle/tetrahedron is inserted into all grid cells it intersects. In practice, an axis-aligned bounding box is placed around the primitive, and its index is inserted into all grid cells its bounding box intersects. This makes insertion and deletion of primitives very quick. In order to insert a triangle the grid coordinates (i_x, i_y and i_z) for all vertices of the primitive are found as follows

$$(i_x, i_y, i_z) = (\lfloor \frac{\boldsymbol{p}_x - min_x}{h} \rfloor, \lfloor \frac{\boldsymbol{p}_y - min_y}{h} \rfloor, \lfloor \frac{\boldsymbol{p}_z - min_z}{h} \rfloor) \tag{3.2}$$

where $\boldsymbol{p}$ is the position coordinate of a vertex. From these the minimum and maximum grid coordinates i_x^{min}, i_y^{min} and i_z^{min} and i_x^{max}, i_y^{max} and i_z^{max} are determined. The eight different combinations of the minimum and maximum coordinates are the eight vertices of the primitive's bounding box. The primitive is then inserted into all grid cells between (and including) i_x^{min} and i_x^{max}, i_y^{min} and i_y^{max}, and i_z^{min} and i_z^{max}. Algorithm 1 shows the primitive insertion routine. Line 1 to 2 initialise the minimum and maximum grid coordinates. Then the grid coordinates for each vertex are found in line 4. Lines 5 to 19 are used to find the minimum and maximum coordinates. In lines 21 to 28 the minimum and maximum coordinates are used to walk through the grid. At each cell the hash key for each bucket is computed (line 24) and the primitive ID is inserted in the bucket with the matching hash key in line 25. Note that the insertion of a new primitive into the hash table has complexity O(1).

Algorithm 1 Insertion of primitives into the hash table

```
1:  i_x^min := nx, i_y^min := ny, i_z^min := nz
2:  i_x^max := 0, i_y^max := 0, i_z^max := 0
3:  for all vertices p do
4:      (i_x, i_y, i_z) := (⌊(p_x − min_x)/h⌋, ⌊(p_y − min_y)/h⌋, ⌊(p_z − min_z)/h⌋)
5:      if i_x < i_x^min then
6:          i_x^min := i_x
7:      else if i_x > i_x^max then
8:          i_x^max := i_x
9:      end if
10:     if i_y < i_y^min then
11:         i_y^min := i_y
12:     else if i_y > i_y^max then
13:         i_y^max := i_y
14:     end if
15:     if i_z < i_z^min then
16:         i_z^min := i_z
17:     else if i_z > i_z^max then
18:         i_z^max := i_z
19:     end if
20: end for
21: for i_x := i_x^min to i_x^max do
22:     for i_y := i_y^min to i_y^max do
23:         for i_z := i_z^min to i_z^max do
24:             key := (i_x · ny + i_y) · nz + i_z
25:             PutInBucket(key)
26:         end for
27:     end for
28: end for
```

3.3.2 Collision Detection with Spatial Hashing

For collision detection each bounding box of an object in the scene is tested for intersection using a ray-cube intersection test. If the ray intersects the bounding box, then the object's spatial Hash table is used to compute the exact intersection point on the object's surface.

A line is represented by its origin $\boldsymbol{o} = (o_x, o_y, o_z)$ and direction $\boldsymbol{v} = (v_x, v_y, v_z)$. The direction is computed by subtracting the lines end position from the origin. A voxel traversal algorithm is used to walk along the line $\boldsymbol{o} + t\boldsymbol{v}$ to find all grid cells it passes through. For this project a voxel traversal algorithm based on 3D Digital Differential Analyzer (3D-DDA) proposed by Amanatides and Woo (1987) is used. Another possible method that can be used to traverse along the line is a 3D Bresenham Algorithm. However, in contrast to the 3D-DDA based voxel traversal algorithm, the Bresenham Algorithm does not guarantee that all voxels along a line are visited, which can lead to small fractions of an object being missed. The 3D-DDA approach by Amanatides and Woo (1987) is therefore more appropriate for this problem.

First the grid coordinates of the starting and end position of the line are computed using Equation 3.2. This gives the grid coordinates (i_x, i_y, i_z) and $(i_x^{max}, i_y^{max}, i_z^{max})$. These coordinates are used to access the buckets containing the surface indices using the hash function described in Equation 3.1. The hash function will return a key to the bucket holding all faces intersecting that grid location. The starting point and other points along the line are likely to be outside the defined grid space of the object. This is not an issue for spatial hashing as the hash function would simply return a key to an empty bucket. Each element in the bucket is tested for an intersection using ray-primitive intersection methods. In this case the primitives are triangles and hence a simple ray-triangle intersection test is performed. If there are intersections within that bucket, the primitive with the closest intersection point is chosen. If an intersection was found in the current bucket, the voxel traversal algorithm is stopped and the intersection information, i.e. the primitive's ID, is returned. If no primitive is found the function returns -1. This is described in Algorithm 2 below, which uses 3D-DDA. Collision detection with spatial hashing can be done to all neighbours of a point in constant time.

The values t_x, t_y and t_z describe the maximum distance one can travel along the line without exiting the current grid cell. dx, dy and dz are used to move along the line in object space and (x_{inc}, y_{inc}, z_{inc}) are used to step into the next grid cell along the line. The loop (see lines 11 to 33) runs until the end of the line is reached or until an intersection is found (see lines 29-32).

Algorithm 2 collision detection using the spatial hashing method with 3D-DDA. A line is represented by its origin $\boldsymbol{o}$ and direction $\boldsymbol{v}$.

1: **initialisation:**
2: $t_x := ((i_x + 1) \cdot h - o_x) \cdot \frac{1}{v_x}$
3: $t_y := ((i_y + 1) \cdot h - o_y) \cdot \frac{1}{v_y}$
4: $t_z := ((i_z + 1) \cdot h - o_z) \cdot \frac{1}{v_z}$
5: $x_{inc} := 1$ **if** $v_x > 0$ **else** $x_{inc} := -1$
6: $y_{inc} := 1$ **if** $v_y > 0$ **else** $y_{inc} := -1$
7: $z_{inc} := 1$ **if** $v_z > 0$ **else** $z_{inc} := -1$
8: $dx := h \cdot x_{inc} \cdot \frac{1}{v_x}$
9: $dy := h \cdot y_{inc} \cdot \frac{1}{v_y}$
10: $dz := h \cdot z_{inc} \cdot \frac{1}{v_z}$
11: **while** $i_x < i_x^{max}$ **and** $i_y < i_y^{max}$ **and** $i_z < i_z^{max}$ **do**
12: **if** $t_x < t_y$ **then**
13: **if** $t_x < t_z$ **then**
14: $i_x = i_x + x_{inc}$
15: $t_x = t_x + dx$
16: **else**
17: $i_z = i_z + z_{inc}$
18: $t_z = t_z + dz$
19: **end if**
20: **else**
21: **if** $t_y < t_z$ **then**
22: $i_y = i_y + y_{inc}$
23: $t_y = t_y + dy$
24: **else**
25: $i_z = i_z + z_{inc}$
26: $t_z = t_z + dz$
27: **end if**
28: **end if**
29: $primitiveId :=$TestIntersections(i_x, i_y, i_z)
30: **if** $primitiveId \geq 0$ **then**
31: **return** $primitiveId$
32: **end if**
33: **end while**

Search in a hash table has a worse case time complexity of $O(N)$ for N primitives. This is true for the case where all primitives are inserted into the same bucket. However, spatial hashing creates the hash table based on the geometry of the object, and as such the worse case scenario should never occur. In practice, spatial hashing has a time complexity of $O(1)$.

Chapter 4

Object Weathering Simulation

4.1 Introduction

When an object is left to the elements it will typically weather and develop blemishes, such as dirt stains, moss growth or rust. These blemishes are visual cues about an object's age, make-up and the environment in which it resides. For example, rust can indicate that an object is made of metal and that it has been left outside for a prolonged period of time. Moss growth can indicate a humid environment. Corpses that are left outside can show moss and mould growth or general staining by mud and sand, depending on their environment. Simulating the appearance of an object surface by external environmental factors, such as rain, mud and moss is important for creating realistic objects that appear affected by their environment. Dirt and moss growth can also help in creating a more realistic and varied appearance of corpses.

The weathering simulation described in this chapter is focused on surface appearance changes caused by environmental factors, rather than the processes internal to the body described Chapters 5 and 6. The underlying particle system driving the surface weathering is based on the work of Chen et al. (2005) and Günther et al. (2012), who propose a general weathering simulation that is able to simulate a variety of weathering effects. There have been a number of surface weathering methods that are concerned with modelling a specific weathering effect, such as dust accumulation by Hsu and Wong (1995), rusting by Mérillou et al. (2001); Chang and Shih (2003) and metallic patinas by Dorsey

and Hanrahan (1996), weathered stone by Dorsey et al. (1999), flow staining by Dorsey et al. (1996), and lichen growth by Desbenoit et al. (2004). In comparison to those, the γ-*ton* tracing approach by Chen et al. (2005) and Günther et al. (2012) allows for combinations of different weathering effects in the same scene and even on the same surface. The method is able to simulate multiple weathering effects such as stain-bleeding. Furthermore, γ-*tons* can also pick up resistant substances, such as heat. When heat carrying γ-*tons* intersect a surface with humidity substances, the heat substance can reduce the effect of the humidity on the surface. These render them a useful tool for simulating general weathering on objects.

A short overview of the γ-*ton* approach was given in Section 2.4 in the literature review chapter. In this section a more detailed explanation of the γ-*ton* approach is provided. Furthermore, suggestions for solving some of its limitations are also presented in this chapter. The content of this chapter was presented in the form of a poster paper at the SIGGRAPH Asia 2014 and was published as an abstract in the conference proceedings, see Frerichs et al. (2014).

4.2 The General Object Weathering Simulation

The general surface weathering approach works by tracing particles through the scene that carry materials. These materials can be deposited or picked up from surfaces hit by those particles. Chen et al. (2005) describe the method as an iterative process with two passes:

1. particle tracing
2. generation of ageing effects

During the first pass, thousands of particles called γ-*tons* are generated and fired into the scene via γ-sources. γ-sources represent sources of certain weathering or ageing phenomena (e.g. rain, sun, etc.) and can be placed at specific points, areas or are environmental. The γ-*tons* are then traced through the scene. Their motion is controlled by motion probabilities stored within the particle itself and by reflectance properties stored at the objects' surfaces. The reflectance properties determine the deflection path of the γ-*ton* at its intersection with the surface. γ-*tons* can deposit substances on the surface they intersect with to induce weathering effects and also pick up blemishes, which enables material transport. The distribution of blemishes on the objects in the scene is recorded in a material atlas as in Günther et al. (2012). A material atlas is a set of textures in which the materials on an object's surface are recorded. As triangles meshes

are used to represent the object's surface throughout this project, texture maps are a better choice than surfels (used by Chen et al. (2005)) to record the material deposits. Furthermore, they allow for the material maps to be created off-line and then applied to the object at run-time. This will enable them to be part of a potential content creation tool with the livor mortis and mummification simulations.

In the second pass the weathering effects are generated. Alongside the reflectance properties, each surface element also holds certain material properties that describe its make-up (e.g. metal, dirt or grass). These material properties determine the effect of the γ-*ton* deposits on the surface appearance. This might modify the material properties of a surface or even the object's geometry. If, for example, the material property at a surface element is metal and a γ-*ton* deposits water, then the material property of metal would turn into the material property rust. Weathering effects can be created using multi-texturing, texture synthesis and displacement maps, depending on the weathering effect.

In this section the particle propagation, material depositing and ageing rules of the general weathering simulation are described. These mostly follow the techniques from Chen et al. (2005) and Günther et al. (2012). The main differences to the method described in this chapter to the ones used in previous methods are the approaches used to record the material deposits. These are described in more detail in Section 4.3. The surface of the object is represented by the surface triangle mesh described as the skin mesh in Chapter 3. All intersections are performed on the object's collision representation with the collision tests described in Chapter 3.

p	position
v	velocity
$\{k_s, k_p, k_f\}$	motion probabilities
$\{s_1, ...s_n\}$	amount of each material $i \in [1, n]$ carried

Table 4.1 The γ-*ton* attributes. Here n is the number of materials represented in the scene.

4.2.1 Particle Propagation

Particle propagation can be divided into the following three tasks:

1. emission of new γ-*tons*
2. tracing of all alive γ-*tons*

3. particle-surface collision detection and collision handling of intersection

First, new γ-*tons* are created and shot into the scene. There are three kind of sources from which new γ-*ton*s are released: points, areas or they can be environmental. Point sources are sources at one specific point in the scene. Area sources release γ-*tons* from a bounded area. This area can be specified by a sphere, circle or other shapes. An environmental source is not bounded, but encompasses the entire scene. The initial position and velocity of each new particle is determined by its emission source. For point sources, the initial position is the same as the point sources position. For the area sources the starting position is a random point within the area bound. γ-*tons* emitted from the environmental source have a random starting position. The initial velocity can be random, predetermined for each source (i.e. in gravity direction), or normal to the bounding shape of an area source.

Each γ-*ton* movement is controlled by motion probabilities stored within each individual particle. Three motion probabilities are described by Chen et al. (2005). These are propagating along a straight line k_s, parabolic curved line k_p or flowing k_f, where $k_s + k_p + k_f \leq 1$. It should be noted these determine the path a γ-*ton* follows. A γ-*ton* can be in one of four movement states. These are one of the three motions described (straight, curved or flowing) plus settled. Settled means the particle has stopped moving and has settled down on a surface and its occurrence probability is $1-(k_s+k_p+k_f)$. The motion probabilities are determined at creation and do not change unless the particle comes in contact with a surface. The particle's motion state is determined using the Russian Roulette technique, i.e. a random number r between 0 and 1 is generated. Then, the motion state m is chosen as follows

$$m(r) = \begin{cases} \text{straight} & \text{if } r < k_s \\ \text{curved} & \text{if } k_s \leq r < (k_s + k_p) \\ \text{flowing} & \text{if } (k_s + k_p) \leq r < (k_s + k_p + k_f) \\ \text{settle} & \text{otherwise} \end{cases} \tag{4.1}$$

For a parabolic curve movement the intersection test is performed for the straight line segments that approximate the curve.

At each simulation step an intersection test is performed for each active γ-*ton*. For a particle moving in a straight line this is a simple ray-triangle intersection test. This method was chosen because it is less computationally expensive to do a single intersection test in the case of straight line movements, than it is to trace the line chronologically at a series of time steps. For parabolic curved and

flow movements, the trajectory is piecewise linearly approximated by straight line movements. A line segment is defined by the particle's velocity $\boldsymbol{v}$ and a fixed user-defined step size (or segment length) h. A line intersection test is performed for this segment. If no intersection along the line $\boldsymbol{x} + h\boldsymbol{v}$ is found, the particle's position $\boldsymbol{x}$ and velocity $\boldsymbol{v}$ are updated to describe the next line segment on the curve. This process is repeated until an intersection is found, the particle moves out of the scene's boundary, or a maximum number of iterations have been performed. Note that, owing to the choice made to use a single ray intersection test for straight line movements, the step size h does not constitute a time step; the particle movements are not temporal, although they are sequential.

The flow movement is handled similarly. Flowing motions only occur if the particle is in contact with a surface and then moves along the surface. Flowing motions are simulated as described by Günther et al. (2012), given an intersection point $\boldsymbol{x}$ on a surface, the particle is moved an offset ϵ away from the surface in the surface normal direction $\boldsymbol{n}$. The particle's velocity $\boldsymbol{v}$ is projected into the surfaces tangent plane which gives a new vector $\boldsymbol{v}_t$. Given the step size h, a new vector $\boldsymbol{w} = \boldsymbol{v} - h\boldsymbol{n}$ is computed. The new position of the flowing γ-*ton* is then found by testing for an intersection along the ray with direction $\boldsymbol{w}$ and origin $\boldsymbol{x}$. If a new intersection is found, the particle is moved an offset ϵ away from the new found intersection point in the surface normal direction $\boldsymbol{n}$ and the process above is repeated until either no intersection is found or the particle has settled down. If no intersection is found, the velocity of the particle is set to point in the gravity direction.

If a particle's trajectory does not intersect a surface it is removed. If a particle hits a surface, its motion probabilities are adjusted according to the surface reflectance properties. The surface reflectance properties of each point on the surface are defined in a texture map. The reflectance properties $\Delta_s, \Delta_p, \Delta_f$ for the three motion probabilities are looked up in the texture map at the intersection point. The new motion probabilities of the γ-*ton* are computed as in Chen et al. (2005):

$$k_s = max(k_s - \Delta_s, 0) \tag{4.2}$$

$$k_p = max(k_p - \Delta_p, 0) \tag{4.3}$$

$$k_f = max(k_f + max(k_p - \Delta_p, 0) - \Delta_f, 0) \tag{4.4}$$

The motion probabilities deteriorate with each surface intersection and, as such, the settling of the particle becomes more and more likely. After the new motion

probabilities have been computed, Russian Roulette is again used to determine the reaction of the particles movement to the surface collision. One in four reactions is possible. The γ-*ton* will either reflect off, bounce off, flow along, or settle on the surface. A reflection puts the particle in the straight movement, bouncing in a curved movement and flow in a flow movement state. For a reflection and bouncing reaction the velocity of the particle is corrected first. Settling puts the particle in the resting state, i.e. velocity is zero and the particle cannot move any more.

4.2.2 Material Transport

γ-*tons* carry certain material, such as water, dirt, grass or rust. The amount a γ-*ton* carries is recorded in its carrier attributes. The materials can effect the motion probabilities of the particle. If, for example, a γ-*ton* carries mostly water, the motion probability for flowing is set higher than the motion probability for a straight line or parabolic curve propagation. Materials are assigned to a γ-*ton* at creation and do not change unless the particle hits a surface. When in contact with a surface a γ-*ton* can deposit or pick up substances from the surface. In this way different substances are transported through the scene to simulate a variety of surface weathering phenomena. Changes in the materials carried by the γ-*ton* can effect changes in the object's motion probabilities.

Material deposits are recorded on the object's surface by adding an amount of the material carried by the particle j ($s_i(j)$) to the same material on the surface ($m_i(\boldsymbol{x})$) at intersection point $\boldsymbol{x}$, i.e for substance i with transfer factor c_i, a material deposit is added to the surface with $m_i(\boldsymbol{x}) = m_i(\boldsymbol{x}) + c_i * s_i(j)$ at the particle j's intersection point $\boldsymbol{x}$ with the surface. The material is then removed from the particle, i.e $s_i(j) = s_i(j) - c_i * s_i(j)$. Here, c_i controls the amount of substance i that is transferred per interaction to account for different volatiles. Similarly, substance pick up from a surface is added to the relevant carrier attribute of the particle ($s_i(j) = s_i(j) + c_i * m_i(\boldsymbol{x})$) and removed from the surface ($m_i(\boldsymbol{x}) = m_i(\boldsymbol{x}) - c_i * m_i(\boldsymbol{x})$). If a particle settles down all of its substances are transferred to the surface. The substances deposited on the object's surface are recorded in the object's material map. This is further explained in Section 4.3.

4.2.3 Ageing Rules

After the first pass is completed, the material properties of the surfaces need to be updated. The interaction of different substances on an object's surface effect different weathering phenomena. An example is a metal surface that also has

water deposits. This should lead to the metal turning into rust over time. These effects are achieved by ageing rules as described by Günther et al. (2012). The ageing rule are functions ($m_i(\boldsymbol{x}, t)$) or rules that define changes over substance distribution on a surface over time. For example, the ageing rule for reducing a substance i on the surface over time would look as follows

$$\frac{\partial m_i(\boldsymbol{x}, t)}{\partial t} = -\lambda_i \tag{4.5}$$

where λ_i controls the amount removed at each simulation step. Another ageing rule can be the creation of a substance l in the presence of two other substances i and j:

$$\frac{\partial m_l(\boldsymbol{x}, t)}{\partial t} = min(m_i(\boldsymbol{x}, t), m_j(\boldsymbol{x}, t))\lambda_l \tag{4.6}$$

where λ_l controls the speed in which the new substance m_l is created from the other too.

These ageing rules are applied during the second pass and result in changes to the surfaces' material properties, such as the creation of rust in the presence of water and metal. Further surface changes, such as updates to the object's bump map or the creation of displacement maps can also be performed in this step.

4.3 Recording Material Depositing

The two previous methods on γ-*ton* tracing differ in the way they are recording the material deposits. Chen et al. (2005) use a surfel representation to record the material deposits, whereas Günther et al. (2012) use texture maps. The surfel approach has a longer execution time compared to using a texture map due to the GPU-unfriendly nature of its representation. The texture maps approach allows for faster localisation of material deposits. Furthermore, the object model used throughout this project involves a triangle mesh as surface representation. Hence the Günther et al. (2012) texture map approach is more suited to this surface weathering simulation than the surfel based approach.

Günther et al. (2012) find the texel in the material texture that corresponds to the intersection point. The colour channel that is used to record the substance to be deposited/removed is then updated accordingly by changing its value as described above. In order to avoid seams at texture coordinate discontinuities

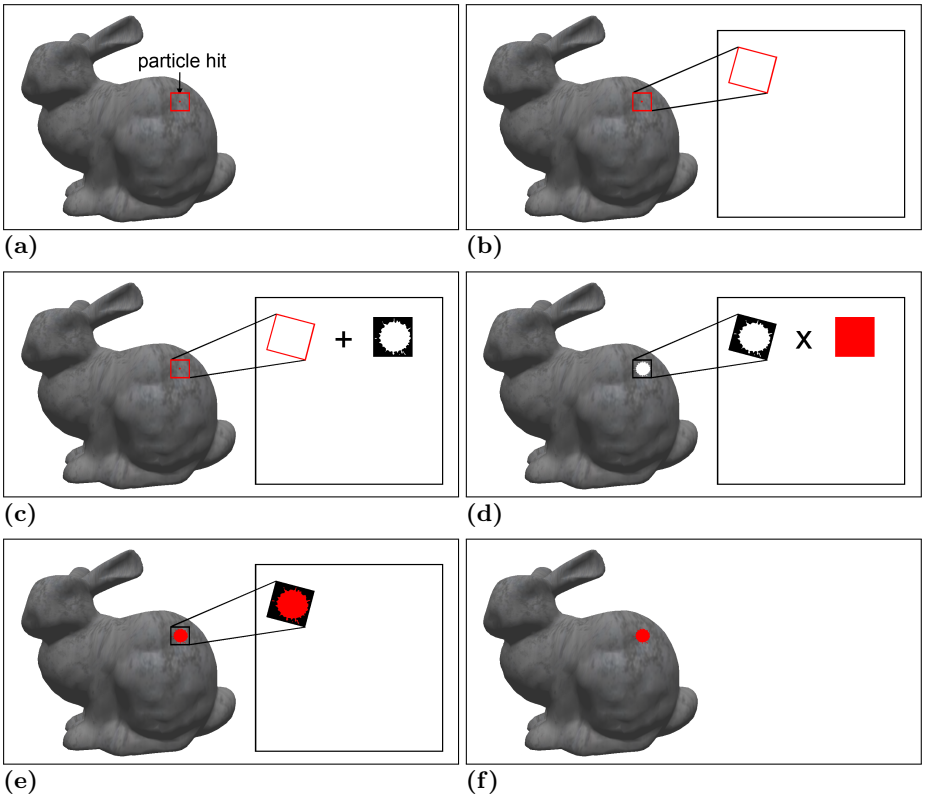


Figure 9: This figure shows the material depositing approach: a) square-projection onto the object, b) mapping of the square into the material map, c) mapping of the splatting texture d-e), assignment of the material colour and f) the resulting deposit.

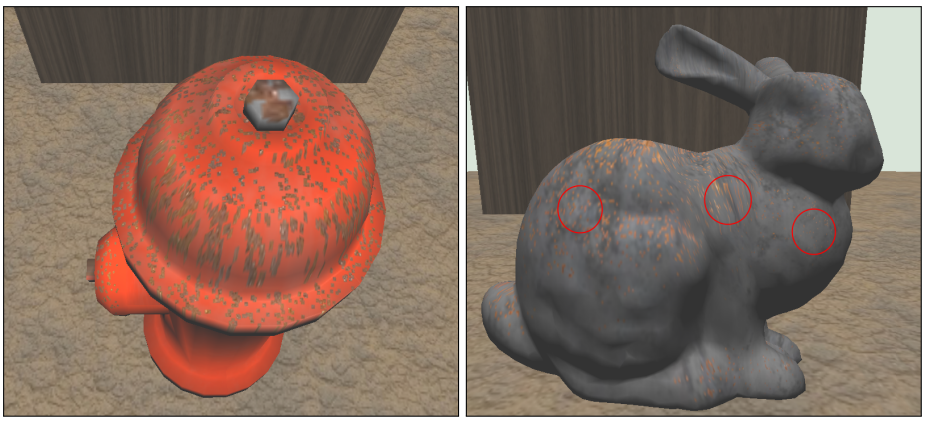
the size of each material recording is one texel. Although this does help in avoiding visual artefacts at texture seams, it does introduce other visual problems that are apparent when texture maps are not uniform over the object. In some cases certain areas of an object require more detail (e.g. face) than the rest of the object. The texture space assigned to those areas is then relatively larger than what is allocated to the rest. This means that a texel on, for example, the face, would be smaller in world space than a single texel on another area of the object, or some texels might be stretched in world space. Figure 10 shows an example of this.

Figure 10a and 10b were generated using the Günther et al. method of using a single texel per deposit. Figure 10a shows a fire hydrant. The texture space covering the iron operating nut on the hydrant's bonnet (top) covers less texels per world space measurement than the texture space on the bonnet itself. Hence, a single deposit on the operating nut is larger than a single deposit on the bonnet. Furthermore, the texture space covering the curved belly of the bonnet is slightly stretched, which results in visually stretched deposits. Similar depositing artefacts can be observed on the Stanford bunny model in Figure 10b. The head and torso are more detailed than the bunny's body, resulting in smaller deposits on the head and torso than on the back. Stretching artefacts can also be observed around the neck area. Aside from visual artefacts, this variation in depositing sizes also has a great impact on the speed at which an object weathers. Areas with relatively detailed texture maps receive less material per deposit per area in world space and thereby weather more slowly than the areas with less detailed textures and thereby greater deposits. This means that the bunny's back will weather quicker than its head and torso, even when receiving the same amount of deposits. This can be avoided by computing the dimensions of the area affected in world space and translating them into texture space.

Figure 9 shows an overview of the recording of a material deposit. This section will explain each step in more detail. This is starting with the depositing square projection (Figure 9a) and the subdivision of the square into independent polygons (Figure 9b). Then the polygons are mapped into the texture space for rasterisation (Figure 9c) using a splatting texture as a stencil (Figure 9d-9e) in order to get the depositing result in the object's surface (Figure 9f).

4.3.1 Depositing Square Projection

For each object a material texture map is created, where each colour-channel is associated with one type of material. The value at each channel represents the amount of material present at that point on the surface. When a particle hits



(a) (b)

Figure 10: This figure shows a simulation result using γ-*ton* with a fixed number of texel deposit as proposed by Günther et al. (2012) method.

an object, it can pick up or deposit certain materials (such as water, rust and dirt) with the deposits then stored in the object's material map. In contrast to the Günther et al. (2012) method, the approach described in this section does not colour a single texel per deposit, but a number of texels that cover a certain area of the surface in world space. Each particle has an additional attribute *size* which controls its depositing dimension in world space. After an intersection $\boldsymbol{x}$ is found, a square with edge lengths of the particle's size is projected onto that surface. For this the tangent space of the intersected triangle is found. The cross product of one of the triangles edges ($\boldsymbol{e}$) and normal ($\boldsymbol{n}$) results in a third vector ($\boldsymbol{v}$). The vectors $\hat{\boldsymbol{e}}$ and $\hat{\boldsymbol{v}}$ are then used to find the four corners of the square with edge length d:

$$\boldsymbol{ul} = \boldsymbol{x} - \frac{d}{2}\hat{\boldsymbol{e}} + \frac{d}{2}\hat{\boldsymbol{v}} \tag{4.7}$$

$$\boldsymbol{ur} = \boldsymbol{x} + \frac{d}{2}\hat{\boldsymbol{e}} + \frac{d}{2}\hat{\boldsymbol{v}} \tag{4.8}$$

$$\boldsymbol{ll} = \boldsymbol{x} - \frac{d}{2}\hat{\boldsymbol{e}} - \frac{d}{2}\hat{\boldsymbol{v}} \tag{4.9}$$

$$\boldsymbol{lr} = \boldsymbol{x} + \frac{d}{2}\hat{\boldsymbol{e}} - \frac{d}{2}\hat{\boldsymbol{v}} \tag{4.10}$$

where $\boldsymbol{ul}$, $\boldsymbol{ur}$, $\boldsymbol{ll}$, and $\boldsymbol{lr}$ are the upper left, upper right, lower left and lower right corners of the square respectively. Using spatial hashing (Section 3.3.2) the intersections of the four corners with neighbouring triangles are quickly found. With the intersection points known, the texture uv-coordinates for each corner can be computed using the barycentric coordinates with respect to the vertices

of the intersected triangle. This results in four vertices in the object's texture space that make up a square. This square (or any polygon) is then divided into triangles and drawn onto the material map of the object. The following shows the attributes of a deposit drawn to the texture map:

n	number of vertices of polygon
$\{v_1, ...v_n\}$, $v_i \in \mathbb{R}^2$	uv-coordinates of polygon vertices
$\{u_1, ...u_n\}$, $u_i \in \mathbb{R}^2$	uv-coordinates of polygon vertices in splatting texture
$colour \in \mathbb{R}^4$	material deposits represented in colour channels

Table 4.2 The deposit element sent to the shader to store a deposit.

The colour attribute is used to record the material. Each of the four colour channels represent one material. The depositing and removing of a material is done by filling the amount to be deposited into the correct colour channel. For removals a negative amount is filled in. The uv-coordinates for the splatting texture are explained in Section 4.3.3. A square deposit has four uv-coordinates. Assuming there are four substances represented in the colour channels of the material map, a square deposit would look as follows:

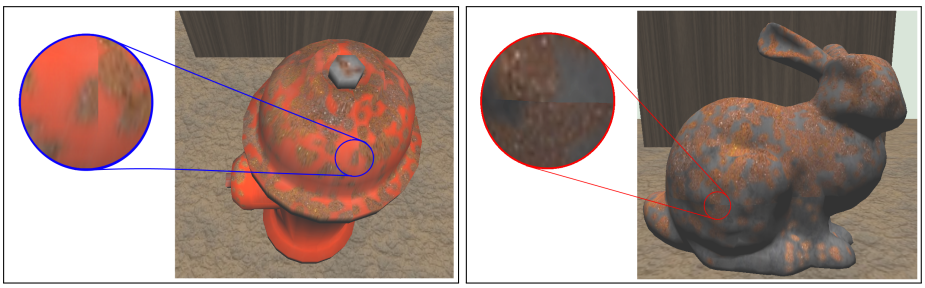
4	number of vertices
$\{u1, u2, u3, u4\}$	uv-coordinates
$\{(0,0), (1,0), (1,1), (0,1)\}$	uv-coordinates splatting texture
$(\frac{dm_1}{dt}, \frac{dm_2}{dt}, \frac{dm_3}{dt}, \frac{dm_4}{dt})$	material deposits

Table 4.3 An example of a square deposit element.

The method described above works relatively well even if the four corners do not lie on the same triangle. However, when the square lies over a texture seam this method breaks down, i.e. it shows the same visual discontinuities of deposits along texture seams that can be observed in the Günther et al. (2012) article when colouring more than one texel per deposit. The next section shows how discontinuities along texture seams can be avoided.

4.3.2 Handling Texture Space Discontinuities

Spatial hashing is used to find the four corners of the depositing squares in world space and also find the triangle faces on which they lie. From this the uv-coordinates for the four vertices can be simply computed. If all vertices lie on the same triangle face, a simple square deposit element as shown in Table 4.3 is created and sent to be drawn to the object's material map. However, if the



(a) (b)

Figure 11: This figure shows examples of splatting textures that can be used as stencils for material deposits.

four vertices do not lie on the same triangle face, creating a single square deposit element in this way can lead to discontinuities along texture seams. In order to avoid these, the projected square is subdivided into smaller polygons, where each polygon lies on a single triangle face and all polygons together form the complete depositing square. Every edge of the depositing polygon either lies on an edge of the triangle it resides on or on the edge of the projected square. Each polygon then becomes a deposit element that can be sent to be drawn in the material map. This works well since any closed polygon can be subdivided into triangles that can then be drawn by any rasterisation algorithm. Each polygon lies on a single triangle face and all its vertices are expressed in relation to the triangle in which they reside. Since no polygon lies across multiple triangles, no polygon can lie across a texture seam and therefore discontinuities are avoided.

The spatial hashing structure can not only be used to find the four corners of the square but also to subdivide the square into smaller polygons in the case the vertices do not lie on a single triangle. Knowing the location of the four vertices in the grid structure representing the object space (see Section 3.3), all triangles lying in the space between the four corners can be quickly obtained by retrieving all triangles that are stored in the grid cells that lie between the (and in the same) cells in which the four corner vertices reside. Having a list of triangles, each is handled individually by determining which part of the depositing square they hold and creating the depositing polygon from it.

First the faces on which the four corners lie on are handled. Each of these triangle faces can have either one, two or three of the square corners which are shown in Figure 12. Note that this method is only used if not all four vertices lie on the same triangle, and, as such, that possibility is ignored at this point. In the case where three corners lie on a single triangle face two intersection points need to be found, as shown in Figure 12c. There are two edges of the depositing square that lead to the square corner outside the triangle. Hence the

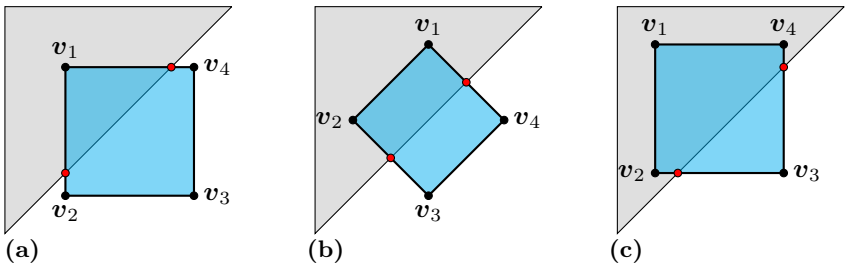


Figure 12: This figure shows the three cases of the square corner positions on a triangle of the object's surface mesh. In a) one corner lies on the triangle face and two intersection points (red) with the triangles edges are found. In b) two corners lie on the triangle face and in c) three corners. Each case is handled separately by finding the intersection points and using these to construct a new polygon.

two intersection points are found by simply finding the intersection of those two edges with the edges of the triangle. Together with the three square corners inside the triangle, there are five vertices that together make the depositing polygon. One or both of the corners may lie on the triangle edge rather than cutting it. In the case when both corners lie on the triangle edge the deposit polygon is made up of three vertices. If only one corner lies on the triangle edge, an intersection must be found, which becomes a fourth vertex of the deposit polygon. When there are two corner points on a single triangle finding the depositing polygon is similar, i.e. the intersection of the two depositing with a triangle edge results in two new vertices (if the two vertices do not lie on the same triangle edge) that are used to create the depositing polygon with the two corner points on the triangle (see Figure 12b). The same process is used in the case of a single corner point on the triangle face. For all three cases a vertex of the triangle may lie inside the depositing square. In this case, that vertex is simply added to the others.

After the triangles holding the corner points have been handled, the next step is to loop through the list of the other triangles that do not hold corner points. For each triangle vertex a simple test is done to see whether they lie in the depositing square. There a four scenarios that can occur: 1) no vertices lie inside, 2) only one vertex lies inside, 3) only two vertices lie inside or 4) all vertices lie inside the square. Three of the possible scenarios are shown in Figure 13. In the case no vertices lie in the square, four line intersection tests between the square and triangle edges need to be performed. If no intersections are found the triangle holds no part of the depositing square and is therefore simply deleted from the list. In the case intersections are found, the intersection points make up the vertices of the depositing polygon. In the fourth case where all vertices lie inside the depositing square (see Figure 13c), the whole triangle itself becomes a

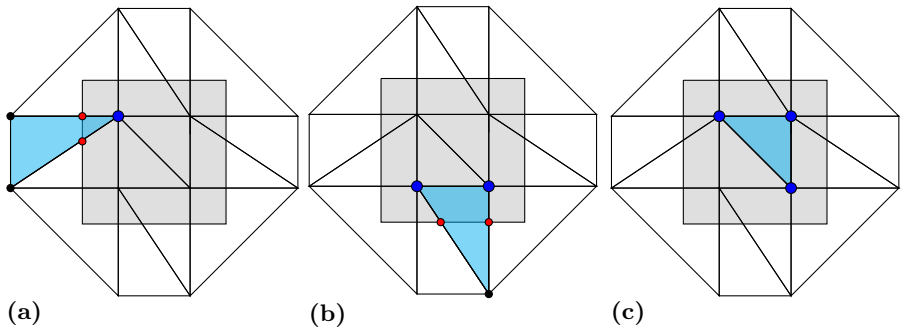


Figure 13: This figure shows the three possibilities of surface triangles (cyan) intersected by the depositing square (grey), that do not hold a square corner. In a) only one vertex (blue) of the triangle lies within the square. A new polygon is created with the three triangle vertices and the two edge intersection points (red). In b) two vertices (blue) lie within the square and in c) all three lie within the square.

depositing polygon. The other two cases are somewhat more complicated. If one triangle vertex lies within the square and the other two outside, as demonstrated in Figure 13a, then that means at least one edge of the square cuts through at least two edges of the triangle. These intersection points of square edges and triangle edges result in new vertices. Together with the triangle vertex that lies within the square a new polygon is created. This is also done for the scenario when two triangle vertices lie within the square and one outside, as shown in Figure 13b. Finding the intersections of the square edge with the two edges leading to the outside triangles results in two more vertices. Together with the two triangle vertices inside the square a new depositing polygon is created.

The computational complexity of the polygonal subdivision algorithm depends on the size of the depositing square and the triangle mesh resolution. Smaller triangles and larger depositing squares require a larger number of depositing elements that need to be constructed by subdivision and drawn into the material map. The complexity depends on the relative size of the depositing square to triangle size.

All depositing polygons created have the same colour attribute. The uv-coordinates for each polygon are found by finding the barycentric coordinates of each vertex of the polygon with respect to the triangle surface it resides on. The barycentric coordinates are then used to interpolate the uv-coordinates for each polygon vertex from the uv-coordinates of the three triangle vertices. The splatting texture coordinates need to also be computed for each vertex. This is explained in the next section.

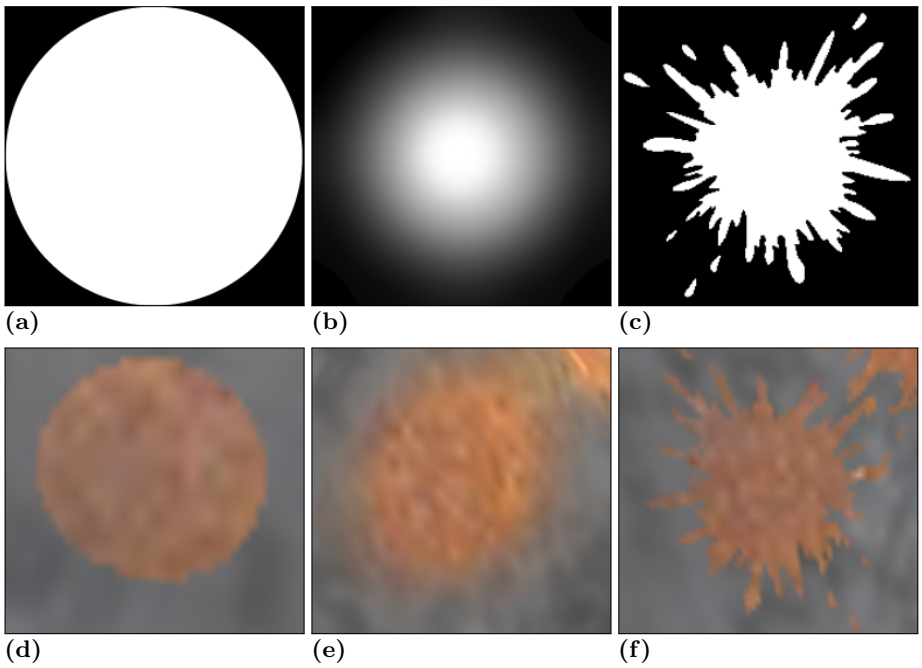


Figure 14: The top row shows examples of splatting textures that can be used as stencils for material deposits. The bottom row shows the resulting appearance of using the splatting textures for rusting.

4.3.3 Texture Splatting

Substance depositing is achieved by projecting a square onto the intersection point on the surface and translating the square into the object's texture space. This involves drawing a number of polygons in the object's texture space that appear as a square shape in world space. As long as the deposit square size is relatively small this looks reasonable. However, with larger deposits the polygonal shape of the deposit with its sharp corners and edges looks unnatural for most substances. In order to avoid this, different splatting textures can be used that control the shape of the deposit on the object's surface.

A splatting texture is a grey scale square image. Some examples of splatting textures are shown in Figure 14. The splatting texture works like a stencil, where white areas receive all the colour, grey areas receive some colour and black areas receive none. The splatting texture is fitted to the depositing square. This means that the corners and edges of the splatting texture correspond to the corners and edges of the depositing square. The upper left corner of the depositing square has the splatting texture uv-coordinates (0.0), the lower right corner has the splatting texture uv-coordinates (1,1) and so on. This means that for any point on the depositing square, a pixel on the splatting texture

can be found easily. When the depositing square is divided into polygons, as described in the previous section, the uv-coordinates (u'_i, v'_i) of the splatting texture need to be recomputed for each polygon. This is done by simply finding the coordinates of the polygon vertices (υ_i, φ_i) with respect to the depositing square using scalar projection:

$$\begin{aligned} \upsilon_i &= \frac{1}{d}(\boldsymbol{ur} - \boldsymbol{ul}) \cdot (\boldsymbol{v}_i - \boldsymbol{ul}) \\ \varphi_i &= \frac{1}{d}(\boldsymbol{ll} - \boldsymbol{ul}) \cdot (\boldsymbol{v}_i - \boldsymbol{ul}) \end{aligned} \tag{4.11}$$

and then dividing it by the depositing square edge length d to get the uv-coordinates:

$$\begin{aligned} u'_i &= \frac{\upsilon_i}{d} \\ v'_i &= \frac{\varphi_i}{d} \end{aligned} \tag{4.12}$$

for all vertices $\boldsymbol{v}_i$ of the polygon. The depositing element for the polygon would then look as shown in Table 4.4 below.

n
$\{v_1, ...v_n\}$, $v_i \in \mathbb{R}^2$
$\{(u'_1, v'_1), ...(u'_n, v'_n)\}$, $u_i \in \mathbb{R}^2$
$(\frac{dm_1}{dt}, \frac{dm_2}{dt}, \frac{dm_3}{dt}, \frac{dm_4}{dt})$

Table 4.4 An example of a polygon deposit element with corrected splatting texture uv-coordinates

In order to draw a polygon deposit onto the object's material map it is first divided into triangles. The triangles are then filled using a rasterisation algorithm. Without a splatting texture, each texel would be filled with the colour given in the deposit element by the rasterisation algorithm. With a splatting texture, each texel (u_t, v_t) in the material map $(\boldsymbol{M})$ filled by rasterisation is instead mapped to a texel (u_s, v_s) in the splatting texture $(\boldsymbol{S})$ using barycentric coordinates. Then the fill colour $\boldsymbol{M}(u_t, v_t)$ at texel (u_t, v_t) is the depositing colour multiplied by the colour in the splatting texture at (u_s, v_s), hence:

$$\boldsymbol{M}(u_t, v_t) = \boldsymbol{S}(u_s, v_s) * (\frac{\partial m_1}{\partial t}, \frac{\partial m_2}{\partial t}, \frac{\partial m_3}{\partial t}, \frac{\partial m_4}{\partial t}) \tag{4.13}$$

for each texel (u_t, v_t) in the polygon deposit area in the object's material map.

4.4 Rendering of Weathering Appearances

The surface weathering simulation will output a weathered colour map and surface normals for each object in the scene that can be used for rendering. The weathered colour map is generated by texture composition created from the object's material atlas. A material atlas is a collection of texture maps for an object which contain the original colour, the surface normals and amounts of materials and substances on the object surface. The material maps are generated by the *γ-ton* simulation as described in the previous sections. The original colour and surface normal maps are predefined. The *γ-ton* simulation might however alter the normal map to account for small scale surface changes caused by weathering as described by Günther et al. (2012). Additional global textures specify the colouration for each material type. If, for example, the scene contains the four substances of metal, rust, water and moss, then there would be four global material maps specifying colourations for metal, rust, wetness and moss. The metal material is not a substance that is carried by the particle but rather tends to be the original material an object is composed of. In this case, there is no global texture map for metal colourations but instead the original texture map of the object is used.

The weathered colour map is created by combining the original colour map with the global material textures using the object's material maps to act as the alpha blending map. In the case of four material types there is one material texture where each of the four colour channels (RGBA) holds one material deposit count. Each texel in the material deposit map is normalised, such that the material deposits amount for each texel add up to one. This is done by adding the RGBA channel values together and dividing them by the sum. These weights are used to create a deposit colour at that texel. The weight w_j that specifies the original colour is excluded from this. If $w_i \neq w_j$ is a alpha blend weight, then the deposit colour is:

$$\boldsymbol{c}_{deposit} = \sum_i max\big(0, w_i \cdot \boldsymbol{c}_i\big) \tag{4.14}$$

where $\boldsymbol{c}_i$ is the colour in the global texture map for material i. The weathered colour can then be computed by combining the deposit colour with the original

colour using the weight of the original material w_j as the blending weight, i.e:

$$\boldsymbol{c}_{weathered} = max\big(0, w_j \cdot \boldsymbol{c}_{original} + (1 - w_j) \cdot \boldsymbol{c}_{deposit}\big) \tag{4.15}$$

where $\boldsymbol{c}_{deposit}$ is the colour in the original colour map of the object.

4.5 Results and Limitations

The aim for the surface weathering simulation is to be able to simulate a variety of surface weathering phenomena, allow for control over the material deposit sizes and appearance while at the same time avoiding visual artefacts of deposits such as inconsistent deposit sizes, stretching and visible discontinuities at texture seams. In this section the surface weathering simulation is demonstrated through a number of example scenarios. Two main models are used for all examples shown in this chapter, the Stanford bunny and a fire hydrant. The resulting material maps from the simulation can be exported and used in other programs.

The problems described with the Günther et al. (2012) method are inconsistent depositing sizes and texture space discontinuities. The method described in this chapter aims to avoid these visual artefacts. The next two sections will compare results generated using the one-texel-deposit approach (Günther et al. (2012)) with results using the deposit square projection approach to avoid inconsistent deposits (see Section 4.3.1) and discontinuities (see Section 4.3.2). The third section shows some examples of how the deposit square approach and splatting textures can be used to control the size and shape of the deposits (Section 4.5.3).

For the examples that are generated in this section, four substances are present in the scene. These are rust, metal, water and moss allocated to the colour channels RGBA respectively. Table 4.5 shows the γ-*ton* attributes for a water particle emitted from position $\boldsymbol{p}_i$.

r_i	radius of particle.
$\boldsymbol{p}$	random position within a spherical area
$(0, -1, 0)$	velocity in gravity direction
$\{0.0, 0.2, 0.8\}$	motion probabilities for water. High flow possibility
$\{0, 0, 1, 0\}$	amount of each material carried (rust, metal, water, moss)

Table 4.5 γ-*ton* attributes for a water particle i, which holds only water at the point of creation.

Similarly, the bunny and hydrant surface were initialised to have the reflection probabilities and materials in Table 4.6.

$\{0.0, 0.4, 0.005\}$	reflection probabilities
$\{0, 1, 0, 0\}$	amount of each material carried (rust, metal, water, moss)

Table 4.6 Surface attributes for the bunny and water hydrant mode at creation.

The particle and surface motion attribute are the same as the ones given by Chen et al. (2005). However, the reflection probabilities on the surface are altered in accordance with material changes. As the surface gets more saturated with water, the flow probability is increased. An ageing rule is applied to turn metal into rust in the presence of water. This involves the following functions:

$$\frac{dm_{rust}(\boldsymbol{x}, t)}{dt} = min(m_{metal}(\boldsymbol{x}, t), m_{water}(\boldsymbol{x}, t))\lambda_{rust} \tag{4.16}$$

which creates rust from water and metal. Here λ_{rust} controls the rusting speed. Then, same amount of metal, as the rust that was created, is removed from the surface to simulate metal turning into rust:

$$\frac{dm_{metal}(\boldsymbol{x}, t)}{dt} = -min(m_{metal}(\boldsymbol{x}, t), m_{water}(\boldsymbol{x}, t))\lambda_{rust} \tag{4.17}$$

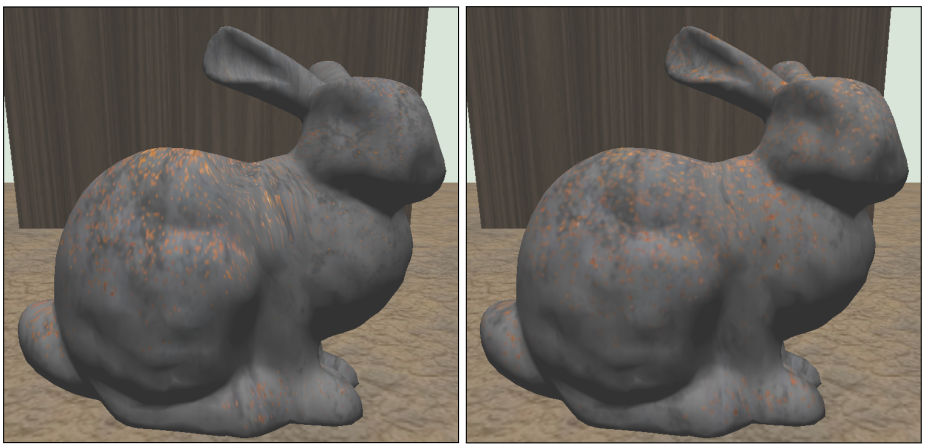
And water is evaporated over time:

$$\frac{dm_{water}(\boldsymbol{x}, t)}{dt} = -\lambda_{water} \tag{4.18}$$

where λ_{water} controls the speed in which water evaporates. It should be noted that for this project the particle simulation was run on the CPU.

4.5.1 Consistent Deposit Size

The texel to world space size is not always uniform over the scene or even over a single object. Some surface areas on an object might require more surface detail than others. In this case more texels are used to cover the detailed area in world space than are used to cover an area of the same dimensions elsewhere. This can be problematic when storing the material deposits in the object's texture maps as this causes inconsistent depositing, with some areas weathering quicker than others even when they have the same particle exposure. Figure 15 compares the one-texel deposit approach described by Günther et al. (2012) (Figure 15a)



(a) (b)

Figure 15: This Figure compares the results of weathering simulations using a) the Günther et al. (2012) method and b) the proposed method.

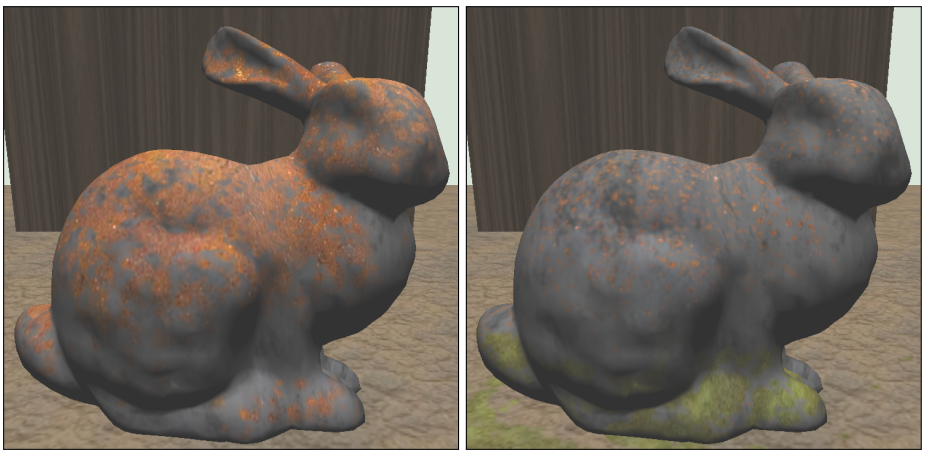
with the depositing square approach described in this chapter (Figure 15b). The square size is chosen to be the size of the largest space a texel from the object's texture map takes up in world space.

The results shown in Figure 15a show larger rust deposits on the bunny's back than on the bunny's chest and head and stretched deposits on the neck. This results in the faster rusting of the back than the head even when hit by a similar amount of particles. The back of the bunny already looks more rusted than the head and ears. In Figure 15b on the other hand, all deposits have a uniform size over the bunny's surface. This also results in a more even rusting. The head and back show equally advanced rusting.

4.5.2 Texture Space discontinuities

The texture maps have been set up to have the texture seams located at areas on the object that are easy to detect in order to demonstrate both the depositing discontinuities along texture seams when using the Günther et al. (2012) with deposits greater than one texel, and the results using the method described in section 4.3.2. There are no visible deposit discontinuities in Figure 15b, 17 and 18, all of which where created using a depositing square.

Figure 16: In this example the depositing dimensions depend on material transported by the *gamma-ton*.



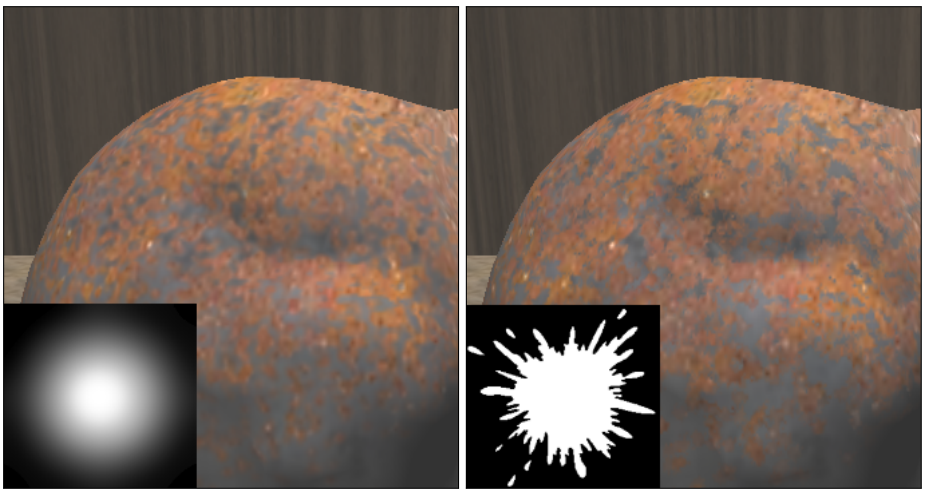
(a) (b)

Figure 17: Different depositing dimensions a) random and b) based on the material being transported.

4.5.3 Controlling the Deposits Size and Shape

Using the depositing square approach allows for direct control over the size and shape of deposits. The dimension of the depositing square is predefined for each particle by the particle's radius (r_i) attribute, which controls the edge length of the depositing square. This radius can be random for each newly created particle or be dependent on the type of material carried by the particle. Figure 17 shows two examples of this. In Figure 17a the depositing size was chosen to be random, which results in a different rusting appearance. Note that this does not create the same varying depositing effects that were observed when using a fixed number of texels per deposits. Even though the deposit sizes vary in Figure 17a, the rusting speeds over the surface is still uniform, as opposed to results observed in Figure 15a which was generated using a fixed number of texels per deposit. In the fixed-texel approach, the size of a deposit is defined by the size of a texel in world space, which can vary over the object's surface. With the random deposit approach, in contrast, the deposit size is defined by the particle's size and is therefore not directly dependant on the object's texture map. Furthermore, with the fixed-texel approach the deposit size variations are fixed. With the square deposit method the deposit size variations are dynamic and controllable.

The depositing size can also be mapped to materials or particle properties. An example of this is shown in Figure 17b. Here the deposit sizes of the water deposits (shown as rust) are small, whereas the depositing size for moss are much larger. This also results in the quicker moss growth compared to rusting.



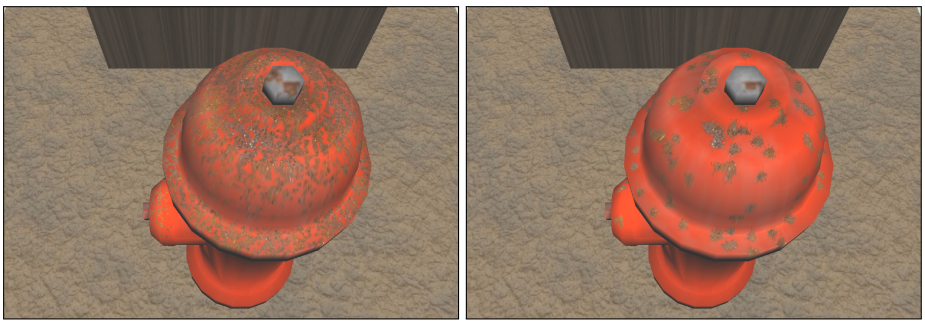
(a) (b)

Figure 18: Surface weathering results generated with different depositing shapes a) round and b) splash.

The splatting texture can be applied to achieve different deposit appearances. Similar to the depositing size, the splatting texture can be assigned to a particle property, such as material carried or motion. As such, a splash (Figure 18b) could be used for a water particle's bounce or reflect motion, whereas a circle 18a could be used for a flowing motion. Different splatting textures lead to different appearances. Comparing the two images in Figure 18 one can observe that the deposits with the blurred circle splatting texture have smooth edges and the deposits with the splash splatting texture appear sharper. Figure 14 shows the difference in using a white circle splatting texture (Figure 14a and 14d) as opposed to a blurred circle splatting texture (Figure 14b and 14e).

4.5.4 Limitations

There are some limitations to the square-deposit method described in this chapter. The complexity of the deposit square subdivision algorithm is dependant on the relative size of the square to the average triangle of the surface mesh, which makes it more computationally expensive than the Günther et al. (2012) one pixel approach. As real-time performance is not the objective, this is a reasonable trade-off. In practice, the deposit square dimensions tend to be smaller than the triangle sizes in the mesh, and therefore do not require a large amount of subdivision. However, the proposed method still requires more drawing instances than the Günther et al. (2012) one pixel approach. The tessellation and drawing of the polygons into the material map is done on the GPU. This is in



(a) (b)

Figure 19: Surface weathering results generated with different depositing shapes on a water hydrant model with a) round and b) splash.

order to make use of its parallelism, which reduces the computational costs.

Another limitation is the size of the depositing square. The smallest possible depositing size depends on the greatest texel-to-world size. In other words, the depositing dimension must be at least as big as the largest area on a surface covered by a single texel. Figure 19a shows an example of a small deposit size on the water hydrant model. The greatest texel-to-world size is on the iron operating nut on the hydrant's bonnet (top). The depositing size in this image was chosen to be smaller than the area covered by a texel on the operating nut. This results in a different appearance of the deposit on the operating nut as compared to the rest of the hydrant. Another problem can occur even if the deposit size is chosen to be the world space dimension of a texel on the operating nut. This is demonstrated in Figure 19b. Here a splash splatting texture was chosen and the deposit dimension equals the world space dimension of a texel on the operating nut. In this case the depositing sizes are uniform, however, the shape of the splatting texture is lost because only a single texel is coloured. In order to use splatting textures with more complicated shapes an even larger deposit size needs to be chosen. Too large depositing sizes can increase the cost of the deposit square subdivision step. Alternatively, a higher resolution texture map can be used. This would decrease the maximum size of a texel in world space. On the other hand, larger texture maps can be expensive.

The depositing square projection assumes a more or less flat surface within the area of the square at intersections. Large or gradual curves are not a problem. However, small concave geometry might cause a deposit to be recorded over the entire square area. This happens if the depth of the concave geometry is larger than the square dimension while its width is smaller. This might result in the deposit being recorded at the concavity's edges, but not in its centre. Another drawback is that this simulation only considers texture changes and

not surface or geometrical deformations. Small scale surface changes could be achieved by creating displacement maps in a similar way as was done with the material deposit maps. Similar work was carried out by Chen et al. (2005).

4.6 Conclusion

In this chapter a general surface weathering approach was presented. Particles that carry materials are tracked in the scene. These particles can pick up and deposit certain materials on a surface, which then induce the weathering effects. The method described in this chapter is intended to improve on a previous approach by Günther et al. (2012), which had visual problems with uneven material deposits and deposit discontinuities along texture seams. These have been shown to lead to sharp edges and inconsistent deposits, which would look particularly out of place on organic objects. The material deposits on an object's surface are recorded in the object's texture map (following the Günther et al. (2012)) as opposed to surfels, which were used by Chen et al. (2005). This decision was made because texture maps are supported by triangle meshes, which are used to represent the surface of the object, as explained in Chapter 3. Furthermore, textures allow faster material lookup and are GPU-friendly, which is an advantage if fast computation speeds are required.

To avoid the uneven material deposits and discontinuities around texture seams associated with fixed-texel deposits, a square-depositing approach was proposed. This is capable of avoiding uneven weathering and visible deposit discontinuities while at the same time allowing for more control over the appearance of the deposits in terms of size and shape. However, the algorithm used to avoid the deposit discontinuities makes the depositing square algorithm more computationally expensive than previous methods. The time complexity scales with the deposit square size and higher triangle mesh resolutions.

In order to avoid uneven deposits, the method's depositing square is limited by the largest area a texel of any texture map covers in world space. If a smaller size is chosen for the depositing square, there will still be uneven depositing sizes throughout the scene. Furthermore, the square projection on the intersected surface assumes a locally flat surface geometry. Small but deep concave geometry might not record a deposit if its size is smaller and deeper than the depositing square's dimension. In comparison to the Günther et al. (2012) method, the surface weathering approach presented in this chapter improved upon the visual appearance of the material deposits.

The particle simulation was run on the CPU. If interactive performance is re-

quired, Günther et al. (2012) has shown that it can be run on the GPU as well. For the surface weathering approach, the object was only defined in terms of its surface - any interior or physical properties defining the surface's, or volume's, composition were ignored. As such, this simulation only models changes to the object's surface appearance. This is acceptable to model some minor surface weathering, but is insufficient to simulate the strong geometry changes associated with human body decomposition. While this method can be used to add moss growth and dirt stain effects on a corpse's surface, for the decompositions processes, such as livor mortis and mummification, which present problems of a greater complexity, different methods are required.

Chapter 5

Biologically Inspired Simulation of Livor Mortis

5.1 Introduction

There are a number of different processes that affect the post-mortem appearance of a body. One of the earliest visual clues of death is livor mortis. Livor mortis (or hypostasis) refers to the early post mortem discolouration of the body caused by blood pooling. In this chapter, a biologically-motivated livor mortis simulation is presented that is capable of modelling the colouration changes in skin caused by blood pooling after death. Livor mortis shows mainly as colouration changes of the skin. In contrast to the surface weathering approach demonstrated in the previous chapter, these changes are driven by internal blood dynamics, rather than external environmental activities. The proposed approach consists of a simulation of post mortem blood dynamics and a layered skin shader that is controlled by the haemoglobin and oxygen levels in blood. The object is represented by a layered data structure made of a triangle mesh for the skin and a tetrahedral mesh on which the blood dynamics are simulated on. This allows the simulation of the skin discolouration caused by livor mortis, including early patchy appearance, fixation of hypostasis and pressure induced blanching. The approach is demonstrated on three different models and scenarios and compare the results to real world livor mortis photographic examples. The content of this chapter has been published in the Visual Computer journal, see Frerichs et al. (2017).

Object weathering and decomposition is an emerging area of research in com-

puter graphics. Research into the area of object decomposition so far shows that there is currently no published output that aims to simulate human body decomposition or livor mortis. Work has however been carried out in simulating other decay and rotting processes, such as corrosion, rotting fruit and withering leafs. Kider et al. Kider et al. (2011) and Liu et al. Liu et al. (2012b) simulate the rotting of heterogeneous organic objects such as fruit. Fruit is made of different layers, similar to a human body, and a layered model is used to represent the fruit in their method. Texture maps that hold nutrient and soft rot information on the object's surface can be used for a reaction diffuse model to guide fungal growths, see Kider et al. (2011). Liu et al. Liu et al. (2012b) simulate the rot spreading into the internal flesh layer starting from the object's surface. The processes causing livor mortis (i.e. blood pooling) are happening inside the body and affect the surface appearance from inside. Liu et al. and Kider et al. simulate dehydration in the internal flesh layer, but this does not follow the same dynamics as the blood pooling in a body after death. Jeong et al. (2013) focus on the withering of leaves. Similar to blood transport in humans, water in the leaf flows through the leaf's veins. In the method described by Jeong et al. the deformation and discolouration of the leaf is controlled by osmotic water flow. A layered model is used to represent the leaf, where the edges are the leaf veins and the vertices hold information on water content and solute concentration. Changes in the water content at each vertex drive the changes of the morphology and shading of the leaf. The dynamics of red blood cells after death are controlled by gravity and do not follow the same fluid dynamics as the one used on the Jeong et al. (2013) method. Furthermore, their approach only considers thin shell objects, whereas the aim for this project is to simulate blood pooling on a volumetric object. Simulating the affects of ageing on akin, such as wrinkling, has also been addressed by Wu et al. (1999) and Boissieux et al. (2000). But they consider skin ageing on a living human body, which does not follow the same dynamics as livor mortis.

This chapter concentrates on simulating the process of skin discolouration after death caused by blood pooling as was described by Prahlow and Byard (2011). The blood flows through the human body via the vascular system, which is made of blood vessels of varying size arranged in an irregular network. This network reaches into the lower layer of the skin. The skin colour is affected by the haemoglobin, a red chromophore found in red blood cells and melanin, a brown chromophore found in the outer skin layer. To model livor mortis the haemoglobin transport after death on a volumetric representation of the body need to be simulated. In addition to this, a skin shader is required, one that is capable of accounting for the colouration change caused by the internal blood

dynamics. The approach presented in this chapter consists of:

- a haemoglobin transport simulation on an irregular tetrahedral mesh
- a layered skin model that accounts for the influences of melanin, haemoglobin and oxygen levels on skin colour.

The internal layers of the human body are represented by a tetrahedral mesh whose edges are used to create a network that loosely represents a vascular system. The tetrahedral mesh allows for a fast and simple haemoglobin transport simulation that is able to capture both the early patchy appearance of the skin and the eventual pooling of red blood cells in the lower lying areas. The skin is represented by an outer melanin layer and an inner haemoglobin layer. Both layers will be rendered individually and then convolved using an approach based on diffusion approximation. The transport simulation affects the haemoglobin saturation in the lower skin layer, which in turn affects the skin colouration. Texture maps are used to specify the absorption and reflection of the melanin layer. The diffuse colour of the haemoglobin layer is obtained from a lookup texture with respect to the haemoglobin saturation and oxygen values. The method is applied to a model of a human arm and two human heads in different positions and scenarios to demonstrate effects such as blanching and fixation of hypostasis during livor mortis. The synthesised images are compared and contrasted to photographs of livor mortis. To the best of the authors' knowledge, there are currently no publications observed in the computer graphics academic literature describing a simulation of livor mortis.

5.2 Biological Background

This section aims to give an overview of the biological make up of blood and skin plus the biological processes behind livor mortis.

5.2.1 Blood Composition

The human body holds between four and six litres of blood, which is spread through the body via the vascular system, see Scanlon and Sanders (2015). Around 55 % of blood is blood plasma, which is mostly water (90 %). The other 45 % of the blood is red blood cells which transport oxygen through the body using the protein haemoglobin. A more detailed blood composition break down is shown in Figure 20. Haemoglobin comes in two oxygen-binding forms, oxygenated (holds oxygen) and deoxygenated (holds no oxygen).

The colour of blood mainly depends on the absorption and scattering properties of the red blood cells, which are reported to depend on the red blood cell's shape, orientation to the light source, haemoglobin concentration and oxygen saturation, see Yim et al. (2012). The optical effect of leucocytes and platelets on skin colour are neglectable according to Yaroslavsky et al. (2002). Blood plasma is a weak absorber and scatterer, but does have some influence on the absorption profile of blood as a whole according to Roggan et al. (1999). Visually, the most important parameters that control the colour of blood are the volume percentage of haemoglobin (haematocrit) and the oxygen saturation in blood as per Yim et al. (2012), where oxygenated blood has a bright red appearance compared to the dark red appearance of deoxygenated blood, as shown in Figure 21.

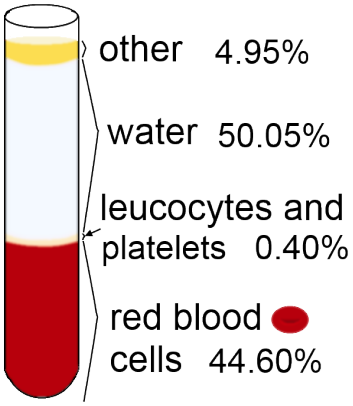
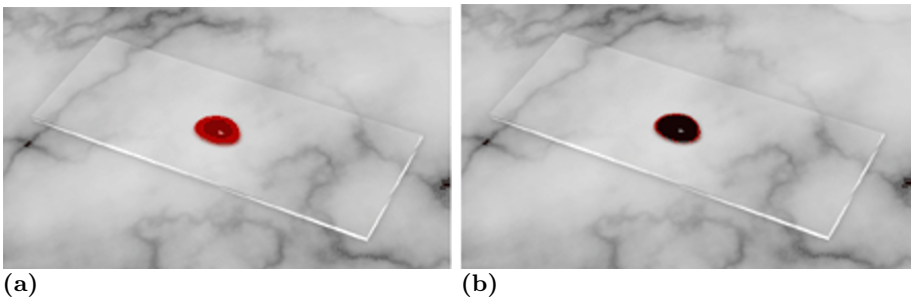


Figure 20: The composition of human blood in percentages.



(a) **(b)**

Figure 21: This image shows render results for a) oxygenated blood and deoxygenated blood by Yim et al. (2012).

5.2.2 Skin Composition

Skin consists of two layers, the epidermis (outer layer) and the dermis (inner layer). The main substances responsible for the skin colouration are the brownish chromophore melanin and the red chromophore haemoglobin. Haemoglobin is transported into the dermis layer of the skin via blood vessels, whereas melanin resides in the epidermis layer. The epidermis can be divided into five sublayers with varying melanin concentration. The melanin concentration and distribution in the epidermis layers determines the skin shade, where more melanin results in a darker skin (The method assumes a uniform distribution of melanin between the epidermis layers) as shown by Igarashi et al. (2007). Haemoglobin on the other hand gives the skin a pink to reddish complexion due to light absorption by haemoglobin being stronger in the blue spectrum than in the red. This is particularly noticeable in light skin with lower melanin content. Light absorption in the red spectral increases with deoxygenated blood. Furthermore, melanin is a high absorber which increases towards shorter wavelength, resulting in more red light being absorbed than blue. This results in a greater decrease of red light compared to blue light when penetrating deep into the tissue. This, together with the decrease of red emission from deoxygenated blood, are the main reasons why deoxygenated blood appears bluish or purple when seen through the skin, see Kienle et al. (1996).

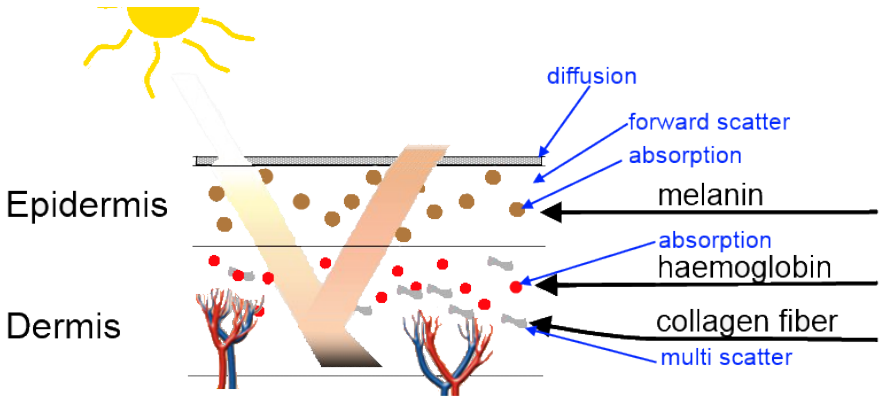


Figure 22: This figure shows the skin composition of the dermis and epidermis layer and the light reflectance and absorption properties. The epidermis is dominated by melanin which is a strong absorber and causes forward scattering. The dermis layer is an absorber (haemoglobin) and strong multi-scatterer (collagen fiber).

5.2.3 Livor Mortis

Livor mortis, also called hypostasis, is one of the earliest signs of death, occurring within a few hours of passing away. The first visual signs, that can appear as early as 30 minutes after death, consist of a patchy appearance of the skin with some areas of pinkish colour and others of pale complexion (see Figure 32a for a photograph showing this). These areas then enlarge to form red/pinkish colouration at low lying areas of the body and a pale one elsewhere, for this see Prahlow and Byard (2011). Figure 33a shows an example of the discolouration called livor mortis. These aforementioned colouration changes are caused by the internal blood dynamics after death. When the heart stops, the red blood cells move downwards under the influence of gravity. This results in the blood pooling in the blood vessels on the lower lying areas of the body, causing discolouration as per Machado (2007). The colour of the blood-filled areas depends on the oxygen saturation of the blood, which decreases over time. Oxygenated blood is bright red, whereas deoxygenated blood is of a darker red, but appears blue or purple through the skin. When pressure is applied to the skin surface, blood is pressed out of affected areas and they appear pale. After around eight to twelve hours, the blood vessels break down and the blood leaks into the surrounding tissue, staining it. At this point the areas of discolouration are fixed and do not change if pressure is applied or the body moved.

5.3 Blood Dynamics

The human body is made of different components that are affected by decomposition in a variety of ways. Flesh decomposes at a higher rate than bones. Skin tends to wrinkle as the internal flesh decays and turns leathery, as described by Dettmeyer et al. (2013). In the case of livor mortis it is essential to differentiate between the different skin layer, flesh and bones, as blood flow does not occur in the bones and the epidermis. Some of the major processes in human body decomposition are internal processes that affect the surface appearance from the inside, such as putrefaction. This is also the case with livor mortis, which is caused by internal blood dynamics which affect the skin appearance. As mentioned above, the colouration of the dermis is greatly effected by the blood content in the tissue. The absorption and scattering properties of blood are dominated by the haemoglobin concentration and its oxygen content. Therefore, the haemocrit and oxygen saturation level are required to simulate the colouration changes in the dermis layer of the skin. The livor mortis is simulated by modelling the changes in haemoglobin concentration and oxygen

saturation after death.

A volumetric representation of the interior is required, which allows the simulation of blood pooling inside the object. The object representation used for livor mortis was presented in Chapter 3. To summarise, the skin is represented by a triangle surface mesh and the internal components, such as flesh and bones, by a surface aligned tetrahedral mesh. Both layers are connected by tracking springs that connect a skin node to the underlying tetrahedral boundary face. For this simulation a body is considered to be made of bones, flesh and skin. For the livor mortis simulation a surface mesh representation has rendering advantages as it allows the use of texture maps to specify skin reflectance and transmittance properties, as well as high surface details. The tetrahedral nodes hold simulation parameters such as blood capacity, haemoglobin content and oxygen saturation. This permits different materials to be represented by varying the simulation parameters. Bones are differentiated from flesh by assigning zero blood capacity to all bone nodes, thus preventing blood from moving through the bone volume.

Tetrahedral edges, where both nodes have a non-zero blood capacity, are able to transport haemoglobin. The network formed by these edges can be thought of as the vascular system, as it allows the transport of haemoglobin through the volumetric object. The edge network does not match a vascular system exactly, but its irregularity results in similar visual effects. Using the same method on a more regular volumetric representation, such as a voxel structure, would result in an even discolouration, which is not representative of the real world phenomenon in question. Tetrahedralisations tend to generate smaller tetrahedra around boundaries and larger elements in the interior of the mesh. This would result in shorter edges (blood vessels) with smaller blood capacity at the boundaries. However, this has been shown not to be an issue as a similar phenomenon can be observed in the vascular system of the human body. Blood vessels inside the body are large and become smaller towards the boundary. The skin layer receives haematocrit and oxygen saturation information from the tetrahedral mesh using the tracking springs (see Section 5.4). A texture map for the skin is required that represents the melanin distribution on the object for rendering. An additional texture map can be used to model the effects of small blood vessels in the dermis layer (Section 5.4.2).

5.3.1 Initial Set-up

The internal volume is represented by a tetrahedral mesh, where each node has a maximum blood capacity and can hold an amount of haemoglobin and oxygen.

At the start of the simulation the initial blood capacity is defined by the user to be between 0 and 1. This allows some nodes to be treated as part of the vascular system (blood capacity > 0) and others not (blood capacity $= 0$), such as in the case of bones. Each node's blood capacity needs to be corrected so that they are representative of the even blood distribution in the body.

For this the method first computes the average volume of all tetrahedra surrounding a node to represent a node's volume share. Blood is then distributed over all nodes relative to both their volume share and the blood capacity set by the user. The average of the volume is used instead of the sum, to avoid boundary nodes incorrectly receiving less blood than internal nodes. The total blood capacity for each node is then:

$$c_i = \frac{V_i M_i}{\frac{n}{\sum_{j=1} V_j}} b \tag{5.1}$$

where c_i is the node's blood capacity, M_i is the initial blood capacity of the node (1 for flesh, 0 for bones), V_i is the average volume of the tetrahedra surrounding the node, b is the total amount of blood and n the number of tetrahedral nodes with non-zero blood capacity. Initially, the haemoglobin content h per node makes up 45 % of the blood, hence:

$$h_i(0) = 0.45 c_i \tag{5.2}$$

where $h_i(0)$ is the haemoglobin content of node i at time 0. Additionally, a texture map can be used to change the blood capacity of boundary nodes to reproduce visual effects caused by the small blood vessels in the dermis. This will be described in more detail in Section 5.4.2.

5.3.2 Haemoglobin Transport

When the heart stops pumping blood through the body the blood plasma ceases to flow through the veins. Since the blood plasma is no longer in motion, advection of the red blood cells by blood plasma can be ignored. This means that the driving factor in the cells' movement is gravity and can therefore be described by diffusional sediment (hillslope) flow. The discolouration of the skin is a result of the chromophore haemoglobin carried by the red blood cells. The haemolgobin transport approach simulates the haemoglobin transport in

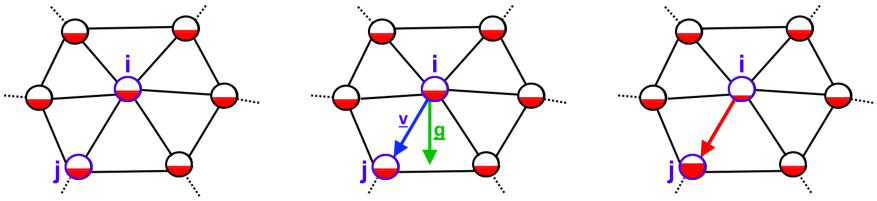


Figure 23: This Figure shows the haemoglobin transport process. The amount of haemoglobin transported from node i to node j (a) is proportional to the negative gradient of v with respect to gravity g (b). The result is shown in c).

the veins by gradually transferring haemoglobin along the tetrahedral edges to approximate diffuse sediment flow.

Each blood vessel node i has one or more blood vessel(s) connecting it to neighbouring nodes $j \in \Omega_i$, where Ω_i is the set of all nodes connected to i by an edge. The haemoglobin transfer between two nodes (see Figure 23a) is governed by the following equations:

$$h_i(t+\Delta t) = h_i(t) + \Delta t\Big(\sum_{j\in\Omega_i} h_{ji}(t+\Delta t) - \sum_{j\in\Omega_i} h_{ij}(t+\Delta t)\Big) \tag{5.3}$$

where $h_i(t+\Delta t)$ is the unfixed haemoglobin content of node i at time $t+\Delta t$ and $h_{ij}(t+\Delta t)$ is the amount of haemoglobin transferred from node i to node j defined as:

$$h_{ij}(t+\Delta t) = min(\lambda_h c_i, h_i(t))\frac{\tau_{ij}}{\sum_{k\in\Omega_i}\tau_{ik}} \tag{5.4}$$

where λ_h is a user defined haemoglobin transport rate. τ_{ij} determines the proportion of haemoglobin node j receives from node i:

$$\tau_{ij} = \begin{cases} \delta_j \dfrac{(\hat{\mathbf{g}}\cdot\hat{\mathbf{v}}_{ij})+1}{2} & \text{if } \hat{\mathbf{g}}\cdot\hat{\mathbf{v}}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

where $\hat{\mathbf{g}}$ is the unity gravity vector and $\hat{\mathbf{v}}_{ij}$ is a unit vector from the position $\mathbf{p_i}$ of node i to the position $\mathbf{p_j}$ of node j. $\delta_j = c_j - h_j(t) - f_j$, where f_j is the fixed haemoglobin in node j explained in Section 5.3.5.

Here, the term $\delta_j(\hat{\mathbf{g}}\cdot\hat{\mathbf{v}}_{ij})$ simulates the movement of haemoglobin due to gravity. This is based on the diffusional sediment flux described in Roering et al. (1999).

Hence the haemoglobin transport h_{ij} is proportional to the negative gradient of $\mathbf{v}_{ij}$ and the space available in node j (see Figure 23b. The red blood cells are more dense than the plasma they sit within. Therefore, to more accurately simulate the movement of the sinking red blood cells through the vascular system, the transport rate is mapped onto the $(0.5, 1)$ range with the term $\frac{(\hat{\mathbf{g}} \cdot \hat{\mathbf{v}}_{ij}) + 1}{2}$. This effects greater horizontal movement to account for the red blood cells that are still suspended and have not sunk to the bottom of the blood vessel (or those being pushed by other red blood cells) and therefore are able to move along horizontal edges more easily.

To begin with, the blood distribution is even over all nodes but, due to the varying blood capacities, some nodes empty or fill up faster than others. This is what causes the patchy appearance at the beginning of the simulation, which has also been observed at the beginning of the livor mortis phenomenon as described in Prahlow and Byard (2011).

The haemoglobin transport rate λ_h is inspired by the transport rate in sediment transport problems by fluid flow or gravity forces. Because of constraints imposed by the complexity of blood flow and haemoglobin transport in veins, the haemoglobin dynamics after death are approximated by Equation 5.4. Since the veins are not modelled as a physically precise pipe system the haemoglobin transport rate was chosen empirically. Furthermore, the haemoglobin transport rate is exposed to the user in order to allow more artistic control over the appearance of livor mortis. Small rates result in slower accumulation of the blood in lower lying areas of the body but will strengthen the patchy appearance of early livor mortis.

5.3.3 Oxygen Dissociation

The oxygen levels in blood reduce over time. This is characterised by the oxyhaemoglobin dissociation curve (ODC). The ODC relates the oxygen saturation of haemoglobin to the partial pressure of oxygen in the blood, which can be described by a sigmoid plot. The Kelman (1966) routine is used to convert the oxygen tension to oxygen saturation:

$$o_i(t + \Delta t) = \frac{a_1 p_i + a_2 p_i^2 + a_3 p_i^3 + p_i^4}{a_4 + a_5 p_i + a_6 p_i^2 + a_7 p_i^3 + p_i^4} \tag{5.6}$$

where p_i is short for $p_i(t + \Delta t)$ and refers to the partial pressure of oxygen at

node i, which is defined as

$$p_i(t + \Delta t) = \rho(t + \Delta t)\frac{h_i(t) + f_i(t)}{c_i} \tag{5.7}$$

with

$$\rho(t + \Delta t) = \rho_i(t) - \Delta t \lambda_o \tag{5.8}$$

where λ_o is a user defined constant that controls the oxygen tension decline rate and $a_1, ..., a_7$ are constants described in Kelman (1966). The oxygen tension decline rate solely exist to control the appearance of livor mortis during deoxygenation. During experimentation it was found that without an oxygen tension decline rate (or when $\lambda_o \geq 1$) the discolouration due to oxygen loss appears early and sudden, which might not be desired by the user. A lower oxygen tension decline rate, on the other hand, allows for a more gradual change in colouration. Figure 27c shows the internal blood dynamics inside the tetrahedral mesh. With the oxygen level in the blood decreasing over time, the blood turns deep red.

5.3.4 Blanching

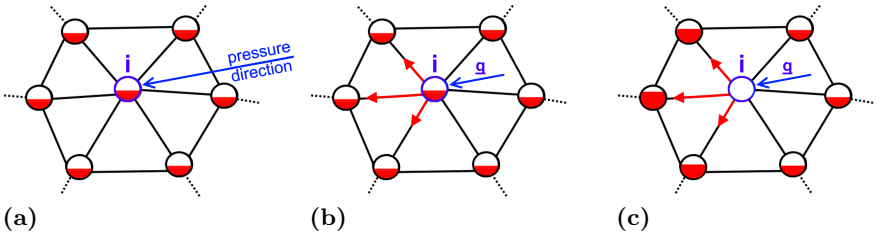


Figure 24: This figure shows the haemoglobin dynamics when applying pressure to a node i (a). The blood is pressed out in the pressure direction q and into the surrounding nodes as shown in b). This results in all haemoglobin from node i being distributed into the surrounding nodes in the pressure direction, where the amount transferred depends on the angle between the edge and the vector q (c).

Pressure induced blanching is the pale discolouration of skin where pressure is applied, for example with a finger or due to contact with a surface. The blood capillaries are compressed which results in blood being forced out. After livor mortis has become fixed (see Section 5.3.5), applying pressure to an area affected by livor mortis will not show any blanching effects. Figure 36 shows a photograph of pressure induced blanching.

Neyret et al. (2002) address blanching effects in surgery simulation, where a medical instrument exerts pressure on an organ's surface. To mimic pressure induced blanching a semi-transparent white disc is drawn into an effects texture at contact point. The effects texture is then combined with the skin and shading textures to achieve the whitening effects. Their method gradually reduced the contribution of blood to the organ's colouration by enlarging the white disc. This work does not model any underlying blood dynamics that would move the blood into the surrounding tissue. This means the blood and its contribution to the skin colouration is lost. Instead, blanching is simulated by reducing the blood capacity of affected nodes which forces the haemoglobin to move into neighbouring ones.

Aside from the blood capacity c_i that represents the maximum blood capacity a node can hold, a second variable a_i specifies the available blood capacity. c_i stays static throughout the simulation, after it has been initialised by Equation 5.1. The available blood capacity a_i is reduced or increased relative to the pressure applied to or relieved from the affected node. During the haemoglobin transport, the available blood capacity a_j is used to determine whether and how much haemoglobin can be moved to node j. As such, δ in Equation 5.5 becomes:

$$\delta_j = \Big(a_j - h_j(t) + min(0, a_j - c_j - f_j)\Big) \tag{5.9}$$

Fixed haemoglobin is not affected by blanching, and can therefore fill up the whole capacity c_j of node j, hence the term $min(0, a_j - c_j - f_j)$, that only considers fixed haemoglobin that exceeds the difference between the maximum and the available capacity.

When pressure is applied, the available capacity of every affected boundary node is set to 0. The available blood capacity of non-boundary nodes is linearly reduced according to their distance to the pressure object. Haemoglobin is pushed out of affected nodes according to the pressure direction. Haemoglobin transport due to pressure is as in Equation 5.4 but with τ_{ij} replaced by

$$\omega_{ij} = \begin{cases} \delta_j(\hat{\mathbf{q}} \cdot \hat{\mathbf{v}}_{ij}) & \text{if } \hat{\mathbf{q}} \cdot \hat{\mathbf{v}}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

where $\hat{\mathbf{q}}$ is the unit pressure direction. This moves haemoglobin along edges that leave node i in the pressure direction $\hat{\mathbf{q}}$ (see Figure 24).

5.3.5 Fixation of Hypostasis

Fixation of hypostasis refers to the fixation of livor mortis due to blood leaking through deteriorated blood vessels into the surrounding tissue. This staining of the tissue results in the fixation of the discolouration that remains even if pressure is applied or the body is moved. This is accounted for by introducing the variable f_i into the simulation, which is the fixed haemoglobin amount at node i that is not moved by the haemoglobin transport simulation.

The haemoglobin h_i at node i is gradually turned into fixed haemoglobin f_i. In addition to this, an amount of its haemoglobin is transported to all neighbouring nodes $j \in \Omega_i$ and fixed there. The amount j receives is related to its distance to node i:

$$f_i(t + \Delta t) = f_i(t) + \Delta t\Big(f_{ii}(t + \Delta t) + \sum_{j \in \Omega_i} f_{ji}(t + \Delta t)\Big) \tag{5.11}$$

$$h_i(t + \Delta t) = h_i(t) - \Delta t\Big(f_{ii}(t + \Delta t) - \sum_{j \in \Omega_i} f_{ij}(t + \Delta t)\Big) \tag{5.12}$$

where f_{ij} is the amount of haemoglobin that leaks from node i into the surrounding tissue of node j. Some haemoglobin at node i is fixed at node i directly (f_{ii}) and some is fixed at neighbouring nodes j (f_{ji}). All haemoglobin transported is removed from node i (Equation 5.12).

$$f_{ij}(t + \Delta t) = min(\lambda_h c_i, h_i(t)) \frac{\upsilon_{ij}}{\sum_{k \in \Omega_i} \upsilon_{ik} + \upsilon_{ii}} \tag{5.13}$$

υ_{ij} determines the proportion of haemoglobin from node i that leaks into the surrounding tissue of node j and is defined as:

$$\upsilon_{ij} = \begin{cases} \delta_j\Big(1 - \dfrac{|p_j - p_i|}{L_i}\Big) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{5.14}$$

where $|p_j - p_i|$ is the length of the edge connecting i and j and L_i is the length of the longest edge leaving node i. The amount of haemoglobin node j receives depends on its available blood capacity and is negatively related to its distance to node i, i.e. the smaller the distance, the more haemoglobin node j receives.

The haemoglobin content of the boundary nodes is used in the skin shading approach discussed in the next section. In skin shading, haemoglobin refers to

the sum of fixed haemoglobin f_i and unfixed haemoglobin h_i.

param	description	range	value
b	total blood amount	$[10^3, 10^8]$	$5000ml$
Δt	time step	$[1, 10^{-4}]$	$1/3min$
λ_h	haemoglobin transport rate	$[0.99, 10^{-6}]$	$0.08m/min$
λ_o	oxygen tension decline rate	$[10^{-3}, 10^{-10}]$	$10^{-6}mmHg/min$

Table 5.1 This table is an overview of the framework variables used for the blood dynamics. The third column holds acceptable ranges for the parameters and the last column shows the values used to generate the results portrayed in section 5.5.

The haemoglobin transport algorithm transfers an amount of haemoglobin along edges connecting two nodes. Hence, the computational complexity of the haemoglobin transport depends on the number of edges that connect two flesh nodes, and can be described as $O(V)$, where V is the number of edges.

5.4 Skin Shading

Skin colour depends mainly on the melanin concentration in the epidermis and haemoglobin concentration in the dermis layer. Livor mortis is visible due to changes in the haemoglobin concentration and blood oxygen saturation in the dermis layer. There have been a number of approaches in skin shading that consider the components responsible for skin colouration, namely melanin and haemoglobin. See, for example, Igarashi et al. (2007). Methods that consider the haemoglobin impact on the skin colour represent the skin in layers with different reflectance and transmittance profiles, such as Krishnaswamy and Baranoski (2004); Donner and Jensen (2005); Donner et al. (2008); Ghosh et al. (2008). These layers usually represent the epidermis (melanin) and dermis/bloody dermis (haemoglobin) layer. They do not consider time-varying haemoglobin distribution. Iglesias-Guitian et al. (2015) concentrate on the optical properties of skin ageing due to changes in the chromosphere concentration that are caused by the thinning of the dermis and epidermis.

When simulating livor mortis the different light reflection and absorption properties of oxygenated and de-oxygenated blood need to be considered, as well as the blood distribution and light attenuation of the outer skin layers. Skin colour can be determined with respect to the haemoglobin and melanin con-

tent using a two-dimensional look up texture as demonstrated by Jimenez et al. (2010). They use texture maps to specify the haemoglobin distribution over a human's face. The skin colour at a given point is retrieved from a texture map using the melanin and haemoglobin amounts as a uv-coordinate. The haemoglobin distribution varies with the emotion the face is displaying. For this, a haemoglobin histogram is constructed from in vivo measurements of haemoglobin distribution during six different emotions. Livor mortis visualisation, on the other hand, requires the haemoglobin distribution to correspond to internal blood dynamics and oxygen levels which are not considered in any of the methods above. As such, a two-dimensional look up texture is not sufficient in the case of colouration changes due to livor mortis.

Instead the model is represented in two layers, similar to the skin model in Donner et al. (2008) and render each layer individually. The first, or outer layer, represents the epidermis. The second, or inner layer, represents the dermis. The skin shader uses a diffuse approximation approach, based on the Jimenez et al. (2015) separable subsurface scattering method to approximate the diffusion profile of skin but apply this to each layer individually in screen space, see Jimenez et al. (2009). This has performance and artist control advantages over more biologically sound methods such as Chen et al. (2015), making it appropriate for use in video game development. The results of the two layers are then convolved in a post-processing step to obtain the final skin colour.

The livor mortis skin shading approach can be summarised as:

1. Render the epidermis diffuse map (Section 5.4.1).
2. Render the dermis diffuse map (Section 5.4.2).
3. Apply the epidermis specific diffusion profile to the rendered epidermis diffuse map (Section 5.4.3).
4. Apply the dermis specific diffusion profile to the rendered dermis diffuse map (Section 5.4.3).
5. Convolve the blurred epidermis and dermis maps (Section 5.4.3).

5.4.1 Render Epidermis

When rendering the epidermis, only the melanin contribution to the skin colour is considered. In the case where a conventional skin albedo map is used, the haemoglobin contribution needs to be removed. Alternatively, highly-detailed melanin maps can be constructed from measured data as in Jimenez et al. (2010), and a look up texture used to determine the skin colour due to melanin.

Alternatively, methods described by Tsumura et al. can be used to extract the melanin and haemoglobin information from the skin. For this see Tsumura et al. (2003). The results presented in this chapter were constructed using manually modified albedo textures.

5.4.2 Render Dermis

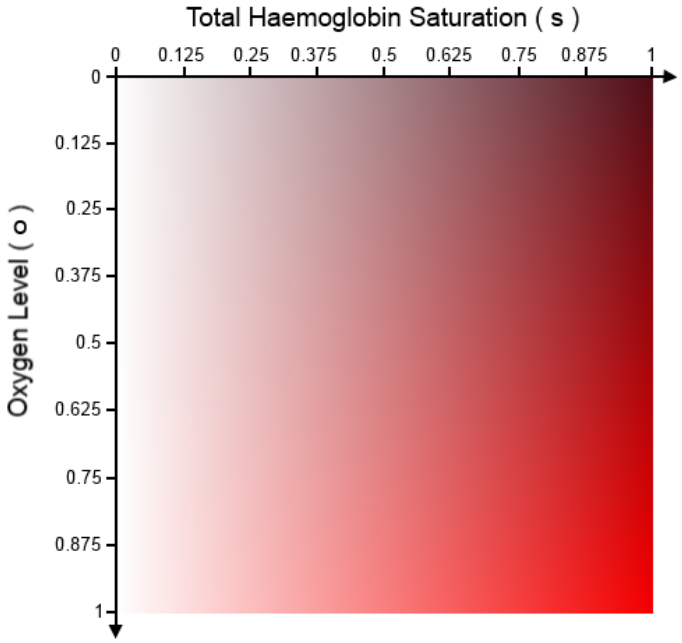
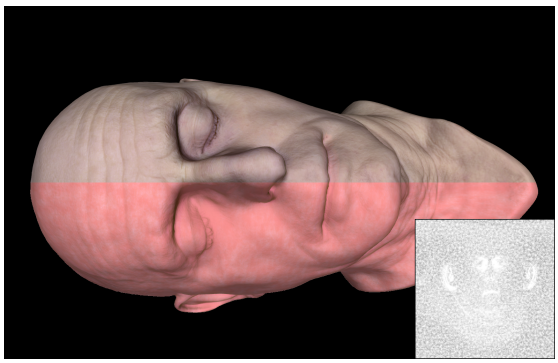


Figure 25: Blood colour lookup texture used in the dermis rendering step. The total haemoglobin saturation s is the fraction of haemoglobin in blood which is obtained using Equation 5.15. The oxygen level in blood o is obtained using Equation 5.6.

Haemoglobin is the main chromophore found in the dermis. It is transported by small blood vessels reaching within the dermis. A greyscale texture map can be used to imitate the colouration effects over the skin caused by these blood vessels. This is achieved in two steps, at the initialisation stage and the rendering stage.

During initialisation the available blood capacity a_i of each tetrahedral boundary node i is modified by the texture map, where white indicates full available capacity and black indicates no available capacity. This will influence the haemoglobin transport described in Section 5.3 for the boundary nodes.

During rendering the haemoglobin saturation (haematocrit) is adjusted. Hae-



(a)

Figure 26: Example of a blood vessel texture is shown in the square on the right lower corner. The dermis results are shown at the bottom and epidermis results at the top.

moglobin saturation $s(x, y)$ and oxygen level $o(x, y)$ determine the colour for each point using a blood colour look up table, as shown in Figure 25. Here, haemoglobin saturation refers to the ratio of total haemoglobin to capacity. The dermis shader receives blood parameters $\left(\frac{h_i+f_i}{a_i}, o_i\right)$ which determine the haemoglobin saturation $h(x, y)$ and oxygen level $o(x, y)$ at each pixel (x, y). In order to vary the dermis colouration over the whole surface according to the texture map the total haemoglobin saturation $s(x, y)$ for each pixel is computed as follows:

$$s(x, y) = h(x, y) \cdot m\big(u(x, y), v(x, y)\big) \tag{5.15}$$

where m is the available capacity ratio that is obtained from the texture map at uv-coordinates $u(x, y)$ and $v(x, y)$ of pixel (x, y). Note that the unfixed haemoglobin saturation given in the blood parameters is $\frac{h_i+f_i}{a_i}$ rather than $\frac{h_i+f_i}{c_i}$. This is done in order to correct the unfixed haemoglobin saturation for pixels within the triangle. Since $a_i = m(u_i, v_i)c_i$, m becomes zero at each vertex. As expected, this results in the saturation being the ratio of total haemoglobin to capacity.

5.4.3 Convolve Layers

In order to obtain the final skin colour, the two layers need to be convolved. The dermis and epidermis have different absorption and reflectance properties. The dermis acts as a strong scatterer mostly due to its thickness and collagen fibre network, whereas multiple scattering in the epidermis is negligible and occurs

skin layers	variance	weights		
		r	g	b
epidermis	0.0064	0.233	0.455	0.649
	0.0484	0.100	0.336	0.344
dermis	0.1870	0.118	0.198	0.000
	0.5670	0.113	0.007	0.007
	1.9900	0.358	0.004	0.000
	7.4100	0.078	0.000	0.000

Table 5.2 Sum-of-Gaussians parameters for the epidermis and dermis. The Gaussian parameters were taken from D'Eon and Luebke (2007).

mainly in the forward and backward direction, for this see Igarashi et al. (2007).

Two convolution kernels are constructed, one for the epidermis and one for the dermis. The convolution kernel for the thin epidermis is constructed in a similar way to Jimenez et al. (2015) but using only the first two Gaussian terms (see Table 5.2).The outgoing radiance of the epidermis is then described by the diffusion profile:

$$\mathbf{R}_e(x) = \sum_{i=1}^{2} \mathbf{w}_i G(v_i, x) \tag{5.16}$$

where $\mathbf{R}_e(x) = [R_r(x), R_g(x), R_b(x)]$ is the convolution profile for the epidermis, $G(v_i, x)$ the Gaussian with variance v_i and $\mathbf{w}_i = [r_i, g_i, b_i]$ are the weights for the rgb channels (See Table 5.2).

The convolution profile of the dermis $\mathbf{R}_d(x)$ is constructed similarly but, since significant multi scattering is happening in the dermis, it is formed from the last four Gaussians:

$$\mathbf{R}_d(x) = \sum_{i=3}^{6} \mathbf{w}_i G(v_i, x) \tag{5.17}$$

$\mathbf{R}_e(x)$ and $\mathbf{R}_d(x)$ are then applied to the rendered epidermis and dermis layer respectively. The resulting blurred maps represent the light reflection of the two layers. Figure 27a shows the results for the epidermis and Figure 27c shows the results for the dermis at different stages of livor mortis. Note that the rgb weights in Table 5.2 are normalised, so that the colour of the skin is controlled by the diffuse skin texture and blood parameters in the dermis. This allows for artist control over the skin colouration. D'Eon and Luebke (2007) provide a discussion on this.

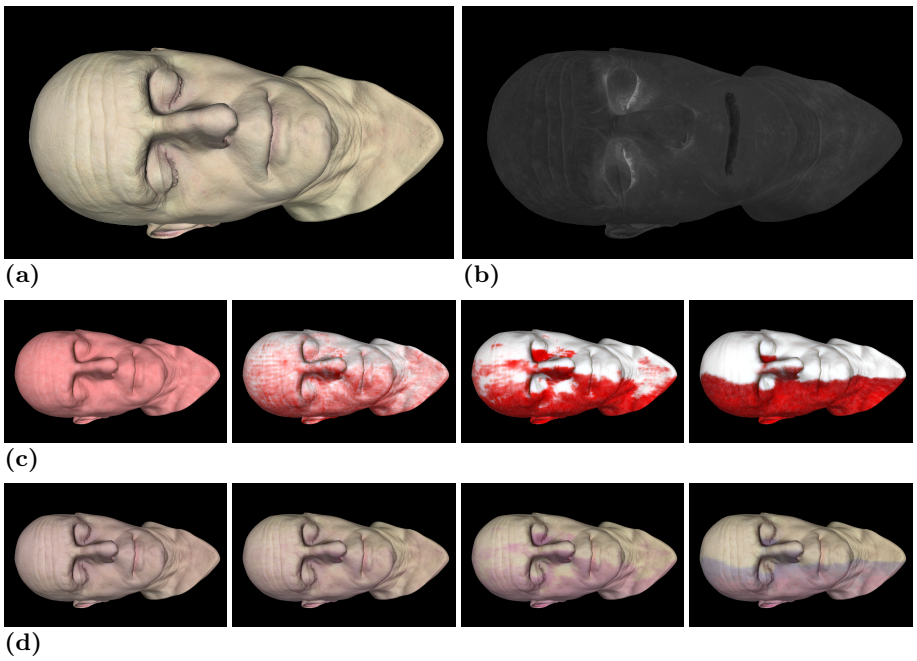


Figure 27: This figure demonstrates the proposed skin shading approach. a) shows the epidermis rendered using Equation 5.16 using a diffuse skin texture and b) epidermis absorption. c) shows the dermis layer render using Equation 5.17 at different stages of livor mortis. d) shows the final skin shading obtained by combining a-c) using Equation 5.18.

The chromophore melanin is highly absorbent and, therefore, the absorption in the epidermis must also be considered. For this the skin shader uses a melanin map (see Figure 27b), which is a greyscale texture showing melanin contribution, from black (no melanin) to white (full melanin). With this the final skin colour can be determined as follows:

$$\mathbf{c} = \mathbf{w}_1\mathbf{L}_e + (1 - \mathbf{A}_e) \circ (\mathbf{w}_2\mathbf{L}_e + \sum_{i=3}^{6} \mathbf{w}_i\mathbf{L}_d) \tag{5.18}$$

where $\mathbf{L}_e$ and $\mathbf{L}_d$ are the reflections of the epidermis and dermis respectively, i.e. the blurred epidermis and dermis render results. $\mathbf{A}_e$ is the absorption by melanin given by a greyscale melanin map. Note that $\circ$ describes the Hadamard product. The first term describes the light that is directly reflected from the epidermis. This helps to preserve the surface details. $(1 - \mathbf{A}_e)$ represents the light that is not absorbed by the melanin layer and therefore reaches into the epidermis layer. Note that light that is reflected from the epidermis does not reach the dermis layer either, but this is accounted for with the Gaussian weights $\mathbf{w}_i$ when summing the reflection contributions for each layer.

5.5 Results and Discussions

In order to demonstrate the livor mortis simulation approach proposed in this chapter a number of examples have been generated of the simulation on different models and scenarios to demonstrate haemoglobin dynamics, skin shading, fixation of hypostasis and, finally, blanching. The results will be compared to photographs of livor mortis. Unfortunately, no time-lapse footage of livor mortis was available for a direct comparison of the different stages. Instead, photographs of different livor mortis stages from different bodies and body parts are used for comparison. No useful data on livor mortis was available for a quantitative evaluation. I used a PC with an Intel Core i7 CPU running at 3.40 GHz, with 16 GB of RAM and a NVIDIA GeForce GTX 760 graphics card. The proposed method was implemented using C++ and DirectX 11. All the tetrahedralisations were generated using the TetGen tool by Si Si (2015). The simulation was run on three different models, each with an internal bone structure. The models include a human arm and two human heads of different ages, as depicted in Figure 28. The examples showing an elderly human head (see Figure 28c) were generated using the free 3D model from TEN24-Digital-Capture (2016). The slower rendering on the head model (old) is due to using a higher resolution and multisampling as opposed to the arm and younger head examples.

See Table 5.3 for simulation statistics of the three models. The parameter values depicted in Table 5.1 were used to generate all results shown in this section. All models took around 11 simulated hours (i.e. 2000 simulation steps) until complete fixation of hypostasis and deoxygenation occurred. Table 5.1 also provides suggested ranges for the parameters at which the simulation generates results correlating to the appropriate physical processes. Lower values can be chosen for the time step, haemoglobin transport rate and oxygen tension decline rate, but they do not visibly improve the results and merely lead to longer running times. Higher values, on the other hand, can generate less realistic results. A high haemoglobin transport rate reduces the early patchy appearance of livor mortis and a high oxygen tension rate leads to the colouration turning purple too early and thereby generating unrealistic results. Upper ranges were chosen by increasing the parameter values until they produced unacceptably unrealistic results. Lower ranges were chosen by reducing the values until no discernible changes in quality were observed. Then, combinations of upper and lower ranges of the different parameters were tested.

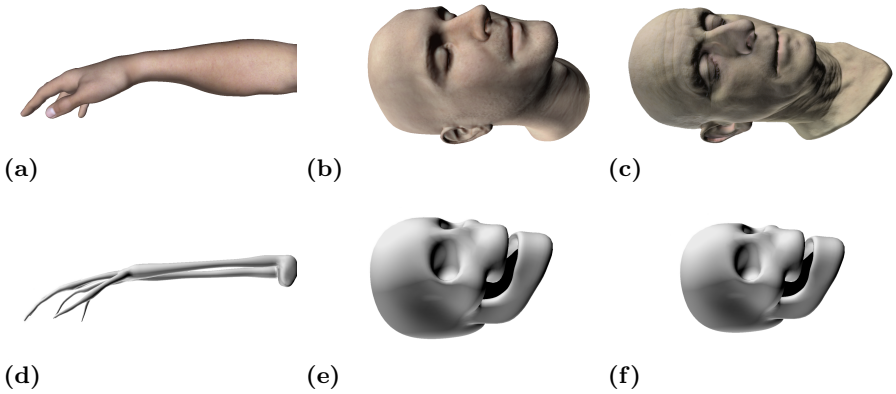


Figure 28: The input meshes used for the simulation. The top row shows the skin layer and the bottom row the bone layer triangle meshes.

	arm model	head model	head model (old)
model size			
tetrahedral node count	29k	29k	85k
flesh node count	28k	23k	83k
tetrahedra count	141k	155k	313k
surface triangle count	19k	10k	122k
timings			
ms per frame	70 ms	43 ms	510 ms
ms per simulation step	59 ms	38 ms	310 ms

Table 5.3 This table shows some geometry information for the two models used in terms of node and polygon counts in thousands (k), plus performance statistics of each livor mortis simulation step and each frame (including skin rendering) in milliseconds (ms).

5.5.1 Haemoglobin Transport

The heavy red blood cells that transport haemoglobin sink down due to gravity. This causes a discolouration of the skin where lower areas turn pink and higher areas turn pale. The haemoglobin dynamics over time can be observed in Figure 27c. The resulting skin discolouration is depicted in Figure 27d, which shows that the internal blood dynamics influence the skin colouration. The results of applying the livor mortis simulation on the lower resolution head model are shown in Figure 29. The top row shows the blood dynamics in the dermis layer, demonstrating that the blood does indeed move to the lower lying areas of the head. The bottom row shows the resulting changes in skin colouration. Figure 30 and Figure 31 show the results of the livor mortis simulation on the model of an arm in a lying and hanging position (respectively). In both cases,

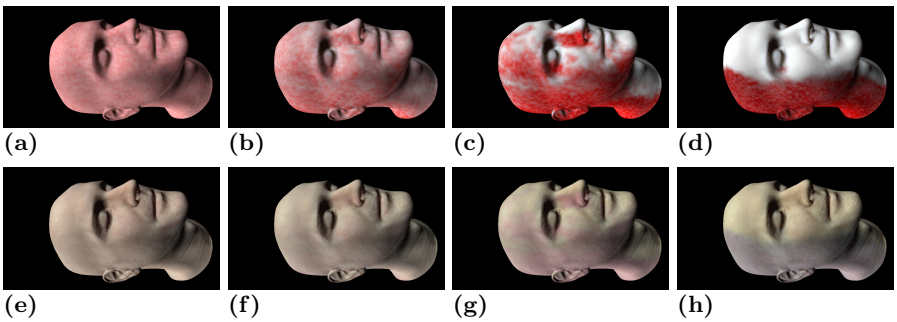


Figure 29: Livor mortis is applied to the model of a human head (young). a) shows the internal blood dynamics and b) the skin shading results.

the livor mortis is visible in the lower lying areas of the model. In the real world the whole forearm in Figure 30c and Figure 30d would be full of blood as it would also contain blood from other parts of the body. For the examples only a model of a forearm was used and, as such, the method can only distribute the blood that is present in the forearm at the start of the simulation resulting in less blood in the arm. Similarly, the head models are closed models that are not connected to the rest of the body. If positioned upright, blood will accumulate in the neck and chin, whereas in a real world body blood would flow out of the head into the rest of the body, leaving the head deprived of blood.

Haemoglobin transport is subject to gravity. This means that when an object is moved before fixation of hypostasis, lividity will change accordingly as demonstrated in Figure 34. Haemoglobin accumulates in the lower areas in Figure 34a. Then, the arm is rotated 180° before livor mortis becomes fixed, which is depicted in Figure 34b. As the arm was turned early on, all haemoglobin moved into the newly lowered areas. This shows how livor mortis is influenced by gravity.

5.5.2 Skin Colouration

The skin colouration is influenced by the internal blood dynamics and the oxygen saturation of haemoglobin. Areas full of blood start turning pinkish and later purple, while the higher lying areas turn pale. Figure 27 shows the different rendering layers during livor mortis. Figure 27c depicts the dermis layer that reflects the internal blood dynamics. A blood vessel texture described in Section 5.4.2 is used in all examples. It was generated using 2D Perlin noise to add roughness to the otherwise smooth appearance of livor mortis. Perlin noise images can be quickly generated by most common image processing applications. Furthermore, Perlin noise has the advantage of being scalable, allowing it

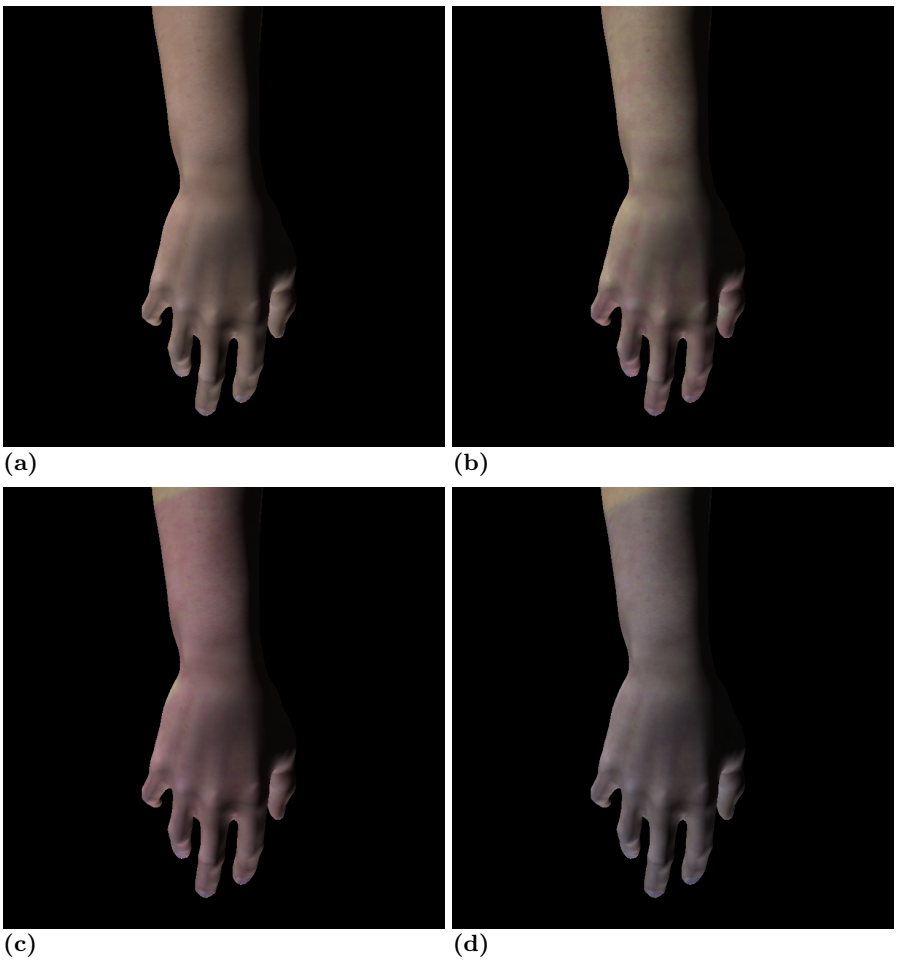


Figure 30: Livor mortis is applied to an arm in hanging position as in a). Early livor mortis results in a patchy discolouration shown in b). c) haemoglobin pools in the hand showing a pink complexion which turns purple with decreasing oxygen as shown in d).

to be fitted to the object's world space dimensions. The vessel texture leads to a smoother transition between haemoglobin rich and haemoglobin deprived areas in order to prevent polygonal edges on the boundary and allow for more artistic control. On the other hand, it can lead to a patchy appearance in haemoglobin rich areas as can be observed in Figure 34d. Figure 32 and Figure 33 compare the skin colouration results from the livor mortis method to photographs of real livor mortis. As an alternative to a Perlin noise texture, artists can generate a more realistic blood vessel texture map by hand, if more realism is required.

At the start of livor mortis the skin looks patchy, as shown in the photograph in Figure 32a. The irregular edge network causes a similar patchy appearance at

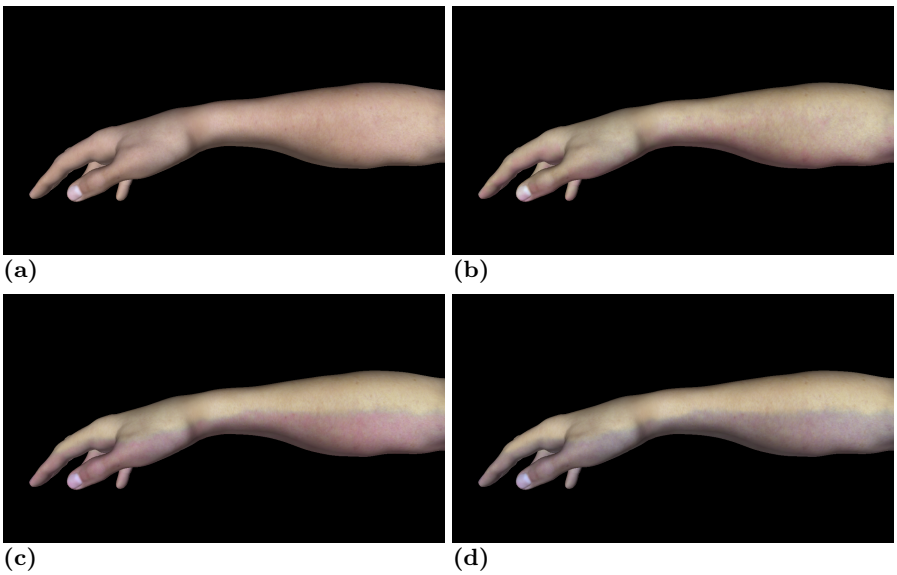
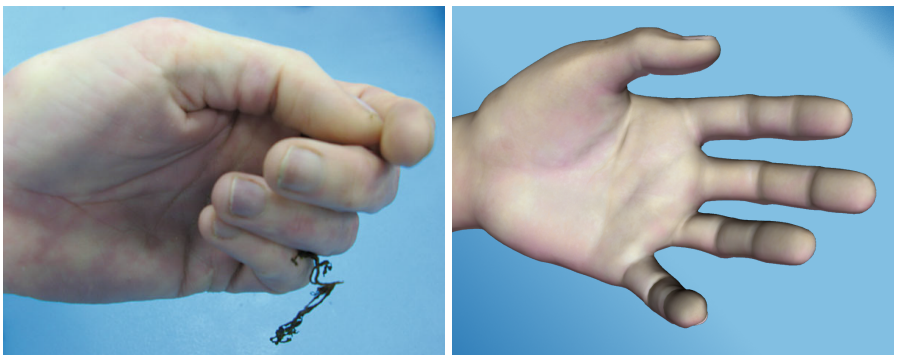


Figure 31: Livor mortis is applied to an arm in lying position as in a). This results in haemoglobin gradually accumulating in the finger tips and lower arm areas shown in b-c). d) shows the bluish discolouration due to oxygen loss.

the start of livor mortis, which is an intended side effect of using the tetrahedral edges as a vascular system. This is particularly visible in Figure 32b, which shows a similar pattern to Figure 32a, but can also be observed in the other results.

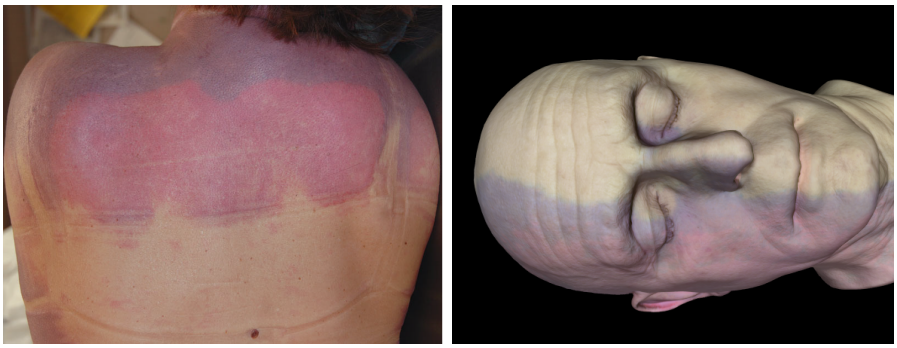
With ongoing livor mortis, haemoglobin accumulates in the lower areas of the object, turning the areas a pinkish colour. Figure 33a shows a photograph of livor mortis, with pink and purple discolourations. Figure 33b shows a simulation result. The purple colour on the top is due to the reduced oxygen in the blood that results in a deeper red colour than oxygenated blood. This can be observed in Figure 27d, which shows how the skin colour turns more purple as the blood turns a deeper red (Figure 27c) due to oxygen dissociation.

The pattern and colouration of livor mortis created by the simulation are very similar to the ones in the photograph. The lividity in Figure 33a has blotchy characteristics that can also be observed in the simulation results (see Figures 27d, 30c, 30d, 31c and 31d). Though more visible in some of the examples than on the photograph, this can be tuned by adjusting the blood vessel texture.



(a) (b)

Figure 32: The patchy appearance of early livor mortis in a) a drowned victim taken from Prahlow and Byard (2011) and b) the simulation results.



(a) (b)

Figure 33: a) shows a photograph that highlights the pink and purple colourations of advanced livor mortis from Prahlow and Byard (2011). Note that the colouration differences on the back are due to changes in the environment temperature during livor mortis. b) shows the colours of advanced livor mortis with high (bottom) and low (top) oxygen saturation.

5.5.3 Fixation of Hypostasis

When the blood vessels decompose, haemoglobin flows out and stains the surrounding tissue. At this point lividity becomes fixed and is unaffected by pressure or movements of the body. Turning the arm displayed in Figure 34a during early livor mortis results in lividity shifting to the newly lowered areas (see Figure 34b). Turning the object during fixation (see Figure 34c) on the other hand shows only a slight reallocation of lividity as some colouration intensity is lost. Turning the object after livor mortis has become fixed, as shown in Figure 34d, shows that lividity is visible at the same areas as in its initial position in Figure 34a, i.e it has not been affected by gravity.

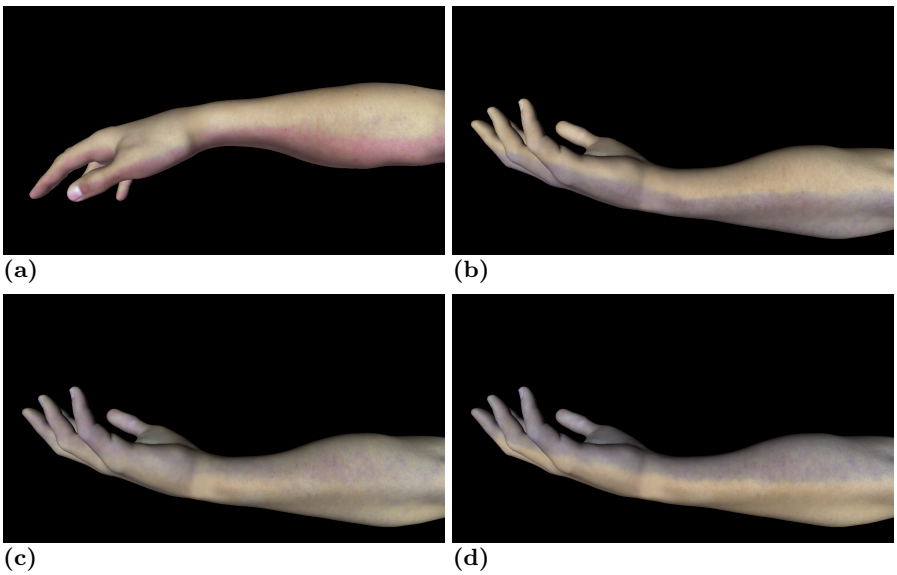


Figure 34: this figure shows an arm with a) early livor mortis in an initial position that was turned 180° at three different stages of livor mortis: b) early livor mortis resulting in haemoglobin accumulating in the back of the arm, c) during fixation of hypostasis which results in some haemoglobin to flow downwards and d) after fixation, which shows no changes in haemoglobin distribution.

5.5.4 Pressure Induced Blanching

Similar results can be observed with blanching. Before the blood vessels break down, applying pressure to an area on the skin results in blanching. The simulation was run four times. Each time, pressure was applied during different livor mortis stages, before livor mortis, during early livor mortis, during fixation of hypostasis and after fixation. The results can be observed in Figure 35. Figure 36 shows two photographs of pressure being applied to an area affected by livor mortis. The area turns white as blood is pressed out. Figure 35a and 35b show results where pressure is applied to the skin before livor mortis is fixed. The resulting blanching effects are similar to the blanching that can be observed in the photograph from Figure 36b. Figure 35c shows the result of pressure being applied during fixation. Some blanching still occurs, though not all colouration disappears. In the simulation where pressure is applied after livor mortis has become fixed no blanching occurs. This is demonstrated in Figure 35d. One can observe some polygonal edges around the blanching area in Figure 35a and 35b. This is due to the blanching being applied on a nodal level.

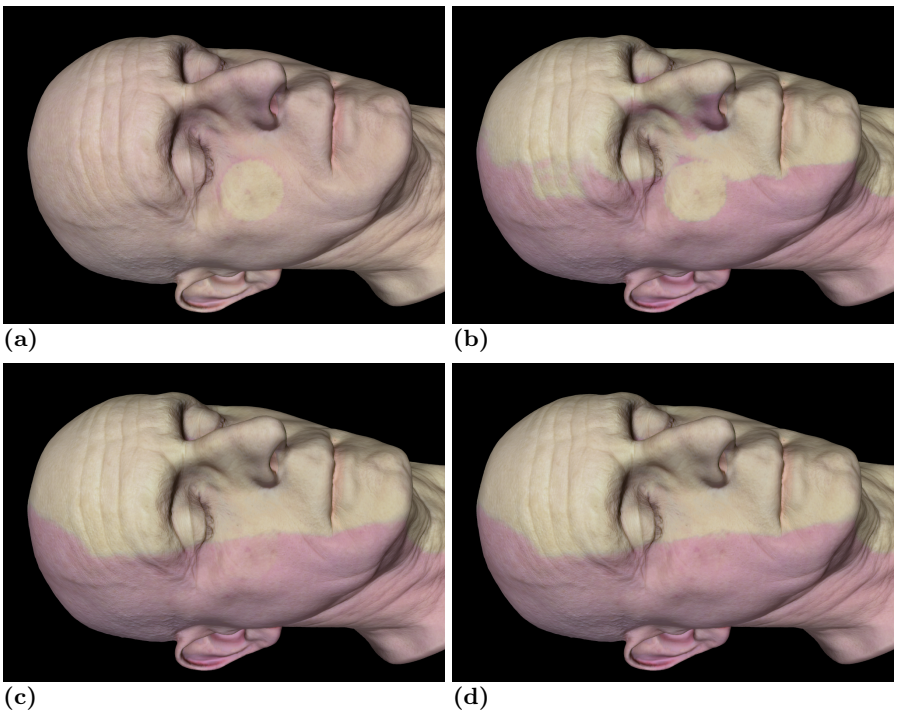


Figure 35: This figure shows the result of applying pressure to an object a) before livor mortis sets in, b) during livor mortis but before fixation of hypostasis, c) during fixation of hypostasis and d) after hypostasis has become fixed.

5.6 Conclusion and Future Work

The objective of the method proposed in this chapter was to create a visually realistic reproduction of the appearance of livor mortis. The livor mortis simulation was able to model the pooling of haemoglobin due to gravity, the fixation of livor mortis due to tissue staining and pressure induced blanching effects. In addition to the above, the presented method includes a skin shader that is able to model colouration changes based on the internal blood dynamics, such as changes in the haemoglobin distribution and oxygen saturation. The tetrahedral representation of the internal body parts allows for the reproduction of the irregular make up of the vascular system and the capturing of the patchy skin appearance of early livor mortis. The proposed skin shader is able to reproduce the changes in skin colour caused by the underlying blood flow and oxygen levels by representing the skin as a melanin and haemoglobin layer.

When compared to the method used in Chapter 4, in which any interior or physical properties defining the surface's composition were ignored, the method proposed in this chapter introduces increased complexity due to the consider-

(a) **(b)**

Figure 36: When pressure is applied to early livor mortis that is not yet fixed a), blanching occurs at the pressure point b). Images taken from Prahlow and Byard (2011)

ation of surface changes caused by the interior properties of the model. This represents an increase in complexity compared to the problem of surface weathering, and therefore requires a different method than the Günther et al. (2012) approach considered in Chapter 4.

Apart from livor mortis, another potential application of this method is bruising, as it is a similar process to fixation of hypostasis. Blood vessels burst with strong impact, which leads to haemoglobin leaking into the surrounding tissue. This would lead to a purple to bluish discolouration of the affected areas as the oxygen content decreases (see Bohnert et al. (2000)). However, in order to simulate the healing of bruises, the breakdown of haemoglobin should also be considered, which leads to the green and yellow colouration. I believe that this model can be used in the entertainment industry to add more realism to the early appearance of corpses that are very commonly seen in modern computer and video games. Another potential application could be as a teaching aid for forensics.

Livor mortis and human body decomposition in general are very complex processes that are affected by many external factors such as temperature and humidity. The approach presented in this chapter does not consider changes in the blood dynamics that are caused by temperature variations. Introducing temperature into the livor mortis simulation would allow for a greater variation in lividity. The examples used simple bone structures for the simulation examples, that led to some haemoglobin accumulating in thin-fleshed areas, as can

be observed where the nose meets the cheek. These can be avoided by creating a more realistic bone structure that better represents the flesh distribution between bone and skin over the whole object. For the best visual results triangles are recommended to be fairly evenly sized over the object's surface with a well fitting internal bone structure.

The skin shading does not consider the oily layer that lies on top of the epidermis and the translucency of thin areas such as ears and nostrils. As the oily layer directly reflects light in all wavelength equally, the simulation results lack the wet and shiny look that can be observed in some of the photographs. Both specular reflection and translucency of thin areas have been considered in skin shading approaches by Jimenez et al. (2009) and D'Eon and Luebke (2007) and can be easily integrated into the proposed skin shading method.

In order to avoid the visual artefact mentioned in Section 5.5.4, the blood vessel texture can be used to record pressure affected areas. The available blood capacity is adjusted in the blood vessel map first and applied to the simulation nodes afterwards. This could yield a smoother result and allow the creation of small-scale blanching caused by belts and strings. For future work, introducing the effects of temperature into the simulation would allow the simulation of a greater variety of lividity. Further greenish-red, brown and blackened skin discolourations are caused by putrefaction and dehydration which also lead to deformations of the skin and internal organs, again, see Prahlow and Byard (2011). Introducing putrefaction and dehydration into the livor mortis simulation would be a particularly interesting area for longer term research and expansions of the existing results demonstrated in this chapter.

The livor mortis vessel network and haemoglobin transportation approach share some incidental similarities with graph algorithms. Looking into graph algorithm techniques used for flow networks might be an appropriate next step in improving the livor mortis simulation. Blood flow has been represented in a similar manner to electrical network or water flow graphs as mentioned in Chapter 2. In contrast to the blood flow dynamics simulated with graph algorithms (see Section 2.9), the blood dynamics that cause livor mortis are not circular. As such, a Directed Acyclic Graph might be a better choice in representing blood flow caused by gravity. On the other hand, this poses the problem of changing edge directions if the the body is moved. Research into the application of graph algorithms to improve the blood dynamics for livor mortis simulations is nonetheless intriguing.

Chapter 6

Simulating Natural Mummification

6.1 Introduction

The pooling of blood responsible for livor mortis is one of the earliest visible signs after death. Although livor mortis is caused by processes happening inside the body, skin colouration changes are the only apparent visual change. The morphology of the body is not altered. However, during later stages in the decomposition, corpses undergo noticeable morphology changes. The most prominent of those processes are autolysis, putrefaction and mummification. This chapter focuses on the simulation of the physical process of natural mummification by dessication and the visual changes it introduces in a corpse's appearance.

Simulating mummification by dessication involves simulating the moisture evaporation from the skin and the moisture diffusion on the internal body layers. The moisture diffusion is simulated on a volumetric tetrahedral mesh using the finite element method (FEM). Changes in the moisture content and distribution in the tissues causes changes in the tissue morphology, such as shrinking and shrivelling of soft tissue. The deformations of the volume structure are also simulated using the FEM. The skin deformation effects, such as shrinking and wrinkling, are simulated on the triangle mesh representing the outer skin layer. For the skin deformation position based dynamics are employed. The skin dynamics are simulated using a cloth simulation approach. Stretching and bending constraints are constructed on the triangle structure to simulate cloth-like deformations. Additional tracking constraints connect the skin mesh to the

underlying tetrahedral volume mesh. As the volume mesh deforms from moisture loss, the skin mesh is pulled along due to the tracking constraints. The colouration changes of the dessicated skin are produced by using a combination of look up textures and subsurface scattering approaches, similar to the one used in the livor mortis simulation.

6.2 Biological Background on Natural Mummification

Livor mortis is an early post mortem process that mostly affects the colouration of the human body. Later processes that drive post mortem changes are autolysis and putrefaction or mummification. In contrast to livor mortis, these lead to structural changes in the body. Enzymes break down the large protein, fat and carbohydrate molecules that are the building blocks of the human body. When the body is still alive these processes are only activated when needed. But, with the loss of oxygen and nutrients after death, the body's cell structure breaks down and releases enzymes, as described by Lynnerup (2007). These enzymes start an autodigestion process, called autolysis. This results in the softening and finally liquefaction or gasification of the body tissue. Putrefaction on the other hand is driven by bacteria. These can be bacteria already inside the body or external ones. These bacteria absorb molecules for nutrition. In order to generate more molecules the bacteria secrete more enzymes which in turn leads to further tissue liquefaction (see Aufderheide (2003)).

Mummification can happen naturally in certain environmental conditions (natural/spontaneous mummification) or by human intervention (artificial/anthropogenic mummification). An example of artificial mummification is Egyptian mummies. Spontaneous mummification can occur for a variety of reasons. A well known example is mummification by dessication in hot and arid environments. Others include (but are not restricted to) freezing, bog bodies or due to exposure to certain metals. Aufderheide (2003) describes a mummy as a corpse or tissue that has been preserved to such a degree that it resembles its living morphology and resists further decay. In order for mummification to happen, the decomposition processes, namely autolysis and putrefaction, need to be prevented.

In order to understand mummification one needs to understand what influences the actions of the enzymes and bacteria in the decay processes. Some factors

(a) **(b)**

Figure 37: Photographs of natural dessicated mummies taken by the author in the Ostkrypta in the St. Petri Cathedral in Bremen, Germany, with a Nikon COOLPIX S3700 and NIKKOR lens. They are estimated to be hundreds of years old.

that influence enzyme and bacterial actions are described by Lynnerup (2007) to be temperature, humidity and PH values. A cold environment restricts enzymatic and bacterial activity that drive the tissue decay processes. Bodies are well preserved in bogs due to its cold, acidic and anaerobic micro-environment that reduces bacterial activity. There are a great number of different causes of mummification. For this project only natural mummification by desiccation is considered. In order to simulate this process an understanding of the visual and morphological changes as well as the processes that drive them need to be understood. For mummification by dessication the evaporation and diffusion of water in the body are important.

About 80% of human tissue is removable water (Aufderheide (2003)). Mummification by desiccation is due to the dehydration of the body tissue through evaporation of the bodily fluids. As enzymatic breakdown and bacterial growth require water, removing bodily fluids can therefore prevent autolysis and putrefaction. Desiccation happens in a dry/warm atmosphere, with free circulation of air. The degree of desiccation often varies over the body depending on the volume to surface area ratio. The rate of water transfer from the body internal tissue to the skin surface is related to the difference in water content, i.e. the water gradient. This is maximised if the skin surface is kept dry. The quickest desiccating mummies are often also the ones with the best-preserved soft tissue in natural mummified bodies. These are commonly the fingers, toes, ear lobes and the other skin areas with a large ratio of skin surface to underlying soft tissue, as reported by Aufderheide (2003). These areas often show a greater preservation of the epidermis. Lynnerup (2007) reports that tissues with a high content of collagen tend to be the best preserved. The dermis, muscle fascia and tendons are tissues with high collagen content and therefore tend to be well

preserved.

(a) **(b)**

Figure 38: Photographs of mummified corpses taken from Prahlow and Byard (2011). The skin is wrinkled in and shows a brownish red colouration.

An object that consists of a high percentage of water undergoes noticeable morphological changes with water loss. This is also observed in desiccated corpses. The gradual removal of the water in human tissue leads to changes in the volume and appearance of the affected tissue. The desiccated soft tissue and internal organs shrink and shrivel then become rigid and brittle. The skin becomes dry and leathery and clings firmly to the body frame. The skin takes on a blackish-brown colour. There is evidence suggesting that the brown colour in dynastic mummies is due to the embalming substances used to cover the corpses with, as per Stani et al. (2014). Spontaneously mummified tissue varies from black-brown to parchment coloured, as shown in Figure 37 and 38. The skin around the groin, neck and armpits can split due to shrinkage strains. The skeletal body features are generally well preserved. Some fat cells in the subcutaneous tissue, that lies underneath the dermis, might break as the skin contracts. This leads to liquid fat smearing the dermis, which then becomes translucent. This is particularly visible in the feet in Figure 38a. During dehydration the connection between the skin layers weakens. This leads to the epidermis separating from the dermis (skin slippage) and the dermis to separate from the underlying tissue.

Papageorgopoulou et al. (2015) describe research on modelling ancient Egyptian mummification and natural mummification by desiccation on two fresh human lower legs. The natural mummification was performed under dry heat using an oven. However, this experiment was stopped after seven days as the leg was showing sign of putrefaction and no onset of mummification. The experiment on artificial mummification was more successful. This was performed by placing

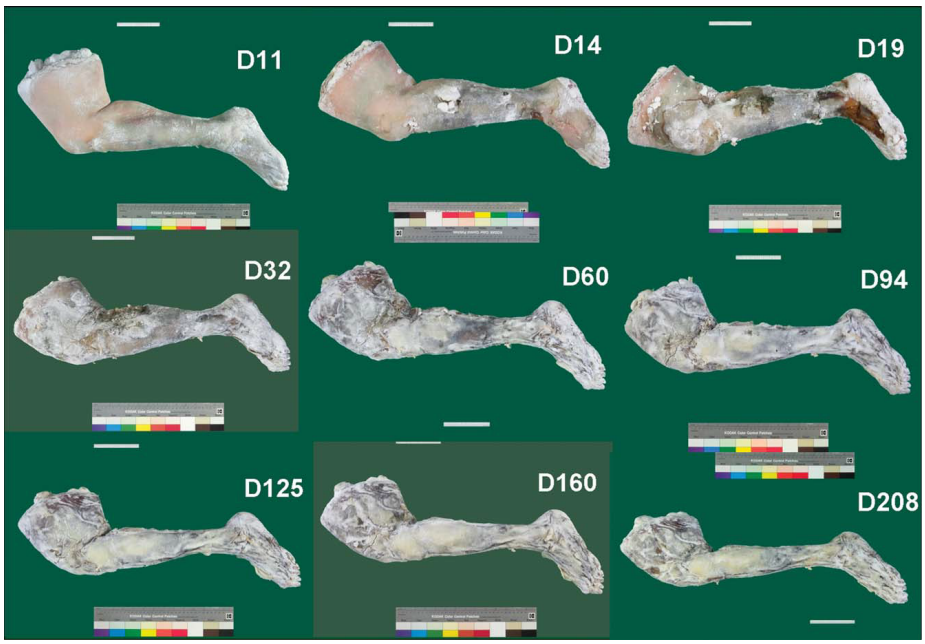


Figure 39: This figure was taken from Papageorgopoulou et al. (2015) and shows photographs of the mummifying lower leg at different days (D).

the leg in an natron mix over 208 days. Over this period, Papageorgopoulou et al. (2015) performed macroscopic and radiological examinations of the skin and muscle samples. In addition; temperature, humidity, PH values and the weight of the lower leg were measured throughout the experiment. They identified humidity, external temperature and PH values as important factors in mummification. Figure 39 shows photographs of the artificially mummified leg at nine different times during the experiment. The visual observations described in the paper are summarised in Table 6.1 for eight different days during the 208 days period of the experiment. Papageorgopoulou et al. (2015) noticed that the leg undergoes drastic colouration changes, starting with a greenish discolouration in some areas that then spreads over the whole leg. The leg then started to turn dark green, greyish and brown and, finally, yellowish with patches of dark green, brown and red. The morphology of the leg also changed noticeably. Skin slippage and detachment started very early on as the epidermis starts separating from the dermis. The feet shrunk at a faster rate than the rest of the leg, turning leathery at around 25 days into the experiment. The rest of the leg took on this leathery appearance after around 60 days. They noticed a significant loss of volume. The weight loss was reported to be 30% during the first two months and 50% by the end of the experiment. The weight and volume of the bones did not change during the experiment. The weight loss of the tissue was

caused by the water loss. At the end of the experiment, the lower leg was nearly completely dehydrated, shrunken and mummified after 208 days. The thigh did not reach that point however.

The visual result of the artificial mummification shows a white crust layer on the lower leg surface. This makes it difficult to observe the skin colouration changes caused by dessication. Similar colouration changes as described in the literature can be observed on the foot in Figure 39 day 19.

day	observation
D5	slight pink discolouration
D11	greenish discolouration covering half the leg skin slippage - the stratum corneum began to separate
D14	greenish discolouration expanded over whole leg more skin slippage - epidermis starts separating from dermis
D19	colour turning dark green, greyish and brown skin completely detached at dorsal part of the foot
D25	leathery appearance at the toes and foot formation of salt layer
D40	yellowish appearance with patches of dark green, brown & red leg not completely dehydrated
D60	leathery appearance over whole leg and brown in colour around 30% weight loss foot shows signs of dehydration
D208	foot has significantly shrunken and dehydrated thigh had not reached complete dehydration & mummification around 50% weight loss

Table 6.1 This table summarises the observations by Papageorgopoulou et al. (2015) during their 208 day mummification experiment.

6.3 The Finite Element Method Background

This section aims to provide the reader background information on the finite element method that is used to simulate the moisture diffusion and deformation of the internal body. For a more in depth exposition on the finite element method see Reddy and Gartling (2010) and Bathe (2006), from which the information in this section is taken.

The finite element method is a numerical technique for solving boundary value problems for a specific field variable, such as moisture or positions. The core idea is to subdivide the problem domain into smaller continuous sub-domains

in which the field variable can be described by an analytical formula. These sub-domains need to cover the entire domain without overlapping. In computer graphics a solid object is often described by a closed surface mesh representing the object's boundary. Any point within the boundary is part of the problem domain. A finite element discretisation is used to express the infinite number of points within the object by a finite number of values. In the case of computer graphics these sub-domains are polygonal elements of finite size. Every point within the polygonal element can then be expressed by an analytical function, called the shape function. The differential equation can then be solved on the vertices of the discretisation.

An alternative method for simulating humidity diffusion and volume deformation is a mass spring system. Moisture can be moved between the vertices in a similar manner to the work done by Aoki et al. (2004) on crack propagation (see Section 2.6.2) to simulate moisture diffusion. Changes in moisture content on the tetrahedral vertices affect the strain parameters of the the springs (i.e. tetrahedral edges) causing the body to deform. Mass-spring systems are generally less computationally expensive than approaches using the FEM. However, they are also less physically accurate, are restricted in the mechanical behaviour and visual quality they can model, and are difficult to tune. For these reasons the FEM has been chosen to simulate the moisture diffusion and volume deformation.

6.3.1 Formulation of Differential Equation

The finite element method is used to find approximate solutions to boundary value problems. As such, consider a physical domain Ω with a boundary Γ on which the following two-dimensional differential equation is defined:

$$\Delta \boldsymbol{u} = \frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \frac{\partial^2 \boldsymbol{u}}{\partial y^2} \tag{6.1}$$

Then, the boundary value problem looks as follows:

$$\begin{aligned} -\Delta \boldsymbol{u} + \rho \boldsymbol{u} &= \boldsymbol{f} && \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{g}_0 && \text{on } \Gamma \\ \partial_{\boldsymbol{n}} \boldsymbol{u} &= \boldsymbol{g}_1 && \text{on } \Gamma \end{aligned} \tag{6.2}$$

where the unknown $\boldsymbol{u}$ can be a scalar valued or vector valued function, depend-

ing on the problem being modelled. It is also called the field variable that varies over time and space. Examples of field variables are displacement, temperature and moisture content. In the case of temperature and moisture content, the field variable is a scalar valued function. Field variables representing displacement tend to be vector valued functions of two- or three-dimensions. ρ is a non-negative constant (usually density), $-\Delta \boldsymbol{u}$ is the diffusion term, $\boldsymbol{f}$ is a function on Ω describing forces etc., $\boldsymbol{g}_0$ and $\boldsymbol{g}_1$ are functions describing the boundary and $\partial_{\boldsymbol{n}} \boldsymbol{u}$ is an exterior normal derivative, where $\boldsymbol{n}$ is a unit vector normal to the boundary. Green's Theorem can be used to rewrite the equation into its weak form as described by Reddy and Gartling (2010):

Find $\boldsymbol{u}$ such that

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{g}_0 && \text{on } \Gamma_D \\ \int_\Omega (\Delta \boldsymbol{u}) \boldsymbol{v} \, d\Omega + \rho \int_\Omega \boldsymbol{u}\boldsymbol{v} \, d\Omega &= \int_\Omega \boldsymbol{f}\boldsymbol{v} \, d\Omega + \int_\Gamma (\partial_{\boldsymbol{n}} \boldsymbol{u}) \boldsymbol{v} \, d\Gamma && \text{on } \Omega \end{aligned} \tag{6.3}$$

where $\boldsymbol{v}$ is called a test function, which "tests" if the equation is satisfied by $\boldsymbol{u}$. $\boldsymbol{v}$ serves as a weight function used to average equations. This means the domain Ω can be discretised into a finite number of sub-domains. $\boldsymbol{u}$ can be approximated at any point within a sub-domain by interpolation using the test function $\boldsymbol{v}$.

In two dimensions the domain can be divided into triangles. Then, a function for each vertex of each triangular element of the form

$$\boldsymbol{N}_i(x, y) = a_i + b_i x + c_i y \tag{6.4}$$

is defined, where $i = 1, 2, 3$ are the three vertices of the triangle. $\boldsymbol{N}$ is linear on each triangle and continuous over the domain. Furthermore, $\boldsymbol{N}$ is a unique function on each vertex with the property

$$\boldsymbol{N}_i(\boldsymbol{p_j}) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{6.5}$$

where $\boldsymbol{p_j}$ is a vertex. Each element can be described by a unique linear combination of the basis functions $\boldsymbol{N}_i$ as follows:

$$\boldsymbol{u} = \sum_{j=1}^{n} \boldsymbol{u}_j \boldsymbol{N}_j \tag{6.6}$$

Hence $\boldsymbol{u}$ is a value of the function on points in Ω. With this the weak formulation can be rewritten as:

$$\begin{aligned} &\text{Find } \boldsymbol{u} \text{ such that} \\ &\boldsymbol{u}(\boldsymbol{p_j}) = \boldsymbol{g}_0(\boldsymbol{p_j}) && \forall j \in Dir \\ &\int_\Omega \Delta\boldsymbol{u} \cdot \Delta\boldsymbol{N}_i \; d\Omega + \rho \int_\Omega \boldsymbol{u}\boldsymbol{N}_i \; d\Omega = \int_\Omega \boldsymbol{f}\boldsymbol{N}_i \; d\Omega + \int_\Gamma \boldsymbol{g}_1\boldsymbol{N}_i \; d\Gamma && \forall i \in Ind \end{aligned} \tag{6.7}$$

The problem to computing $\boldsymbol{u}$ has been reduced to computing its value on the vertices of the triangulation. This means that there is now a finite number of unknowns and a finite number of equations to solve. Considering Equation 6.6, the above can be approximated by

$$\sum_j (\int_\Omega \Delta\boldsymbol{N}_j \cdot \Delta\boldsymbol{N}_i \; d\Omega + \rho \int_\Omega \boldsymbol{N}_j\boldsymbol{N}_i \; d\Omega)\boldsymbol{u}_j = \int_\Omega \boldsymbol{f}\boldsymbol{N}_i \; d\Omega + \int_\Gamma \boldsymbol{g}_1\boldsymbol{N}_i \; d\Gamma \tag{6.8}$$

A closer look at the new equation reveals that it can be expressed by two matrices and a vector. These are:

the stiffness matrix,

$$\boldsymbol{K}_{ij} = \int_\Omega \Delta\boldsymbol{N}_j \cdot \Delta\boldsymbol{N}_i \; d\Omega \tag{6.9}$$

the mass matrix:

$$\boldsymbol{M}_{ij} = \int_\Omega \boldsymbol{N}_j\boldsymbol{N}_i \; d\Omega \tag{6.10}$$

and the force vector:

$$\boldsymbol{f}_i = \int_\Omega \boldsymbol{f}\boldsymbol{N}_i \; d\Omega + \int_\Gamma \boldsymbol{g}_1\boldsymbol{N}_i \; d\Gamma \tag{6.11}$$

The following shows Equation 6.8 in matrix form:

$$\sum_{j \in Ind} (\boldsymbol{K}_{ij} + \rho\boldsymbol{M}_{ij})\boldsymbol{u}_j = \boldsymbol{f}_i - \sum_{j \in Dir} (\boldsymbol{K}_{ij} + \rho\boldsymbol{M}_{ij})\boldsymbol{g}_0(\boldsymbol{p_j}) \tag{6.12}$$

6.3.2 Finite Element Discretisation

A vital step in the finite element method is to divide the problem space into subspaces on which the problem is simpler to solve. This is called finite element discretisation. In 2D space the problem domain can be divided into different 2-dimensional polygons, such as triangles and quads. A volumetric object in 3-dimensional space is divided into a set of volume elements, such as tetrahedra and voxels. For this simulation the object's boundary is represented by a tetrahedral mesh. This triangle surface mesh is used for the skin deformation. The internal volume is represented by tetrahedra. This representation can be used as finite element discretisation to solve the moisture diffusion and dynamic deformations of the object. This section will look at triangular and tetrahedral domains for the finite element discretisation and their shape function, also called test or interpolation function.

For a triangle element in 2-dimensional space the interpolation function looks as follows:

$$N_i(x, y) = a_i + b_i x + c_i y \tag{6.13}$$

for the three vertices $i = 1, 2, 3$ of the triangle, with

$$N_i(\boldsymbol{p_j}) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{6.14}$$

so

$$\begin{aligned}
N_1(x_1, y_1) &= a_1 + b_1 x_1 + c_1 y_1 &&= 1 \\
N_1(x_2, y_2) &= a_1 + b_1 x_2 + c_1 y_2 &&= 0 \\
N_1(x_3, y_3) &= a_1 + b_1 x_2 + c_1 y_3 &&= 0 \\
N_2(x_1, y_1) &= a_2 + b_2 x_1 + c_2 y_1 &&= 0 \\
N_2(x_2, y_2) &= a_2 + b_2 x_2 + c_2 y_2 &&= 1 \\
N_2(x_3, y_3) &= a_2 + b_2 x_2 + c_2 y_3 &&= 0 \\
N_3(x_1, y_1) &= a_3 + b_3 x_1 + c_3 y_1 &&= 0 \\
N_3(x_2, y_2) &= a_3 + b_3 x_2 + c_3 y_2 &&= 0 \\
N_3(x_3, y_3) &= a_3 + b_3 x_2 + c_3 y_3 &&= 1
\end{aligned} \tag{6.15}$$

(6.16)

This can be written in matrix form as

$$\underbrace{\begin{bmatrix} 1 & x1 & y1 \\ 1 & x2 & y2 \\ 1 & x3 & y3 \end{bmatrix}}_{\boldsymbol{P}} \underbrace{\begin{bmatrix} a1 & a2 & a3 \\ b1 & b2 & b3 \\ c1 & c2 & c3 \end{bmatrix}}_{\boldsymbol{N}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{I}} \tag{6.17}$$

Hence,

$$\boldsymbol{N} = \boldsymbol{P}^{-1} \tag{6.18}$$

An alternative way of constructing the shape function is by computing the area coordinates L_1, L_2 and L_3, which will result in the same interpolation function.

The interpolation function for a tetrahedral element in 3-dimensional space is similar:

$$N_i(x, y) = a_i + b_i x + c_i y + d_i z \tag{6.19}$$

and considering the condition described in Equation 6.14:

$$\underbrace{\begin{bmatrix} 1 & x1 & y1 & z1 \\ 1 & x2 & y2 & z2 \\ 1 & x3 & y3 & z3 \\ 1 & x4 & y4 & z4 \end{bmatrix}}_{\boldsymbol{P}} \underbrace{\begin{bmatrix} a1 & a2 & a3 & a4 \\ b1 & b2 & b3 & b4 \\ c1 & c2 & c3 & c4 \\ d1 & d2 & d3 & d4 \end{bmatrix}}_{\boldsymbol{N}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{I}} \tag{6.20}$$

and hence,

$$\boldsymbol{N} = \boldsymbol{P}^{-1} \tag{6.21}$$

An important property of a shape function is that the value at a point on a triangle edge depends only on the value of the two vertices forming the edge. Similarly, the value of a point lying on a triangle surface of a tetrahedra depends only on the three vertices forming the triangle.

6.3.3 Matrix Assembly

The shape functions can be used to compute the integrals in Equation 6.8 in a process called assembly. As the name suggest, the elemental mass matrices (Equation 6.10), stiffness matrices (Equation 6.9) and force vectors (Equation 6.11) are computed and assembled into a global mass matrix, stiffness matrix and force vector respectively.

For triangle elements, let

$$\boldsymbol{B}_i = \Delta \boldsymbol{N}_i = \begin{bmatrix} \frac{\partial \boldsymbol{N}_i}{\partial x} & \frac{\partial \boldsymbol{N}_i}{\partial y} \end{bmatrix} = \begin{bmatrix} b_i & c_i \end{bmatrix} \tag{6.22}$$

$\boldsymbol{B}^e$ is a constant matrix holding the partial derivatives of the shape functions of the triangle element e. Using $\boldsymbol{B}^e$ a 3x3 stiffness matrix for triangle element e can be constructed using

$$\boldsymbol{K}^e = \int_\Omega \Delta \boldsymbol{N}^T \Delta \boldsymbol{N} \; dA = \int_\Omega \boldsymbol{B}^T \boldsymbol{B} \; dA = A^e \boldsymbol{B}^T \boldsymbol{B} \tag{6.23}$$

The integration of the $\boldsymbol{N}^T \boldsymbol{N}$ term in the 3x3 elemental mass matrix $\boldsymbol{M}^e$ (see Equation 6.10) and the 3x1 elemental force vector $\boldsymbol{f}^e$ (see Equation 6.11) can be carried out by using a formula by Eisenberg and Malvern (1973) which states the following:

$$\int_A \boldsymbol{N}_1^m \boldsymbol{N}_2^n \boldsymbol{N}_3^p \; dA = \frac{m!n!p!}{(m+n+p+2)!} 2A \tag{6.24}$$

for integration over an area and

$$\int_l \boldsymbol{N}_1^m \boldsymbol{N}_2^n \; dA = \frac{m!n!}{(m+n+1)!} l \tag{6.25}$$

for integration over a line. Hence, the elemental mass matrix can be calculated as follows

$$\boldsymbol{M}^e = \rho \int_\Omega \boldsymbol{N}^T \boldsymbol{N} \; dA = \frac{\rho A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{6.26}$$

and the nodal force vectors $\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T \, dl$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T|_{1-2} \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} \, dl = \frac{1}{2} l_{1-2} \begin{Bmatrix} f_x \\ f_y \\ f_x \\ f_y \\ 0 \\ 0 \end{Bmatrix} \tag{6.27}$$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T|_{2-3} \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} \, dl = \frac{1}{2} l_{2-3} \begin{Bmatrix} 0 \\ 0 \\ f_x \\ f_y \\ f_x \\ f_y \end{Bmatrix} \tag{6.28}$$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T|_{1-3} \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} \, dl = \frac{1}{2} l_{1-3} \begin{Bmatrix} f_x \\ f_y \\ 0 \\ 0 \\ f_x \\ f_y \end{Bmatrix} \tag{6.29}$$

for edge $(\boldsymbol{p}_1, \boldsymbol{p}_2)$, $(\boldsymbol{p}_2, \boldsymbol{p}_3)$ and $(\boldsymbol{p}_1, \boldsymbol{p}_3)$ respectively. Similarly, for a tetrahedral element e in 3 dimensions

$$\boldsymbol{B}_i = \Delta \boldsymbol{N}_i = \begin{bmatrix} \frac{\partial \boldsymbol{N}_i}{\partial x} & \frac{\partial \boldsymbol{N}_i}{\partial y} & \frac{\partial \boldsymbol{N}_i}{\partial z} \end{bmatrix} = \begin{bmatrix} b_i & c_i & d_i \end{bmatrix} \tag{6.30}$$

and thus

$$\boldsymbol{K}^e = \int_\Omega \Delta \boldsymbol{N}^T \Delta \boldsymbol{N} \, dV = \int_\Omega \boldsymbol{B}^T \boldsymbol{B} \, dV = V^e \boldsymbol{B}^T \boldsymbol{B} \tag{6.31}$$

The Eisenberg and Malvern (1973) formula for integrating shape functions over volume elements is:

$$\int_V N_1^m N_2^n N_3^p N_4^q \, dV = \frac{m!n!p!q!}{(m+n+p+q+9)!} 6V \tag{6.32}$$

Hence, the elemental mass matrix for a tetrahedral element e is

$$\boldsymbol{M}^e = \rho \int_\Omega \boldsymbol{N}^T \boldsymbol{N} \; dV = \frac{\rho V}{20} \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \tag{6.33}$$

and the nodal force vectors

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f} \boldsymbol{N}^T|_{123} \begin{Bmatrix} f_x \\ f_y \\ f_z \end{Bmatrix} \; dA = \frac{1}{3} A_{123} \begin{Bmatrix} f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{6.34}$$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f} \boldsymbol{N}^T|_{124} \begin{Bmatrix} f_x \\ f_y \\ f_z \end{Bmatrix} \; dA = \frac{1}{3} A_{124} \begin{Bmatrix} f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \\ 0 \\ 0 \\ 0 \\ f_x \\ f_y \\ f_z \end{Bmatrix} \tag{6.35}$$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T|_{234} \begin{Bmatrix} f_x \\ f_y \\ f_z \end{Bmatrix} \, dA = \frac{1}{3} A_{234} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \end{Bmatrix} \tag{6.36}$$

$$\boldsymbol{f}^e = \int_\Gamma \boldsymbol{f}\boldsymbol{N}^T|_{134} \begin{Bmatrix} f_x \\ f_y \\ f_z \end{Bmatrix} \, dA = \frac{1}{3} A_{134} \begin{Bmatrix} f_x \\ f_y \\ f_z \\ 0 \\ 0 \\ 0 \\ f_x \\ f_y \\ f_z \\ f_x \\ f_y \\ f_z \end{Bmatrix} \tag{6.37}$$

for the tetrahedral faces $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$, $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_4)$, $(\boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4)$ and $(\boldsymbol{p}_1, \boldsymbol{p}_3, \boldsymbol{p}_4)$, respectively.

With these, the global 3n x 3n matrices $\boldsymbol{K}$ and $\boldsymbol{M}$ and the global 3n x 1 vector $\boldsymbol{f}$ can be assembled to construct the linear equation.

$$\boldsymbol{K}_{n_i n_j} = \sum_{e \in \Omega} \boldsymbol{K}^e_{ij} \tag{6.38}$$

$$\boldsymbol{M}_{n_i n_j} = \sum_{e \in \Omega} \boldsymbol{M}^e_{ij} \tag{6.39}$$

$$\boldsymbol{f}_{n_i} = \sum_{e \in \Omega} \boldsymbol{f}^e_{ij} \tag{6.40}$$

where n_i and n_j are the global vertex indices of the vertex i and j of element e respectively. The resulting set of linear equations can then be solved for $\boldsymbol{u}$ using different numerical integration methods such as Runge-Kutta Integration (Gibson and Mirtich (1997)), Verlet Integration (Jakobsen (2001)), Newton-Raphson Solver (Barbič and James (2005)) or Explicit/Implicit Euler Integration (Gibson and Mirtich (1997)). For this project the Implicit Euler Integration has been used. Implicit Euler (or Backward Euler) is stable and allows for larger time-steps than explicit Euler. Note that all elements $\boldsymbol{K}_{ij}$ and $\boldsymbol{M}_{ij}$ are zero except where vertices i and j belong to the same tetrahedron. This means that the global 3n x 3n matrices $\boldsymbol{K}$ and $\boldsymbol{M}$ are sparse, which makes them less computationally intensive to solve than dense matrices. The assembly of the global matrices has complexity $O(E)$, where E is the number of tetrahedral elements. The elemental matrix for each tetrahedral element is computed (in constant time) and then inserted into the corresponding global matrix.

6.4 Humidity Diffusion

The deformations observed in mummification by dessication are driven by the changes in the tissue moisture content. As such, the first part of the mummification simulation is a simulation of the moisture evaporation and diffusion within the object. The drying process has been simulated on a slice of a potato using the FEM by Yang et al. (2001). Liu et al. (2012b) use a similar method to simulate fruit rotting. Water evaporation is caused by differences in the water content of air and the object's surface. A similar process happens inside the body: the moisture moves to areas with less water content. The next section will formulate the boundary value problem describing moisture diffusion. This is then formulated as a Finite Element problem on the tetrahedral mesh that describes the object's internal volume. Moisture diffusion follows similar mechanics as heat diffusion. A FEM approach for heat transfer is described by Reddy and Gartling (2010).

6.4.1 Mathematical Background

The humidity differences in the air and object cause vapour pressure differences in the air and boundary of the object. This leads to moisture exchange between the air and the object's surface. If the humidity in the air surrounding the object is lower than the humidity in the object itself, water leaves the object due to evaporation. Inside the object the water moves to the more dehydrated

areas. Let $\phi(x, y, z, t)$ be the moisture content at any point (x, y, z) at a time t. Then, the moisture diffusion inside the object can be expressed by the following boundary value problem:

Find ϕ such that

$$\rho \frac{\partial \phi}{\partial t} = \Delta \cdot (w \rho \Delta \phi) \qquad \text{moisture diffusion} \qquad (6.41)$$

$$\Delta \phi \cdot \boldsymbol{n} = \frac{h_m(\phi_e - \phi)}{w\rho} \qquad \text{boundary condition} \qquad (6.42)$$

where ρ is the density of the solid, w is the moisture diffusion coefficient, ϕ_e is the equilibrium moisture content in the surrounding air, h_m is the mass transfer coefficient and $\boldsymbol{n}$ is the normal of the boundary surface. $\Delta = \left\{ \frac{\partial}{\partial x} \quad \frac{\partial}{\partial y} \quad \frac{\partial}{\partial z} \right\}$ is the spatial gradient operator and $\Delta\phi$ is the diffusion term, i.e. the gradient of the moisture content. The equation describes how moisture diffusion happens in the direction of the gradient, i.e. how moisture flows to areas with lower humidity. Note that ρ and w are functions of time that need to be recalculated at each time step:

$$w(t) = w_0 \exp(-\frac{\phi 0}{\phi}) \qquad (6.43)$$

$$\rho(t) = \frac{\rho_0}{S_V} \qquad (6.44)$$

where w_0 and ρ_0 are the initial moisture diffusion and initial density respectively. $S_V = \mu \frac{\phi}{\phi_0} + \lambda$ is the volumetric moisture shrinking coefficient that describes how much the element shrinks based on its moisture loss. Here μ is the fraction of volume that is affected by water loss (i.e. the maximum volume the object can lose) and λ the fixed volume, hence $\lambda + \mu = 1$. This is described further in Section 6.5.3.

The mass transfer coefficient h_m controls the rate at which moisture leaves the object's boundary into the surrounding air. In order to model a physically precise mass transfer coefficient, environmental activities such as heat and air flow around the model would need to be simulated. However, environmental influences are omitted in this model. Instead, the mass transfer coefficient h_m and moisture diffusion coefficient w_0 were initially chosen from literature on the drying of meat based products (see Gou et al. (2002) and Simal et al. (2003)). The coefficient then needed to be fine tuned to produce the desired

results. As the mass transfer coefficient in this model is not controlled by environmental factors, such as heat and air ventilation, it is instead exposed to the user. This allows the user to control the dehydration process by adjusting the mass transfer coefficient h_m and equilibrium moisture content in the air ϕ_e to simulate mummification supporting or suppressing environmental conditions. Similarly, adjustments to the moisture diffusion coefficient w_0 allow for artistic control over the visual appearance of the mummifying body.

6.4.2 FEM Formulation of Moisture Diffusion Equations

The object's internal volume is divided into tetrahedral volume elements. The moisture content ϕ is defined at each vertex of the tetrahedralisation and represented by the vector $\boldsymbol{\phi} \in \mathbb{R}^{1\times n}$, where n is the number of vertices in the volume mesh. Let Ω be the set of tetrahedral volume elements and Γ be the set of tetrahedral boundary elements. Note that $\Gamma \subset \Omega$. Let $\boldsymbol{N}$ be the shape function as defined by Equations 6.21 and 6.19. Making use of Green's Theorem, as stated in Section 6.3, the Boundary Value Problem from Equation 6.41 can be rewritten as follows:

$$\int_{\Omega^e} \Delta\phi \Delta \boldsymbol{N}_i \, dV + \int_{\Omega^e} \rho \frac{\partial \phi}{\partial t} \boldsymbol{N}_i \, dV + \int_{\Gamma^e} h_m \phi \boldsymbol{N}_i \, dA - \int_{\Gamma^e} h_m \phi_e N_i \, dA \quad (6.45)$$

for all vertices i. Using Equation 6.6 this becomes

$$\begin{aligned} &\sum_{j \in vertices} \Big(\int_{\Omega^e} \Delta \boldsymbol{N}_j \cdot \Delta \boldsymbol{N}_i \, dV + \int_{\Gamma^e} h_m \boldsymbol{N}_j \cdot \boldsymbol{N}_i \, dA \Big) \phi_j \\ + &\sum_{j \in vertices} \Big(\int_{\Omega^e} \rho \boldsymbol{N}_j \cdot \boldsymbol{N}_i \, dV \Big) \frac{\partial \phi_j}{\partial t} - \int_{\Gamma^e} h_m \phi_e \boldsymbol{N}_i \, dA \end{aligned} \quad (6.46)$$

Or in matrix form:

$$\sum_{j \in vertices} \Big(\boldsymbol{M}_{ij} \frac{\partial \phi_j}{\partial t} + (\boldsymbol{K}_{ij} + \boldsymbol{H}_{ij}) \phi_j \Big) - \boldsymbol{f}_i \quad (6.47)$$

where

$$\boldsymbol{K}^e_{ij} = \int_{\Omega^e} \Delta \boldsymbol{N}_j \cdot \Delta \boldsymbol{N}_i \, dV \quad (6.48)$$

$$\boldsymbol{M}_{ij}^{e} = \int_{\Omega^e} \rho \boldsymbol{N}_j \cdot \boldsymbol{N}_i \; dV \tag{6.49}$$

$$\boldsymbol{H}_{ij}^{e} = \int_{\Gamma^e} h_m \boldsymbol{N}_j \cdot \boldsymbol{N}_i \; dA \tag{6.50}$$

$$\boldsymbol{f}_{i}^{e} = \int_{\Gamma^e} h_m \phi_e \boldsymbol{N}_i \; dA \tag{6.51}$$

The integral in $\boldsymbol{K}^e$ can be solved as demonstrated in Equation 6.31. The Integral in $\boldsymbol{M}^e$ can be solved as described in Equation 6.33. $\boldsymbol{H}^e$ on the other hand involves an area integral in 3-dimensional space. The integral can be solved using the Eisenberg and Malvern (1973) approach for an area integral as was described in Equation 6.24. This results in

$$\boldsymbol{H}^e = h_m \int_{\Omega} \boldsymbol{N}^T \boldsymbol{N}|_{ijk} \; dA = \frac{h_m A_{ijk}}{12} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.52}$$

for the tetrahedral boundary surface made of the vertices i,j,k. Note that the row and column of the non-boundary vertex of a tetrahedron have zero entries. The other rows and columns have a value of two on the diagonal and a value of one on the off-diagonal. The vector $\boldsymbol{f}^e$ is constructed similar to Equations 6.34 to 6.37. For a boundary surface made of tetrahedral nodes i,j,k

$$\boldsymbol{f}^e = \int_{\Gamma} h_m \phi_e \boldsymbol{N}^T|_{ijk} \; dA = \frac{h_m \phi_e A_{ijk}}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{Bmatrix} \tag{6.53}$$

The global matrices and vectors can then be assembled as shown in Equations 6.38 to 6.40.

6.4.3 Solving Humidity Diffusion Equations

The moisture diffusion is an evolutionary equation, i.e. it depends on time as shown in the left hand side term ($\frac{\partial \phi}{\partial t}$) in Equation 6.41. In order to simulate the moisture dynamics inside the object, the time dependent moisture content needs to be computed at every point. The integrals solved in the assembly step

are integrals over space (i.e. volume and area). In order to simulate moisture diffusion, the time dependent part in the differential equation needs to be solved as well. After the assembly the humidity diffusion is described by the following linear equation:

$$\boldsymbol{M}\frac{\partial \boldsymbol{\phi}}{\partial t} + (\boldsymbol{K} + \boldsymbol{H})\boldsymbol{\phi} = \boldsymbol{f} \tag{6.54}$$

at any time t. In order to compute ϕ at any time the equation needs to be integrated over time. This can be expressed as

$$\boldsymbol{\phi}^t = \boldsymbol{\phi}_0 + \int_{t_0}^{t} \frac{\partial \boldsymbol{\phi}^t}{\partial t}\,\mathrm{d}t \tag{6.55}$$

where $\boldsymbol{\phi}_0$ are the initial moisture contents. Let t_0 be the time at the start of the simulation. The time from t_0 can be partitioned into time intervals $0 = t_0 < t_1 < t_2 < ... < t_n < t_{n+1} < ....$ Then the time step is $\Delta t = t_{n+1} - t_n$. Using time intervals $\frac{\partial \boldsymbol{\phi}}{\partial t}$ can be expressed as follows:

$$\frac{\partial \boldsymbol{\phi}^{t+1}}{\partial t} = \frac{\boldsymbol{\phi}^{t+1} - \boldsymbol{\phi}^t}{\Delta t} + O(\Delta t^2) \tag{6.56}$$

Applying this to Equation 6.54 yields:

$$\frac{1}{\Delta t}\boldsymbol{M}(\boldsymbol{\phi}^{t+1} - \boldsymbol{\phi}^t) + (\boldsymbol{K} + \boldsymbol{H})\boldsymbol{\phi}^{t+1} = \boldsymbol{f}^{t+1} \tag{6.57}$$

which can be rearranged to get the following

$$(\boldsymbol{M} + \Delta t(\boldsymbol{K} + \boldsymbol{H}))\boldsymbol{\phi}^{t+1} = \boldsymbol{M}\boldsymbol{\phi}^t + \Delta t\boldsymbol{f}^{t+1} \tag{6.58}$$

which results in the equation used by Liu et al. (2012b) for their fruit rotting simulation. The above equation can be simplified into the linear equation

$$\boldsymbol{A}\boldsymbol{\phi} = \boldsymbol{b} \tag{6.59}$$

where

$$\boldsymbol{A} = \boldsymbol{M} + \Delta t(\boldsymbol{K} + \boldsymbol{H}) \tag{6.60}$$

$$\boldsymbol{b} = \boldsymbol{M}\boldsymbol{\phi}^t + \Delta t \boldsymbol{f} \tag{6.61}$$

$$\boldsymbol{\phi} = \boldsymbol{\phi}^{t+1}$$

Now the linear equation can be solved for the unknown $\boldsymbol{\phi}$ at each time step. For this simulation a conjugate gradient solver was used to solve Equation 6.59 for $\boldsymbol{\phi}$ at each time step. The complexity of the construction of the matrix A and vector b depend on the number of vertices N and the connectivity of each vertex. As such the complexity is $O(M)$, where M is the number of non-zero elements in the global matrices. The complexity of the CGS is described as $O(M\sqrt{K})$. Here, K is the maximum number of iterations specified for the CGS.

6.5 Deformation Volume

Corpses that have been mummified due to dessication show great structural changes. This can be observed in the Figures in Section 6.2. Water loss causes the tissue to shrink, resulting in a visible loss of volume. The soft tissue, such as skin and flesh, get compressed whereas bones do not change. This section deals with the dynamic deformation caused by the moisture loss described in the previous section. Nealen et al. (2006) and Gibson and Mirtich (1997) provide a good overview of physically based deformation techniques. For the mummification method described here a differential equation is constructed from Newton's second law of motion that describes the dynamic deformation of an object based on Hooke's law. This is then formulated in a FEM framework, similar to the humidity diffusion in the previous section.

6.5.1 Physical Background

The object to be mummified is a continuous connected volume. The deformation of a continuous object can be described by continuum mechanics in terms of forces, stresses and strains. Deformation of an object occurs when a force is applied to its surface which, in turn, exerts strain on the affected material. The deformation behaviour of a material can be described by Hooke's law (Nealen et al. (2006)) for Hookean materials. Hooke's law states that the greater a force f applied to an area A, the greater the relative expansion of an object. For a

beam of length l, Hooke's law states that

$$\frac{f}{A} = E\frac{\Delta l}{l} \tag{6.62}$$

where Δl is the elongation of the beam. This can also be expressed in terms of strain and stress.

$$\sigma = E\varepsilon \tag{6.63}$$

Here $\sigma = \frac{f}{A}$ is applied stress, $\varepsilon = \frac{\Delta l}{l}$ is the resulting strain and E is Young's modulus which describes the stiffness of the material. Stress leads to strain within the object, which in turn causes the object to deform.

For the deformation simulation, the object is considered to be a continuous three-dimensional elastic solid. The force in three dimensions is described by $\boldsymbol{f} \in \mathbb{R}^3$ and therefore elasticity, stress and strain can no longer be described by a scalar value.

Stress is the result of a force acting on a surface. There are two kinds of stress, normal stress and shear stress. Normal stress is the stress that is perpendicular to the surface, whereas sheer stress in tangent to the surface. Take for example the surface x created by the yz axis. The normal stress is the stress perpendicular to the yz-plane and, as such, runs along the x-axis. The shear stress lies on the yz-plane, and hence runs along the y-axis and z-axis. This is also true for the xy-plane and xz-plane. This can be expressed in the following matrix

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix} \tag{6.64}$$

The first subscript x of ε_{xy} describes the surface on which the stress is acting (i.e. yz-plane) whereas the second subscript represents the direction of the stress. Hence the diagonal represents the normal stress components ($\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}$) and the off-diagonal describes the shear stress components ($\varepsilon_{xy}, \varepsilon_{yz}, \varepsilon_{xz}$). Note that the stress tensor is symmetrical and therefore has only six distinct values and is therefore often expressed by the 6-dimensional vector below.

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z & \sigma_{xy} & \sigma_{yz} & \sigma_{zx} \end{bmatrix} \tag{6.65}$$

Strain can also be expressed in a 3 by 3 matrix as shown below.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{bmatrix} \tag{6.66}$$

Strain is the relative expansion or compression of a material due to stress. Similar to stress, the diagonal holds the normal strains and the off-diagonal the shear strains and its six distinct values can be represented by the six dimensional vector below.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z & \varepsilon_{xy} & \varepsilon_{yz} & \varepsilon_{zx} \end{bmatrix} \tag{6.67}$$

The strain generally varies within an object and hence needs to be defined at each point $\mathbf{x}$ inside the object. No deformation causes no strain and constant deformation of the entire object (i.e. a translation and rotation) also yields no strain. The strain must therefore depend on the spatial variation of the displacements. There are different formulas to compute the strains. For this project the Cauchy linear strain tensor is used to compute the strain tensor which looks as follows

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + [\nabla \boldsymbol{u}]^T) \tag{6.68}$$

with

$$\nabla \boldsymbol{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \tag{6.69}$$

where $\boldsymbol{u}$ is the displacement.

This yields the generalised form of Hooke's law

$$\boldsymbol{\sigma} = \boldsymbol{E}\boldsymbol{\varepsilon} \tag{6.70}$$

Now $\boldsymbol{E} \in \mathbb{R}^{6\times 6}$ is no longer expressed by a single scalar but instead by the 6 by

6 dimensional stiffness matrix below

$$\boldsymbol{E} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu) \end{bmatrix} \tag{6.71}$$

where E is still Young's modulus describing the material stiffness, and $\nu \in [0...\frac{1}{2})$ is the Poisson ratio, which describes the material's volume conservation properties. Strain and stress are also no longer scalars.

This strain-stress relationship described by Hooke's law can be used to simulate the dynamics of an elastic object. Newton's second law of motion for an object with constant mass states that $\boldsymbol{f} = m\boldsymbol{a}$, where m is the object's mass. This means that the sum of all forces $\boldsymbol{f}$ acting on an object equals the object's mass m times its acceleration $\boldsymbol{a} = \ddot{\boldsymbol{x}}$. Dividing both sides by the element's volume yields the following equation of body forces acting on the element at point $\mathbf{x} \in \mathbb{R}^3$:

$$\rho\ddot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{6.72}$$

Here $\boldsymbol{f}(\boldsymbol{x})$ includes all external forces $\boldsymbol{f}(\boldsymbol{x})_{ext}$ and the internal forces $\boldsymbol{f}(\boldsymbol{x})_{int}$ that are caused by deformations. Hence

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})_{ext} + \boldsymbol{f}(\boldsymbol{x})_{int} \tag{6.73}$$

The internal forces acting on $\boldsymbol{x}$ are due to internal stress and can be expressed by

$$\boldsymbol{f}_{int} = \nabla\sigma \tag{6.74}$$

Using Hooke's law from Equation 6.62 yields the following equation for describing the internal forces

$$\boldsymbol{f}_{int} = \nabla(\boldsymbol{E}\boldsymbol{\varepsilon}) \tag{6.75}$$

and Newton's second law of motion can now be described in terms of strain by:

$$\rho\ddot{\boldsymbol{x}} = \nabla(\boldsymbol{E}\boldsymbol{\varepsilon}) + \boldsymbol{f}_{ext} \tag{6.76}$$

This equation can be used to compute the changes in positions $\dot{\boldsymbol{x}}$ when external forces $\boldsymbol{f}_{ext}$ are applied to the object's surface. This is carried out using the Finite Element Method.

6.5.2 FEM Formulation of Deformation

The deformation of an object is described by a vector field $\boldsymbol{u}(\boldsymbol{x})$ described as $\boldsymbol{u}(\boldsymbol{x}) = [u(\boldsymbol{x}), v(\boldsymbol{x}), w(\boldsymbol{x})]^T$ for every point $\boldsymbol{x}$ in the continuous object. In order to model the deformation, the displacement $\boldsymbol{u}$ for any point of the object needs to be determined from which the new positions $\boldsymbol{x}$ can be computed. This can be achieved with the help of the FEM in a similar manner to the humidity diffusion described in Section 6.4. Equation 6.76 will then only need to be solved at a finite number of points, namely the vertices of the tetrahedralisation. The Finite Element discretisation and shape function are the same as the ones described in the humidity diffusion simulation.

Using Equations 6.6 and 6.30, strain can be expressed as $\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{u}$ and stress becomes $\boldsymbol{\sigma} = \boldsymbol{E}\boldsymbol{\varepsilon} = \boldsymbol{E}\boldsymbol{B}\boldsymbol{u}$. The internal strain described in Equation 6.75 reduces to the following sum:

$$\boldsymbol{f}_{int} = \sum_j \Big(\int_\Omega \boldsymbol{B}_j \boldsymbol{E} \boldsymbol{B}_i \, dV \Big) \boldsymbol{u} \tag{6.77}$$

for all vertices i. This means that the internal elastic forces acting on the nodes of an element e depend on the strain. This is similar as shown in Equation 6.23, this holds for each continuous element e

$$\boldsymbol{f}^e_{int} = \boldsymbol{K}^e \boldsymbol{u}, \tag{6.78}$$

with

$$\boldsymbol{K}^e = \boldsymbol{B}^{eT} E \boldsymbol{B}^e V^e. \tag{6.79}$$

Here $\boldsymbol{K}^e$ is the elemental stiffness matrix. The left hand side of Equation 6.72

can also be expressed as a sum over the object's elements:

$$\rho\ddot{\boldsymbol{x}} = \sum_j \Big(\rho \int_\Omega \boldsymbol{N}_j \boldsymbol{N}_i \, dV\Big)\ddot{\boldsymbol{x}} = \tag{6.80}$$

This integral is an elemental mass matrix $\boldsymbol{M}^e$ as described in Equation 6.33 and can be described by

$$\boldsymbol{M}^e = \rho V^e \boldsymbol{N}^{eT} \boldsymbol{N}^e \tag{6.81}$$

The global stiffness matrix $\boldsymbol{K}$ and the global mass matrix $\boldsymbol{M}$ can be assembled as described in Equation 6.38 and 6.39 from Section 6.3 respectively.

For this project the following Lagrangian dynamic deformation equation is used to describe the elastic deformation of a material.

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{K}(\boldsymbol{x} - \boldsymbol{x}_0) = \boldsymbol{f}_{ext} \tag{6.82}$$

Here $\boldsymbol{D} = \alpha\boldsymbol{M} + \beta\boldsymbol{K}$ is the Rayleigh damping matrix, which is a linear combination of the mass and stiffness matrix. Note that $\boldsymbol{x} - \boldsymbol{x}_0 = \boldsymbol{u}$, so Equation 6.78 can be rewritten as

$$\boldsymbol{f}_{int} = \boldsymbol{K}^e(\boldsymbol{x} - \boldsymbol{x}_0). \tag{6.83}$$

Let $f_0 = -\boldsymbol{K}\boldsymbol{x}_0$, then the following system of second order ordinary differential equations (ODEs) describes the dynamic deformation of the object.

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} + \boldsymbol{f}_0 = \boldsymbol{f}_{ext} \tag{6.84}$$

Equation 6.84 is a linear system of $3n$ equations, where n is the number of position vectors making up the tetrahedralisation. Linear elastic forces only require the stiffness matrix $\boldsymbol{K}$ to be computed once at initialisation. They also work well with an implicit Euler solver, which is what is used for this project. There is a drawback when using linear stiffness however; this does not work well under large deformations. Müller and Gross (2004) describe the artefacts that can appear under large rotational deformation when using linear stiffness, such as an unrealistic growth in volume. Müller et al. (2002) and Müller and Gross (2004) introduce an approach called elemental-warped stiffness that allows the use of a constant stiffness matrix while avoiding the artefacts caused by large

rotational deformations associated with linear stiffness.

This is achieved by extracting the rotational part $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ from the tetrahedron's deformation. The deformed position $\boldsymbol{x}$ can be rotated back to its un-rotated position with $\boldsymbol{x}_i\boldsymbol{R}^{-1}$. The force from Equation 6.83 is computed with the un-rotated $\boldsymbol{x}$ and then rotated back, as shown below:

$$\boldsymbol{f}_{int} = \boldsymbol{R}\boldsymbol{K}^e(\boldsymbol{x}\boldsymbol{R}^{-1} - \boldsymbol{x}_0) \tag{6.85}$$

This is done for all four vertices of the tetrahedra using the tetrahedral rotation matrix $\boldsymbol{R}^e$. The elemental stiffness matrix used to perform stiffness warping is then expressed as

$$\boldsymbol{K}^{'e} = \boldsymbol{R}^e\boldsymbol{K}^e(\boldsymbol{R}^e)^{-1} \tag{6.86}$$

and the $\boldsymbol{f}_0$ from Equation 6.84 becomes

$$\boldsymbol{f}^{'e} = -\boldsymbol{R}^e\boldsymbol{K}^e\boldsymbol{x}_0, \tag{6.87}$$

where $\boldsymbol{R}^e$ is the 12 by 12 matrix, which has four copies of the rotation matrix $\boldsymbol{R}$ along its diagonal and is zero everywhere else. Then f_{int} becomes

$$\boldsymbol{f}_{int} = \boldsymbol{K}^{'e}\boldsymbol{x} + \boldsymbol{f}^{'e}. \tag{6.88}$$

Müller and Gross (2004) proposed the following way of extracting the rotation matrix from the deformation. Let $\boldsymbol{p1}$, $\boldsymbol{p2}$, $\boldsymbol{p3}$ and $\boldsymbol{p4}$ be the positions of the four vertices on the undeformed tetrahedron and $\boldsymbol{q1}$, $\boldsymbol{q2}$, $\boldsymbol{q3}$ and $\boldsymbol{q4}$ the positions of the same vertices on the deformed tetrahedron. Then, the two matrices below describe the tetrahedron in the undeformed and deformed state respectively.

$$\boldsymbol{P} = \begin{bmatrix} p1x & p2_x & p3x & p4_x \\ p1y & p2_y & p3y & p4_y \\ p1z & p2_z & p3z & p4_z \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{6.89}$$

$$\boldsymbol{Q} = \begin{bmatrix} q1x & q2_x & q3x & q4_x \\ q1y & q2_y & q3y & q4_y \\ q1z & q2_z & q3z & q4_z \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{6.90}$$

The deformation of the tetrahedron can be described by the deformation of its four vertices and a transformation matrix $\boldsymbol{T} \in \mathbb{R}^{4\times4}$.

$$\boldsymbol{Q} = \boldsymbol{TP} \tag{6.91}$$

Hence, the deformation matrix $\boldsymbol{T}$ can be obtained by

$$\boldsymbol{T} = \boldsymbol{QP^{-1}} \tag{6.92}$$

where $\boldsymbol{T}$ has the form

$$\boldsymbol{T} = \begin{bmatrix} & & & t_x \\ & \boldsymbol{RS} & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.93}$$

Here $\boldsymbol{RS}$ is a 3 by 3 matrix describing the rotation and stretching transformations. The rotation matrix $\boldsymbol{R}$ can be extracted from $\boldsymbol{RS}$ using Polar Decomposition.

There are some other advantages to using stiffness warping reported by Müller and Gross (2004). These include avoiding ghost forces, higher stability, faster simulation and better quality. Please see the Müller and Gross (2004) work for a fuller description of this.

6.5.3 Humidity interface

The dynamic deformation of the volume is controlled by changes in the object's moisture content. The removal of moisture from an object creates an imbalance between the object's internal and external pressure. This pressure imbalance creates contraction stresses that in turn lead to the shrinking of the object observed in drying material. In the early stages of dehydration of a material with high moisture content the material's volume decreases nearly linearly with its water loss. At some point the material loses enough water to change from the "rubbery" to the "glassy" stage at which point the amount of water loss can be notably higher than the corresponding volume loss. Mayor and Sereno (2004) assume that the total mass of an object consists of dry solid, water and air. The air contribution is generally required to compute the porosity of the material, which is not considered in this approach (see below). For this project the air contribution to an object's mass has been omitted for simplicity.

Shrinkage refers to a dimensional reduction in an object's volume, area or thickness. In the case of volume this is often described by the volumetric shrinkage coefficient $Sv = \frac{V}{V_0}$, where V and V_0 are the current and initial volume of the object respectively. In order to model shrinkage the correlation between volume loss to moisture content needs to be described. Mayor and Sereno (2004) give an overview of different empirical and fundamental shrinkage models, most of which describe the shrinkage of vegetables. There is however, a model for describing the volume shrinkage of a slab of beef, which is

$$S_v = \frac{1}{1-\varrho}\Big(1 + \frac{\rho_0(\phi - \phi_0)}{\rho_w(1+\phi_0)}\Big) \tag{6.94}$$

or with the Mayor and Sereno (2004) correction,

$$S_v = \frac{1}{1-\varrho}\Big(1 + \frac{\rho_0(\phi - \phi_0)}{\rho_w(1+\phi_0)} - \varrho_0\Big) \tag{6.95}$$

Here ϱ_0 and ϱ are the initial and current porosity of the material and ρ_w is the density of water. As in the previous section ϕ_0, ϕ and ρ_0 are the initial water content, the current water content and the solid density. Porosity refers to the ratio of air volume to the total volume, described as

$$\varrho = \frac{V_a}{V_s + V_w + V_a} \tag{6.96}$$

where V_a, V_s and V_w are the volumetric air, solid and water content of the material.

Note that if the porosity term is kept constant at 0 and $\rho_0 = \rho_w$ is assumed, then the above becomes

$$S_v = 1 + \frac{\phi - \phi_0}{1+\phi_0} = \frac{\phi}{1+\phi_0} + 1 - \frac{\phi_0}{1+\phi_0} = \Big(\frac{\phi_0}{1+\phi_0}\frac{\phi}{\phi_0}\Big) + \Big(1 - \frac{\phi_0}{1+\phi_0}\Big) \tag{6.97}$$

where the term $\frac{\phi_0}{1+\phi_0}$ describes the fraction of the initial moisture content in the material. Let $\mu = \frac{\phi_0}{1+\phi_0}$ and $\lambda = 1 - \frac{\phi_0}{1+\phi_0}$ then this resembles the equation to calculate the volume shrinkage coefficient described in Liu et al. (2012b), when a moisture content of 87.35% is used. In contrast to the Liu et al. (2012b) method, the density term $\frac{\rho_0}{\rho_w}$ is considered for this simulation when computing the volume shrinkage coefficient. The porosity term is kept at 0 throughout the simulation, as its effect on visual realism is negligible. With the humidity

values known for each time step (Section 6.4), the volume shrinkage coefficient is calculated using Equation 6.95.

The volume shrinkage coefficient is used to calculate the shrinkage strains that drive the object's deformation as described in Liu et al. (2012b) and Yang et al. (2001):

$$\boldsymbol{\varepsilon_{ml}} = \gamma \begin{bmatrix} S_x & S_y & S_z & 0 & 0 & 0 \end{bmatrix}^T \tag{6.98}$$

Let $\boldsymbol{u}_{ml}$ be the deformation caused by moisture loss. Then, using Equation 6.88 the internal forces due to moisture loss can be described by

$$\begin{aligned} \boldsymbol{f}^e_{ml} &= \boldsymbol{R}^e \boldsymbol{K}^e \boldsymbol{u}_{ml} \\ &= \boldsymbol{R}^e \boldsymbol{K}^e \boldsymbol{B}^{e-1} \boldsymbol{\varepsilon}_{ml} \\ &= \boldsymbol{R}^e (V^e \boldsymbol{B}^{eT} \boldsymbol{E} \boldsymbol{B}^e) \boldsymbol{B}^{e-1} \boldsymbol{\varepsilon}_{ml} \\ &= \boldsymbol{R}^e V^e \boldsymbol{B}^{eT} \boldsymbol{E} \boldsymbol{\varepsilon}_{ml} \\ &= \boldsymbol{R}^e \boldsymbol{W}^e \varepsilon_{ml} \end{aligned} \tag{6.99}$$

where

$$\mathbf{W}^e = V^e \boldsymbol{B}^{eT} \boldsymbol{E} \tag{6.100}$$

The total strain is the sum of the elastic strain and the strain caused by water loss, i.e. $\boldsymbol{\varepsilon}_{total} = \boldsymbol{\varepsilon}_{elastic} + \boldsymbol{\varepsilon}_{ml}$. Then, $\boldsymbol{\varepsilon}_{elastic} = \boldsymbol{\varepsilon}_{total} - \boldsymbol{\varepsilon}_{ml}$ and hence the system of second order ODEs from Equation 6.84 becomes

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{K}'\boldsymbol{x} + \boldsymbol{f}'_0 - \boldsymbol{f}_{ml} = \boldsymbol{f}_{ext} \tag{6.101}$$

which considers the forces $\boldsymbol{f}_{ml}$ from the changes in water content.

6.5.4 Solving The Equation of Volume Deformation

The deformation of an object is described by the changes in the positions $\boldsymbol{x}$ of its vertices over time. To simulate this the system of second order ODEs described in Equation 6.101 is first rewritten as a system of coupled first order ODEs as shown below

Find $\boldsymbol{x}$ such that (6.102)

$$\dot{\boldsymbol{x}} = \boldsymbol{v}$$
$$\boldsymbol{M}\dot{\boldsymbol{v}} = -\boldsymbol{D}\boldsymbol{v} - \boldsymbol{K}\boldsymbol{x} - \boldsymbol{f}_0 + \boldsymbol{f}_{ml} + \boldsymbol{f}_{ext}$$

As the positions $\boldsymbol{x}$ need to be computed at any time t they become functions of time, i.e. $\boldsymbol{x}(t)$. The analytical solution for the above coupled ODE then looks as follows:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{v}(t)\ dt \tag{6.103}$$

$$\boldsymbol{M}\boldsymbol{v}(t) = \boldsymbol{M}\boldsymbol{v}_0 + \int_{t_0}^{t} (-\boldsymbol{D}\boldsymbol{v}(t) - \boldsymbol{K}\boldsymbol{x}(t) - \boldsymbol{f}_0 + \boldsymbol{f}_{ml} + \boldsymbol{f}_{ext})\ dt \tag{6.104}$$

with initial conditions $\boldsymbol{x}_0 = \boldsymbol{x}(0)$ and $\boldsymbol{v}_0 = \boldsymbol{v}(0)$. This can be approximated by a numerical solution. For this project the implicit Euler method is used to compute $v(t)$ followed by $x(t)$ at each time step t.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \Delta t \boldsymbol{v}^{(t+1)} \tag{6.105}$$

$$\boldsymbol{M}\boldsymbol{v}^{(t+1)} = \boldsymbol{M}\boldsymbol{v}^{(t)} + \Delta t(-\boldsymbol{D}\boldsymbol{v}^{(t+1)} - \boldsymbol{K}\boldsymbol{x}^{(t+1)} - \boldsymbol{f}_0 + \boldsymbol{f}_{ml} + \boldsymbol{f}_{ext}) \tag{6.106}$$

Note that (using Equation 6.105) $\boldsymbol{K}\boldsymbol{x}^{(t+1)}$ can be expressed as $\boldsymbol{K}\boldsymbol{x}^{(t+1)} = \boldsymbol{K}\boldsymbol{x}^{(t)} + \Delta t \boldsymbol{K}\boldsymbol{v}^{(t+1)}$. The above system of ODEs in Equation 6.106 can be rearranged in terms of the unknown $\boldsymbol{v}^{(t+1)}$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \Delta t \boldsymbol{v}^{(t+1)} \tag{6.107}$$

$$(\boldsymbol{M} + \Delta t \boldsymbol{D} + \Delta t^2 \boldsymbol{K})\boldsymbol{v}^{(t+1)} = \boldsymbol{M}\boldsymbol{v}^{(t)} - \Delta t(\boldsymbol{K}\boldsymbol{x}^{(t)} + \boldsymbol{f}_0 - \boldsymbol{f}_{ml} - \boldsymbol{f}_{ext}) \tag{6.108}$$

which is the system of equations

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \Delta t \boldsymbol{v}^{(t+1)} \tag{6.109}$$

$$\boldsymbol{A}\boldsymbol{v}^{(t+1)} = \boldsymbol{b} \tag{6.110}$$

where the $3n$ by $3n$ matrix $\boldsymbol{A} = \boldsymbol{M}+\Delta t\boldsymbol{D}+\Delta t^2\boldsymbol{K}$ and the $3n$-dimensional vector $\boldsymbol{b} = \boldsymbol{M}\boldsymbol{v}^{(t)}-\Delta t(\boldsymbol{K}\boldsymbol{x}^{(t)}+\boldsymbol{f}_0-\boldsymbol{f}_{ml}-\boldsymbol{f}_{ext})$. The system of Equation 6.110 can then be solved for $\boldsymbol{v}^{(t+1)}$ using a conjugate gradient solver and the results can be used to compute $\boldsymbol{x}^{(t+1)}$ for each time step. The computational complexity for solving the volume deformation is as mentioned in Section 6.4.3. The computational cost, however, is greater. This is mainly due to the larger number of non-zero matrix entries caused by deformation having three degrees of freedom (x, y, z) and humidity diffusion only one (moisture content).

6.6 Deformation of the Skin

The previous section described the deformation of the internal volume (flesh) due to dehydration. As the internal volume shrinks, the skin undergoes similar structural changes. The skin compresses due to water loss but is also pulled along the deformation of the volume mesh. This results in the skin clinging firmly to the underlying layers and wrinkling, as can be observed in Figures 39, 38 and 37. Similar effects have been achieved when simulating fruit rotting as described by Liu et al. (2012b) and Kider et al. (2011) by defining the skin mesh as a mass-spring system. However, the deformation behaviour is very dependent on the spring network structure and tuning the constants in mass spring systems can be difficult.

In order to simulate the deformation of the skin in this mummification, simulation position based dynamics are employed. Position based dynamics, as described by Müller et al. (2007), work directly on the positions of the vertices. This allows for direct manipulation of the object vertices during the simulation. Furthermore, position based dynamics allow for stable simulations. In the position based dynamic framework the skin is represented by a set of N particles (vertices) and M constraints. Each vertex of the triangle mesh acts as a particle. Table 6.2 shows the attributes of each particle and constraints are defined by Table 6.3

m_i	mass
$\boldsymbol{x}_i$	position
$\boldsymbol{v}_i$	velocity

Table 6.2 The attributes of each particle $i \in [1, ...N]$ in the position based dynamics framework. The positions are the vertex coordinates of the triangle mesh representing the skin.

n_j	cardinality
$C_j : \mathbb{R}^{3n_j} \rightarrow \mathbb{R}$	scalar constraint function
$\{i_1, ...i_{n_j}\}, i_k \in [1, ...N]$	set of particle indices
k_j	stiffness parameter

Table 6.3 The attributes of each constraint $j \in [1...M]$ in the position based dynamics framework.

The dynamic motion of the triangle mesh is simulated by estimating new particle positions due to external forces. The new positions are then corrected to satisfy some predefined constraints. Below is an overview of the Müller et al. (2007) algorithm (Algorithm 3). First, the positions $\boldsymbol{x}_i$, velocities $\boldsymbol{v}_i$ and weights $w_i = 1/m_i$ are initialised for all vertices as shown in Lines 1 to 3. Explicit Euler integration is used to compute new approximate positions. At each time step (Δt) the new velocity due to external forces ($\boldsymbol{f}_{ext}$) and pulling forces ($\boldsymbol{f}_{pull}$) of each particle is computed (lines 6-7). Damping is applied to the velocities at this point, as shown in line 8. The damping method proposed by Müller et al. (2007) is used for this project. The method is described in algorithm 4. The damped velocities are then used to compute approximated positions ($\boldsymbol{p}_i$) in line 11. With the approximated positions known, the collision can be detected and collision constraints generated (line 14). The estimated positions need to be corrected such that the $M + M_{coll}$ constraints are satisfied (lines 16-18). This is done iteratively using a Gauss-Seidel type solver explained below. Finally, the positions and velocity are both updated as shown in lines 19 to 22. The velocities need to be updated to account for collisions (line 23).

The skin deformation is treated as a cloth-like simulation driven by the deformation of the underlying volume mesh by tracking constraints (Section 6.6.3). In order to simulate the cloth-like dynamics, stretching, bending and collision constraints need to be constructed and projected. In contrast to cloth dynamics, skin has lower stretching stiffness. Furthermore, the stiffness changes with the moisture loss.

6.6.1 The Constraint Solver

Applying the external forces to the vertices of the skin mesh results in a set of predicted positions $\boldsymbol{p}_i$, which need to be corrected to fulfil the constraints. This is done by moving the vertices until they satisfy the constraints. This is called constraint projection. Müller et al. (2007) propose a method for constraint projection that conserves both linear and angular momentum. $\nabla \boldsymbol{p}_i$ is the displacement of a vertex i by projection, called the correction vector. Given,

Algorithm 3 Position based dynamics algorithm based on Müller et al. (2007)

for all vertices i **do**
    initialise $\boldsymbol{x}_i = \boldsymbol{x}_i^0$, $\boldsymbol{v}_i = \boldsymbol{v}_i^0$, $w_i = \frac{1}{m_i}$
end for
loop
for all vertices i **do**
    $\boldsymbol{v}_i := \boldsymbol{v}_i + \Delta t w_i \boldsymbol{f}_{ext}(\boldsymbol{x}_i)$
    $\boldsymbol{v}_i := \boldsymbol{v}_i + \Delta t \boldsymbol{f}_{pull}(\boldsymbol{x}_i)$
    dampVelocities$(\boldsymbol{v}_1, ..., \boldsymbol{v}_N)$
end for
for all vertices i **do**
    $\boldsymbol{p}_i := \boldsymbol{x}_i + \Delta t \boldsymbol{v}_i$
end for
for all vertices i **do**
    generateCollisionConstraints$(\boldsymbol{x}_1, ..., \boldsymbol{x}_N, \boldsymbol{p}_1, ..., \boldsymbol{p}_N)$
end for
loop solverIterations **times**
ProjectConstraints$(C_1, ..., C_{M+M_{coll}}, \boldsymbol{p}_1, ..., \boldsymbol{p}_N)$
endloop
for all vertices i **do**
    $\boldsymbol{v}_i := \frac{\boldsymbol{p}_i - \boldsymbol{x}_i}{\Delta t}$
    $\boldsymbol{x}_i := \boldsymbol{p}_i$
end for
velocityUpdate$(\boldsymbol{v}_1, ..., \boldsymbol{v}_N)$
endloop

Algorithm 4 The damping algorithm by Müller et al. (2007). Here $\tilde{\boldsymbol{r}}_i \in \mathbb{R}^{3\times3}$ has the property $\tilde{\boldsymbol{r}}_i \boldsymbol{v} = \boldsymbol{r}_i \times \boldsymbol{v}$. $k_{damping}$ is the damping coefficient. See Müller et al. (2007) for more information.

$\boldsymbol{x}_{cm} := \Big(\sum_i \boldsymbol{x}_i m_i\Big) / \Big(\sum_i m_i\Big)$
$\boldsymbol{v}_{cm} := \Big(\sum_i \boldsymbol{v}_i m_i\Big) / \Big(\sum_i m_i\Big)$
for all vertices i **do**
    $\boldsymbol{r}_i := \boldsymbol{x}_i - \boldsymbol{x}_{cm}$
end for
$\boldsymbol{L} := \Big(\sum_i \boldsymbol{r}_i \times (m_i \boldsymbol{v}_i)\Big)$
$\boldsymbol{I} := \Big(\sum_i \tilde{\boldsymbol{r}}_i \tilde{\boldsymbol{r}}_i^T \times m_i\Big)$
$\boldsymbol{\omega} := \boldsymbol{I}^{-1} \boldsymbol{L}$
for all vertices i **do**
    $\Delta \boldsymbol{v}_i := \boldsymbol{v}_{cm} + \boldsymbol{\omega} \times \boldsymbol{r}_i - \boldsymbol{v}_i$
    $\boldsymbol{v}_i := \boldsymbol{v}_i + k_{damping} \Delta \boldsymbol{v}_i$
end for

$\boldsymbol{p} \in \mathbb{R}^{3N}$ which is a vector of the concatenated approximations, a global correction vector $\nabla \boldsymbol{p}$ needs to be found such that $C(\boldsymbol{p}+\nabla \boldsymbol{p}) = 0$. This means a set of $M+M_{coll}$ equations with $3N$ unknowns needs to be solved. Generally $3N < M$, and hence the the system is over-determined. Furthermore, the equations are generally non-linear (Müller et al. (2007)).

Müller et al. (2007) propose a Newton-Raphson iteration type method to solve the non-linear system. At first, a guess of a solution is taken. In this case these are the estimated positions $\boldsymbol{p}$. Each constraint function C_k is then linearised in the neighbourhood of the current solutions. This is described by the following equation:

$$C(\boldsymbol{p} + \nabla \boldsymbol{p}) = C(\boldsymbol{p}) + \nabla_p C(\boldsymbol{p}) \cdot \nabla \boldsymbol{p} + O(|\nabla \boldsymbol{p})|^2) = 0, \tag{6.111}$$

which yields the linear system

$$\begin{aligned} \nabla_p C_1(\boldsymbol{p}) \cdot \nabla \boldsymbol{p} &= -C_1(\boldsymbol{p}) \\ &... \\ \nabla_p C_M(\boldsymbol{p}) \cdot \nabla \boldsymbol{p} &= -C_M(\boldsymbol{p}). \end{aligned} \tag{6.112}$$

Here $\nabla_p C_j(\boldsymbol{p})$ is a $1 \times N$ dimensional vector holding the derivatives of C_j with respect to $\boldsymbol{p}_1, ..\boldsymbol{p}_N$. These need to be solved for $\nabla \boldsymbol{p}$.

The solver proposed by Müller et al. (2007) is a Gauss-Seidel-type (GS) iteration solver. It resembles a GS solver in that it solves each constraint independently. This works by repeatedly iterating through the constraints and projecting the particles to a position satisfying the constraint. For this the global correction vector $\nabla \boldsymbol{p}$ is computed for each constraint at each iteration. In order to solve Equation 6.111 for $\nabla \boldsymbol{p}$, $\nabla \boldsymbol{p}$ is restricted to the direction of $\nabla_p C_j$. The following formula can be used to compute the correction vector for a particle i and constraint j:

$$\nabla \boldsymbol{p}_i = -s w_i \nabla_{p_i} C_j(\boldsymbol{p}), \tag{6.113}$$

where

$$s = \frac{C_j(\boldsymbol{p})}{\sum_j w_j |\nabla_{p_i} C_j(\boldsymbol{p})|^2} \tag{6.114}$$

and $w_i = \frac{1}{m_i}$ are the weights controlling the effect of the constraint on each of the vertices involved based on their relative masses m_i. With this $\nabla \boldsymbol{p}$ can be computed for each constraint individually. Then, the estimated positions $\boldsymbol{p}$ are corrected by simply applying the correction, i.e. $\boldsymbol{p} = \boldsymbol{p} + k' \nabla \boldsymbol{p}$. Here $k' = 1 - (1 - k_s)^{(n_s)}$ is the stiffness term with k_s being the stiffness parameter of the constraint and n_s the number of iterations of the Gauss-Seidel solver. Note that k' is used instead of directly using the stiffness parameter k_s. This is done in order to avoid an error otherwise introduced by the iterative solver (see Müller et al. (2007) for more details). The positions are updated for each constraint projection before projecting the next constraint. This will influence the projection of the next constraint.

The masses m_i of each vertex i are computed by taking the sum of the masses of each adjacent triangle divided by three. The mass of a triangle is the skin density times the area of the triangle.

6.6.2 Constraints for the Skin Dynamics

There are two constraints that handle the cloth-like skin deformation, stretching constraints and bending constraints. The constraints for both types are constructed at the start of the simulation. Additional collision constraints are constructed at each iteration of the solver.

Stretching Constraints

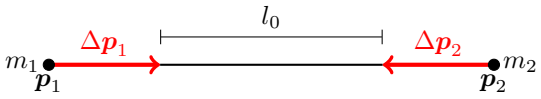


Figure 40: A stretching constraint between two points, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ with an initial distance of l_0. m_1 and m_2 are the masses of node 1 and 2 respectively. $\Delta \boldsymbol{p}_i$ are the corrections for each node, weighted by their masses $w_i = \frac{1}{m_i}$.

Stretch constraints are constructed for each edge $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ of the skin mesh. Figure 40 shows two points with predicted positions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ connected by an edge acting as a stretch constraint. The initial length of the connecting edge is l_0 and the current distance between the predicted positions is $|\boldsymbol{p}_1 - \boldsymbol{p}_2|$. The constraint function for a stretching constraint can thereby be described by

$$C_{stretch}(\boldsymbol{p}_1, \boldsymbol{p}_2) = |\boldsymbol{p}_1 - \boldsymbol{p}_2| - l_0 \tag{6.115}$$

l_0 is also called the rest length. In order to project the stretching constraint the two correction vectors $\Delta \boldsymbol{p}_1$ and $\Delta \boldsymbol{p}_2$ need to be computed and applied to the predictive positions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. The correction vectors can be computed with Equation 6.113. For this the derivatives of the stretch constraint C_j with respect to $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are needed. These are $\nabla_{\boldsymbol{p}_1} C_j(\boldsymbol{p}) = \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)}{|\boldsymbol{p}_2 - \boldsymbol{p}_1|} = \boldsymbol{n}$ and $\nabla_{\boldsymbol{p}_2} C_j(\boldsymbol{p}) = -\frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1)}{|\boldsymbol{p}_2 - \boldsymbol{p}_1|} = -\boldsymbol{n}$. Now inserting the Equation 6.115 and its derivatives into Equation 6.114 yields:

$$s = \frac{1}{w_1 + w_2}(|\boldsymbol{p}_2 - \boldsymbol{p}_1| - l_0) \tag{6.116}$$

and the correction vector Equation 6.113 for stretching constraints becomes:

$$\nabla \boldsymbol{p}_1 = -\frac{w_1}{w_1 + w_2}(|\boldsymbol{p}_1 - \boldsymbol{p}_2| - l_0)\frac{\boldsymbol{p}_1 - \boldsymbol{p}_2}{|\boldsymbol{p}_1 - \boldsymbol{p}_2|} \tag{6.117}$$

$$\nabla \boldsymbol{p}_2 = +\frac{w_2}{w_1 + w_2}(|\boldsymbol{p}_1 - \boldsymbol{p}_2| - l_0)\frac{\boldsymbol{p}_1 - \boldsymbol{p}_2}{|\boldsymbol{p}_1 - \boldsymbol{p}_2|} \tag{6.118}$$

Bending Constraints

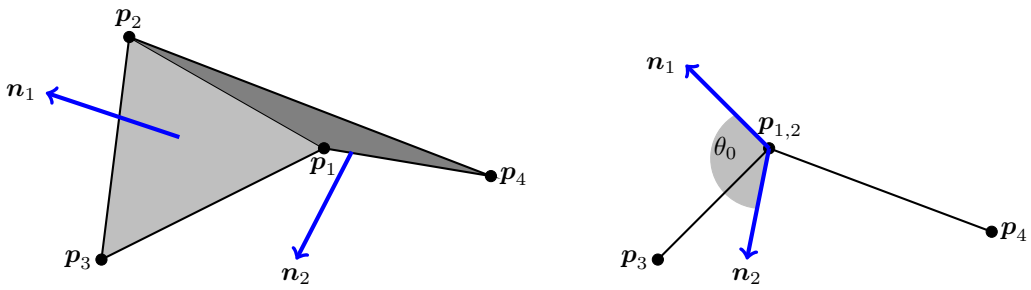


Figure 41: A bending constraint between two triangle surfaces with surface normals $\boldsymbol{n}_1$ and $-\boldsymbol{n}_2$. The bending is controlled by the initial dihedral angle θ_0, which is the angle between $\boldsymbol{n}_1$ and $-\boldsymbol{n}_2$.

Bending constraints are constructed for triangle pairs that share an edge. Figure 41 shows two adjacent triangles $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ and $(\boldsymbol{p}_1, \boldsymbol{p}_4, \boldsymbol{p}_2)$, that share the edge $(\boldsymbol{p}_1, \boldsymbol{p}_2)$ and have surface normals $\boldsymbol{n}_1$ and $-\boldsymbol{n}_2$ respectively. The bending is described by changes in the angle θ, that lies between the face normals. As such, the bending constraint function is:

$$C_{bending}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4) = \arccos(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) - \theta_0 \tag{6.119}$$

where θ_0 is the initial dihedral angle between the two triangles and

$$\boldsymbol{n}_1 = \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)}{|(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)|} \tag{6.120}$$

$$\boldsymbol{n}_2 = \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_4 - \boldsymbol{p}_1)}{|(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_4 - \boldsymbol{p}_1)|} \tag{6.121}$$

In order to compute the correction vector the derivatives of the constraint function with respect to the four points need to be found. Without loss of generality assume $\boldsymbol{p}_1 = 0$. Then $\boldsymbol{n}_1 = \frac{\boldsymbol{p}_2 \times \boldsymbol{p}_3}{|\boldsymbol{p}_2 \times \boldsymbol{p}_3|}$ and $\boldsymbol{n}_2 = \frac{\boldsymbol{p}_2 \times \boldsymbol{p}_4}{|\boldsymbol{p}_2 \times \boldsymbol{p}_4|}$. Also, note that $\frac{d}{dx} \arccos(x) = -\frac{1}{\sqrt{1-x^2}}$ With this the derivatives of the constraint function with respect to the four positions can be expressed by:

$$\nabla \boldsymbol{p}_i = -\frac{w_i \sqrt{1 - d^2}(arccos(d) - \theta_0)}{\sum_j w_j |\boldsymbol{q}_i|^2} \tag{6.122}$$

where $d = \boldsymbol{n}_1 \cdot \boldsymbol{n}_2$ and

$$\boldsymbol{q}_2 = \frac{(\boldsymbol{p}_3 \times \boldsymbol{n}_2) + (\boldsymbol{n}_1 \times \boldsymbol{p}_3)d}{|\boldsymbol{p}_2 \times \boldsymbol{p}_3|} - \frac{(\boldsymbol{p}_4 \times \boldsymbol{n}_1) + (\boldsymbol{n}_2 \times \boldsymbol{p}_4)d}{|\boldsymbol{p}_2 \times \boldsymbol{p}_4|} \tag{6.123}$$

$$\boldsymbol{q}_3 = \frac{(\boldsymbol{p}_2 \times \boldsymbol{n}_2) + (\boldsymbol{n}_1 \times \boldsymbol{p}_2)d}{|\boldsymbol{p}_2 \times \boldsymbol{p}_3|} \tag{6.124}$$

$$\boldsymbol{q}_4 = \frac{(\boldsymbol{p}_2 \times \boldsymbol{n}_1) + (\boldsymbol{n}_2 \times \boldsymbol{p}_2)d}{|\boldsymbol{p}_2 \times \boldsymbol{p}_4|} \tag{6.125}$$

$$\boldsymbol{q}_1 = -\boldsymbol{q}_2 - \boldsymbol{q}_3 - \boldsymbol{q}_4 \tag{6.126}$$

Collision Constraints

Figure 42: A collision constraint between a point $\boldsymbol{q}$ and a triangle surfaces $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ with surface normal $\boldsymbol{n}$. h is the skin thickness. $\boldsymbol{q}$ is required to stay outside the triangle a distance h away in the normal direction.

The collision detection of every point of the skin mesh is done using spatial hashing as described in Section 3.3.2. A vector from the current vertex position $\boldsymbol{x}_i$ to the predicted position $\boldsymbol{p}_i$ is created. Then spatial hashing is used to find a ray-triangle intersection. This can be done using the collision object representation of the skin mesh (self collision) or any other collision object in the scene whose boundary box is intersected by the vector $(\boldsymbol{p}_i - \boldsymbol{x}_i)$. If an intersection is found, the intersected triangle and the barycentric coordinates of the intersection point are returned. These are used to calculate the intersection point $\boldsymbol{q}_i$ of the predicted position $\boldsymbol{p}_i$. The intersected triangle is described by its three vertex positions $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$, face normal $\boldsymbol{n}$ and a thickness h (see Figure 42). For the skin simulation the thickness equals the epidermis thickness, which is set to $0.1mm$ for the skin deformation. In the case of an intersection, a collision constraint is created. The constraint function of the collision constraint looks as follows.

$$C_{coll}(\boldsymbol{q}, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) = (\boldsymbol{q} - \boldsymbol{p}_1) \cdot \boldsymbol{n}' - h \tag{6.127}$$

with

$$\boldsymbol{n}' = \begin{cases} \frac{(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)}{|(\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)|} h & \text{if } (\boldsymbol{q} - \boldsymbol{p}_1) \cdot \boldsymbol{n} \geq 0 \\ \frac{(\boldsymbol{p}_3 - \boldsymbol{p}_1) \times (\boldsymbol{p}_2 - \boldsymbol{p}_1)}{|(\boldsymbol{p}_3 - \boldsymbol{p}_1) \times (\boldsymbol{p}_2 - \boldsymbol{p}_1)|} & \text{otherwise} \end{cases} \tag{6.128}$$

Only the intersecting point $\boldsymbol{q}$ is corrected, not the vertices of the intersected triangle. The weights for the triangle vertices are therefore set to zero. Note that $\nabla_q C_j(\boldsymbol{q}_i, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) = \boldsymbol{n}'$. Since $|\boldsymbol{n}'| = 1$, Equation 6.114 for collision constraints is simply $s = C_j(\boldsymbol{q}_i, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$. This means the correction vector $\Delta \boldsymbol{q}_i$ for collision point i can be described by the following equation.

$$\nabla \boldsymbol{q}_i = -C_j(\boldsymbol{q}_i, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)\boldsymbol{n}' \tag{6.129}$$

As $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are not corrected, only $\nabla \boldsymbol{q}_i$ needs to be computed and applied.

6.6.3 Connection between skin and volume

In a living body the epidermis is attached to the underlying dermis by the basement membrane. As the internal parts of the body (subcutaneous tissue and muscle tissue) shrink due to dehydration (see Section 6.5) so does the skin. The internal body parts are represented by a tetrahedral volume mesh. For the skin a highly detailed triangle mesh is used to model the epidermis. The

boundary of the volume mesh acts as the dermis layer. The attachment of the two layers are modelled by connecting every vertex of the skin mesh to the underlying volume mesh using tracking points. These are located on the internal volume's boundary faces. A vertex's tracking point is the point at which a vector along the negative normal direction of the vertex intersects the volume boundary. The tracking point is described by the three vertices of the triangle face it intersects, using barycentric coordinates. This means that, as the internal volume mesh deforms, the tracking points located on its boundary move along. Additionally, the initial distance between the vertex and tracking point is stored for each connection. The new positions of the tracking points and the initial distances can then be used to compute the new positions of the skin vertices using position based dynamics. See Figure 43 for a diagram of a tracking connection between the surface mesh and the volume mesh.

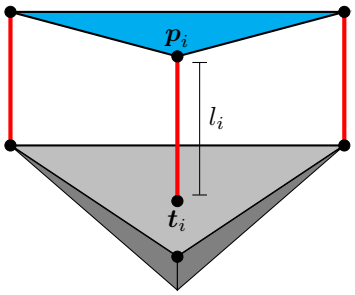


Figure 43: The grey triangle face is a boundary face of the tetrahedral volume mesh. The blue triangle is a surface element from the skin mesh. Every vertex $\boldsymbol{p}_i$ on the skin mesh has a tracking point $\boldsymbol{t}_i$ on the boundary faces of the underlying volume mesh. The red lines are springs of length l_i connecting each vertex to its tracking point.

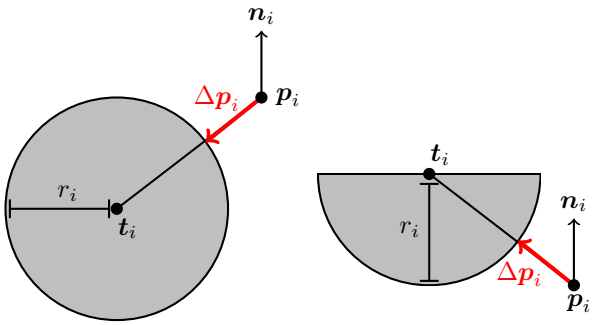


Figure 44: left: A tracking constraint of a node i with position $\boldsymbol{p}_i$ and tracking point $\boldsymbol{t}_i$ on the volume mesh boundary. A radius r_i is assigned to the constraint, which acts as the maximum distance $\boldsymbol{p}_i$ can move away from $\boldsymbol{t}_i$. $\boldsymbol{n}_i$ is the vertex normal of vertex $\boldsymbol{p}_i$. right: one sided attachment constraint to avoid collision of the surface vertex $\boldsymbol{p}_i$ with the volume boundary.

Müller and Chentanez (2010) propose an improvement to position based dynamics that is able to add more wrinkles to the cloth or skin animations. The method works by using position based dynamics to simulate cloth dynamics on a coarse surface mesh. A higher resolution surface mesh is attached to the coarse mesh. The vertices of the high resolution mesh are pulled along the coarse mesh, but are allowed to deviate a certain distance from the coarse mesh. A similar technique is used here to drive the deformation of the surface mesh representing the epidermis through deformation of the internal volume. The connection of the two layers by tracking points has been explained above. These can now be used to construct tracking constraints similar to the attachment constraints described by Müller and Chentanez (2010). Figure 44 shows a diagram describing a tracking constraint. The vertex with position $\boldsymbol{p}_i$ is connected to the tracking point $\boldsymbol{t}_i$. An initial radius r_j controls the maximum distance that $\boldsymbol{i}_i$ is allowed to move away from $\boldsymbol{t}_i$. The correction vector $\Delta\boldsymbol{p}_i$ is only calculated and applied if $|\boldsymbol{t}_i - \boldsymbol{p}_i| > r_j$. The tracking constraint is similar to the stretch constraint in Section 6.6.2, but with only $\boldsymbol{p}_i$ being allowed to move. As such, the constraint function for a tracking constraint j is described by

$$C_{tracking}(\boldsymbol{p}_i, \boldsymbol{t}_i) = |\boldsymbol{t}_i - \boldsymbol{p}_i| - r_j \tag{6.130}$$

The derivative of the constraint function with respect to $\boldsymbol{p}_i$ is similar to the stretching constraint, i.e. $\nabla_{p_i} C_j(\boldsymbol{p}_i, \boldsymbol{t}_i) = \frac{\boldsymbol{t}_i - \boldsymbol{p}_i}{|\boldsymbol{t}_i - \boldsymbol{p}_i|}$. The correction vector for a tracking constraint of a position $\boldsymbol{p}_i$ is then

$$\nabla\boldsymbol{p}_i = \frac{\boldsymbol{t}_i - \boldsymbol{p}_i}{|\boldsymbol{t}_i - \boldsymbol{p}_i|}(|\boldsymbol{t}_i - \boldsymbol{p}_i| - r_j) \tag{6.131}$$

The radius r_j controls the wrinkling strength and formation. In order to induce the wrinkle effects in the Müller and Chentanez (2010) method, this radius needs to be painted on the surface mesh. In the method proposed here, a maximum radius was chosen for the entire skin surface mesh, i.e. each vertex connects to the volume mesh with constant distance. The radius controls the maximum distance that the skin layer is allowed to separate from the underlying volume mesh. Initially the length of the tracking constraints will be close to zero, but will increase as the volume shrinks.

The tracking constraint is divided into two cases: the surface constraint is pointing outward (i.e. along the vertex normal direction) or inward (along reverse normal direction), assuming a directional constraint pointing from the surface vertex towards the tracking point on the volume boundary as shown in Figure 44. These two cases are handled separately.

An outward facing tracking constraint indicates that the vertex has penetrated the boundary of the volume mesh. In this case, the tracking constraint is treated as a one sided attachment constraint (see the right diagram in Figure 44). For one sided attachment constraints, the stiffness is set to 1 and the radius r_j is kept constant throughout the simulation. This forces a surface vertex that has penetrated the volume to move outwards again. The radius controls how deep the vertex is allowed to move into the volume, before it is moved outwards again. Allowing the vertex to move slightly within the volume allows for stronger wrinkle formation. The internal radius can be set to zero if the surface mesh is not allowed to intersect the volume mesh.

Initially, the inward facing constraint was handled similarly, with the difference that the constraint's stiffness and radius are not constant but change with the loss of moisture. However, using a uniform radius approach in combination with a tracking constraint resulted in smooth skin deformation with very little wrinkling formation. Therefore, a different approach was chosen for inward facing constraints. Instead of treating the inward facing constraints as normal tracking constraints, they are instead used to compute a pulling force in the direction of the tracking constraint. The strength of the force is controlled by the stiffness parameters and is only triggered if the constraint's length exceeds the radius. Let r_{max} be the maximum radius a vertex is allowed to move away from the volume boundary, then the radius r_j for tracking constraint j is computed as follows

$$r_j = (1 - k_j) r_{max} \tag{6.132}$$

where k_j is the constraint's stiffness. The stiffness parameter is affected by moisture changes during the dessication simulation. At the start, when the skin is still fully moisturised, the stiffness is large, causing the radius to be small. With the loss of moisture the radius gets larger, allowing the skin to separate more and more from the boundary of the volume mesh. This is further explained in the next section. The pulling force $\boldsymbol{f}_{pull,j}$ corresponding to tracking constraint j is computed as follows;

$$\boldsymbol{f}_{pull,j} = \begin{cases} \nabla \boldsymbol{p}_i & \text{if } |\boldsymbol{t}_i - \boldsymbol{p}_i| > r_j \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{6.133}$$

where $\nabla \boldsymbol{p}_i$ is the correction vector described in Equation 6.131. The pulling force is added to the velocity after the external forces but before the damping (see line 7 in Algorithm 3). The subsequent damping step is important

when using the pulling force as it directly affects the wrinkling formation. A low damping coefficient results in strong wrinkles, but can cause a wave like movement of the skin layer if set too small. A higher damping value removes this effect, but at the same times reduces the strength of the wrinkles. Using this method allows for greater control over the strength of wrinkling effects. Additionally, it grants an easy way to simulate the separation of the epidermis from the dermis associated with mummification by dessication (as described by Papageorgopoulou et al. (2015)) by loosening the tracking constraints.

The computational complexity of the skin deformation algorithm depends on the number of nodes N and number of constraints. There are $\frac{3}{2}T$ bending, $\frac{3}{2}T$ stretching and N tracking constraints, where T is the number of tetrahedra. The complexity of the skin dynamics is therefore $O(K(T+N))$, where K is the number of solver iterations (see line 16 in Algorithm 3).

6.6.4 Effects of Moisture Content on Skin Deformation

The skin deformation is mainly driven by the shrinking of the underlying volume mesh. However, dehydration also has a more direct effect on the skin structure, such as the thinning of the epidermis layer and the decrease in skin elasticity. These effects can be modelled by modifying the stiffness parameters of the different constraints so that they depend on the vertices' moisture content. Loss in skin elasticity can be modelled by increasing the stiffness of the stretch constraint. In order to account for the affects of epidermis thinning, the stiffness of the bending constraints can be decreased. A low stiffness in bending constraints results in more narrow folds, which gives the impression of a thinner material. However, altering the stiffness parameters of the bending and stretching constraints did not show any significant visual differences compared to using a high stretching stiffness and a very small bending stiffness from the onset.

Another effect of mummification by dessication is the separation of the epidermis from the dermis, called skin slippage described by Papageorgopoulou et al. (2015). To model the skin slippage the tracking constraints are modified. When the moisture content on a surface node reaches a predefined threshold, the stiffness of its tracking constraint is reduced in accordance with the water loss, until it reaches zero. Now the skin node's dynamics are no longer affected by the volume mesh, but, instead, driven entirely by external forces such as gravity and the internal forces of the surface mesh caused by stretching, bending and collision constraints. The stiffness k_j of tracking constraint is computed

as follows.

$$k_j = \begin{cases} \frac{m_i k_t}{\tau_d} & \text{if } m_i \leq \tau_d \\ 0 & \text{if } m_i < \tau_n \\ k_t & \text{otherwise} \end{cases} \tag{6.134}$$

where, m_i is the moisture content of surface node i, k_t is the initial tracking stiffness, and τ_d and τ_n are the moisture content thresholds from which to decrease and nullify the stiffness value respectively. The stiffness parameters are updated before the skin deformation loop.

6.7 Skin Shading

Not only the skin structure but also its surface appearance changes greatly during mummification. An example of a mummifying leg was shown in Figure 39. The leg shown in the Figure was mummified by being placed in a natron mix over a period of 208 days. On day 19 one can observe the brown leathery appearance of the skin on the foot. The lower leg shows some blackish discolouration under the white crust. During later stages the skin appears black and light lime green. It is not clear how much of the discolouration is caused by the natron rather than the dehydrated skin. Papageorgopoulou et al. (2015) report an early pinkish discolouration of the skin which turns greenish. After around 19 days the skin takes on a dark shade. The toes and foot appear leathery after around 25 days. After 40 days a dark green, brown and red appearance has been reported. On the sixtieth day the leg has taken on a brown leathery appearance. Figure 39 shows an older mummy. Here the skin looks very dark, though with a dust layer covering it. In some places, e.g. on the wrist, a more brownish colour is shining through.

I have not been able to find concrete explanations on what causes the changes in skin colouration during desiccation. There have been attempts to model the properties of ageing skin, which involves some skin drying, although to a much lesser degree. Iglesias-Guitian et al. (2015) concentrate on simulating the optical properties of skin ageing. The skin is represented as a five layer structure, each of which reflects the light differently due to the difference in chromophore concentrations. Three main substances are considered to affect skin appearance in the method described by Iglesias-Guitian et al. (2015), namely melanin, haemoglobin and water. Ageing skin undergoes a few structural changes. These are described by Iglesias-Guitian et al. (2015) as the thinning of the dermis and

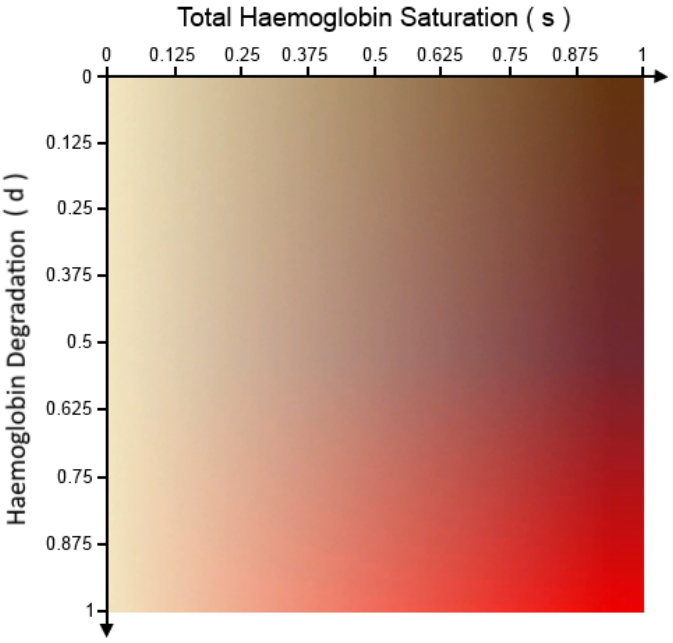


Figure 45: Dermis colour lookup texture used in the dermis rendering step. The total haemoglobin saturation s is the fraction of haemoglobin in blood which is obtained using Equation 6.136. The haemoglobin degradation of the blood d is obtained using Equation 6.135.

epidermis, the drying out of the skin, the flattening of the junction between the epidermis and dermis, roughening of the epidermis, decrease in melanin and haemoglobin contents and deterioration of collagen fibers. It is further reported that water loss causes reduced light reflection and increases absorption, especially in the dermis layer. A decrease in haemoglobin causes higher light reflectance in the dermis layer. Furthermore, the form of the light scattering profile changes with a wider radial distance of the lower wavelengths.

Although mummifying skin undergoes greater optical changes than ageing skin, some of the observations in Iglesias-Guitian et al. (2015) can be used as a guideline for the mummification skin shader. In particular the effects of water loss on the light reflectance and absorption properties of skin. Furthermore, there are reports on the changes in colour that occur by Papageorgopoulou et al. (2015) and Aufderheide (2003). For the reasons above an ad hoc method rather than a chemical based method is employed. The skin shading method is based on texture maps.

The skin shading approach for the mummification is an extension of the layered skin shader used in the livor mortis chapter in Section 5.4. The changes in skin

colour are driven by the changes in humidity within the skin vertices. One theory for the colouration changes caused by dehydration might be the degradation (or catabolism) of haemoglobin. The first phase of this (deoxygenation) was employed in the livor mortis dermis shading approach in terms of the oxygen content. Later stages in the haemoglobin degradation show as greenish and yellow skin colouration, as can be observed in the healing of bruises. The idea of haemoglobin degradation is used to control the dermis colouration during mummification.

The skin shader used for the livor mortis simulation is extended as follows. The dermis shader takes an additional parameter, namely the humidity content value. The humidity content is given alongside the haemoglobin saturation and oxygen level parameters provided by the livor mortis simulation. The new dermis shader receives the blood parameters $\left(\frac{h_i+f_i}{a_i}, o_i, m_i\right)$, where m is the moisture (or water) content. Furthermore, the blood colour look up texture (see Figure 25) used to retrieve the dermis colour from haemoglobin and oxygen values has been modified to include colouration changes caused by humidity diffusion. The new look up texture is shown in Figure 45. Instead of taking an oxygen saturation parameter, the new look up texture takes the haemoglobin degradation level (d) as the v-coordinate. The haemoglobin degradation level is a combination of the oxygen level and humidity levels and is computed as follows:

$$d(x,y) = \frac{1}{2}\Big(o(x,y) + m(x,y)\Big); \tag{6.135}$$

where $m(x,y)$ is the water content at pixel (x,y). Note that in this simulation, the humidity diffusion starts after livor mortis is completed and therefore $o(x,y) = 0$ if $m(x,y) < 1$. The haemoglobin saturation $s(x,y)$ in Section 5.4 is adjusted by using the available blood capacity ration from the texture maps as shown as in Equation 5.15. However, with water loss the haemoglobin concentration in the dermis layer would increase. As such, the equation is corrected to consider moisture loss:

$$s(x,y) = min\Big(1.0, \Big(h(x,y) \cdot M\big(u(x,y), v(x,y)\big)\Big) + \frac{1}{2}(1.0 - m(x,y))\Big) \tag{6.136}$$

The method proposed to compute the haemoglobin degradation for mummification decolouration is, it should be noted, not physically or chemically correct. The new dermis colour look up texture has been constructed to recreate the desired colouration during mummification, using the colouration of the foot on

day 19 in Figure 39 as a guideline. Thinking of the haemoglobin degradation as the v-coordinate in the look up texture serves as a guidance for controlling the colouration.

Different weights are used when combining the dermis and epidermis layers for mummification. New weights for the state of total dehydration are chosen and a linear interpolation between the old and new weights is used. The interpolation is controlled by the water content. When hydrated, the Gaussian weights from the livor mortis simulation are used. The final skin colour can then be computed as follows:

$$\mathbf{c} = \omega_1 \mathbf{L}_e + (1 - \mathbf{A}_e) \circ (\omega_2 \mathbf{L}_e + \omega_3 \mathbf{L}_d) \tag{6.137}$$

where $\mathbf{L}_e$ and $\mathbf{L}_d$ are the reflections of the epidermis and dermis respectively and ω_i are the weights defined as:

$$\begin{aligned} \omega_1 &= m\mathbf{w}_1 + (1 - m)\boldsymbol{\psi}_1 \\ \omega_2 &= m\mathbf{w}_2 + (1 - m)\boldsymbol{\psi}_2 \\ \omega_3 &= m\sum_{i=3}^{6} \mathbf{w}_i + (1 - m)\boldsymbol{\psi}_3, \end{aligned} \tag{6.138}$$

where m is short for $m(x, y)$, $\mathbf{w}_i$ are the Gaussian weights as in Equation 5.18, and $\boldsymbol{\psi}_i$ are the weights for fully dehydrated skin. For the results shown in the next section $\boldsymbol{\psi}_1 = [0.070, 0.083, 0.100f]$, $\boldsymbol{\psi}_2 = [0.100, 0.100, 0.100]$, and $\boldsymbol{\psi}_3 = [0.830, 0.817, 0.800]$ where used. The values were chosen to best recreate the skin shading associated with dehydration described by Iglesias-Guitian et al. (2015). The lower values in $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$, as compared to $\mathbf{w}_1$ and $\mathbf{w}_2$ lead to less contribution of the epidermis to the skin colouration. This is done to account for the thinning of the epidermis, which makes it more translucent. Colouration changes within the dermis were simulated using the look up texture in Figure 45. The changes in dermis colouration are caused by the higher absorption of light due to water loss, modelled here by the orange-brown discolouration.

The absorption by melanin $\mathbf{A}_e$ for mummification is controlled by the melanin map similarly to the one used in Equation 5.18, with the difference that two colour channels are used to specify the absorption strength for both the fully hydrated and dehydrated stage. The absorption values $\mathbf{A}_e$ are then computed with respect to the moisture content by linearly interpolating between the hydrated and dehydrated values, as with the weights shown in Equation 6.138.

6.8 Results and Discussions

The mummification simulation consists of four components, the moisture diffusion, volume deformation, skin deformation and skin shading. Each of these will be examined individually before looking at the result of the four components combined. For this the simulation was run on the two models, a human arm and a human head. Both models consist of an internal bone layer inside a flesh volume and a skin layer, which are the same as shown in Figure 28. The skin layer on the head model is of higher resolution than the boundary of the underlying volume mesh. The geometry information for those can be seen in Table 6.4.

The results shown in this chapter were generated using a time step Δt of $\frac{1}{6}day$. Both models took around 360 simulated days (i.e. around 2200 simulation steps) to fully mummify. The simulation step timings for both models are shown in Table 6.4. A PC with an Intel Core i7 CPU running at 3.40 GHz, with 16 GB of RAM and a NVIDIA GeForce GTX 760 graphics card was used for the examples. The simulation was implemented using C++ and DirectX 11.

	head model	arm model
model size		
volume node count	35k	29k
flesh node count	33k	28k
tetrahedra count	149k	141k
surface triangle count	122k	19k
timings		
s per frame	11.093 s	3.146 s
s per simulation step dehydration	1.216 s	0.372 s
s per simulation step volume deformation	8.591 s	2.563 s
s per simulation step skin deformation	1.246 s	0.173 s

Table 6.4 This table shows the geometry information for the two models used in terms of node and polygon counts in thousands (k), plus performance statistics of each of the three mummification simulation steps and each frame (including skin rendering) in seconds (s). The volume node count is the total number of vertices of the volume mesh, whereas the flesh node count excludes the vertices of the bone mesh.

6.8.1 Humidity Diffusion

The first part of the mummification by desiccation algorithm is the moisture diffusion. Water evaporates from the object's surface, causing a difference in water content at its surface compared to its interior. The internal water then

moves towards more dehydrated areas (i.e. the surface), causing a difference in water content further inside the object. Like this, all moisture gradually moves towards the object's surface from where it will evaporate, leaving the entire object dehydrated. Unfortunately, no temporal data describing the moisture evaporation from a human body part was available for a quantitative comparison. Papageorgopoulou et al. (2015) provide information about the approximate weight loss after two months and at the end of the experiment. However, two data points are too few from which to build a meaningful water evaporation model. In the absence of useful quantitative data, visual comparison with photographs and detailed descriptions of the process are the alternative used for evaluating the moisture diffusion simulation.

Areas with a high surface to flesh ratio are expected to dry out quicker as mentioned by Aufderheide (2003). The fingers are a very thin structure and as such have the highest surface to flesh ratio and are expected to dry out quickest. This is similar to what was reported by Papageorgopoulou et al. (2015). The toes of the mummifying lower leg dried out first, whereas the thigh dries out slowest, as it has the highest flesh volume to skin surface ratio. Similar developments are expected to happen with the model of the arm.

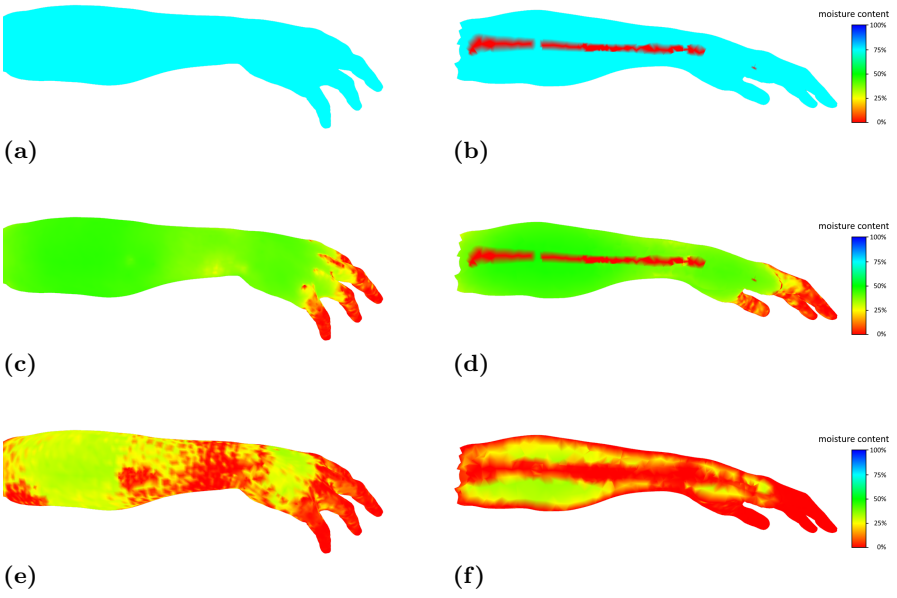
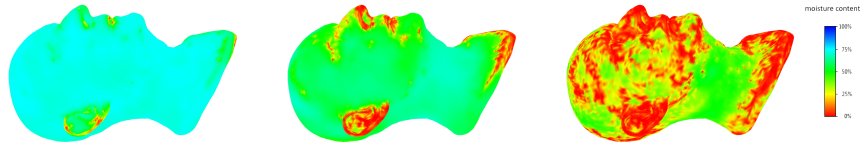


Figure 46: This figure shows the results of the moisture diffusion simulation applied to the model of an arm with an internal bone structure at three different time steps. The left column shows the outside of the object, whereas the right column shows a cross cut of the arm, revealing the internal bone.

Figure 46 shows the outside (left column) and a cross section (right column) of an arm during the humidity diffusion simulation. Blue areas are saturated and red areas are deprived of all water, as described by the the colour scale on the right. The red line within the arm in the right column is a bone. It is shown as red due to its initial moisture content being zero. As expected, the fingers dry out quickest as shown by the yellow to red colourations in Figure 46c and 46d, and are already nearly completely dehydrated. The forearm and wrist on the other hand still contain about two third of the initial water content. The dehydrated areas spread from the fingers and also become more apparent at the wrist. The forearm, on the other hand, is still mostly hydrated. Papageorgopoulou et al. (2015) observed in their experiment that the thigh on the lower leg had not fully dehydrated by the end due to its greater volume. This is in agreement with the result displayed in Figure 46e and 46f. Papageorgopoulou et al. (2015) terminated the experiment before full dehydration of the leg. Some results shown in this section reach full dehydration.

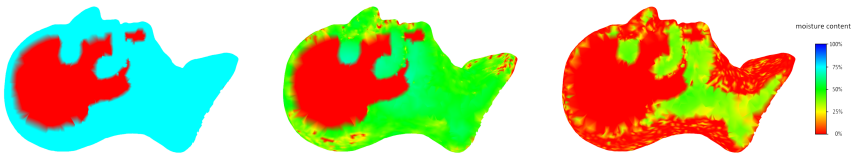


(a) **(b)** **(c)**

Figure 47: This figure shows the results of the moisture diffusion simulation applied to the model of a head with an internal skull at three different time steps. a) and b) show the moisture diffusion at early time steps and c) at an advanced stage of dehydration.

Figure 47 shows results of the moisture diffusion applied to a model of a human head at different time steps. Note that for this simulation the head model has an internal bone structure resembling a skull, as depicted in Figure 28. Figure 47 shows an external view. The first two images show early stages, with the head still mostly hydrated. Similarly to the hand model, one can observe that thin areas and areas with a high surface to underlying volume ratio are already more dehydrated than other areas. This can be seen in the nose, ear and lips, as expected. Note that the throat is still very much hydrated. This is due to the model missing the bone structure and hollow interior of a human throat, resulting in a greater flesh volume than it does on a real neck.

The internal moisture diffusion of the same head model is shown in Figure 48. Figure 48a shows the initial state with 75% moisture content for the flesh and 0% for the bone (shown in red). In Figure 48c the scalp shows advanced dehydration. This is due to the internal bone structure. The skull around that area



(a) **(b)** **(c)**

Figure 48: The images show a cross cut of the head, revealing the internal skull. a) shows the initial moisture diffusion, where the red areas are part of the skull. b) and c) show later steps in the dehydration process.

lies very close to the surface, and as such, there is very little flesh underneath the surface. This results in the area drying out quickly. The skull geometry greatly influences the moisture distribution. The concave structure in the eye socket and the hollow area in between the jaw bones dry out slower than the rest of the head. The moisture flow is restricted by solid bone in most directions, which means that less moisture leaves these concave areas. This can be observed in Figure 48b and 48c. In Figure 48c the eye socket and jaw area still hold about two third of their initial moisture content, whereas most of the head is dehydrated. As above, the neck shows very slow dehydration.

6.8.2 Volume Deformation

Mummification by dessication is often accompanied by a significant volume loss. The change in the moisture content causes internal body strains that lead to the deformation of the flesh volume. Aufderheide (2003) and Papageorgopoulou et al. (2015) point out the difference in mummification speed over a body. Papageorgopoulou et al. (2015) report that the toes and feet shrunk faster than the rest of the leg. Aufderheide (2003) mentions that areas of a large skin surface to underlying soft tissue, such as fingers, toes and ears often show greater preservation of soft tissue. He attributes that to the faster dessication rate of those areas. Following the moisture diffusion update for each time step, the volume deformation solver is applied to the volume mesh. This section will show some generated results on the models of a human arm and head with internal bone structures. As no statistical data on volume loss from water evaporation for human body parts could be found, the simulation results are compared visually to photographs and descriptions of the process.

Figure 49 shows the deformation of an arm with water loss. The colouration represents the moisture content distribution at the time the images were taken, as shown in the colour scale on the right. The black shape around the arm in

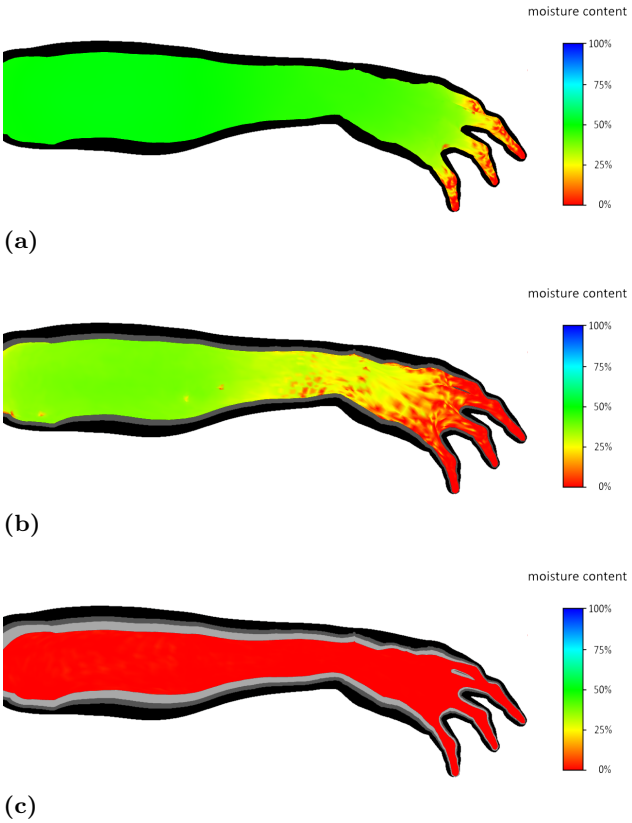
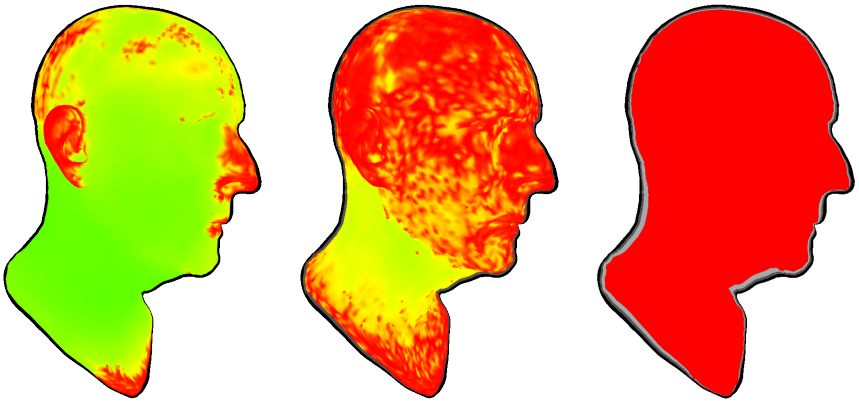


Figure 49: The three images show the deformation caused by moisture diffusion on the arm at three different stages. The black shape around the arm is its initial shape when fully hydrated. The grey shapes in figure b) and c) are the shapes of the previous steps. These allow for a direct comparison of volume loss between the stages.

Figure 49a is the outline of the hydrated arm. The grey shapes in the other two images are the outline of the previous figures. These shapes visualise the volume loss relative to the previous time steps. The hand and fingers show a greater black area than grey. This shows that most volume in the fingers and hand was lost early in the dehydration process. In contrast to this, the volume loss on the forearm is more gradual. The results are comparable to the experiment on mummification by Papageorgopoulou et al. (2015) whose results are shown in Figure 39. Similarly to the simulation results, Papageorgopoulou et al. (2015) report the toes and foot reaching dehydration faster than the rest of the leg, as can also be observed in the photographs they supply (see Figure 39).

Towards the end of the Papageorgopoulou et al. (2015) experiment, the soft tissue on the foot, toes and lower leg had shrunken so much that the skin was clinging to the underlying bone structure. This can be observed in D160 and



(a) (b) (c)

Figure 50: The three images show the deformation caused by moisture diffusion on the head at three different stages. The black shape around the head is its initial shape when fully hydrated. The grey shapes in figure b) and c) are the shapes of the previous steps. These allow for a direct comparison of volume loss between the stages.

D208 in Figure 39. They also report that the thigh has not fully dehydrated and shows less volume loss then the lower leg. This is similar to what can be observed in Figure 51, which shows the simulation result of the arm from different angles. Note that no skin shading or skin wrinkling has been applied at this stage. Figure 51a shows the initial arm with 75% moisture content and Figures 51b- 51d show the fully dehydrated arm. The arm has shrunken significantly and the bone can be seen underneath the skin (see Figure 51b).

The moisture diffusion and deformation were also simulated on the model of a human head with an internal skull. Figure 50 shows three timelapse images of the moisture diffusion simulation and the resulting deformation. The colouration represents the moisture content on the head's surface, similar as in Figure 48. As expected, the ears and nose dry out quickest and start to shrink early on, whereas the neck is still very much unscathed. The scalp does not deform much. This is due to the skull bones' close proximity to the skin surface. The neck shows the greatest volume loss, due to the lack of bone structure in the neck. Most of the neck's deformation occurs later on. This is shown by the light grey shape, which has a greater area around the neck than the black shape. The nose on the other hand has a larger black than grey area, which implies that most volume was lost early in the dehydration processes.

Figure 52a and 52b show photographs of the heads of desiccated mummies. Figure 52c and 52d show two simulation results of a mummified head from different angles for comparison. The hollowed eyes sockets and the sunken

cheeks can be observed on both mummies in the photograph. The nose has shrunk significantly with the nostrils collapsed and only a thin bridge of the nose remaining. The lips have shrivelled and retreated from the mouth. On the whole the features of the mummies are nonetheless well preserved due to the small amount of soft tissue between the skull and the outer skin layer. The results depicted in Figure 52 show some similar developments. The cheek is collapsed with the cheek bones showing more dominantly. The eyes have retreated deeper into the eye sockets. The eye details, such as the eyelashes and eye-line, are still visible in the simulation results. This is due to the texture map of the epidermis and the normal map, which are static throughout the simulation. Furthermore, eyes have a high water content (Aufderheide (2003)) which generally results in a greater volume collapse. For this simulation the volume underneath the eyelid, which would be the eye ball in a real head, was treated as simple flesh and therefore received the same amount of initial water content as the rest of flesh parts of the head. This accounts for the reduced collapse in the eye sockets. The ears of the simulation model have thinned similarly to the photographs, but do not show the same collapse as observed in the real mummies. The thin ears are still represented by a volume mesh, rather than as a thin shell object, which makes thin shell deformation, such as buckling and folding, more difficult to simulate. The lips have lost volume, but are not pulled back as in photographs. The simulation described in this chapter does not support splitting of the mesh which would be required in order to simulate this appearance. Overall, the head has shrunk and the skin clings firmly to the bones. Most facial features have been preserved.

6.8.3 Skin Deformation

The simplest way to generate the skin deformation caused by the volume mesh is to pull its vertices along with the volume mesh. This works well in cases where the bone is in close proximity to the skin, e.g. on the head. The results in Figure 52 were generated using this method. The result is a smooth skin appearance that looks stretched over the skull, with some unnaturally sharp edges, as can be seen on the nose. Figure 51 shows a similar result. The soft tissue of the hand has compressed to only a thin skin layer which is suspended across the bones of the hand. This is especially visible on the back of the hand in the simulation results in Figure 51b.

Two photographs of a desiccated mummy can be seen in Figure 53. Compared to the simulation results in Figure 51 and 52 the skin of the real mummy shows noticeable wrinkling and folding over the body and on the arms. In order to

achieve more realistic skin dynamics, particularly on the arm model, the internal skin dynamics were simulated using position based dynamics. First the tracking points for each skin vertex were used to compute the tracking forces. Tracking forces were then applied to the skin. The skin dynamics are applied to correct the skin appearance using stretching and bending constraints. This results in a more wrinkled look. An example of this is shown in Figure 55. Additional tracking constraints are used to avoid and control the separation of the skin and volume mesh.

The results of the wrinkling methods can be seen in Figure 54. Figure 54a shows the undeformed head model and Figures 54b - 54d show the mummified head. Strong wrinkles are noticeable on the neck of the model, where most of the deformation has taken place. This is due to the lack of underlying bone structure in the neck compared to the head. The skin around the mouth and on the nose also shows some wrinkling deformations. The other areas of the head model show a more pitted skin deformation, as the loss in the underlying volume is not great enough to produced strong wrinkles. This can also be observed in the skin appearance of the forehead and the wrinkles on the neck in the photograph from Figure 37b. The mummy heads in the photographs from Figure 52 on the other hand show a smooth skin appearance over the skull, but still strong wrinkles on the neck. The appearance of mummified skin depends upon a number of physical and environmental factors and can vary greatly between mummies.

The wrinkle results on the arm model are shown in Figure 55. The figure shows the simulation results with skin wrinkling from different angles. Note how the skin folds around the lower arm/wrist area. Similar skin folding can be observed in the photograph of a mummy's foot shown in Figure 53b. The thicker area further up the arm shows more of a buckling effect. Figure 55c shows protruding bone. The fingers and thumb of the hand in Figure 56 show some very strong wrinkling effects. Figure 56a shows a zoom in on the hand from that simulation. This is compared to the fingers of the mummy shown in the photograph in Figure 56b and simulation results are comparable. The fingers in the simulation results (Figure 56a) and photograph (Figure 56b) show similar wrinkling formation. The mummy in the photograph is much older and the skin appears somewhat polished. Further dust deposits fill out some of the concavities created by the wrinkles which causes a divergence in appearance from the simulation result. However, the overall geometry of the skin in the two images is alike. The nails shown in the photograph have yellowed. This yellowing is not visible in the simulation results. The reason for this is that the nail parts of a finger are treated the same as the skin covering the rest of the

arm and, as such, are rendered as skin.

The position based dynamic approach allows for easy control of the wrinkling strength during dessication. Figure 57 shows some simulation results on the arm model with different parameters for the skin deformation solver, shown beneath each picture. This causes different wrinkling effects with different wrinkling strengths. The first image (Figure 57a) shows very little wrinkling, followed by results with increased wrinkling effects. The amount of wrinkles in Figure 57b and Figure 57c are comparable, however, the wrinkles in Figure 57c are larger. This can be observed around the stump and back of the arm. The photographs of mummies all show varying wrinkle strength and forms, from fine wrinkles, to buckling and big skin folds. As such, control over the wrinkling strength not only allows for more user control over the mummy's appearance, but also allows for more variety. The wrinkle appearance depends greatly on the triangle mesh, both in terms of where these appear and how fine they are (i.e. surface triangle count). Wrinkles form along triangles edges and are therefore dependent on the triangle layout. The size and frequency of wrinkles depend on the surface triangle count. This means that a detailed triangle mesh is required to model fine wrinkles. For very fine wrinkles a wrinkle map would be required. This would also allow for fine wrinkles to occur on the face of a triangle, as opposed to only along its edges.

6.8.4 Skin Shading

Figure 39 shows photographs of a leg mummification experiment by Papageorgopoulou et al. (2015). The leg was placed in a natron mix in order to artificially mummify it by dehydrating the skin. The natron mix might be the cause of the white crust layer on the leg, which is not observed on the natural mummified corpses in Figures 52, 53, and 56. The photographs of the natural mummified bodies show dust or sand accumulation on the mummy. The white crust on the leg photographs makes it difficult to examine the skin colouration underneath. However, some colouration changes can be observed in some areas with less white crust. On day 19, a brownish colouration can be seen on the foot and Papageorgopoulou et al. (2015) report that the skin takes on a brownish and later black colour.

Figure 58 shows time-laps images of a mummifying arm with skin rendering. The top image shows the fully hydrated arm after fixation of hypostasis (see Section 5.3.5) which leads to the blue colouration at the bottom and pale complexion at the top. In Figure 58b the arm has started to dehydrate, showing a brown colouration on the fingers that spreads to the rest of the arm in Fig-

ure 58c and finally covers the entire arm as shown in Figure 58d. The areas that are full of blood have turned a darker brown than the blood deprived areas. The arm and back of the hand in Figure 58b appear more waxy than the skin in the top image.

The skin shading results of the mummified arm are compared to the photograph of a leg in Figure 59. The Figure provides a side by side view of the leg from day 19 and the skin shading result of a partially mummified arm. The black colouration in Figure 59a are the areas that the blood pooled during livor mortis, whereas the brown areas are the blood deprived areas. The brownish colouration on the arm is similar to the colour that can be seen on the leg. Some blackened skin can also be observed in both images. However, the simulation is not able to capture the greenish colouration shown on the thigh in Figure 59b. The literature was not clear on the chemical causes of the green discolouration on the skin. A possible explanation might be haemoglobin degradation, which leads to greenish and yellow discolouration observed in healing bruises (see Bohnert et al. (2000)). The greenish discolouration has not been described by the other literature discussed in Section 6.2.

The mummification method plus skin shading was also applied to the model of the human head. The results are shown in Figure 60. Figure 60a to Figure 60c shows the changes in the skin's colouration during the mummification process, where Figure 60a shows the time just after livor mortis has fixed and the haemoglobin has oxygenated. Figure 60b shows some areas dehydrated, and as such, are appearing drier and darker. Other areas, such as the neck and cheeks, are still mostly hydrated. Figure 60c and Figure 60d show the skin at the fully dehydrated stage from two different angles. The entire head looks more leathery now.

6.9 Conclusion and Future Work

The objective of the method proposed in this chapter was to create a visually realistic reproduction of the appearance of mummification by natural dessication. Desiccation is not the exclusive cause of mummification. Instead, there are a wide variety of reasons tissue can mummify, some of which were mentioned in Section 6.2. Even mummification by dessication can occur under different circumstances (e.g. spontaneous and artificially) and under varying environmental conditions. Mummification by dessication is a complex process and mummified bodies can vary greatly in appearance. Nonetheless, some common appearance patterns can be observed on dessicated mummies. These are the volume loss

of the soft tissue, often to the degree that the bones are visibly outlined under the skin. The skin can become wrinkly or appear folded. Although the degree and form of the skin wrinkling varies between mummies and areas on the same mummy.

The method proposed in this chapter introduces further complexity when compared to the methods used in Chapter 4, in which any interior or physical properties defining the surface's composition were ignored, and in Chapter 5, in which no deformation of the object takes place. This is due to the consideration of volume changes caused by moisture diffusion in the interior of the model. This represents an increase in complexity compared to the problems of surface weathering and livor mortis and therefore requires a different methodology than those applied in Chapters 4 and 5.

The great variety in the appearance of dessicated mummies makes visual evaluation difficult. In this case the focus is on reproducing some of the most pronounced visual mummification characteristics. These include: the shrinking of the soft tissues such that the underlying bone structure can be discerned, the wrinkling and folding effects of the skin and the brownish to black colouration changes of the soft tissue. The bones are generally not affected by mummification.

In this chapter a method to recreate the visual appearance changes of desiccating mummies was proposed, which is able to believably recreate the soft tissue shrinkage, skin wrinkling and skin colouration effects observed in dessicated mummies. The volume representation of the internal soft tissue and bone structure allowed for the simulation of humidity diffusion within the body and the resulting contraction of the dehydrated soft tissue. There are other factors that influence the dehydration and shrinking of the tissue, such as environmental temperature, air circulation around the body, and whether some skin is covered by cloth. These have not been considered in this approach. Introducing these would create more realistic and varying mummies and would be an interesting extension to the proposed mummification method. The high resolution triangle mesh representing the object's skin layer made it possible to create detailed skin wrinkling formations. The use of position based dynamics allowed for fast and controllable skin wrinkling driven by tracking points on the volume mesh. Although this method allows the manipulation of the strength and appearance of the wrinkles, it causes strong wave motions when watched in motion. This could be reduced and in some cases eliminated by damping the velocity term, but this causes a reduction in the strength of the wrinkling as well. A possible approach to avoid this is by manually painting the maximum radius each skin node is allowed to separate from the volume mesh directly onto the surface mesh. The

wrinkles are then induced with attachment constraints (see Müller and Chentanez (2010)), similar to the tracking constraints proposed in this method. This would also allow for more artistic control.

For the skin shading, per vertex moisture content information was given to the shader. The moisture content information was used to look up the dermis skin colouration and to interpolate between saturated and dehydrated RGB weights when combining the two skin layers. They attempt to simulate the changes in the light's reflectance and absorption properties in the skin due to water loss. The skin shader applied is an ad hoc method and does not represent the chemical actions that cause the skin colouration changes in mummies. A more chemically accurate method would be desirable to create more realistic skin shading results, such as the method applied by Iglesias-Guitian et al. (2015) for skin ageing. The lack of information on chemical changes in mummified skin made this difficult. There are however some aspects of optical changes in skin due to dehydration that could be modelled without fully understanding the chemical changes in skin due to mummification. An example would be the effects of light reflectance properties caused by water loss as was done in Iglesias-Guitian et al. (2015) and more accurate changes of light transmittance caused by the thinning and separation of the skin layers. Implementing a more accurate model to describe the light reflection properties of mummifying skin would nonetheless be an interesting subject for future research.

The contraction of the soft tissue and drying of the skin can lead to ripping of the skin, which would reveal the underlying soft tissue and bones. In many cases mummies are not intact due to damage to their soft tissue, either before or after mummification. This can be observed in Figure 61. These reveal some soft tissue and bones underneath the epidermis layers. Skin ripping was not considered in the mummification approach described in this chapter. Simulating skin ripping would involve detecting where the skin's deformation stress exceeds a certain ripping threshold, identifying the split path and potentially re-meshing around the split for better visual quality. A similar approach has been described by Müller and Gross (2004) to simulate the fracturing of meshes. Fracturing of multi-layered thin shell object with adaptive re-meshing was described by Busaryev et al. (2013); Pfaff et al. (2014). All three approaches use the finite element method to solve the elastoplastic equation driving the deformation. The approach described in this chapter used position based dynamics to describe skin deformation, as this is more efficient. Introducing skin ripping would improve the realism of the mummification simulation and would be an intriguing idea for future research. This could be done with ideas from Busaryev et al. (2013) and Pfaff et al. (2014), which would involve replacing the position

based dynamics with a finite element approach. Another consideration when implementing skin ripping is the modelling and rendering of the underlying soft tissue and bones. In Figure 61, the underlying soft tissue becomes visible and as such need to be modelled in greater detail, which can be difficult considering its complex fibrous geometry.

(a)

(b)

(c)

(d)

Figure 51: These images show the deformation caused by moisture diffusion on the arm model shown in a). b-d) show the fully dehydrated arm from different viewing angles.

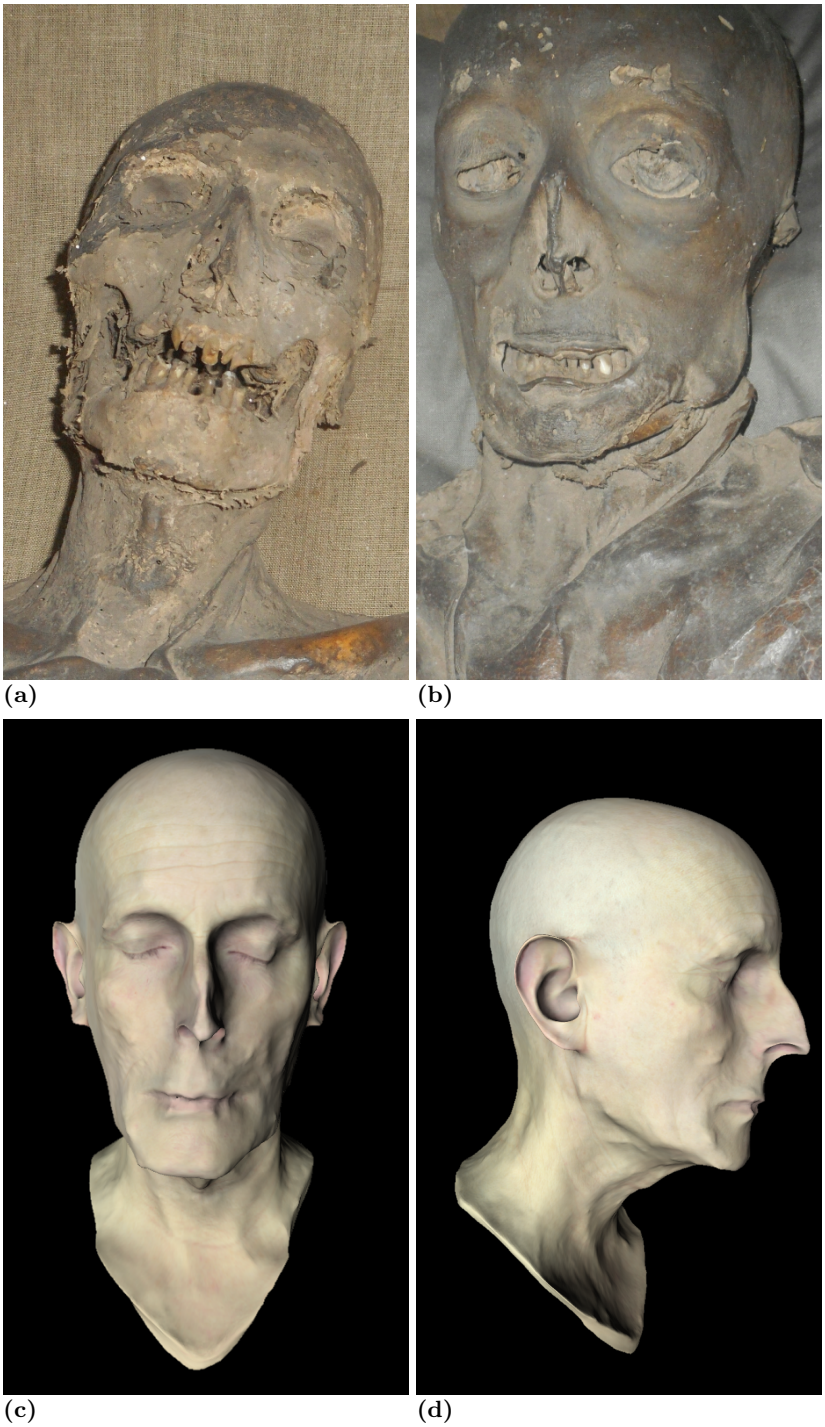
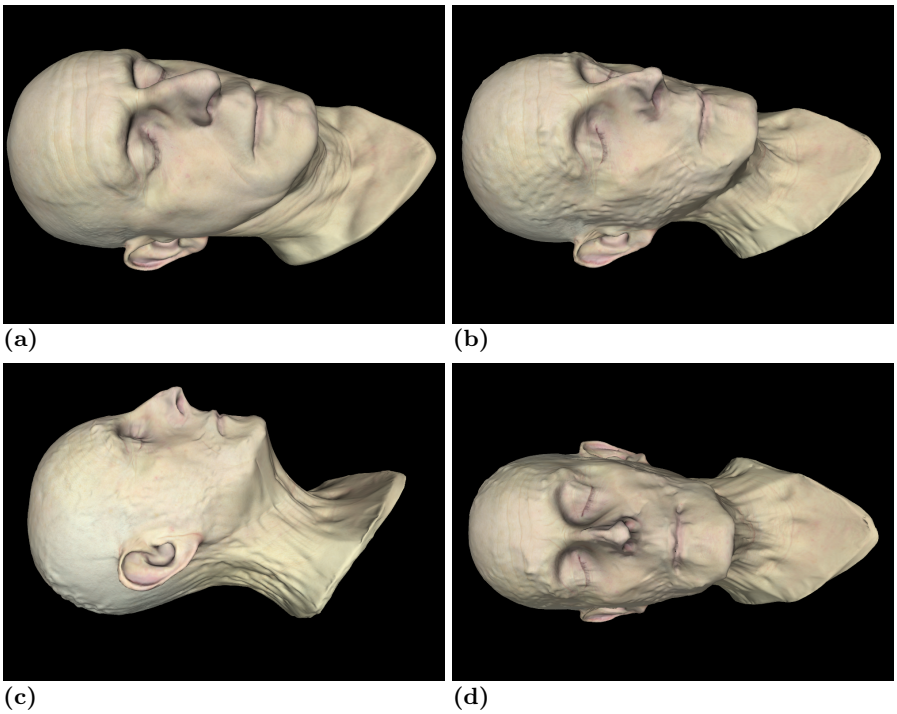


Figure 52: The photographs in the top row show the heads of naturally desiccated mummies taken by the author in the Ostkrypta in the St. Petri Cathedral in Bremen, Germany, with a Nikon COOLPIX S3700 and NIKKOR lens. The bottom row shows the simulation results of the mummification technique on a human head model.

(a) **(b)**

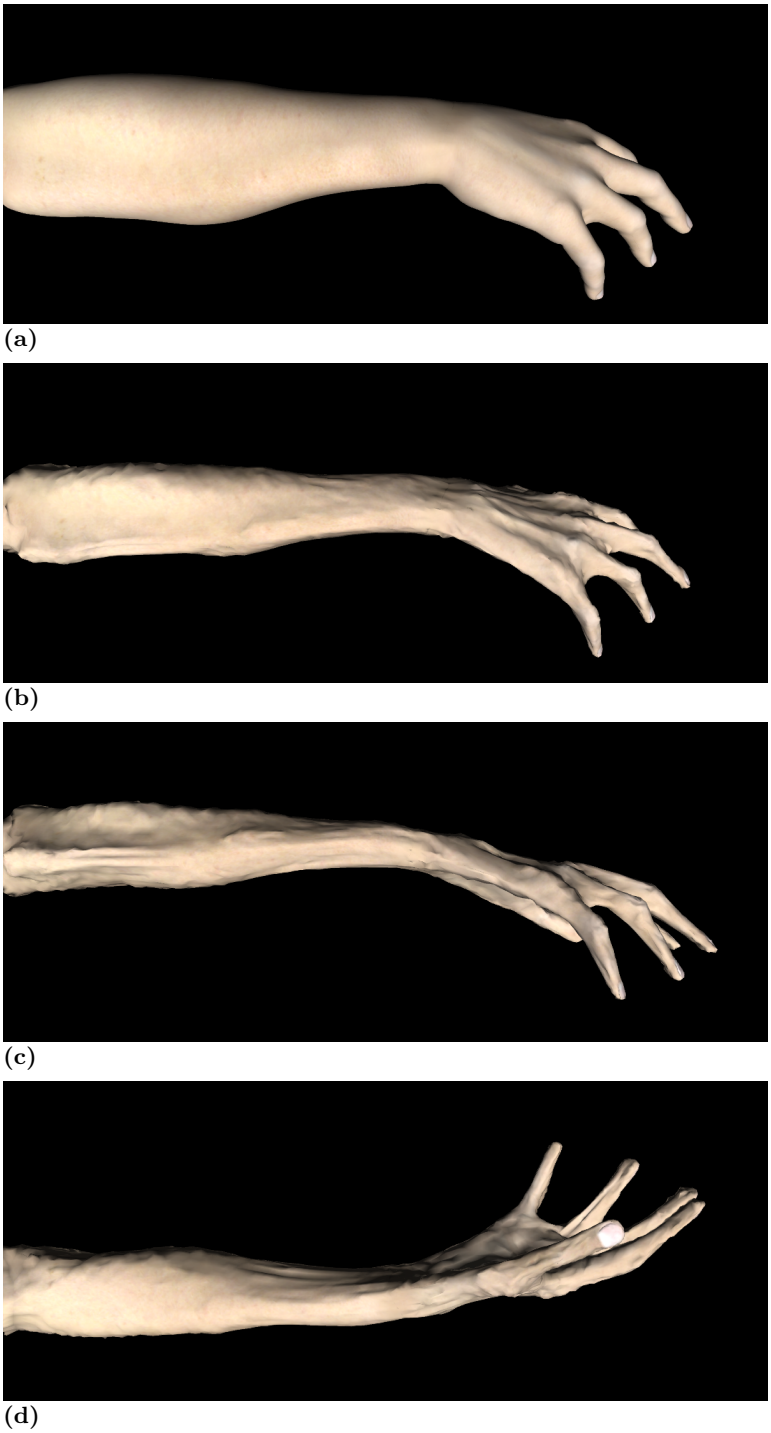
Figure 53: Photographs of a mummy and a mummified foot with skin wrinkles. The photographs were taken by the author in the Ostkrypta in the St. Petri Cathedral in Bremen, Germany, with a Nikon COOLPIX S3700 and NIKKOR lens.



(a) **(b)**

(c) **(d)**

Figure 54: This figure shows mummification results with skin wrinkling. Figure a) shows the hydrated head and Figures b)-d) show the fully dehydrated head from different angles.



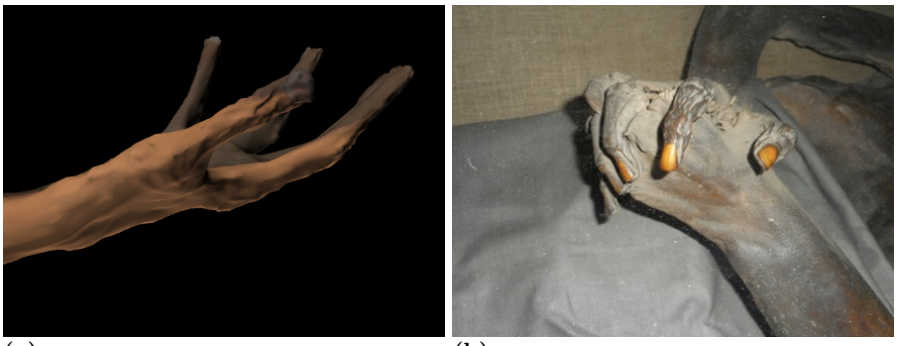
(a)

(b)

(c)

(d)

Figure 55: Skin deformation result on the model of an arm from different angles.



(a) **(b)**

Figure 56: This figure compares the simulation results to a photograph of a mummified arm which was taken by the author in the Ostkrypta in the St. Petri Cathedral in Bremen, Germany, with a Nikon COOLPIX S3700 and NIKKOR lens.

(a) $r_{max} = 0.001$, $k_{damping} = 0.5$

(b) $r_{max} = 0.002$, $k_{damping} = 0.2$

(c) $r_{max} = 0.004$, $k_{damping} = 0.08$

Figure 57: This figure shows mummification results with different wrinkles strengths controlled by the damping coefficient and tracking constraint radius.

.

(a)

(b)

(c)

(d)

Figure 58: This shows the skin shading results at different mummification stages. Image a) shows the fully hydrated stages just after fixation of hypostasis. Image b) shows the partial dehydration and c) shows full dehydration.

(a)

(b)

Figure 59: a) shows the skin shading results due to mummification. b) shows a photo of a mummifying leg taken from Papageorgopoulou et al. (2015).

Figure 60: This figure shows the skin shading results at different mummification stages. Image a) shows the skin shading just before dehydration starts. Image b) shows the partial dehydration and c-d) show full dehydration from different angles.

Figure 61: This figure shows a natural mummified body with a post-mortem injury on its arm, revealing the soft tissue under the epidermis. The Photograph was taken by the author in the Ostkrypta in the St. Petri Cathedral in Bremen, Germany with a Nikon COOLPIX S3700 and NIKKOR lens.

Chapter 7

Conclusion

There is a trend in video games towards greater realism in term of both graphics and behaviour of the game environment. In a believable environment corpses are expected to decay over time, however, this element has, so far at least, been neglected. The goal of this research was to develop methods to recreate post-mortem appearance changes due to organic decomposition that are able to produce results comparable to their real world equivalents.

A study of literature in the area of object deformation and decomposition has shown an increasing trend towards computer generation of object weathering, ageing and decomposition effects. Considerable work has been done in creating ageing and weathering effects that mainly affect the surface appearance, such as cracks,by using a particles to carry fractures, corrosion, rusting, patinas, and staining. Other degenerating processes that affect the entire geometry of the object have also been attempted, but these consist mainly of approaches which remove material from the object's surface, as is the case with erosion, burning and melting. Work on simulating organic decomposition, on the other hand, has been limited.

Organic decomposition, and in particular human body decomposition, is a complex process that comprises a number of different biological, chemical and physical activities which are still not fully understood. This vast number of processes and their complexities makes a comprehensive model difficult to approximate. The focus of this work is on recreating the visual appearance changes of three particular processes:

1. surface weathering effects caused by environmental activities
2. livor mortis

3. natural mummification by dessication

Other activities, such as putrefaction and autolysis, have not been attempted in this thesis.

The surface weathering effects are achieved by using particles to carry materials through the scene, which are deposited on intersected objects. The main contribution of this work is the depositing square approach that is used to record material deposits. A square (the depositing square) is projected onto the object's surface to specify the area affected by a particle's material deposit. The depositing approach of previous methods could result in deposit sizes varying on different objects or even on different areas of the same object. This in turn can cause some areas to weather quicker than others, even with the same particle exposure. The approach demonstrated as part of this doctoral thesis is able to successfully reduce these effects of inconsistent texel-to-world-space sizes on the visual appearance of material deposits. Furthermore, potential deposit discontinuities on texture seams are avoided by subdividing the depositing square polygons, each of which is confined to a single triangle face. Splatting textures were introduced that allow the user to control the deposits shape and to avoid unnatural polygonal shaped deposits. The method was demonstrated by using examples of rusting and moss growth. The generated examples showed that with this method inconsistent depositing sizes can be avoided without introducing deposit discontinuities at texture seams. The surface weathering effects generated as example were rusting and moss growth. Rusting was chosen instead of dirt or dust deposits because the ageing rules are more complex. This made it a more appropriate choice to show the potential of the approach. Integrating dust and dirt deposits into the current surface weathering framework is straightforward as shown by Chen et al. (2005) and Günther et al. (2012).

Although the method is able to reduce the visual and methodological problems inherent in previous methods, it is still limited by the maximum size of a texel in world space. The method is able to create a wide variety of visual weathering effects affecting the object's surface. It is not able to effect changes to its geometry. Furthermore, it is only able to model changes inflicted from the outside, yet the major influences of human body decomposition are processes internal to the body. A different approach is therefore required to simulate livor mortis and mummification. For these processes a model that is able to represent the internal volume and differentiate between the different components of the body (i.e. flesh and bones) is required.

For this project the human body was represented by a layered model consisting of a tetrahedral mesh representing the flesh and bone components and a triangle

surface mesh representing the skin. The volume representation is required for the simulation of the decomposition processes, as they are acting on the body's internal structures. The tetrahedral mesh is able to represent the volumetric structure and allows for the differentiation between flesh and bone. A triangle mesh was chosen to represent the skin for simulation and rendering reasons. Human skin is a relatively thin structure and tetrahedra are unsuitable for modelling very thin structures. A triangle mesh allows for cloth-like deformations to be used to generate the skin wrinkling effects. Tracking points connect each vertex of the skin mesh to the underlying tetrahedral boundary, which allowed the use of a high resolution skin mesh with a lower resolution tetrahedral boundary mesh. This meant that the skin could be rendered at high resolution while the computations could be performed on a tetrahedral mesh created from a lower resolution triangle mesh. A lower resolution leads to larger and fewer tetrahedra and, therefore, a faster simulation of the blood dynamics.

Livor mortis is caused by post mortem blood dynamics. Simulating the rheology of blood is complex and, as such, the modelling of all states was not attempted. The blood dynamics that were modelled are the sinking of red blood cells in the blood plasma due to gravity and the tissue staining caused by vessel wall break down. For this model blood was represented by red blood cells (or haemoglobin) in liquid. The dynamics of the blood were simulated by transferring haemoglobin between tetrahedral vertices along connecting edges. The transport is controlled by gravity and surface pressure. The haemoglobin can be fixed (an approximation of clotting or staining) at a certain time in the simulation. As such, two stages (liquid and clotting) of blood were approximated in this approach to simulating livor mortis. No viscoelastic behaviour is modelled, such as might be seen in a two-phase suspension such as blood, e.g. red blood cell aggregates and consequences of these. However, this does not prevent the model from producing a realistic appearance of livor mortis, which is its goal.

The method presented is able to simulate the pooling of haemoglobin in the lower lying areas as well as pressure induced blanching effects and fixation of hypostasis. Here, the tetrahedral mesh has shown to be a good choice of representation on which to simulate the post mortem blood dynamics. Its irregular layout was able to produce the patchy appearance of early livor mortis. An edge network resulting from a regular mesh, such as a voxel mesh, is not easily able to create the same result.

Currently, the haemoglobin content is given on a per-vertex basis and is interpolated over the triangle faces. This can lead to unnatural polygonal edges around the blanching. It also makes it unsuitable to model thin and detailed blanching effects, such as can be caused by cloth folds, jewellery or belts. A superior place

for recording the area affected by pressure is possibly the object's dermis map. Recording the pressure points on a per-pixel basis can avoid sharp polygonal edges and allow the modelling of very thin and detailed blanching effects. This is a potential area of further research.

In addition to the blood dynamics, the oxygen decrease in the blood was modelled, which drives the colouration changes in the blood. This was done by approximating the oxyhaemoglobin dissociation curve. The results show that the blood colour turns a deeper red with ongoing livor mortis with haemoglobin poor areas releasing oxygen faster.

For the visualisation a layered skin shader was proposed. The skin shader is controlled by haemoglobin concentration and oxygen saturation levels at each point on the skin mesh. This allowed to simulate the colouration changes on the skin referred to as livor mortis or hypostasis. This approach is able to successfully reproduce the colouration changes of the skin caused by the blood dynamics after death. These include the early patchy appearance, the pinkish discolouration where the blood pools, the pressure induced blanching effects, the fixation of the colourations due to tissue staining, and the gradual discolouration of blood filled area to purple and blue due to oxygen loss.

The livor mortis simulation is sensitive to the accuracy of the inner bone structures. Holes and disconnected bone fragments can divert the path of the haemoglobin. This results in incorrect pooling effects. Instead of blood accumulating on top of the bones, it will sink through holes and gaps. The opposite can happen if the bone is set too close to the skin, causing a bottle neck effect. This leads to pink colourations in areas that should be deprived of blood. A more precise bone mesh is therefore advisable to avoid these problems. However, more detailed bone structures could result in smaller tetrahedra in the interior of the model and therefore slower computation.

The same object representation was used to simulate the mummification by dessication. The tetrahedral mesh representing the object's volume was used as a finite element discretisation. This allowed the finite element method employed on the volume to simulate moisture diffusion and the resulting deformation of the internal volume substrate. The FEM was used to simulate the evaporation of moisture from the object's surface and the concurrent dessication of the body's flesh parts. The resulting humidity diffusion patterns correspond with those observed in mummification literature, i.e. body parts with high surface-to-flesh ratio (e.g. fingers, ears), dried out quicker than areas with large underlying flesh parts (upper arm, thigh etc.).

The changes in moisture content were used to compute shrinking strain, which

drives the volume deformation. This caused the flesh volume to shrink significantly, while the bones retained their shape. The resulting geometry shows the similar characteristics as real life dessicated mummies. The bone structure is visible under the skin due to the great loss of flesh volume. Thin areas, such as fingers and ears shrink first as was observed with the toes in the mummification experiment by Papageorgopoulou et al. (2015). As with the livor mortis simulation, the accuracy of the bone structure affects the visual appearance of the mummified object. As the volume shrinks down to the bone and the skin is pulled along, the outlines of the bones can be discerned, and so can any of the imperfections in their modelling.

The skin wrinkling effects were modelled on the triangle mesh using position based dynamics to model cloth like deformations. Wrinkles were formed by applying pulling forces to the nodes. These allowed for direct control over the vertex positions and better control over the wrinkling dynamics. Although this creates some realistic wrinkles, it causes undesirable wave motions of the skin when the mummification is watched in progression. The wave effects were reduced using damping, but at some cost in wrinkle strength. An alternative approach exits where the maximum wrinkle radii are painted on the surface. The radii control the wrinkle strength an appearance. This approach would also allow for greater artistic control. On the other hand it would require greater manual work in preparation for each model.

For the skin shading an ad hoc approach was proposed, which is an extension of the livor mortis skin shader. A look up texture was used to determine the dermis colouration. In contrast to the livor mortis shader an additional parameter, the moisture content, is used to determine the dermis colouration. The skin shader allowed me to recreate similar colouring to those that could be observed on the photographs of mummies provided. The skin shader has some restrictions however. It was not able to model the greenish discolouration that were observed in a mummifying leg or the polished appearance of older mummified skin. Furthermore, the epidermis layer is not as translucent as it appears in some of the photographs. The appearance of mummified skin varies greatly, even among the photographs that were displayed as examples in this document. Furthermore, there is a lack of literature on the chemical processes behind the skin colouration changes that could be used to create a more chemically correct skin shading model for mummification.

The main focus of this work has been on simulating human body decomposition, which, to the best of my knowledge, has not been attempted before. This did pose some problems in terms of the validation of the proposed methods, as no prior work was available for a direct comparison. For this reason, a

visual comparison of the computer generated results with photographs of the equivalent processes was used.

The proposed approach is able to model changes to both the geometry and surface appearance of the body, in a way that resembles the appearance of the natural post mortem processes livor mortis and natural mummification by desiccation. A review of literature revealed no prior model to simulate the post mortem appearance changes in human bodies. The results demonstrated in this doctoral thesis show a clear resemblance to a selection of the natural process of human body decomposition. They represent a valuable starting point towards creating realistic appearances of corpses for use in the entertainment industry.

Other applications for simulating human body decomposition could be in the forensic training area. This might be in the form of an interactive training tool to help the users familiarise themselves with the visual processes a body undergoes after death, and how moving the body, applying pressure or changing environment variables (e.g. moisture content in the air) effect the appearance of the corpse. Alternatively, a similar tool could be developed to test the user's ability to estimate time of death, derive information about the corpse's environment from the appearance of the corpse, or recognise whether the body has been moved based on the distribution and strength of lividity. The presence of moss can indicate a wet environment whereas mummification can indicate a dry environment. Blanching effects on the body can give clues about the clothing the corpse might have worn that were removed after fixation of hypostasis occurred. The application of such a tool is not restricted to forensics. It could also be used as a teaching tool for non-specialists that are interested in broadening their knowledge of human body decomposition. Any of the suggested applications would benefit from simulating additional decomposition processes such as rigor mortis, autolysis, putrefaction and maggot activity.

7.1 Limitations

Due to lack of previous work on human body decomposition, direct comparison with other approaches was not possible. Furthermore, although there is a body of work on the biological processes behind organic decomposition, the processes involved are very complex and not fully understood. For these reasons reports on the structure and appearance changes as well as images for the simulated processes were used for comparison with the results. Validation by visual comparison is not ideal as it is susceptible to subjective bias on the part of the observer. It may also be difficult to measure consistently and accurately, both

in terms of colouration and deformation behaviour. The qualitative approach of visual comparison of the realism of the appearance of the decomposing corpse could be improved by employing an expert panel for the visual evaluation. The panel could consist of a combination of forensic experts and artists.

Visual comparison of colouration is subjective. Perceived colour differs between different observers and is also influenced by environmental factors such as lighting. A more quantitative measurement of colouration differences is desirable. The colour of materials can be represented in a coordinate system, such as the Commission International de L'Eclairage colour system (CIELAB). This allows for comparison between colours in terms of lightness, intensities, and direction of green-red and blue-yellow coordinates. Differences in colouration between real world data (e.g. photographs) and simulation results could be evaluated using principal component analysis (PCA). However, photographs are usually not true to colour. This means that the comparison of photographs with simulation results is not ideal. Colour coordinate data obtained from colourimetry measurements would serve as a superior ground truth for comparison with simulation results.

Aside from colouration changes, mummification also causes deformation of the object. As the goal of the method was realistic visual appearance, the results were evaluated in terms of the shrinkage behaviours and timings of different parts of the object. A more quantitative evaluation could involve comparing weight or volume loss of the simulation results with measurements of real corpse dehydration. Weight loss has been measured in mummification experiments, such as the one performed by Papageorgopoulou et al. (2015). This or similar data can be used for comparison. However, weight loss only indicates the amount of moisture that was lost, but not the total volume loss. Volume loss of the mummified corpse has not been reported in any of the literature that has been examined. As in the case of weight loss, there is quantitative data available that describes water volume loss in the object, but this does not correspond precisely to the total volume loss as the air volume inside the object can change.

The project represented in this document did not aim to model all the processes that influence post mortem appearance of human bodies. Considering all possible combinations and their effects on the body's appearance is too complex for the purpose of this doctoral thesis. Furthermore, the livor mortis and mummification approaches are not able to accurately simulate human body decomposition. Instead, the focus was on recreating the visual appearance of corpses under certain conditions for their use in the entertainment industry. The physical, biological and chemical models used to drive these appearance changes are strongly simplified and, as such, the resulting model can not be

used for fully accurate simulations of livor mortis and mummification. This makes it unsuitable for any forensic or biological studies beyond the use of a training tool.

The target industry for this technique is clearly the games industry. However, the physical processes driving the decomposition simulation are too expensive for real-time applications. The long term goal is therefore to create content creation tools to reduce the work load of artists, with the help of which they will be able to produce believable corpses.

7.2 Future Work

7.2.1 Extending Current Models

The livor mortis simulation is able to simulate the blanching effects caused by pressure to the skin surface. However, pressure is applied on a per vertex basis, and, as such, the method is not able to model fine scale pressure effects that are often caused by belts, jewellery and other clothing. Extending the pressure induced blanching effects to work on a per-surface point basis, for example by recording them in a texture map would improve the realism of the livor mortis appearance. There are other skin conditions that sometimes appear during livor mortis aside from the one covered in the proposed methods. These include skin marbling and Tardieu Spots (see Prahlow and Byard (2011)). Skin marbling can occur during later stages of livor mortis due to haemoglobin breakdown. Tardieu spots are caused by burst capillaries. Simulating these additional livor mortis appearances would allow for a greater variety of post mortem appearances.

The proposed mummification simulation is driven by moisture evaporation due to differences in humidity between the air and the object's surface, which is a very simplified form of the real process. In the physical world, temperature, air circulation, and whether the corpse is covered by cloth also influence the speed and appearance of dessicated mummies. Introducing the effects of temperature and air circulation would be an interesting next step in extending the mummification simulation. Another compelling subject for future work is introducing skin ripping. As previously pointed out, this might involve solving the skin deformation using the finite element method instead of the position based dynamics currently employed. Furthermore, the internal soft tissue and bones would need to be modelled in more detail as they will be visible underneath the ripped skin. The dried soft tissue has a complex fibrous structure that might be both difficult and expensive to model. The bone surface could be rendered using

texture based approaches similar as the ones used for the skin. The internal soft tissue on the other hand is more difficult to render.

Dessication is probably one of the best known causes of mummification. There are other causes however. Mummies can appear in very cold and dry climate and in bogs. Metals have also been reported to halt bacterial actions and thereby aid mummification (see Aufderheide (2003) for a more complete list of mummification mechanics). Simulating some of these other mummification processes as well as artificial mummification is worth considering for future work.

7.2.2 Simulating Putrefaction and Autolysis

The various effects of human body decomposition dealt with in this doctoral thesis are not exhaustive. There are further decomposition processes that greatly affect the appearance of corpses. The most influential ones are putrefaction and autolysis. These lead to the liquefaction of the soft tissue resulting in strong deformation of the corpse. Putrefaction starts at the lower right abdominal quadrant and spreads from there to areas that are easily affected by putrefaction. This produces gases that lead to the bloating of the body. Ongoing putrefaction causes soft tissue to break down further such that the gases are released. This results in the collapse of the body, as observed in Prahlow and Byard (2011). Putrefaction and autolysis have a great impact on the morphology and appearance of the corpse. Simulating the effects would involve both expansions and concurrent deflation of the body, which involves simulating the occurrence of gases and their bloating effects on the body.

The liquefied soft tissue escapes through body openings such as nostrils, mouth and ripped skin. Simulating putrefaction could involve modelling the liquid flow. The liquefaction process resemble the phase changes, such as melting, that were described in the literature review in Section 2.7. The methods used to simulate phase changes could be used as a starting point to simulate the liquefaction process caused by putrefaction and autolysis. Alternatively a FEM similar to the mummification one could be used to simulate the bloating due to gas creation and the subsequent collapse of the skin due to gas escaping. Whatever methods are chosen, they need to be able to handle skin decomposition and ripping as well as the greenish to black discolouration of the skin. Other appearances connected to putrefaction are skin slippage and extensive skin blisters.

7.2.3 Real-time

The livor mortis and mummification approaches are too computationally expensive to be usable in real time applications and, as such, can currently not be used at runtime within the game environment. The immediate goal was to create a content creation tool for artists. In order to use the resulting decomposition objects in game, a blend shape needs to be created from the surface mesh of the final stage at decomposition. The decomposition stages can then be approximated in-game by blending between the two meshes. For more accurate results, more blend shapes from the intermediate steps can be created. The skin shading could be reproduced at run-time, using the moisture, haemoglobin and oxygen saturation information from the mesh surface. These would also need to be interpolated between the blend shapes.

For future work, it would be practical to find ways of encoding the different decompositions stages (i.e. in form of blend shapes/morph targets), such that they can be cheaply reproduced at run-time. As skin-ripping is not simulated in the current approach, the surface triangle mesh is therefore sufficient for constructing an encoding for the decomposition stages. The tetrahedral mesh is not required for the encoding. For the skin shading, the haemoglobin, oxygen and moisture content information can either be encoded with the triangle mesh, or the resulting textures can be encoded separately. For this a technique based on Space-Time Appearance Factorization (STAF) by Gu et al. (2006), might be appropriate in encoding the skin appearance changes over time.

Bibliography

Amanatides, J. and Woo, A. (1987). A fast voxel traversal algorithm for ray tracing. In *In Eurographics '87*, pages 3–10.

Amarasinghe, D. and Parberry, I. (2011). Towards fast, believable real-time rendering of burning objects in video games. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, FDG '11, pages 256–258, New York, NY, USA. ACM.

Amarasinghe, D. and Parberry, I. (2013a). Real-time rendering of burning solid objects in video games. In *Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational Serious Games (CGAMES), 2013 18th International Conference on*, pages 139–143.

Amarasinghe, D. and Parberry, I. (2013b). Real-time rendering of melting objects in video games. In *Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational Serious Games (CGAMES), 2013 18th International Conference on*, pages 154–158.

Antiga, L. and Steinman, D. (2017). The vascular modeling toolkit. `http://www.vmtk.org/`.

Antoniades, T. (2016). Creating a live real-time performance-captured digital human. In *ACM SIGGRAPH 2016 Real-Time Live!*, SIGGRAPH '16, pages 36:21–36:21, New York, NY, USA. ACM.

Aoki, K., Dong, N. H., Kaneko, T., and Kuriyama, S. (2004). Physically based simulation of cracks on drying 3d solids. In *Proceedings of the Computer Graphics International*, CGI '04, pages 357–364, Washington, DC, USA. IEEE Computer Society.

Aufderheide, A. C. (2003). *The scientific study of mummies*. Cambridge University Press.

Bang-Jensen, J. and Gutin, G. Z. (2008). *Digraphs: theory, algorithms and applications*. Springer Science & Business Media.

Bao, Z., Hong, J.-M., Teran, J., and Fedkiw, R. (2007). Fracturing rigid materials. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):370–378.

Barbič, J. and James, D. L. (2005). Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.*, 24(3):982–990.

Bathe, K.-J. (2006). *Finite element procedures*. Klaus-Jurgen Bathe.

Beardall, M., Farley, M., Ouderkirk, D., Reimschussel, C., Smith, J., Jones, M., and Egbert, P. (2007). Goblins by spheroidal weathering. In *Proceedings of the Third Eurographics Conference on Natural Phenomena*, NPH'07, pages 7–14, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Becket, W. and Badler, N. I. (1990). Imperfection for realistic image synthesis. *The Journal of Visualization and Computer Animation*, 1(1):26–32.

Beneš, B. (2007). Real-Time Erosion Using Shallow Water Simulation. pages 43–50, Dublin, Ireland. Eurographics Association.

Beneš, B. and Forsbach, R. (2001). Layered data representation for visual simulation of terrain erosion. In *Computer Graphics, Spring Conference on, 2001.*, pages 80–86. IEEE.

Beneš, B. and Forsbach, R. (2002). Visual simulation of hydraulic erosion.

Beneš, B., Těšínský, V., Hornyš, J., and Bhatia, S. K. (2006). Hydraulic erosion. *Computer Animation and Virtual Worlds*, 17(2):99–108.

Bézin, R., Crespin, B., Skapin, X., Terraz, O., and Meseure, P. (2014). Generalized maps for erosion and sedimentation simulation. *Computers and Graphics*, (0):–. Available from here: `http://dx.doi.org/10.1016/j.cag.2014.07.001`.

Bezin, R., Peyrat, A., Crespin, B., Terraz, O., Skapin, X., and Meseure, P. (2010). Interactive hydraulic erosion using cuda. In Bolc, L., Tadeusiewicz, R., Chmielewski, L., and Wojciechowski, K., editors, *Computer Vision and Graphics*, volume 6374 of *Lecture Notes in Computer Science*, pages 225–232. Springer Berlin Heidelberg.

Bohnert, M., Baumgartner, R., and Pollak, S. (2000). Spectrophotometric evaluation of the colour of intra- and subcutaneous bruises. *International Journal of Legal Medicine*, 113(6):343–348.

Boileau, E., Nithiarasu, P., Blanco, P. J., Mller, L. O., Fossan, F. E., Hellevik, L. R., Donders, W. P., Huberts, W., Willemet, M., and Alastruey, J. (2015). A benchmark study of numerical schemes for one-dimensional arterial blood flow modelling. *International Journal for Numerical Methods in Biomedical Engineering*, 31(10):e02732–n/a. e02732 cnm.2732.

Boissieux, L., Kiss, G., Thalmann, N. M., and Kalra, P. (2000). *Computer Animation and Simulation 2000: Proceedings of the Eurographics Workshop in Interlaken, Switzerland, August 21–22, 2000*, chapter Simulation of Skin Aging and Wrinkles with Cosmetics Insight, pages 15–27. Springer Vienna, Vienna.

Boulos, P. and Altman, T. (1993). Explicit calculation of water quality parameters in pipe distribution systems. *Civil Engineering Systems*, 10(3):187–206.

Bro-Nielsen, M. and Cotin, S. (1996). Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer graphics forum*, volume 15, pages 57–66. Wiley Online Library.

Busaryev, O., Dey, T. K., and Wang, H. (2013). Adaptive fracture simulation of multi-layered thin plates. *ACM Trans. Graph.*, 32(4):52–52.

Carlson, M., Mucha, P. J., Van Horn, III, R. B., and Turk, G. (2002). Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 167–174, New York, NY, USA. ACM.

Chang, Y.-X. and Shih, Z.-C. (2003). The synthesis of rust in seawater. *The Visual Computer*, 19(1):50–66.

Chen, T. F., Baranoski, G. V. G., Kimmel, B. W., and Miranda, E. (2015). Hyperspectral modeling of skin appearance. *ACM Trans. Graph.*, 34(3):31:1–31:14.

Chen, Y., Xia, L., Wong, T.-T., Tong, X., Bao, H., Guo, B., and Shum, H.-Y. (2005). Visual simulation of weathering by gamma-ton tracing. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1127–1133, New York, NY, USA. ACM.

Cheng, S.-W., Dey, T. K., and Shewchuk, J. (2012). *Delaunay mesh generation*. CRC Press.

Chiba, N., Muraoka, K., and Fujita, K. (1998). An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation*, 9(4):185–194.

Cotin, S., Delingette, H., and Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. *IEEE transactions on Visualization and Computer Graphics*, 5(1):62–73.

Courtecuisse, H., Allard, J., Kerfriden, P., Bordas, S. P., Cotin, S., and Duriez, C. (2014). Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical image analysis*, 18(2):394–410.

D'Eon, E. and Luebke, D. (2007). Advanced techniques for realistic real-time skin rendering. in nguyen (ed.), gpu gems 3.

Desbenoit, B., Galin, E., and Akkouche, S. (2004). Simulating and modeling lichen growth. *Computer Graphics Forum*, 23(3):341–350.

Desbenoit, B., Galin, E., and Akkouche, S. (2005). Modeling cracks and fractures. *The Visual Computer*, 21(8-10):717–726.

Dettmeyer, R., Verhoff, M. A., and Schütz, H. F. (2013). *Forensic medicine: fundamentals and perspectives*. Springer Science & Business Media.

Donner, C. and Jensen, H. W. (2005). Light diffusion in multi-layered translucent materials. *ACM Trans. Graph.*, 24(3):1032–1039.

Donner, C., Weyrich, T., d'Eon, E., Ramamoorthi, R., and Rusinkiewicz, S. (2008). A layered, heterogeneous reflectance model for acquiring and rendering human skin. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 140:1–140:12, New York, NY, USA. ACM.

Dorsey, J., Edelman, A., Jensen, H. W., Legakis, J., and Pedersen, H. K. (1999). Modeling and rendering of weathered stone. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 225–234, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Dorsey, J. and Hanrahan, P. (1996). Modeling and rendering of metallic patinas. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 387–396, New York, NY, USA. ACM.

Dorsey, J., Pedersen, H. K., and Hanrahan, P. (1996). Flow and changes in appearance. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 411–420, New York, NY, USA. ACM.

Dorsey, J., Rushmeier, H., and Sillion, F. (2010). *Digital modeling of material appearance*. Morgan Kaufmann.

Eisenberg, M. A. and Malvern, L. E. (1973). On finite element integration in natural co-ordinates. *International Journal for Numerical Methods in Engineering*, 7(4):574–575.

Fan, D., Liu, S., and Wei, Y. (2013). Fruit ring rot simulation based on reaction-diffusion model. In *Proceedings of the 2013 International Conference on Virtual Reality and Visualization*, ICVRV '13, pages 199–205, Washington, DC, USA. IEEE Computer Society.

Formaggia, L., Quarteroni, A., and Veneziani, A. (2010). *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, volume 1. Springer Science & Business Media.

Frerichs, D., Vidler, A., and Gatzidis, C. (2014). Object weathering simulation avoiding texture space stretching and discontinuities. In *SIGGRAPH Asia 2014 Posters*, SIGGRAPH ASIA '14, pages 37–37, New York, NY, USA. ACM.

Frerichs, D., Vidler, A., and Gatzidis, C. (2015). A survey on object deformation and decomposition in computer graphics. *Computers & Graphics*, 52:18 – 32.

Frerichs, D., Vidler, A., and Gatzidis, C. (2017). Biologically inspired simulation of livor mortis. *The Visual Computer*, 33(11):1453–1466.

Fujisawa, M. and Miura, K. T. (2007). Animation of ice melting phenomenon based on thermodynamics with thermal radiation. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 249–256, New York, NY, USA. ACM.

Fujishiro, I. and Aoki, E. (2001). Volume graphics modeling of ice thawing. In *Proceedings of the 2001 Eurographics Conference on Volume Graphics*, VG'01, pages 67–79, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Fulkerson, D. R. (1961). An out-of-kilter method for minimal-cost flow problems. *Journal of the Society for Industrial and Applied Mathematics*, 9(1):18–27.

Ghosh, A., Hawkins, T., Peers, P., Frederiksen, S., and Debevec, P. (2008). Practical modeling and acquisition of layered facial reflectance. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 139:1–139:10, New York, NY, USA. ACM.

Gibson, S. F. F. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics. Technical Report TR97-19, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139.

Gingold, Y., Secord, A., Han, J. Y., Grinspun, E., and Zorin, D. (2004). A discrete model for inelastic deformation of thin shells.

Glondu, L., Muguercia, L., Marchal, M., Bosch, C., Rushmeier, H., Dumont, G., and Drettakis, G. (2012). Example-based fractured appearance. *Computer Graphics Forum*, 31(4):1547–1556.

Gödde, R. and Kurz, H. (2001). Structural and biophysical simulation of angiogenesis and vascular remodeling. *Developmental Dynamics*, 220(4):387–401.

Goldberg, A. V. and Tarjan, R. E. (1989). Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)*, 36(4):873–886.

Gou, P., Comaposada, J., and Arnau, J. (2002). Meat ph and meat fibre direction effects on moisture diffusivity in salted ham muscles dried at 5c. *Meat Science*, 61(1):25 – 31.

Gu, J., Tu, C.-I., Ramamoorthi, R., Belhumeur, P., Matusik, W., and Nayar, S. (2006). Time-varying surface appearance: Acquisition, modeling and rendering. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 762–771, New York, NY, USA. ACM.

Günther, T., Rohmer, K., and Grosch, T. (2012). GPU-accelerated Interactive Material Aging. *VMV 2012: Vision, Modeling and Visualization.*

Hegemann, J., Jiang, C., Schroeder, C., and Teran, J. M. (2013). A level set method for ductile fracture. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 193–201, New York, NY, USA. ACM.

Hirota, K., Tanoue, Y., and Kaneko, T. (2000). Simulation of three-dimensional cracks. *The Visual Computer*, 16(7):371–378.

Hong, S. M., Simpson, B., and V. G. Baranoski, G. (2005). Interactive venation-based leaf shape modeling: Natural phenomena and special effects. *Comput. Animat. Virtual Worlds*, 16(3-4):415–427.

Hsu, S.-C. and Wong, T.-T. (1995). Simulating dust accumulation. *IEEE Computer Graphics and Applications*, 15(1):18–22.

Igarashi, T., Nishino, K., and Nayar, S. K. (2007). The appearance of human skin: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(1):1–95.

Iglesias-Guitian, J. A., Aliaga, C., Jarabo, A., and Gutierrez, D. (2015). A biophysically-based model of the optical properties of skin aging. *Computer Graphics Forum*, 34(2):45–55.

Im, J., Park, H., Kim, J.-H., and Kim, C.-H. (2013). A particle-grid method for opaque ice formation. *Computer Graphics Forum*, 32(2pt3):371–377.

Ito, T., Fujimoto, T., Muraoka, K., and Chiba, N. (2003). Modeling rocky scenery taking into account joints. In *Computer Graphics International, 2003. Proceedings*, pages 244–247.

Iwasaki, K., Uchida, H., Dobashi, Y., and Nishita, T. (2010). Fast particle-based visual simulation of ice melting. *Computer Graphics Forum*, 29(7):2215–2223.

Jakobsen, T. (2001). Advanced character physics. In *Game developers conference*, volume 3.

Jeong, S., Kim, T.-h., and Kim, C.-H. (2011). Shrinkage, wrinkling and ablation of burning cloth and paper. *Vis. Comput.*, 27(6-8):417–427.

Jeong, S., Park, S.-H., and Kim, C.-H. (2013). Simulation of morphology changes in drying leaves. *Computer Graphics Forum*, 32(1):204–215.

Jimenez, J., Scully, T., Barbosa, N., Donner, C., Alvarez, X., Vieira, T., Matts, P., Orvalho, V., Gutierrez, D., and Weyrich, T. (2010). A practical appearance model for dynamic facial color. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 141:1–141:10, New York, NY, USA. ACM.

Jimenez, J., Sundstedt, V., and Gutierrez, D. (2009). Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept.*, 6(4):23:1–23:15.

Jimenez, J., Zsolnai, K., Jarabo, A., Freude, C., Auzinger, T., Wu, X.-C., von der Pahlen, J., Wimmer, M., and Gutierrez, D. (2015). Separable subsurface scattering. *Computer Graphics Forum*, 34(6):188–197.

Jones, M. D., Farley, M., Butler, J., and Beardall, M. (2010). Directable weathering of concave rock using curvature estimation. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):81–94.

Kelley, A. D., Malin, M. C., and Nielson, G. M. (1988). Terrain simulation using a model of stream erosion. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 263–268, New York, NY, USA. ACM.

Kelman, G. R. (1966). Digital computer subroutine for the conversion of oxygen tension into saturation. *Journal of Applied Physiology*, 21(4):1375–1376.

Kider, J. T., Raja, S., and Badler, N. I. (2011). Fruit senescence and decay simulation. *Computer Graphics Forum*, 30(2):257–266.

Kienle, A., Lilge, L., Vitkin, I. A., Patterson, M. S., Wilson, B. C., Hibst, R., and Steiner, R. (1996). Why do veins appear blue? a new look at an old question. *Appl. Opt.*, 35(7):1151–1160.

Kim, T., Adalsteinsson, D., and Lin, M. C. (2006). Modeling ice dynamics as a thin-film stefan problem. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 167–176, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Kirbas, C. and Quek, F. K. (2003). Vessel extraction techniques and algorithms: A survey. In *Bioinformatics and bioengineering, 2003. Proceedings. Third IEEE symposium on*, pages 238–245. IEEE.

Klein, M. (1967). A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220.

Krishnaswamy, A. and Baranoski, G. V. (2004). A biophysically-based spectral model of light interaction with human skin. *Computer Graphics Forum*, 23(3):331–340.

Krištof, P., Beneš, B., Křivánek, J., and Ďtáva, O. (2009). Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2):219–228.

Larboulette, C., Quesada, P., and Dumas, O. (2013). Burning paper: Simulation at the fiber's level. In *Proceedings of Motion on Games*, MIG '13, pages 25–25, New York, NY, USA. ACM.

Lenaerts, T. and Dutr, P. (2009). Mixing fluids and granular materials. *Computer Graphics Forum*, 28(2):213–218.

Lii, S.-Y. and Wong, S.-K. (2014). Ice melting simulation with water flow handling. *The Visual Computer*, 30(5):531–538.

Liu, S., An, T., Gong, Z., and Hagiwara, I. (2012a). Transactions on edutainment vii. chapter Physically Based Simulation of Solid Objects' Burning, pages 110–120. Springer-Verlag, Berlin, Heidelberg.

Liu, S. and Fan, D. (2015). Computer modeling and simulation of fruit sunscald. *International Journal of Image and Graphics*, 15(03):1550013.

Liu, S., Liu, Q., An, T., Sun, J., and Peng, Q. (2009). Physically based simulation of thin-shell objects' burning. *Vis. Comput.*, 25(5-7):687–696.

Liu, Y., Chen, Y., Wu, W., Max, N., and Wu, E. (2012b). Physically based object withering simulation. *Comput. Animat. Virtual Worlds*, 23(3-4):395–406.

Liu, Y., Yang, X., Cao, Y., Wang, Z., Chen, B., Zhang, J., and Zhang, H. (2015). Dehydration of core/shell fruits. *Computers & Graphics*, 47:68 – 77.

Löffler, F., Müller, A., and Schumann, H. (2011). Real-time rendering of stack-based terrains. pages 161–168.

Losasso, F., Irving, G., Guendelman, E., and Fedkiw, R. (2006). Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):343–352.

Lynnerup, N. (2007). Mummies. *American journal of physical anthropology*, 134(S45):162–190.

Machado, C. (2007). *Brain death: a reappraisal*. Springer Science & Business Media.

Mandelbrot, B. (1982). The fractal geometry of nature . san franciso.

Matsumura, M. and Tsuruno, R. (2005). Visual simulation of melting ice considering the natural convection. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA. ACM.

Mayor, L. and Sereno, A. (2004). Modelling shrinkage during convective drying of food materials: a review. *Journal of Food Engineering*, 61(3):373–386.

Mei, X., Decaudin, P., and Hu, B.-G. (2007). Fast hydraulic erosion simulation and visualization on gpu. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 47–56.

Melek, Z. and Keyser, J. (2003). Interactive simulation of burning objects. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 462–, Washington, DC, USA. IEEE Computer Society.

Melek, Z. and Keyser, J. (2004). Modeling decomposing objects under combustion. In *Proceedings of the Conference on Visualization '04*, VIS '04, pages 598–, Washington, DC, USA. IEEE Computer Society.

Melek, Z. and Keyser, J. (2005). Multi-representation interaction for physically based modeling. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, pages 187–196, New York, NY, USA. ACM.

Melek, Z. and Keyser, J. (2006). Bending burning matches and crumpling burning paper. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, New York, NY, USA. ACM.

Melek, Z. and Keyser, J. (2007). Driving object deformations from internal physical processes. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, pages 51–59, New York, NY, USA. ACM.

Mérillou, S., Dischler, J.-M., and Ghazanfarpour, D. (2001). Corrosion: Simulating and rendering. In *No Description on Graphics Interface 2001*, GRIN'01, pages 167–174, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

Mérillou, S. and Ghazanfarpour, D. (2008). Technical section: A survey of aging and weathering phenomena in computer graphics. *Computers & Graphics*, 32(2):159–174.

Miller, G. S. P. (1986). The definition and rendering of terrain maps. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 39–48, New York, NY, USA. ACM.

Molino, N., Bao, Z., and Fedkiw, R. (2005). A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. ACM.

Muguercia, L., Bosch, C., and Patow, G. (2014). Fracture modeling in computer graphics. *Computers & Graphics*, 45.

Müller, L. O. and Toro, E. F. (2014). A global multiscale mathematical model for the human circulation with emphasis on the venous system. *International journal for numerical methods in biomedical engineering*, 30(7):681–725.

Müller, M. and Chentanez, N. (2010). Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 85–92, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Müller, M., Chentanez, N., and Kim, T.-Y. (2013). Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Trans. Graph.*, 32(4):115–115.

Müller, M., Dorsey, J., McMillan, L., Jagnow, R., and Cutler, B. (2002). Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 49–54, New York, NY, USA. ACM.

Müller, M. and Gross, M. (2004). Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118.

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

Müller, M. T. B. H. M., Pomeranets, D., and Gross, M. (2003). Optimized spatial hashing for collision detection of deformable objects. Technical report, Technical report, Computer Graphics Laboratory, ETH Zurich, Switzerland.

Musgrave, F. K., Kolb, C. E., and Mace, R. S. (1989). The synthesis and rendering of eroded fractal terrains. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 41–50, New York, NY, USA. ACM.

Nagashima, K. (1998). Computer generation of eroded valley and mountain terrains. *The Visual Computer*, 13(9-10):456–464.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836.

Neidhold, B., Wacker, M., and Deussen, O. (2005). Interactive physically based fluid and erosion simulation. pages 25–32.

Neyret, F., Heiss, R., and Sénégas, F. (2002). Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation. *The Visual Computer*, 18(3):135–149.

Ninja Theory Ltd. (2014a). Hellblade. `http://www.hellblade.com`.

Ninja Theory Ltd. (2014b). The Independent AAA Proposition. `http://www.hellblade.com/?p=16972`.

O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. (2002). Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294.

O'Brien, J. F. and Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Onoue, K. and Nishita, T. (2003). Virtual sandbox. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 252–, Washington, DC, USA. IEEE Computer Society.

Paiva, A., Petronetto, F., Lewiner, T., and Tavares, G. (2006). Particle-based non-newtonian fluid animation for melting objects. In *Computer Graphics and Image Processing, 2006. SIBGRAPI '06. 19th Brazilian Symposium on*, pages 78–85.

Papageorgopoulou, C., Shved, N., Wanek, J., and Rühli, F. J. (2015). Modeling ancient egyptian mummification on fresh human tissue: macroscopic and histological aspects. *The Anatomical Record*, 298(6):974–987.

Parker, E. G. and O'Brien, J. F. (2009). Real-time deformation and fracture in a game environment. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 165–175, New York, NY, USA. ACM.

Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., and Guibas, L. J. (2005). Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 957–964, New York, NY, USA. ACM.

Peytavie, A., Galin, E., Grosjean, J., and Merillou, S. (2009). Arches: a framework for modeling complex terrains. *Computer Graphics Forum*, 28(2):457–467.

Pfaff, T., Narain, R., de Joya, J. M., and O'Brien, J. F. (2014). Adaptive tearing and cracking of thin sheets. *ACM Trans. Graph.*, 33(4):110–110.

Pons, J.-P. and Boissonnat, J.-D. (2007). Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8.

Prahlow, J. A. and Byard, R. W. (2011). *Atlas of forensic pathology: for police, forensic scientists, attorneys, and death investigators.* Springer Science & Business Media.

Price, E. and Ostfeld, A. (2014). Optimal water system operation using graph theory algorithms. *Procedia Engineering*, 89:502–508.

Quarteroni, A. (2015). *Modeling the heart and the circulatory system*, volume 14. Springer.

Reddy, J. and Gartling, D. (2010). *The Finite Element Method in Heat Transfer and Fluid Dynamics, Third Edition*. Computational Mechanics and Applied Analysis. Taylor & Francis.

Reichold, J., Stampanoni, M., Keller, A. L., Buck, A., Jenny, P., and Weber, B. (2009). ascular graph model to simulate the cerebral blood flow in realistic vascular networks. *Journal of Cerebral Blood Flow & Metabolism*, 29(8):1429–1443. PMID: 19436317.

Roering, J. J., Kirchner, J. W., and Dietrich, W. E. (1999). Evidence for nonlinear, diffusive sediment transport on hillslopes and implications for landscape morphology. *Water Resources Research*, 35(3):853–870.

Roggan, A., Friebel, M., Dörschel, K., Hahn, A., and Muller, G. (1999). Optical properties of circulating human blood in the wavelength range 400-2500 nm. *Journal of biomedical optics*, 4(1):36–46.

Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A.-G., and Prusinkiewicz, P. (2005). Modeling and visualization of leaf venation patterns. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 702–711, New York, NY, USA. ACM.

Scanlon, V. C. and Sanders, T. (2015). *Essentials of anatomy and physiology*. FA Davis Company.

Si, H. (2015). Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36.

Simal, S., Femenia, A., Garcia-Pascual, P., and Rossell, C. (2003). Simulation of the drying curves of a meat-based product: effect of the external resistance to mass transfer. *Journal of Food Engineering*, 58(2):193 – 199.

Skorkovská, V. and Kolingerová, I. (2015). Multiple material meshes for erosion simulation. In *Proceedings of SIGRAD 2015, June 1st and 2nd, Stockholm, Sweden*, number 120, pages 5–8. Linköping University Electronic Press.

Skorkovská, V. and Kolingerová, I. (2016). Complex multi-material approach for dynamic simulations. *Computers & Graphics*, 56:11 – 19.

Stani, M., Baraldi, A., Boano, R., Cinquetti, R., and Bridelli, M. G. (2014). Study of skin degradation in ancient egyptian mummies: complementarity of fourier transform infrared spectroscopy and histological analysis. *Journal of*

Biological Research-Bollettino della Società Italiana di Biologia Sperimentale, 87(1).

Su, J., Schroeder, C., and Fedkiw, R. (2009). Energy stability and fracture for frame rate rigid body simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 155–164, New York, NY, USA. ACM.

Sumner, R. W., O'Brien, J. F., and Hodgins, J. K. (1999). Animating sand, mud, and snow. *Computer Graphics Forum*, 18(1):17–26.

TEN24-Digital-Capture (2016). TEN24 free 3d model. `http://ten24.info/tag/free-3d-model/`. Copyright ©2016 TEN24 Media Ltd.

Tsumura, N., Ojima, N., Sato, K., Shiraishi, M., Shimizu, H., Nabeshima, H., Akazaki, S., Hori, K., and Miyake, Y. (2003). Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 770–779, New York, NY, USA. ACM.

Tychonievich, L. and Jones, M. (2010). Delaunay deformable mesh for the weathering and erosion of 3d terrain. *The Visual Computer*, 26(12):1485–1495.

Valette, G., Prévost, S., and Lucas, L. (2006). A generalized cracks simulation on 3d-meshes. In *Proceedings of the Second Eurographics Conference on Natural Phenomena*, NPH'06, pages 7–14, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Št'ava, O., Beneš, B., Brisbin, M., and Křivánek, J. (2008). Interactive terrain modeling using hydraulic erosion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 201–210, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Wang, C., Zhang, Q., Xiao, H., and Shen, Q. (2012). Simulation of multiple fluids with solidliquid phasetransition. *Computer Animation and Virtual Worlds*, 23(3-4):279–289.

Wang, J., Tong, X., Lin, S., Pan, M., Wang, C., Bao, H., Guo, B., and Shum, H.-Y. (2006). Appearance manifolds for modeling time-variant appearance of materials. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 754–761, New York, NY, USA. ACM.

Wei, X., Li, W., and Kaufman, A. (2003). Melting and flowing of viscous volumes. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, CASA '03, pages 54–, Washington, DC, USA. IEEE Computer Society.

Whipple, K. X. and Tucker, G. E. (1999). Dynamics of the stream-power river incision model: Implications for height limits of mountain ranges, landscape response timescales, and research needs. *Journal of Geophysical Research: Solid Earth*, 104(B8):17661–17674.

Wicke, M., Hatt, P., Pauly, M., Müller, M., and Gross, M. (2006). Versatile virtual materials using implicit connectivity. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'06, pages 137–144, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Wicke, M., Steinemann, D., and Gross, M. (2005). Efficient animation of point-sampled thin shells. *Computer Graphics Forum*, 24(3):667–676.

Wojtan, C., Carlson, M., Mucha, P. J., and Turk, G. (2007). Animating corrosion and erosion. In *Proceedings of the Third Eurographics Conference on Natural Phenomena*, NPH'07, pages 15–22, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Wu, Y., Kalra, P., Moccozet, L., and Magnenat-Thalmann, N. (1999). Simulating wrinkles and skin aging. *The Visual Computer*, 15(4):183–198.

Yang, H., Sakai, N., and Watanabe, M. (2001). Drying model with non-isotropic shrinkage deformation undergoing simultaneous heat and mass transfer. *Drying Technology*, 19(7):1441–1460.

Yaroslavsky, A., Priezzhev, A., Rodriguez, J., Yaroslavsky, I., and Battarbee, H. (2002). Optics of blood, handbook of optical biomedical diagnostics, valery v. tuchin ed.

Yim, D., Baranoski, G., Kimmel, B., Chen, T., and Miranda, E. (2012). A cell-based light interaction model for human blood. *Computer Graphics Forum*, 31(2pt4):845–854.

Yin, X., Fujimoto, T., and Chiba, N. (2004). Cg representation of wood aging with distortion, cracking and erosion. *The journal of the Society For Art and Science*, 3(4):216–223.

Zhang, N., Zhou, X., Shen, Y., and Sweet, R. (2010). Volumetric modeling in laser bph therapy simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1405–1412.

Zhao, Y., Wei, X., Fan, Z., Kaufman, A., and Qin, H. (2003). Voxels on fire [computer animation]. In *Visualization, 2003. VIS 2003. IEEE*, pages 271–278.

Zhu, J., Chang, Y., and Wu, E. (2011). Realistic, fast, and controllable simulation of solid combustion. *Computer Animation and Virtual Worlds*, 22(2-3):125–132.

Glossary

3D-DDA	3D Digital Differential Alnalyzer
3-G-map	Three-Dimensional Generalized Map
BCS	Body-Centred Square
BRDF	Bidirectional Reflectance Distribution Function
BSP	Binary Space Partitioning
CPU	Central Processing Unit
DDM	Delaunay Deformable Model
EngD	Engineering Doctorate
FEM	Finite Element Method
FFD	Free Form Deformation
GPU	Graphics Processing Unit
GS	Gauss-Seidel-type
LBM	Lattice Boltzmann Method
MAC	Marker-And-Cell
MLS	Moving Least Square
MRI	Magnetic Resonance Imaging
ODC	Oxy-haemoglobin Dissociation Curve
ODE	Ordinary Differential Equations
phyxels	Physical Volume Element
RGBA	Red Green Blue Alpha
SPH	smooth particle hydrodynamics
STAF	Space-Time Appearance Factorization
surfels	Surface Element
VACD	Volumetric Approximate Convex Decomposition
VOF	Volume-Of-Fluid