Simulation of 3D Model, Shape, and Appearance Aging by Physical, Chemical, Biological, Environmental, and Weathering Effects

Joseph T. Kider
University of Pennsylvania, kiderj@seas.upenn.edu

Follow this and additional works at: http://repository.upenn.edu/edissertations

Part of the Computer Sciences Commons

Recommended Citation
http://repository.upenn.edu/edissertations/526

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/edissertations/526
For more information, please contact libraryrepository@pobox.upenn.edu.
Simulation of 3D Model, Shape, and Appearance Aging by Physical, Chemical, Biological, Environmental, and Weathering Effects

Abstract
Physical, chemical, biological, environmental, and weathering effects produce a range of 3D model, shape, and appearance changes. Time introduces an assortment of aging, weathering, and decay processes such as dust, mold, patina, and fractures. These time-varying imperfections provide the viewer with important visual cues for realism and age. Existing approaches that create realistic aging effects still require an excessive amount of time and effort by extremely skilled artists to tediously hand fashion blemishes or simulate simple procedural rules. Most techniques do not scale well to large virtual environments. These limitations have prevented widespread utilization of many aging and weathering algorithms.

We introduce a novel method for geometrically and visually simulating these processes in order to create visually realistic scenes. This work proposes the ‘`mu-ton’ system, a framework for scattering numerous mu-ton particles throughout an environment to mutate and age the world. We take a point based representation to discretize both the decay effects and the underlying geometry. The mu-ton particles simulate interactions between multiple phenomena. This mutation process changes both the physical properties of the external surface layer and the internal volume substrate. The mutation may add or subtract imperfections into the environment as objects age.

First we review related work in aging and weathering, and illustrate the limitations of the current data-driven and physically based approaches. We provide a taxonomy of aging processes. We then describe the structure for our ‘`mu-ton’ framework, and we provide the user a short tutorial how to setup different effects. The first application of the `‘mu-ton’ system focuses on inorganic aging and decay. We demonstrate changing material properties on a variety of objects, and simulate their transformation. We show the application of our system aging a simple city alley on different materials. The second application of the `‘mu-ton’ system focuses organic aging. We provide details on simulating a variety of growth processes. We then evaluate and analyze the `‘mu-ton’ framework and compare our results with `‘gamma-ton’ tracing. Finally, we outline the contributions this thesis provides to computer-based aging and weathering simulation.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Norman I. Badler

Keywords
Aging, Appearance Change, Graphics, Simulation, Weathering

This dissertation is available at ScholarlyCommons: http://repository.upenn.edu/edissertations/526
Subject Categories
Computer Sciences

This dissertation is available at ScholarlyCommons: http://repository.upenn.edu/edissertations/526
SIMULATION OF 3D MODEL, SHAPE, AND APPEARANCE AGING 
BY PHYSICAL, CHEMICAL, BIOLOGICAL, ENVIRONMENTAL, 
AND WEATHERING EFFECTS

Joseph T. Kider Jr.

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2012

Supervisor of Dissertation

Norman I. Badler, Professor,
Computer and Information Science

Graduate Group Chair

Jianbo Shi, Associate Professor,
Computer and Information Science

Dissertation Committee

David P. Pope, Professor,
Materials Science and Engineering

Kostas Daniilidis, Professor,
Computer and Information Science

Julie Dorsey, Professor,
Yale University
(External Committee Member)

Stephen H. Lane,
Associate Professor of Practice,
Computer and Information Science
Acknowledgements

I am first and foremost grateful to my lovely wife, Teresa, for all her love and support. I could not ask for a better friend and partner to help me through my time at Penn. I am also very thankful to all my family, especially my mother, and grandparents who inspired me in my educational pursuits. I additionally would like to thank Dr. Norman I. Badler who has been a great mentor and friend. His wisdom and guidance has tremendously helped my research work and career development.

I am also grateful to all the members in my dissertation committee: Stephen Lane, Kostas Daniilidis, Julie Dorsey, and David Pope. Their insights and suggestions have helped guide my dissertation. I am also grateful to Alla Safonova for all her help and guidance to constantly push me to be better.

During my time at Penn, I have been privileged to work with very talented and friendly labmates in the Center for Human Modeling and Simulation (HMS). I would like to thank Amy Calhoun for being the best listener and for all the moral support. I am externally grateful for all our long conversations. Liming Zhao for being a sincere friend both inside and outside the lab, and my fellow PhD students Jan Allbeck, Aline Normoyle, Cory Boatright, Ben Sunshine-Hill, Pengfei Huang, John Drake, and Alex Shoulson.

I would like to thank Samantha Raja, Yining (Karl) Li, and Kaitlin Pollock for all their help shading, modeling, and general Maya questions. Without their help my images would not be as realistic as they appear. I also want to thank all the the Digital Media Design and Computer Graphics and Game Technology students I have had the pleasure to work with during my time at Penn. They have all been a true inspiration.
ABSTRACT
SIMULATION OF 3D MODEL, SHAPE, AND APPEARANCE AGING
BY PHYSICAL, CHEMICAL, BIOLOGICAL, ENVIRONMENTAL,
AND WEATHERING EFFECTS
Joseph T. Kider Jr.
Supervisor : Norman I. Badler

Physical, chemical, biological, environmental, and weathering effects produce a range of 3D model, shape, and appearance changes. Time introduces an assortment of aging, weathering, and decay processes such as dust, mold, patina, and fractures. These time-varying imperfections provide the viewer with important visual cues for realism and age. Existing approaches that create realistic aging effects still require an excessive amount of time and effort by extremely skilled artists to tediously hand fashion blemishes or simulate simple procedural rules. Most techniques do not scale well to large virtual environments. These limitations have prevented widespread utilization of many aging and weathering algorithms.

We introduce a novel method for geometrically and visually simulating these processes in order to create visually realistic scenes. This work proposes the “µ-ton” system, a framework for scattering numerous µ-ton particles throughout an environment to mutate and age the world. We take a point based representation to discretize both the decay effects and the underlying geometry. The µ-ton particles simulate interactions between multiple phenomena. This mutation process changes both the physical properties of the external surface layer and the internal volume substrate. The mutation may add or subtract imperfections into the environment as objects age.

First we review related work in aging and weathering, and illustrate the limitations of the current data-driven and physically based approaches. We provide a taxonomy of aging processes. We then describe the structure for our “µ-ton” framework, and we provide the user a short tutorial how to setup different effects. The first application of the “µ-ton”
system focuses on inorganic aging and decay. We demonstrate changing material properties on a variety of objects, and simulate their transformation. We show the application of our system aging a simple city alley on different materials. The second application of the “μ-ton” system focuses organic aging. We provide details on simulating a variety of growth processes. We then evaluate and analyze the “μ-ton” framework and compare our results with “γ-ton” tracing. Finally, we outline the contributions this thesis provides to computer-based aging and weathering simulation.
Contents

Acknowledgements ii

Acknowledgements ii

Contents v

List of Tables ix

List of Figures x

1 Introduction 1

1.1 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.2 Aging Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.2.1 μ-ton System Framework . . . . . . . . . . . . . . . . . . . . . . . 6

1.2.2 Organic Material Simulation . . . . . . . . . . . . . . . . . . . . . 7

1.2.3 Inorganic Material Simulation . . . . . . . . . . . . . . . . . . . . 7

1.2.4 Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

1.3 Organization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2 Related Work 9

2.1 Rule-based Visual Simulation Techniques . . . . . . . . . . . . . . . . 9

2.2 Physically-based Simulation Methods . . . . . . . . . . . . . . . . . . 12

2.3 Chemically-based Simulation Methods . . . . . . . . . . . . . . . . . 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Organically-based Simulation Methods</td>
<td>14</td>
</tr>
<tr>
<td>2.5</td>
<td>Data-driven Simulation Methods</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Taxonomy of Aging Phenomena</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>Organic Phenomena</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Environmental Phenomena</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>Chemical Phenomena</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>Physical Phenomena</td>
<td>24</td>
</tr>
<tr>
<td>3.5</td>
<td>Weathering Phenomena</td>
<td>25</td>
</tr>
<tr>
<td>3.6</td>
<td>Multiple Interacting Phenomena</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>$\mu$-ton Simulation Framework</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>$\mu$-ton - Point Based Representation</td>
<td>28</td>
</tr>
<tr>
<td>4.2</td>
<td>$\mu$-ton Source Emitters</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>$\mu$-ton Propagation</td>
<td>35</td>
</tr>
<tr>
<td>4.4</td>
<td>$\mu$-ton Intersection</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>$\mu$-ton Distribution Maps</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>$\mu$-tation (Mutation)</td>
<td>40</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Geometric Material Update</td>
<td>40</td>
</tr>
<tr>
<td>4.6.2</td>
<td>$\mu$-ton Particle Attribute Update</td>
<td>43</td>
</tr>
<tr>
<td>4.6.3</td>
<td>$\mu$-ton Material Properties</td>
<td>44</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Constraint Simulation</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>A $\mu$-ton Tutorial Example</td>
<td>49</td>
</tr>
<tr>
<td>5.1</td>
<td>Setting up an Aging Effect</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>Inorganic Aging</td>
<td>55</td>
</tr>
<tr>
<td>6.1</td>
<td>Geometric Decay</td>
<td>55</td>
</tr>
<tr>
<td>6.2</td>
<td>Simulating Different Materials</td>
<td>57</td>
</tr>
<tr>
<td>6.3</td>
<td>Corrosion Example</td>
<td>58</td>
</tr>
</tbody>
</table>
6.4 Inorganic Results .................................................. 61

7 Organic Aging ......................................................... 63
  7.1 Implementation Overview ..................................... 63
  7.2 Algorithm Details ............................................... 65
  7.3 Organic Decay Results ........................................ 67

8 Evaluation and Analysis ........................................... 68
  8.1 Implementation ................................................ 69
    8.1.1 User Interface Tool .................................... 69
  8.2 Performance Testing .......................................... 71
    8.2.1 Memory .................................................. 72
  8.3 Comparison with γ-ton tracing ............................... 73
  8.4 Comparison with a Simple Erosion system ................ 76
  8.5 Perceptual User-Study ....................................... 77
  8.6 Discussion .................................................... 80

9 Summary of Contributions ........................................ 81
  9.1 Future directions ............................................. 83
    9.1.1 Practical Use ........................................... 83

A Fruit Senescence and Decay Simulation ......................... 84
  A.1 Introduction .................................................. 84
  A.2 Biological Background of Fruit Decay ..................... 86
  A.3 Fruit Senescence and Decay ................................. 89
    A.3.1 Distribution Maps ...................................... 89
    A.3.2 Internal Volume ......................................... 91
    A.3.3 External Skin ........................................... 92
    A.3.4 Presets and Parameters ................................ 93
    A.3.5 Rendering ................................................ 93
List of Tables

3.1 Organic phenomena classes of interactions. ........................................ 19
3.2 Environmental phenomena classes of interactions. ............................ 21
3.3 Chemical phenomena classes of interactions. .................................... 23
3.4 Physical phenomena classes of interactions. ..................................... 25
3.5 Weathering phenomena classes of interactions. ................................. 26

4.1 This table provides some basic material properties for a variety of different materials. [2]. ................................................................. 45

7.1 This table provides the biologically inspired parameters [35] for our reaction-diffusion mathematical model. ............................................. 67

A.1 This table provides the weight, water content, and the transpiration coefficient rates (K) for several fruits. [4, 15]. ................................. 88
A.2 This table provides the input parameters for the different fruits in our results. The temperature was set to 72 degrees and humidity to 50%. .... 93
List of Figures

1.1 Examples of decay phenomena: An example of a bridge losing paint and rusting, a tree changing color, fungus and mold forming on left over spaghetti, and mold and rot on a lemon. .................................................. 2

1.2 Examples of decay simulation from the $\mu$-ton System Framework: An example of a tomato, apple, and mold formation. ................................. 5

1.3 An example illustration of the $\mu$-ton System Framework. ............................ 6

2.1 Examples of rule based simulations: (a) Blemishes created from simple distribution models [5], (b) Cracks simulated by 3D surface automata [30], (c) A reaction diffusion technique for generating textures for arbitrary surfaces [71], (d) $\gamma$-ton tracing [16] multiple phenomena on a fire hydrant. . 11

2.2 Examples of physical, chemical, and organic simulation. (a) Simulating metallic patinas [22], (b) Simulating weathered stone [21], (c) A physically-based model for rendering realistic scratches [12], (d) Simulating and propagating lichen growth [19]. .................................................. 13

2.3 Examples of data-driven simulation methods: (a) image based material weathering [77], (b) synthesis of material drying history controlled by a drying curve [46] (c) simulating crack patterns by observing and transferring material histories [29] (d) a photograph of the multi-light source multi-camera dome used to obtain time-varying surface appearances [32]. 16
3.1 Organic Phenomena: moss on a tree, algae on rocks, lichen on rocks, moss on a branch, rotten pear, tomato fungi, lemon mold, and noodles.

3.2 Environmental Phenomena: dust accumulation, leaves, leaves and trash, trash accumulation, human paths, graffiti, and a abandoned shed.

3.3 Chemical Phenomena: rust formation from oxidation and leaf color change.

3.4 Physical Phenomena: leaf crumbling, glass shattering, newspaper tearing, paint peeling, rock eroding, ice melting, paint chipping, and ice calving.

3.5 Weathering Phenomena: snow accumulation, temperature and radiation from the sun, rain, and snow.

3.6 Multiple Phenomena: Wood chipping, fading, moss growth, paint peeling, fading, and rusting, mold, fungi, and color change, bark peeling, moss forming, and fungi.

4.1 This figure outlines the $\mu$-ton tracing framework. The $\mu$-ton tracing framework is similar to photon tracing, however it affects the underlying geometry and diffuses and reacts with the surface.

4.2 This figure illustrates the application of different $\mu$-ton emitters inside our Maya plug-in. Here are examples of different users setting up an area, a shape, and hose to fit their production needs. This shows just a few ways users can cast $\mu$-tons into the scene.

4.3 This figure illustrates the different types of $\mu$-ton emitters: (a) point, (b) directional, (c) spot, (d) area, (e) volume, and (f) ambient.

4.4 This figure illustrates material $\mu$-tons scattered inside an object. The left image shows there is a single decay $\mu$-ton (yellow) above the cube, the decay $\mu$-ton intersects the geometry forcing local re-meshing (middle), and the newly emitted $\mu$-tons are shown on the right. This is another type of emitter.
4.5 This figure shows $\mu$-tons propagating and intersecting in a test scene. Here the yellow particles represent the $\mu$-tons, and the red points show intersection points. .......................... 36

4.6 This figure shows a simple k-d tree the $\mu$-ton system uses to partition space to accelerate intersection queries and to find the nearest material or surface $\mu$-tons in a scene. .......................... 37

4.7 This figure illustrates different types of surface point based $\mu$-ton distribution maps on a bench. $\mu$-tons on the surface store properties for various aging effects. These maps are the primary data-structure that stores various aging properties for the object as it changes over time. .......................... 38

4.8 This figure illustrates various distribution map styles on a sphere. These include ambient occlusion, normal map, user defined pattern, multiple types of mold growth, soft rot, and fungus. .......................... 39

4.9 This figure breaks down 3 pieces of geometry into their voronoi regions and gives each a unique random color for visualization. Different $\mu$-ton particles are scattered in the space and store various material properties at every location for the cell. .......................... 40

4.10 This figure illustrates the underlying geometry. Here the system consists of $\mu$-ton particles and a constraint lattice spanning neighboring faces. .......................... 41

4.11 This figure illustrates two simple Voronoi cells on a sphere and a constraint on the neighboring face. .......................... 42

4.12 This figure illustrates the geometric update that happens upon a change occurring in the material. Notice how the top of the right image has more material $\mu$-tons and therefore more local tessellation only where aging is occurring. .......................... 43
4.13 This figure illustrates mutating the geometry. As the $\mu$-ton particles fall, they reconfigure the geometry at the intersection point. Here the particle emits 3 new particles, calculates the $L_1$ norm which is the Voronoi diagram. Then we take the dual which is the Delaunay triangulation of the geometry. The process repeats as more $\mu$-ton particles are emitted and more intersections occur.

4.14 This figure shows a variety of material updates. The top row shows the closest $\mu$-ton material particle updating with an aging process update and turning white. The top row left, shows a more gradual update. In the middle row the size of the $\mu$-ton decay particle is changed so it has a higher sphere of influence than the top row. The middle row right shows aging phenomena being carried away from the intersection area. The bottom row shows a different color representation of many interacting effects.

4.15 This figure illustrates the comparison of different materials’ Young’s Modulus and density [43].

4.16 This figure illustrates the relationship between stress and strain, and where the fracture point occurs. [43]

5.1 This figure shows our Maya plug-in in action: a hemispherical $\mu$-ton source, decay $\mu$-tons visualized in yellow, and the plane to age selected in green.
5.2 This figure shows the various stages of the mold / wall example. On the
top left is the nutrient map which was created with 2 passes of a noise
function, and the top right image is the distribution map calculated from
the $\mu$-ton hits. The second row shows the various stages of the $\mu$-ton
hits. The left image shows the intersection maps show the first time $\mu$-ton
decay particles interact with the wall, the right image shows the beginning
of growth from the reaction-diffusion process, and the left bottom image
is the final growth pattern. Finally, the lower right image is the final result
rendered onto the wall.

6.1 This figure shows a more complex street scene from an alley way. Here
there are multiple geometry mutations happening at once. The curb is
breaking off, pieces are falling to the ground, and the paint on the road is
chipping and peeling away.

6.2 This figure shows a more detailed view of the street scene from an alley
way. Here there are multiple geometry mutations happening at once. The
curb is breaking off, pieces are falling to the ground, and the paint on the
road is chipping and peeling away.

6.3 This figure shows a very simple cube with only 10 voronoi regions. The
gometry starts to erode away from multiple decay $\mu$-ton hits.

6.4 This figure shows different runs of the $\mu$-ton framework on a cube (top
left). The constraint strength parameters are changed in the various simu-
lations which lead to a wide range of mutation effects.

6.5 This figure shows a photograph of pitted corrosion (JEI Metal Inc).
6.6 This figure shows the result of inorganic decaying over time of paint chipping, rust, and pitted corrosion. The left top image is the original geometry in Maya. The next two on the top row show the paint chipping away, and the pitted corrosion forming. The bottom left image is the distribution map of the rust flowing down and away from the holes in the paint, then all the geometry together, and then final rendered frame. 61

6.7 This figure shows the result of two rendered frames of the inorganic aging process. 62

6.8 This figure shows the result of the street scene with paint chipping away and the pavement crumbling. 62

7.1 This figure shows the result of an organic aging processes in the \( \mu \)-ton framework. The left is a plane log, and the next two images in the growth of moss over time. 67

8.1 This figure show images of an OpenGL (left) implementation and a Maya plug-in (right) of the framework. 68

8.2 A screen shot of our user interface tool in Autodesk’s Maya. Our plug-in tool allows users to choose presets and parameterize the simulation to generalize output. The user imported her own geometry and is currently displaying the nutrient map on the object. 70

8.3 Performance testing of the major bottlenecks for the \( \mu \)-ton framework. Here we tested 4 objects: Object 1 (8 vertices and 12 faces), Object 2 (382 vertices and 760 faces), Object 3 (1562 vertices and 3120 faces), and Object 4 (6242 vertices and 12,480 faces). The graph plots number of particles scattered vs time to compute on a log scale. For every particle scattered the Voronoi cell computation and boolean operation were timed. 72
8.4 Performance testing of the second slowest bottleneck for the $\mu$-ton framework. Here we tested 4 objects: Object 1 (8 vertices and 12 faces), Object 2 (382 vertices and 760 faces), Object 3 (1562 vertices and 3120 faces), and Object 4 (6242 vertices and 12,480 faces). The graph plots number of particles scattered vs time to compute on a log scale. This calculates the time to build the k-d tree acceleration structures to help speed up the collision detection and finding the nearest neighbors.

8.5 This figure illustrates the differences and advantages of the $\mu$-ton tracing framework. (Left) $\gamma$-ton tracing [16] framework. (Right) $\mu$-ton tracing framework. On the right the $\mu$-ton tracing framework updates and mutates the underlying geometry. The framework also accounts for more complex emission sources.

8.6 Rendered log with moss. On the left is the result of $\gamma$-ton tracing. $\gamma$-ton tracing transported moss into the cracks of the log and blended the textures together. The next two images are frames from the $\mu$-ton framework where the moss grows over time from reaction-diffusion. Here the simulation adds substance to the scene.

8.7 This figure shows a rendered sphere with geometry changes. On the top is the result of $\gamma$-ton tracing, which uses displacement maps. The bottom image is the result of the $\mu$-ton framework. Here the actually geometry is changed, and pieces break off the object over time and gather near the sphere. This is not possible with $\gamma$-ton tracing.

8.8 This figure shows a rendered sphere with geometry changes. Here we simulated a simple erosion system, and compared the results to the $\mu$-ton system. Our simulation allows for more localized effects, and less tessellation then subdividing the entire mesh at once in the beginning.
8.9 Sample screen shots from one of the videos shown. The top row is the reference time-lapse video of a decaying tomato. The bottom row is a comparison of a simulated decaying tomato.

8.10 An example set of images shown in the user-study. A simulated wall of concrete, simulated mold growth, and a real photograph of mold growth [Basement Dehumidifiers].

8.11 Perceptual study responses plotted on a five-point Likert scale for the perceived realism of the simulation videos an images. Higher numbers are better.

8.12 Perceptual study responses plotted on a five-point Likert scale for how disgusting are the simulation videos an images. Higher numbers are better.

A.1 Photographs of real fruit decay. (a) Penicillium digitatum formation on an orange. (b) Neofabraea formation on an apple. (c) Rhizopus formation on a tomato. (d)-(f) Time lapse of fungal growth on the bottom point of contact between a tomato and a glass platform.

A.2 The system framework highlighting the important steps in our fruit decay pipeline. The user provides the input mesh and parameters to the system. Our model calculates the growth and decay and then renders the final image.

A.3 Results of the final maps: Values range between \{0 \ldots 1\} for the phenomenon concentration values. These processes change the physical properties of the surface and internal volume substrate. (a) Shows nutrient depletion, (b) Fungal colony growth, (c) Soft rot spread, (d) Final rendered frames.

A.4 A composite image of a cross-section view of our fruit’s anatomy. (Right) shows a final rendered frame. (Left) shows the skin and deformable body of our substrate.
A.5 Left: Example result of a simulated tomato decaying. Here multiple mold and rot processes affect the fruit. Right: Example result of a bowl of oranges decaying. Here mold and rot start at the contact points.

A.6 A comparison of real photographs (right), and our simulation results (left) of a tomato and orange.

A.7 Simulation results of a tomato and an orange under three different parameter settings set by three users from our user interface. Simulation (a) and (b) set two different pressure points and produce different results. (c) changes parameters in the mold growth to simulate different growth rates.

B.1 This is a demonstration of a tomato and bowl fruit aging on a table scene.

B.2 This is a demonstration of a tomato being aged.

B.3 This is a demonstration of a second tomato being aged.

B.4 This is a demonstration of a third tomato being aged.

B.5 This is a demonstration of a final apple being aged.

B.6 This is a demonstration of a second apple being aged.

B.7 This is a demonstration of an orange being aged.

B.8 This is a demonstration of a second orange being aged.

B.9 This figure shows a log with moss growth.

B.10 This figure shows a log with more moss growth.

B.11 This figure shows a log with more moss growth.

B.12 This figure is a blank wall with mold growth and propagation.

B.13 This figure is a concrete wall with mold growth and propagation overtime.

B.14 This figure is a 2nd run of the mold growing on a wall.

B.15 This figure shows the bowl after the aging process and mold has grown inside of it.

B.16 This figure shows a close up image of the mold rendered on the bowl.

B.17 This figure shows paint chipping away, rust, and pitted corrosion.

B.18 This figure shows paint chipping away, rust, and pitted corrosion.
B.19 This figure shows small a test scene of the curb crumbling. 107

B.20 This figure shows our street scene of a curb crumbling and paint chipping away. 107

B.21 A rendered alley scene with all the previous effects simulated. 108
Chapter 1

Introduction

Blockbuster movies, video games, and virtual reality applications increasingly utilize computer generated models to create complex and realistic environments. From a realistic “middle-earth” in the *Lord of the Rings*, to detailed “trash dumpsters” in *Toy Story 3*, these virtual worlds provide users with a perception of immersion in an imaginary environment.

Unfortunately, many computer generated objects and materials often seem too clean and flawless to actually look real. In computer generated scenes, blemish and weathering effects provide an important visual cue of realism to the audience. A metal object begins to rust over time, leaves change color, mold begins to grow on plates, and fruit decomposes (Figure 1.1). All objects age and the nature of the materials change. These shifts in appearance give important visual cues of age that can enhance the realism of a synthetic scene. Therefore, there is a clear need for automatic methods to generate realistic decay phenomena. Aging phenomena changes the material properties of objects over time and helps scenes appear more realistic.

While many methods exist for producing aging and decay phenomena, generating realistic effects, where the processes interact both on the surface and substrate, remains an immensely difficult task. Time-varying aging phenomena are complicated because their interactions often happen externally at a global level or internally at a complex molecular level. Materials and substances frequently have intricate compositions. Many material
scientists and chemists still do not understand all of the stages in decay processes. Dorsey and Hanrahan [22] describe how the development and breakdown of surface coatings of copper patina are not fully solved in chemistry. In addition, the chemical and physical processes of aging and surface breakdown are scientifically interesting mainly from a prevention perspective, rather than the computer graphics generative view. We want materials to rot and decay. Humans are highly proficient in perceiving anything that appears unnatural. When an object is aged incorrectly, or not aged at all, human observers quickly spot the problems. Aging materials and objects need to balance realism, computation, and accuracy to immerse viewers and to ensure that presence is not broken in the artificially generated environment.

Developing intuitive tools for both professional modelers and novice 3d software users continues to be a challenge. Many Hollywood and game studios manually hand-fashion textures and utilize intricate shading networks to place blemishes in their virtual scenes. The artist has full control of the style, amount, and location of decay in the shot. An artist fine tunes the details and iteratively works with the director to achieve the look and feel they desire. This approach suffers from three fundamental flaws: the technique is artistically rigorous, manually exhaustive, and monetarily expensive to create the amount of textures needed for movies and games; the creation of these textures is subjective and
lacks any sense of realism or methodology; and this approach maintains a static look that
does not change over time. When artists attempt to hand-fashion a decay process, they
have trouble maintaining the temporal coherence from shot to shot. The subtle complexi-
ties of many blemish and weathering effects often prove difficult to correctly replicate by
hand.

With the growing complexity of movies and demanding directors, simulation offers a
straightforward solution. These techniques rapidly create production materials that tech-
nical artists fine-tune. These simulations created salads in *Ratatouille*, the stained objects
in *King Kong*, and help rust the character Mater in *Cars*. For simulations to further spread
into the studio production pipeline, they must combine the best of what a computer auto-
matically generates with the best of what the technical artists produce. Technical artists
will always want control over the final appearance of frames in a movie; that fact is un-
avoidable. However, simulating blemish animations may simplify an artist’s workload to
maximize their creative utility and resources elsewhere in the production pipeline.

In this thesis we address the fundamental difficulty in simulating multiple interacting
phenomena. We aspire to construct cheaper, quicker, and more controllable tools and
techniques to realistically simulate dynamic decay. First, we provide a general framework
to age a variety of materials types. We present our $\mu$-ton (pronounced Mu-ton) system.
This framework scatters numerous $\mu$-ton particles throughout an environment to change
and mutate the world. We take a point-based representation to discretize both the decay
effects and the underlying geometry. The $\mu$-ton particles simulate the interactions between
multiple aging and weathering phenomena. This mutation process changes both the physi-
cal properties of the external surface layer and the internal volume substrate. The mutation
may add, subtract, or transform imperfections in the environment. This approach is a gen-
eral method that simulates different decay phenomena classes, and provides a rich set of
tools to help artists author these effects.

The first application of this method addresses the change and decay of inorganic mate-
rials over time. The visual simulation of metal corroding and changing over time requires
that the underlying geometry to changes and breaks off. Again, an artist could use simple
techniques, such as hand painting a displacement map, to achieve this effect. This ad-hoc
approach does not achieve the fine grain detail of watching objects change and break off
over time, nor the artist-driven method produce the small flakes of debris commonly found
next to aging materials.

The second application of this method addresses the rather unexplored area of organic
aging. The visual simulation of organic aging process requires attention not only to aes-
thetic changes, but also to the underlying biology of senescence (aging) and decay. While
artists can add visual imperfections to their models and environments through traditional
texture mapping and physics-based modeling methods, these may be based on observa-
tion, intuition, and artistic skill. Parameterized procedural decay methods which are built
on scientific principles could be useful in making artistic rendering more precise. These
growth patterns are easily extended to mold and moss under the $\mu$-ton system.

1.1 Simulation

Simulation automates the problematic tasks of hand fashioning blemishes and yields a
broad assortment of appearance controls. Simulations using parametrically defined tex-
tures can be iteratively tuned and can alleviate most hand painting. A designer merely
adjusts the simulation parameters rather than recreating an entire scene over and over.
In order to achieve a simulation the underlying physical processes of natural phenomena
must be modeled by rules, physically based methods, or collected data. The simulation
represents key features and behaviors. Simulation tools also tend to make certain assump-
tions to approximate how complex surface imperfections form and change over time. As a
drawback, artists tend to have less fine grained control with most simulations due to their
complex nature. Many simulations provide a set of initial parameters, and then run an un-
derlying mathematical model. The particular construction of the simulation is extremely
relevant to producing realistic results. In this thesis, we provide the user with a set of
tools that both allow a simulation to create accurate decay phenomena, and allow an artist to tune the simulation to produce their desired effect. The artistic tools may relax some of the biological, chemical, and physical accuracy of the result in favor of control only when specifically required. At any point in time, an artist may hand paint more detail on distribution maps to guide the simulation to their desired configuration.

Several surveys [24, 52] summarize aging and weathering phenomena simulation in computer graphics. Current research presents three simulation techniques to synthesize realistic appearing models: rule-based texture synthesis, physically-based texture synthesis, and data-driven texture synthesis. However, these methods neither directly include, nor can they be readily extended to, the complex interacting processes associated with decomposition that affects both the surface and substrate of a large number of objects.

1.2 Aging Simulation

This thesis provides a framework for simulating multiple interacting aging phenomena. An aging simulation is an off-line simulation which mutates objects in the world in order to maximize the visual perception time has passed.

An aging simulation must provide the following features to produce realistic and compelling results:
1. Simulate multiple interacting phenomena that affect both the surface properties and the substrate properties of an object.

2. Generalize to multiple classes of decay phenomena to simulate a variety of decay effects, in which the multiple effects interact with each other to produce more dynamic simulations.

3. Simulate virtual worlds with many substances and objects efficiently, so there is a rich and robust environment.

The $\mu$-ton framework has the following features, which will be discussed in detail in future chapters of this thesis.

### 1.2.1 $\mu$-ton System Framework

The $\mu$-ton system framework (Figure 1.3) is a technique for simulating a variety of classes of decay phenomena. This method generalizes to a variety of material types, and simulates
desired effects across different aging classes from a provided taxonomy. The main objective of this system is to provide a general system that can produce several aging effects and combine multiple effects together. The goal was not to produce a perfectly physically correct system, however we do provide a large degree of accuracy with many of our results since we ground our growth patterns and material properties in science. This framework can easily be integrated into a variety of 3D software, and we demonstrate integration with Autodesk’s Maya 2012 and an OpenGL framework.

1.2.2 Organic Material Simulation

We have developed a system for simulating organic growth. This system is based on a biological background for simulating organic classes of decay phenomena. This simulation creates multiple types of substances that directly interact with one another. The simulation affects both the surface and the substrate. We have applied this to fruit senescence and decay, moss, and mold simulations. Experimental results show that our approach and user tools achieve much better and more robust results than standard hand-painting and simple rule-based methods.

In our approach, we use a reaction-diffusion model to create complex biological processes, such as the growth of fungus and bacteria, colony formation, and soft rot. Fungal colonies diffuse radially, interacting with each other and the nutrients on the fruit surface. The decay affects both the surface and internal substrate of the material. Our model parameters for decaying organic substances are biologically-derived and present a novel visual simulation method that has not been covered in previous approaches. This is visualized in Figure 1.2.

1.2.3 Inorganic Material Simulation

We have developed a system for simulating a variety of inorganic materials. This system is based on material science for simulating different material properties. This simulation
changes the underlying mesh of the object, and breaks off small debris over time.

1.2.4 Evaluation

We demonstrate and evaluate the $\mu$-ton system framework on a variety of organic and inorganic substances. We compare our results to $\gamma$-ton tracing [16] which is the most similar in nature to our approach. $\gamma$-ton tracing is the only other major work that attempts to simulate multiple again phenomena. We also conducted a small user study to compare and evaluate our simulation to determine if we produced realistic results.

1.3 Organization

In the remainder of this thesis, we will first review past work directly summarizing decay simulation. We will then describe a taxonomy of decay phenomena that provides a general classification scheme to all these various aging effects. We first explain the general technique for aging objects - our $\mu$-ton system framework. This system provides details on the process of integrating multiple interacting phenomena. We apply the framework in a short tutorial that explains how to set up the system for a mold decay effect. We then discuss how the $\mu$-ton system framework can simulate both inorganic and organic substances. We analyze the performance and compare our work with other approaches. Finally, we summarize the thesis work, and the main contributions this thesis provides to computer-based modeling and aging simulation communities.

Additionally, we have provide two appendixes for the reader. Appendix A discusses details of our fruit senescence and decay simulation, which presents a novel physically-based approach for simulating the organic class of decay phenomena. This work directly simulates the fruit’s surface and substrate. Appendix B provides high resolution production images of the 17 different aging effects found throughout this dissertation simulated on 10 different environments.
Chapter 2

Related Work

This chapter reviews existing work in aging simulation. Section 2.1 gives an overview of rule based techniques. Section 2.2 focuses on physically based aging. Section 2.3 describes previous chemically based techniques. Section 2.4 outlines organic approaches. Section 2.5 describes pure data-driven approaches to aging and decay simulations.

2.1 Rule-based Visual Simulation Techniques

Rule based techniques refer to a broad spectrum of generating textures and displacement maps from mathematically based models onto an object. Blemishes are not entirely random; there usually exists a fundamental logical distribution. For example: exposed surfaces accumulate dust, trash piles against walls, and paint cracks over time. The user adjusts a set of parameters that change the outcome of the simulation based on the predetermined mathematical rules. Later on, a technical artist fine tunes small details in the resulting output to meet specific demands. Rule based simulations do not fully guarantee physical accuracy, however they do provide a rapid approach to model visually plausible surface imperfections.

Early work on creating imperfections focused on procedural texture synthesis. These procedurally generated textures enrich a model’s detail. Cook [17] proposed surfaces have
more complex shading distinctiveness with procedural textures. Ebert et al.’s [25] classic book on procedural texturing and modeling provides even further detail on the usefulness of procedural rule based methods. Perlin [62] introduced a pseudo-random ‘noise’ function to create more realistic looking textures. ‘Perlin noise’ is a simple yet powerful technique still used in many blemish simulations to simulate an inherent randomness with some useful ability to control the local appearance through a discrete assignment of desired gradients or features at selected grid points. Turk [71] and Witkin and Kass [73] proposed using reaction-diffusion methods for texture synthesis (Figure 2.1). Reaction-diffusion textures come from a biochemically inspired set of partial differential equations. This non-linear interaction process creates natural patterns, such as stripes and spots. Later, Zhang et al. [79] presented a method for using progressively-variant textures to depict local surface variations, such as orientation, color, and shape in more complex animal coat patterns.

The simulation of natural phenomena from a rule-based approach dates back to many of James Blinn’s early papers [9, 7, 8] and SIGGRAPH panel [6]. This work illustrated to the graphics community the complexity of time-varying phenomena as a complex problem in the graphics community. This work drew attention to surface imperfections, however it was primarily concerned with the visual appearance of the blemishes (such as dust accumulation and wrinkles) and did not focused on how to appropriately distribute imperfections in a scene. Becket and Badler [5] proposed using simple distribution models with a simple natural language interface that formed rules to simulate various surface imperfections (Figure 2.1 (a)). This approach helped to illustrate the need for aging processes. Their system only generated a 2D texture which did not account for environmental reactions, intrinsic geometry, or surface irregularities. Hsu and Wong [34] modeled dust accumulation based on mathematical rules derived from the surface of an object. These methods used simple distribution models to calculate the aging effect’s distribution pattern. Gobron and Chiba [30, 31] modeled crack propagation based on a set of semi-physical rules.
Though rule-based simulation appears rather subjective, there are a few rule selections that appear unavoidable, because they exploit the structure of an object to produce a more realistic appearance. For example, Wong et al. [75] demonstrated that the fundamental geometry of an object affects the tendency towards blemish distributions. Dust accumulation is dependent upon surface inclination and stickiness. Miller [54] computed surface accessibility and used the metric to produce metallic patina imperfection on an object’s surface. Rule-based simulation techniques must exploit the inherent structure in a scene so the aging and weathering effects appear less random and more realistic.

γ-ton tracing [16] is a recent rule-based approach to simulate aging phenomena. This work is the most clearly related research to the topics of this dissertation. γ-ton tracing introduces a rule-based simulation approach developed from techniques used in photon tracing (developed for image synthesis) to produce various blemish effects. γ-ton particles use motion probabilities and carrier attributes to affect the local aging properties. Figure 2.1 (d) illustrates an example of a simulation including multiple phenomena interacting on a fire hydrant.

Rather than a probabilistic approach proposed by γ-ton, the µ-ton system spreads particles on the surface and substrate of the object. For example, we employ a reaction-diffusion system to produce aging phenomena on multiple textures and mesh layers in our pipeline. In the physical world, for example, we find that as nutrients diminish, mold
growth stops. Mold does not just transport, it grows. This result would be difficult to recreate with $\gamma$-ton tracing. Furthermore, in the $\mu$-ton framework the underlying geometry is mutated and changed over time. $\gamma$-ton tracing presented the notion of using displacement maps to change the geometry, however method would simply remove but not flake off discrete amounts of debris which commonly happens in the real world. Such results can be easily generated with our system since the underlying geometry is broken up with particles, not just the surface.

2.2 Physically-based Simulation Methods

Several researchers have attempted physically-based simulation methods to automate the creation of realistic aging and imperfections. Physically based simulation techniques are based primarily on underlying physical properties rather than on somewhat arbitrary statistical distributions and ad hoc mathematic equations. This technique attempts to approximate tangible physical laws into computational models to realistically simulate aging phenomena and aging effects. Here too, the modeler adjusts a set of parameters that change the outcome of the simulation based on the predetermined mathematical rules. Physically based rules are more rigidly designed since they hope to ensure some approximation of physical accuracy. Modelers and artists tend to have less control on the final output of the simulation, however they may achieve more natural appearing results then previous visual simulation techniques.

Hsu and Wong [34] integrated surface properties such as inclination and stickiness to simulate dust accumulation. Dust fell on areas determined by these surface properties and changed across an object. Paquette and Drettakis [60] simulated paint cracking and peeling and surface aging by impacts [59]. This was some of the first work to change the underlying geometry and not focus solely on the surface. Bosch and his colleagues [12] simulated the formation of scratches and rendered their visual behavior under a variety of lighting conditions (Figure 2.2). Cracks and fractures play an important role in aging
solid structures [20, 31]. Paquette and Drettakis modeled spectrum stress as an additional technique to generate cracks in a semi-physical approach. Other researchers proposed methods to simulate appearance-changing flows [23] to weathered stone [21]. Cutler et al. [18] create a procedural approach for modeling solid objects. They simulated tree growth by displacing geometry and dirt bricks upward around the tree roots. There has been a wide range of work simulating hydraulic and rainfall erosion [50, 38, 40, 56].

Physically-based method produce more realistic results than simple rule-based simulations since they incorporate tangible physical laws into the simulation. As an object ages or wears, small pieces break off and fall to the side. Water tends to flow over a surface causing local staining and erosion. These physical properties increase a simulation’s realism.

2.3 Chemically-based Simulation Methods

Chemically-based simulation methods age materials by approximating the molecular changes in the atomical structure. These methods establish underlying chemical properties and change the material. Chemically based simulation rules also achieve some approximation of accuracy. Many chemically-based methods do not provide users with many controls. These simulations only let the user modify some initial conditions before running the simulation. Some types of chemical aging, such as patina, tend to strengthen and improve
Dorsey and Hanrahan [22] modeled and rendered a metallic patina. This work showed the patination process heavily depends on environmental factors. They integrated these parameters into their phenomenological model to produce a variety of results. An example render of their work is illustrated in Figure 2.2(a). Several other corrosion models have been proposed that extend this work to more general conditions. Merillou et al. [53] extended patina models to better simulate new forms of corrosion by using a better model of atmospheric condition predictions. Chang and Shih [13] modeled patina for Chinese bronze statues buried for thousands of years. Shahidi et al. [66] simulated efflorescence in fire-clay brick. They modeled the transport and crystallization of salts in materials. Merillou et al. [51] simulate the interactions between materials and pollution on buildings. They simulate color blackening and small scale geometric changes. Newer work focuses on how salt decay ages buildings and varies over time and space.

### 2.4 Organically-based Simulation Methods

Organically-inspired techniques attempt to simulate natural scenery based on interactions from living organisms. Most simulations account for the weather and environmental factors. These approaches grow biological phenomena. Desbenoit et al. [19] grew lichen in a virtual scene to help create outdoor environments. Figure 2.2(d) shows lichen growth simulated on a stone surface. The lichen grows to try to favor light and water in the scene. Yin and his colleagues [78] proposed a model for wood aging based on biological deterioration. Ogasawara et al. [57] simulate moss growth taking into account local temperature and humidity. Boissieux et al. [10] simulated skin aging and wrinkles based on insight from cosmetics. Wu et al. [76] simulated skin aging and wrinkling using B-spline patch muscles for contraction forces. Jiao and Wu et al. [37] perform multiple weathering effects on fur.
To our knowledge, little work has been done so far to simulate organically-based decay simulations, and no work (other than our own [39]) on aging and decaying fruit. The \(\mu\)-ton system is able to simulate organic growth propagation by spreading particles derived from the underlying biology. The \(\mu\)-ton system handles mold growth and fruit aging and presents a novel visual simulation technique that has not been covered in previous approaches by others.

### 2.5 Data-driven Simulation Methods

Data-driven simulation techniques rely on carefully captured images and measurements to produce time-varying surface imperfections. These methods avoid approximating physically based models or arbitrarily creating rules. Data-driven systems emerged as very promising techniques for automatic aging synthesis. They guarantee a higher degree of realism since they directly capture reality. Several techniques have been developed over the past few years that exploit a database of images, video, or precisely captured measurements. Many databases use a light dome to carefully capture temporal changes under a variety of lighting conditions. The data is simplified to a set of key frame images, or perhaps a bidirectional reflectance distribution function (BRDF). A technical artist manipulates the simplified data and the model’s parameters to control the simulation. Data-driven approaches tend to look more photorealistic out of all the approaches discussed.

Georghiades et al. [29] captured changes in particular materials over time. They utilize the collected data to produce models composed of similar materials and transfer the changes in the material appearance (Figure 2.3(c)). Lu et al. [46] developed spatio-temporal variations from wetting and drying. The captured data was reduced to a set of two parameters used to control wetting and drying on similar materials by a drying curve (Figure 2.3(b)). Enrique et al. [28] collected a small set of time-lapse controlled photographs (paint drying, grass growing, snow accumulating) to form a synthesis algorithm similar to image quilting [26] tailored to time varying textures. Gu et al. [32] acquired a
database of time-varying images using a multi-camera/light device scene in Figure 2.3(d). Their work looked at apples and banana skin captures. However, their approach simulates only the visual effect of the fruits’ aging over time and not the biological factors behind the process.

Wang et al. [72] use a single image to construct an appearance manifold to simulate aging phenomena. An appearance manifold is a visual simulation method for modeling the local distributions of weathering degrees of materials. The technique is a data-driven approach because it requires a single blemished data image and uses the single instant in time to calculate time variant and spatial variant surface imperfections. Xue et al. [77] improve the transfer of weathering from the manifold between images (Figure 2.3(a)). Our work differs from the appearance manifold approach since many phenomena age in complex stages. One image will not capture the dynamics of the entire aging process. Our technique models multiple types of appearance change on the surface and alters the underlying geometry. Lu et al. [45] captured and transferred weathering effects from real to synthetic shapes using a context-aware texturing approach. Sun et al. [68] address the acquisition, analysis, modeling, and rendering of time-varying BRDFs by a complex data acquisition method. Bosch et al. [11] recently proposed extracting parameters and detail maps from images to simulate and render flow phenomena.
Example-based texturing synthesis techniques generate new textures from an input texture data samples. There are many approaches that create larger textures on models from small input sources [49, 41, 42]. This work develops static textures that do not dynamically change over time. Matusik et al. [48] presented a data-driven model for isotropic BRDF functions based on measured reflectance data. Previously, BRDF functions only utilized physical or empirical equations to generate light reflection on surfaces.

Data-driven simulation methods have drawbacks. Acquiring sufficient spatiotemporal image data for several blemishes may require hours, days, or even years. For example, fruit ages in stages over the course of multiple weeks. The patina formed on a copper dome of a building takes years to form. This leads to the second problem of necessitating a comprehensive large database to accurately capture every blemish and multi-imperfection reaction. Most of this data is captured in very fixed laboratory conditions. These conditions may not mimic an outdoor and chaotic environment. Even moderately sized databases may overwhelm a technical artist. Therefore, we develop and utilize methods that simplify data and interpolate information between key points.
Chapter 3

Taxonomy of Aging Phenomena

In this chapter we discuss a taxonomy of aging phenomena that provides a general classification scheme. Aging and decay modify and decompose materials in a variety of methods. These aging methods consist of types of aging, aging properties, and material interactions. We categorize aging into five classifications: organic (biological), chemical, physically (mechanical), weathering, and environmental. Phenomena classes share certain important properties and characteristics. These classes define the type of interaction and mechanism of aging. No two substances age in the same fashion due to their own unique material properties, however most undergo the same broad chronological phases within a class. This classification helps to provide structure and general categories in which to group various aging phenomena so that their individual class properties and interaction may be better understood.

We also categorize these phenomena into three interaction types: additive phenomena, subtractive phenomena, and transformative phenomena. Additive phenomena insert new matter onto a material or into an environment. Subtractive phenomena remove matter away from a material or scene. Transformative phenomena change matter in some manner either at the physical or chemical level. These processes affect both the surface and substrate properties.
3.1 Organic Phenomena

Organic phenomena occur as a result of interactions with living organisms with the environment. These interactions may be physical and/or chemical. The organisms include plants, insects, microorganisms, diseases, fungi, and animals. Table 3.1 categorizes these phenomena. Some of these effects are additive. Moss, lichen, and fungi add new substances to the environment. Weeds and vegetation grow over time. Other effects are subtractive, e.g., insects or animals eating substances and mold or fungi consuming nutrients. Over time most organic substances decay and decompose as they age.

<table>
<thead>
<tr>
<th>Organic Type</th>
<th>Additive</th>
<th>Subtractive</th>
<th>Transformative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mold Growth</td>
<td>fruit, trees,...</td>
<td>X</td>
<td>fruit, trees,...</td>
</tr>
<tr>
<td>Fungus Growth</td>
<td>fruit, trees,...</td>
<td>X</td>
<td>fruit, trees,...</td>
</tr>
<tr>
<td>Rot Growth</td>
<td>fruit, trees,...</td>
<td>X</td>
<td>fruit, trees,...</td>
</tr>
<tr>
<td>Lichen</td>
<td>forest</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Moss</td>
<td>forest</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Algae</td>
<td>rocks</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wilting</td>
<td>water loss</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Diseases</td>
<td>leaves, plants...</td>
<td>leaves, plants...</td>
<td>leaves, plants...</td>
</tr>
<tr>
<td>Consuming</td>
<td>worms, bugs...</td>
<td></td>
<td>worms, bugs ...</td>
</tr>
<tr>
<td>Decompose</td>
<td>X</td>
<td>X</td>
<td>composite</td>
</tr>
</tbody>
</table>

Table 3.1: Organic phenomena classes of interactions.

While disease may strike substances at any stage of life, most substances naturally become more vulnerable to organic phenomena over time. Organic phenomena typically initiate when a point on the substance’s surface is compromised. The nutrients and proteins of the organic matter decompose into smaller organic parts. Once an initial infectious agent like bacteria has attacked, it further weakens the surface and leaves it susceptible to infection by other organisms. The broken-down matter serve as building blocks for other fungi and bacteria. This is the reason that mold often forms in areas already infiltrated by bacteria. The soft rot serves as an entry point for fungal spores. Examples include
Figure 3.1: Organic Phenomena: moss on a tree, algae on rocks, lichen on rocks, moss on a branch, rotten pear, tomato fungi, lemon mold, and noodles.

Penicillium digitatum, Penicillium expansum, and Rhizopus mold forming (respectively) on an orange, apple and tomato.

There is also a large amount of fruit-to-fruit spread of bacteria and fungus. Moisture often is trapped between the fruit making an ideal environment for rot and decay. As fruit puts pressure on other fruit, wounds and bruises start to form. These start to form clusters of decayed fruit and spread rapidly.

3.2 Environmental Phenomena

Environmental phenomena occur as a result of some interaction and movement in the environment. Table 3.2 categorizes these phenomena. For example, along roadsides and alleyways, litter tends to accumulate. Leaves fall off trees. Weeds grow. People mark walls and structures with words, images, and elaborate graffiti. Dust collects on objects. The majority of this class of effects are additive. New items join the scene, such as trash, leaves, and dust. Over time, more and more of these substances pile up. Environmental phenomena provide an important visual cue that an area is inhabited and used (or abused). These additive items also tend to move around the scene and accumulate in a non-random
Table 3.2: Environmental phenomena classes of interactions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additive</th>
<th>Subtractive</th>
<th>Transformative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic Growth</td>
<td>weeds, bushes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Trash accumulation</td>
<td>alleys, streets, rooms, ..</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Leaf, branch piles</td>
<td>forests, streets,...</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Dirt and dust</td>
<td>all objects</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Object movements</td>
<td>X</td>
<td>cracking, chipping...</td>
<td>X</td>
</tr>
<tr>
<td>Human paths / tracks</td>
<td>dirt carried</td>
<td>plant life destroyed</td>
<td>wear and tear</td>
</tr>
<tr>
<td>Graffiti</td>
<td>graffiti paint</td>
<td>X</td>
<td>surface color</td>
</tr>
</tbody>
</table>

Figure 3.2: Environmental Phenomena: dust accumulation, leaves, leaves and trash, trash accumulation, human paths, graffiti, and a abandoned shed.

manner. People will drop trash where they walk, and the wind will blow trash and leaves around based on the layout of the environment. These additive items do not scatter randomly, however they scatter based on the structure and purpose of the environment.

Not all the environmental phenomena add substances or materials into an environment. Some environmental phenomena subtract and transform the environment over time. Humans cause damage due to their interactions with environments. For example, crowds make paths over grass as shortcuts, and oil and tire marks accumulate on road surfaces. Shopping carts chip paint at particular levels and areas of supermarkets where they are
used. Street and sidewalks buckle and crack in high traffic areas. These subtractive and transformative environmental phenomena provide visual cues of environmental usage. Areas with more of these features show more activity and interaction.

### 3.3 Chemical Phenomena

Chemical phenomena occur by the transformation of the chemical bonds and structures of a material. The most common chemical decay reactions include: oxidation, carbonation, hydrolysis, and carbonic acid etching. Table 3.3 categorizes these phenomena. These phenomena primarily occur on the molecular level and produce a new substance. For example, during oxidation oxygen molecules and different materials interact. As electrons are added and subtracted as these substances interact new chemical species occur such as rust, patina, and anodized aluminum. Matter itself is never destroyed or added during chemical phenomena. Only the atomic structure is rearranged. These reactions change the density, bulk, and stress of objects. The new material may have a different phase. For example, two liquids may combine to form a gas. These changes may visibly alter appearance such as color and reflectivity, and also affect physical structure and integrity. These new materials may have dramatically different properties as they age, e.g., as items start to rust and leaves change color.

Often the new material has dramatically different material properties as they age. The material may grow stronger or grow weaker as it ages. For example, as items start to rust, the integrity of the metal is diminished and the object becomes weaker over time. In contrast, as copper starts to age patina forms a protective barrier which becomes stronger as it oxidizes and ages.

Chemical phenomena also include color changes: fruit turns brown, autumn leaves change color, and certain materials stain. Chlorophyll in leaves breaks down during autumn producing a variety of color changes. The sun sends out light and heat energy that is absorbed by materials. This process produces a chemical change and breakdown in the
colorants making many materials fade in color over time. Old objects appear much more faded than newer ones.

<table>
<thead>
<tr>
<th>Chemical Phenomena Classes of Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Corrosion</td>
</tr>
<tr>
<td>Oxidation</td>
</tr>
<tr>
<td>Fading</td>
</tr>
<tr>
<td>Hydrolysis</td>
</tr>
<tr>
<td>Hydration</td>
</tr>
<tr>
<td>Color Change</td>
</tr>
<tr>
<td>Combustion</td>
</tr>
<tr>
<td>Staining</td>
</tr>
<tr>
<td>Pollution</td>
</tr>
</tbody>
</table>

**Table 3.3:** Chemical phenomena classes of interactions.

**Figure 3.3:** Chemical Phenomena: rust formation from oxidation and leaf color change.
3.4 Physical Phenomena

Physical phenomena occur from some change in the object by entirely mechanical and physical methods. Forces may originate both within an object and external to an object. These forces include gravity, buoyancy, collisions, and pressure. Some examples of this class include: glass shattering, paint peeling, rope fraying, and metal bending. Table 3.4 categorizes these phenomena. Unlike chemical phenomena, the underlying atomic structure and chemical bonds remain unchanged with physical phenomena. Objects experience change by physical force, rather than molecular changes. Only the condition of the material changes. These phenomena might happen quickly, such as fractures, impacts and cracks, or more slowly, such as rock erosion, pressure or stress cracks, and (accumulated) abrasions. Changes in state or phase are also physical phenomena. Examples include: melting, freezing, and condensation.

These physical phenomena vary based on the material properties of the object. The composition and properties change the types of forces and interactions. For example, different materials have different tensile strengths. Ductile materials exhibit an elastic behavior under strain. Brittle materials fail at small strains. Therefore, over time the structure and the material properties of objects determine how they will decay.

Figure 3.4: Physical Phenomena: leaf crumbling, glass shattering, newspaper tearing, paint peeling, rock eroding, ice melting, paint chipping, and ice calving
<table>
<thead>
<tr>
<th>Physical Type</th>
<th>Additive</th>
<th>Subtractive</th>
<th>Transformative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cracking</td>
<td>X</td>
<td>X</td>
<td>stones, paints,...</td>
</tr>
<tr>
<td>Fracturing</td>
<td>X</td>
<td>X</td>
<td>stones, paints,...</td>
</tr>
<tr>
<td>Wrinkling</td>
<td>X</td>
<td>X</td>
<td>cloth, skins, paints...</td>
</tr>
<tr>
<td>Scratching</td>
<td>X</td>
<td>glass</td>
<td>X</td>
</tr>
<tr>
<td>Bending-warping</td>
<td>X</td>
<td>X</td>
<td>stone, metals,...</td>
</tr>
<tr>
<td>Chipping</td>
<td>X</td>
<td>paints, stones, ...</td>
<td>X</td>
</tr>
<tr>
<td>Peeling</td>
<td>X</td>
<td>paints</td>
<td>X</td>
</tr>
<tr>
<td>Punctures</td>
<td>X</td>
<td>X</td>
<td>clothes, metals, ...</td>
</tr>
<tr>
<td>Shattering</td>
<td>X</td>
<td>X</td>
<td>glass, ceramics,...</td>
</tr>
<tr>
<td>Bending</td>
<td>X</td>
<td>X</td>
<td>metals, plastics...</td>
</tr>
<tr>
<td>Curing</td>
<td>X</td>
<td>X</td>
<td>cements</td>
</tr>
<tr>
<td>Rip/Tear</td>
<td>X</td>
<td>X</td>
<td>cloth, paper...</td>
</tr>
<tr>
<td>Fraying</td>
<td>X</td>
<td>X</td>
<td>rope, string, ...</td>
</tr>
<tr>
<td>Abrasion</td>
<td>X</td>
<td>surface grinding</td>
<td>X</td>
</tr>
<tr>
<td>Erosion</td>
<td>X</td>
<td>soil, rocks, ...</td>
<td>X</td>
</tr>
<tr>
<td>Impacts</td>
<td>X</td>
<td>X</td>
<td>dents, impacts</td>
</tr>
<tr>
<td>pressure</td>
<td>X</td>
<td>X</td>
<td>sinking, compression,...</td>
</tr>
<tr>
<td>Drying</td>
<td>X</td>
<td>X</td>
<td>paint, clay, water spots</td>
</tr>
<tr>
<td>Melting</td>
<td>X</td>
<td>X</td>
<td>ice, metals...</td>
</tr>
<tr>
<td>Freezing</td>
<td>X</td>
<td>X</td>
<td>ice, metals</td>
</tr>
<tr>
<td>Wetting</td>
<td>water</td>
<td>X</td>
<td>slaking</td>
</tr>
</tbody>
</table>

Table 3.4: Physical phenomena classes of interactions.

3.5 Weathering Phenomena

Weathering phenomena occur from some change in atmospheric conditions. In this case, “weathering phenomena” is not a general phrase, but particularly refers to a class type. This class includes precipitation types: snow, rain, hail, and sleet. Table 3.5 categorizes these phenomena. Precipitation results when water droplets become too heavy to remain in the cloud and fall toward the earth. The various forms of precipitation can produce different interactions with the environment. Rain will flow water across the environment, and may also leave flow and stains marks behind. Sleet and hail might dent and impact surfaces. Snow accumulates and blows based on surface exposure. Snow may force bending...
and collapsing due to the weight during accumulation.

Weathering phenomena also refers to changes in temperature and humidity. Temperature directly impacts expansion and contraction of many objects. Over time, objects begin to warp and crack from the stress of heat and ice. Water falls into cracks on the sidewalk. The ice expands as the temperature falls below freezing and expands the cracks. Humidity affects moss, lichen, and fugal growth. As humidity rises, there is more moisture in the environment. This water provides ideal conditions for growth. Both temperature and humidity may curb growth of many biological substances at their extremes. This subtracts items from the environment as mold, fungus, and plant life die.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additive</th>
<th>Subtractive</th>
<th>Transformative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rain</td>
<td>water</td>
<td>erosion</td>
<td>flows, stains</td>
</tr>
<tr>
<td>Ice</td>
<td>ice</td>
<td>X</td>
<td>slippery surface</td>
</tr>
<tr>
<td>Snow</td>
<td>snow flakes</td>
<td>X</td>
<td>bending with weight</td>
</tr>
<tr>
<td>Hail - Sleet</td>
<td>hail and sleet pellets</td>
<td>X</td>
<td>impact dents</td>
</tr>
<tr>
<td>Sun</td>
<td>X</td>
<td>X</td>
<td>fading, buckling, peeling</td>
</tr>
<tr>
<td>Temperature</td>
<td>fungus, mold</td>
<td>curb growth</td>
<td>melting, freezing</td>
</tr>
<tr>
<td>Humidity</td>
<td>more fungus, mold</td>
<td>curb growth</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 3.5: Weathering phenomena classes of interactions.

The sun greatly affects the environment. Specifically, exposed surface colors start to fade from constant exposure from the sun. Paint starts to dry and peel from the increase in temperature. Water freezes and melts at different rates from the sun’s radiation and temperature changes. Plant life tends to grow where the sun shines (and mold grows where it does not) The structure of the environment defines what surfaces are exposed to the sun and provides important cues. Areas with an abundance of plant growth imply the area has high exposure to the sun. Minerals expand and contract by different amounts with the surface temperature fluctuations.
3.6 Multiple Interacting Phenomena

Decay phenomena involve a multitude of interactions occurring simultaneously. A piece of plastic left out in the sun will fade (chemical and weathering) and bend (physical). An autumn leaf will change color (chemical), rot (organic), collapse and wrinkle (physical), and be blown somewhere to decompose (environmental). Figure 3.6 shows how interactions can be additive, subtractive, and transformative, often all at the same time. Therefore, a complete system needs to account for the various decay classes, interactions, and transport as they mutate the environment.

Figure 3.6 illustrates examples of multiple interacting phenomena. As the building begins to decay, the wood starts to decay and flake away. Since there is high humidity, lichen and moss spores begin to take root. On the metal bar, the rust forms where the paint has chipped off. Here the metal starts to change composition and soon becomes weaker. The lemon grows blue mold, fungi, and darkens color over time. The old tree loses bark from disease, and has moss growing only in areas where the bark remains. Many additive, subtractive, and transformative changes occur simultaneously on these materials.
Chapter 4

µ-ton Simulation Framework

In this chapter, we describe the technique for µ-ton decay simulation. We present a novel approach for simulating the various categories of aging and decay that is grounded in scientific research. This method is a general attempt to simulate multiple interacting processes simultaneously. The µ-ton framework mutates the world. The simulation affects both the surface textures and underlying geometry. Figure 4.1 provides an overview of the µ-ton system framework. We directly compare our work to γ-ton tracing [16] to highlight the differences between the two approaches. Both µ-ton tracing and γ-ton tracing both resemble the classical approach of photon tracing and photon mapping. Many features of the µ-ton framework parallel features in these traditional photon based approaches.

4.1 µ-tons - Point Based Representation

The µ-ton framework utilizes a point-based representation (PBR) of the world. These points, or µ-tons, discretize the world. There are three main types of µ-tons: an emitter particle, a decay particle, and a material particle. At any point in time µ-ton particles may mutate into different particle types. The µ-ton particles contain a variety of attributes which define their type, decay class, interaction rules, and material properties. All the particles derive from the same class and implement the same interface. All particles must
Figure 4.1: This figure outlines the µ-ton tracing framework. The µ-ton tracing framework is similar to photon tracing, however it affects the underlying geometry and diffuses and reacts with the surface.

implement the following methods: compute, propagate, visualize, emit, and update. So when a decay phenomena is coded, this interface defines the interaction with the larger system. Algorithm 1 shows a sample implementation of the µ-ton class, and Algorithm 2 shows how the class is called. A new type of decay may extend the interface with novel interactions, or derive from a previous particle implementation with new features. For example one can make a mold µ-ton which implements the main µ-ton interface, and change a variety of attributes to get different effects. Then one may define different mold subtypes which extend the mold µ-ton class and perhaps define different interactions. Every particle is registered in the µ-ton system factory which processes all the particles at every time
step. Later, in chapter 5, we provide an example of setting up a mold substance in this framework.

A material \( \mu \)-ton particle represents a piece of material in or on an object. This comes in two subtypes: geometry and texture. A geometry \( \mu \)-ton represents a piece of geometry, usually a center point. The framework samples an object’s continuous geometry to a set of discretized points. Initially, there are only a few points, and these points are uniformly distributed and constructed with a 3D constrained Voronoi cell method. A texture \( \mu \)-ton is a point location on the texture which is mapped to an object. This type of \( \mu \)-ton is primarily a feature to increase performance. Some surfaces have many very thin layers, for example a surface with many layers of paint. A user may compose very thin layers of geometry \( \mu \)-tons or utilize texture \( \mu \)-tons to represent these layers. These material particles contain the material properties for the object. This is a more powerful approach than just composing shaders (a la Rendeman) for layer effects.

A decay \( \mu \)-ton particle represents a specific decay type. These particles propagate through a scene producing various aging phenomena, and are the main driving force of the method. These particles carry the type of decay, attributes for the decay process, interaction rules, and motion information. A \( \mu \)-ton source emitter creates a set of new \( \mu \)-tons. This feature is covered in detail in Section 5.2 below. Emitter \( \mu \)-tons contain attributes that describe what \( \mu \)-tons should be emitted, how often they should be emitted, and when they should decay. The emitters may be set up in a variety of different configurations. An emitter may also exist on the surface or interior of an object if the aging phenomena needs to diffuse. Two examples are mold propagation or rust weakening an object.

The last feature \( \mu \)-ton particles contain are constraints. These are springs that connect certain material \( \mu \)-ton points in various lattices. These \( \mu \)-ton material particles represent the underlying mesh and usually share a face between the constraint. Our system solves physical or chemical reaction equations and modifies the constraint forces. The constraint
Algorithm 1 $\mu$-ton Representation

$\mu$-ton class{

Attributes:
- $c_r$, $c_g$, $c_b$, $c_a$ // color attributes
- $a_1$, $a_2$, $a_3$ // decay $a_1$ attributes
- ... // other attributes for the emitter, material, and decay particles

Motion Trajectory:
- $u_x$, $v_y$, $w_z$ // motion velocity
- $x_x$, $y_y$, $z_z$ // positional information

Constraints:
- $C_{vector}$ // vector listing constraints

Compute(); // compute the $\mu$-ton decay process
Propagate(); // move the $\mu$-ton (does not have to move)
Visualize(); // visualize the $\mu$-ton
Emit(); // emit $\mu$-tons if this is an emitter
Update(); // update the $\mu$-ton attributes, trajectory, and constraints
}

Algorithm 2 $\mu$-ton Update

Result: Propagate $\mu$-tons and simulate interactions

initialization();

for all $\mu$-tons in the $\mu$-ton system factory do
read current $\mu$-ton: ($M_n$);
$M_n$.Emit();
Propagate();
CheckIntersections();
UpdateDistributionMaps();
$M_n$.Compute();
ComputePhysics();

if materialChange then
    UpdateGeometrTexture();
    MutateGeom();
end if

$M_n$.Update();
VisualizeChange();
end for
force specifies when an material $\mu$-ton will mutate, collapse, or break off. The particle-constraint representation lets the $\mu$-ton framework simulate impact forces and various decay phenomena. We demonstrate this approach in the fruit decay where a deformable object shrinks and collapses to simulate the internal substrate. Rot affected various spring constants at different rates causing local collapses and wrinkles. Finally, we demonstrate different materials chipping and breaking off.

### 4.2 $\mu$-Ton Source Emitters

A $\mu$-ton’s life begins at the $\mu$-ton source emitter. Since the environment is a point based world, every $\mu$-ton has the ability to mutate into an emitter particle. For every $\mu$-ton source in the scene, the emitter creates a set of $\mu$-tons. A $\mu$-ton emitter divides the overall power of the decay effect amongst the child $\mu$-tons and casts them into the scene until the emitter loses all its energy. $\mu$-ton emitters with more energy release more $\mu$-tons then dimmer emitters. The aging parameter may change if the emitter receives an additional intersection with an object, and the material $\mu$-ton transfers energy to the decay particle. Energy may be subtracted or added. Emitters also have a time associated with them. This determines how often they should emit $\mu$-tons into a scene. Some effects, such as rain, emit $\mu$-tons every frame. Other effects, such as a leaky pipe, emit $\mu$-tons at a predetermined rate.
There are several types of \( \mu \)-ton source emitters: point, directional, spot, area, volume, and ambient. Figure 4.2 and Figure 4.3 illustrate the various types of emitters. A point source emits \( \mu \)-tons evenly in a uniform pattern in all directions. Emitters may emit \( \mu \)-tons uniformly, based on a distribution function, or use some type of sampling method to randomly distribute particles. A directional emitter casts \( \mu \)-tons in a particular direction with a specified size. A spot emitter casts \( \mu \)-tons into a scene defined by a cone. Within the cone’s specified range, \( \mu \)-tons are cast evenly. An area source emits \( \mu \)-tons based on some shape and size in all directions. A volume emitter has a visible range of influence. The \( \mu \)-ton decays linearly from the emission point to the specified volume boundary. Ambient sources are complex and powerful emitters that cast \( \mu \)-tons on all surfaces. This may perpetuate some global and general change into the scene. This type of emitter is used sparingly to mimic air pollution and other general effects. The \( \mu \)-ton emitter types parallel lighting design on purpose. Since \( \mu \)-ton tracing directly parallels photon tracing, these emitter types provide artists with a similar tool to creatively define a scene’s mood by adjusting \( \mu \)-ton emitters in the same fashion they would adjust the lighting of a scene. The purpose of emitters is that they are extendable to the needs of the user. So new types of emitter objects may be added in the future, and will be treated under the same set of rules.

\[ \text{Figure 4.3: This figure illustrates the different types of } \mu \text{-ton emitters: (a) point, (b) directional, (c) spot, (d) area, (e) volume, and (f) ambient.} \]

\( \mu \)-ton emitters may be specified by a user to start the decay process. For an example, a user may set up an area hemisphere emitter to cast \( \mu \)-tons into a scene. A user may set up a directional emitter to simulate a leaky pipe. Additionally, \( \mu \)-ton intersections automatically may create \( \mu \)-ton emitters. For example, during a collision a \( \mu \)-ton emitter casts \( \mu \)-tons into the object which in turn is based on a Box-Muller transform random
number distribution. This distribution of $\mu$-tons may be uniform, around a point, around the shape of a line, on a shape of a plane, or around/on the shape of a mesh depending on the collision type. Another example is during corrosion; ambient air $\mu$-tons and directional water $\mu$-tons interact with the metal surface and cast new $\mu$-tons into the object which will cause a chemical reaction. This allows corrosion and other decay effects to simulate multi-layer and complex interactions.

Figure 4.2 illustrates a few sample users setting up a variety of different emitters inside the $\mu$-ton Maya plug-in. In the first image, the user creates a piece of geometry to surround an object and cast $\mu$-tons into the scene. The second user just used a simple plane to achieve a rain effect. The third user wanted to create a hose to simulate water gushing out of a pipe. The system is robust enough to support a variety of different emitters to suit production needs.

Figure 4.4: This figure illustrates material $\mu$-tons scattered inside an object. The left image shows there is a single decay $\mu$-ton (yellow) above the cube, the decay $\mu$-ton intersects the geometry forcing local re-meshing (middle), and the newly emitted $\mu$-tons are shown on the right. This is another type of emitter.

Lastly, emitters may form inside an object. Figure 4.4 illustrates after a $\mu$-ton has hit the surface of an object it radially ($r$-spread) scattered $n$ more $\mu$-tons into the object. This subdivided the object, and constants and spread pattern are different based on the aging effect and material properties. For example, white mold spreads differently on citrus fruit than on non-citrus fruit. Rust has different effects based on the underlying material
type. Such dependencies improve the efficiency of the system. Since the original object only needs as many material $\mu$-tons to define the object’s strength and material properties. These other material $\mu$-tons are added later down the simulation pipeline to provide local detail where needed, and add detail only where there is change.

Figure 4.4 shows a simple cube whose geometry has 8 voronoi regions and each with a single $\mu$-ton particle. As the particle collides with the mesh in the middle, more $\mu$-tons are emitted into the voronoi region, in this case at random. The figure on the right shows just the $\mu$-tons in the scene. These parameters can be adjusted to determine how to cast $\mu$-tons into geometry based on a user’s needs.

4.3 $\mu$-ton Propagation

The emitted $\mu$-tons are scattered and travel within the scene. At every time step, $\mu$-tons update their motion and carrier attributes. Eventually, the $\mu$-tons are absorbed or decay. The previous gamma-ton tracing method [16] only accounted for four possible motion types: reflect, bounce, flow, and settle. Additionally, these types are stochastically determined and are not determined by the phenomena type. $\mu$-tons propagate in more complex motion patterns. For example, fluid $\mu$-ton particles follow a Lagrangian smooth particle hydrodynamic (SPH) method for simulating flow. $\mu$-tons that propagate into geometry due to an impact or decay effect follow different strain tensors and material properties. Organic $\mu$-tons in fruit follow the reaction-diffusion equations. These effects account for the particular decay class and material properties of the scene. Organic particles follow organic trajectories based on diffusion. The particles follow a physically based simulation. $\mu$-tons exploit the scene and material structure to better propagate $\mu$-tons.

At anytime in the simulation, the user may change how $\mu$-tons scatter and propagate. If the user wants to add randomness, they may easily incorporate the 4 possible motion types of $\gamma$-ton tracing. They may also want to define new simulation types for their novel phenomena. The $\mu$-ton propagation framework is a general framework that attempts to
match a $\mu$-ton’s path with some scientific foundation. This framework may be modified to produce a desired look, however the scientific grounding for the aging effect is relaxed.

The majority of $\mu$-ton particles integrate the equations of motion using Runge-Kutta 4 (RK4). The RK4 integrator samples derivatives at several points per timestep. This solves the intimal value problem (Equation 4.1) set out in a fashion similar to Baraff and Witkin’s method for solving particle systems using RK4 [74].

Figure 4.5 shows a sample emitter object and a bunny in Maya. The yellow points represent the traveling $\mu$-tons in the scene, which are simply falling towards the ground from gravity using the RK4 particle system. The intersection points are represented by red points in the space in the right image. Here the $\mu$-tons flow over the bunny, and bounce on the ground which has been represented with a more elastic material.

4.4 $\mu$-ton Intersection

$\mu$-tons propagate throughout the environment until they interact with an object. The simulator must detect a collision, then must respond in a realistic fashion. Collisions are detected by checking the position of a particle against the plane made by the polygon similar to Baraff and Witkin’s method [74]. If $P$ is a point on the plane, $N$ is the normal, then
the sign of

\[(X - P) \cdot N\]  \hspace{1cm} (4.1)

determines whether there is a collision of a point X with the plane.

Once a collision is detected the system must respond to the collision. The collision response depends on the material and decay properties. For example, the particle may be absorbed by the material particle, and turned into an emitter (in the case of an impact collision). This casts particles into an object based on the material’s strain tensor to simulate an elastic collision. Some \(\mu\)-ton particles are reflected away from the object’s intersection point. Here the velocity and force vectors are negated. The user programs the proper collision response based on the type of decay and material properties of the object.

\[\text{Figure 4.6: This figure shows a simple k-d tree the } \mu\text{-ton system uses to partition space to accelerate intersection queries and to find the nearest material or surface } \mu\text{-tons in a scene.}\]

To accelerate intersection tests and point queries the triangle mesh and 2 triangles for the ground plane use a k-d tree acceleration structure. A simple k-d tree is shown in Figure 4.6. This partitions the face in an axis-aligned binary search tree way.

\section*{4.5 \(\mu\)-ton Distribution Maps}

During the propagation process, \(\mu\)-ton particles settle throughout the scene. Distribution maps are a sequence of time-varying maps which account for the \(\mu\)-ton distribution. The
distribution maps range $X$ between $[0 ... 1]$, with white representing a maximum density of 1, and black representing a density of 0. These maps guide the formation of decay processes. These maps may be stored as 2D textures, or the distribution maps may be a collection of points in 3D space (a point cloud). These 3D distribution point maps are stored in the $kd$-tree. Each node of the $kd$-tree stores the information for a $\mu$-ton intersection $(x, y, z)$ and a pointer to the intersecting $\mu$-ton particle.

Distribution maps may be changed by a variety of means, but their main utility is to provide a data-structure to capture the time-varying aging processes in the simulation. One approach is $\mu$-ton particles intersect with an object and change the underlying distribution maps and $\mu$-ton material particles. Another approach is lighting calculations that directly change the underlying distribution maps. For example, the ambient occlusion (or other
complex shadow methods) is baked into a scene. This approximates the way light radiates and creates shadows. Ambient occlusion is often related to accessibility shading. Additionally, a normal map may be baked into a scene. This information provides how likely certain aging phenomena flow or stick to a surface. Using ambient occlusion, upward facing normals, and the normal map provide information to quickly find tops, corners, and crevices. These three features are critical structural features in aging. Figure 4.8 shows some example distribution maps.

The key feature of the distribution map data structure is that these maps govern the interactions between aging effects. A new set of distribution maps is generated and stored at each frame. Before a $\mu$-ton particle interacts with the scene, the particle uses its own attributes, the attributes of the interacting $\mu$-ton particle, and the value of the distribution map to govern the interaction. This method allows multiple phenomena to efficiently
interact. In the case of fruit aging, nutrient distribution map levels deplete over time as fungal colonies grow and use up resources, slowing down the growth process.

**Figure 4.9:** This figure breaks down 3 pieces of geometry into their voronoi regions and gives each a unique random color for visualization. Different µ-ton particles are scattered in the space and store various material properties at every location for the cell.

### 4.6 µ-tation (Mutation)

#### 4.6.1 Geometric Material Update

A key feature of our approach is the geometric µ-tation process. The µ-ton framework mutates the underlying geometry during the decay simulation. As objects age their geometric material properties change. Fruit gets softer and loses volume. Copper becomes stronger as it forms a patina. Paint chips off from heat and environmental use.

The initial condition of our geometry is to uniformly scatter a few material particles inside the volume of the initial object. (In practice, this has been a triangulated polygonal object.) There can be many particles or a few particles, depending on the production needs of the user, and the amount of detail the user wishes to provide for the object. To do this we first calculate the bounding box for the object. This is found by looking for the minimum and maximum points of the vertices in the storage vector. We add a minor offset to the bounding box dimensions (usually 0.001) in all directions. We then cast rays from the surface of the bounding box. The rays can be cast randomly biased by some distribution
or based on user-input. When there is an even number of intersections, the positions on
the line-segment between the intersection points are randomly sampled. (An even number
of intersections ensures the line-segment is inside the object.) This provides us the volume
samples inside the object. Figure 4.4 shows some randomly scattered material particles
inside a cube.

Every intersection point - from the ray casting step in the bounding box volume -
also provides the framework with random surface samples. Figure 4.7 shows examples of
different sampling methods. The left image is biased to form more samples on the legs
(which may rust), while the one on the right has a uniform distribution of samples. To
simulate complicated surface effects, one may add thin layers of points for paint or other
thin films, or use a multi-layer texturing approach of various distribution maps to store
aging phenomena information per pixel.

The volume material particles represent the center points for a Voronoi representation
for the geometry. Figure 4.9 shows 3 different objects with random particles scattered
inside the objects, and their subsequent Voronoi regions. To efficiently calculate this, the
μ-ton framework calculates the Voronoi regions using the bounding box information from
the previous step. This step calculates the Voronoi regions inside a cube. A cube has more
trivial cutting planes and requires less computation. A boolean operation (intersection) is
used on each Voronoi cell and the original object. This yields the additional cutting planes
to clip the Voronoi regions of the object, and in general is much more computationally
efficient than calculating the Voronoi region of (possibly) concave geometry, such as the
bunny object in Figure 4.9.

Figure 4.11: This figure illustrates two simple Voronoi cells on a sphere and a constraint
on the neighboring face.

We connect these particles together and form a constraint lattice similar to Smith et al.
[67] of the connected faces. The constraint map is simply the Delaunay representation of
the Voronoi center points. The Delaunay and Voronoi representations are geometric dual
dual of each other. Figure 4.10 visualizes this representation, and shows some simple cells with
the constraints that connect them together.

The underlying lattice provides constraints, or the glue, that holds the various pieces
of geometry in place. These constraints initially are very stiff springs. These connections
represent the strength of the bond inside the material. For example, metal would have
stronger connection strengths than glass. So upon an impact, the metal is less likely to
shatter then the glass. Various aging effects modify the spring constants of the material
connections. Once a particular strain has been reached, the $\mu$-ton breaks all connections,
Figure 4.12: This figure illustrates the geometric update that happens upon a change occurring in the material. Notice how the top of the right image has more material $\mu$-tons and therefore more local tessellation only where aging is occurring.

and may drastically change position based on the physically based rigid body simulation. A $\mu$-ton may also break connections if a chemical reaction weakens the spring constants. Various material strengths are determined from the material science literature to provide a scientific basis for the material interactions.

The last major step to mutating the geometry happens after $\mu$-ton decay particles hit the surface, update the distribution maps, constraint map, and volume particles enough times to cause a change. Once this happens, as each cell changes $\mu$-ton material particles are emitted in the changed cell to further subdivide the regions. Figure 4.12 shows single $\mu$-ton particles hitting a cube like rain, and the mesh mutating over time as the processes age the object. The amount of times to subdivide an object can be tuned based on the strength properties of the material being simulated.

### 4.6.2 $\mu$-ton Particle Attribute Update

$\mu$-tons will mutate during decay processes and interactions. At a simple level $\mu$-tons can change types from a decay particle to a material particle to an emitter particle. For
example, upon a collision the decay particle may turn into an emitter and propagate new particles to simulate cracking. At every time-step certain attributes get updated, such as the motion trajectory information. Also the amount in the decay attributes of the particle may change as the \(\mu\)-ton acquires or loses carrier attributes during transportation. This change occurs due to collisions and interactions.

In the \(\mu\)-ton framework, update is not synonymous with change. \(\mu\)-tons may receive an update but not actually change any information. This happens with many material \(\mu\)-tons. They remain static until other substances produce a reaction. The key point with the \(\mu\)-ton framework is that change can occur at any point and at any time-step. This follows the same complexity of a normal point-constraint system. This framework provides a general approach to capture all types of aging phenomena and interactions.

### 4.6.3 \(\mu\)-ton Material Properties

There are a variety of material properties which need to be accounted for to produce effective decay phenomena. Physical properties include: density, mass, state change temperatures. Mechanical properties include: elastic modulus, viscosity, shear modulus, strength
properties, and elongation. Thermal properties include thermal expansion and conductivity. Table 4.1 lists just a couple of these properties for a few materials. Using these various properties obtained from the literature helps different materials break, melt, and decay in different ways. All objects decay in different ways; accounting for the various material properties during the decay effect ensures some degree of physical validity to the simulation. This simulation views an elastic solid as a bundle of ideal springs with constraints whose constants are derived from material science.

Later chapters describe examples for fruit, mold, corrosion, and some toy examples which use a variety of these (and other) properties.

<table>
<thead>
<tr>
<th>Material</th>
<th>Density $kg/m^3$</th>
<th>Melting Point (K)</th>
<th>Boiling Point (K)</th>
<th>Elasticity Modulus (GPa)</th>
<th>Thermal Expansion ($/K$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aluminum</td>
<td>2700</td>
<td>933.47</td>
<td>2792.15</td>
<td>62.053</td>
<td>2.310</td>
</tr>
<tr>
<td>Copper</td>
<td>8960</td>
<td>1357.77</td>
<td>2835.15</td>
<td>110.316</td>
<td>1.650</td>
</tr>
<tr>
<td>Iron</td>
<td>7870</td>
<td>1811.15</td>
<td>3134.15</td>
<td>193.053</td>
<td>1.180</td>
</tr>
<tr>
<td>Tin</td>
<td>7260</td>
<td>505.08</td>
<td>2875.15</td>
<td>41.369</td>
<td>2.200</td>
</tr>
</tbody>
</table>

Table 4.1: This table provides some basic material properties for a variety of different materials. [2].

Figure 4.14 shows various examples of $\mu$-ton material property updates. The $\mu$-ton decay particles are visualized in yellow, and the material $\mu$-ton particles start all black. As the properties change, the color turns to white to illustrate an aging effect has been transported. There can be an immediate jump or gradual jump based on how the user tunes the amount parameter on the decay $\mu$-ton particle. Figure 4.14 also shows material being moved around, as the surface material $\mu$-tons are transported throughout the scene. $\mu$-tons have a size associated with them. This size parameter determines the sphere of influence for the decay effect. Therefore, one may use fewer $\mu$-tons in a scene to speed up calculations.
Figure 4.14: This figure shows a variety of material updates. The top row shows the closest µ-ton material particle updating with an aging process update and turning white. The top row left, shows a more gradual update. In the middle row the size of the µ-ton decay particle is changed so it has a higher sphere of influence than the top row. The middle row right shows aging phenomena being carried away from the intersection area. The bottom row shows a different color representation of many interacting effects.

4.6.4 Constraint Simulation

The constraint simulation strives to simulate some basic fundamental mechanical and physical properties of materials. Material strength and ductility depend on the forces which bind atoms. Stress-strain curves, such as Figure 4.16, provide the correlation between stress (the load applied $\sigma$) vs. the strain (the deformation $\epsilon$). Each material has a unique curve based on their material properties. Young’s modulus quantifies the stiffness of elastic materials. Young’s modulus is derived by the ratio of stress vs. strain where
Figure 4.15: This figure illustrates the comparison of different materials’ Young’s Modulus and density [43]

Hooke’s law applies. \( \text{stress}(\sigma) = \frac{\text{force}(F)}{\text{area}(A)} \) and \( \text{strain}(\epsilon) = \frac{\text{elongation or compression}(\Delta L)}{\text{length}(L_0)} \) [43]

Young’s modulus is defined by:

\[
E = \frac{\sigma}{\epsilon} = \frac{FL_0}{A\Delta L}
\]  
(4.2)

Now Hooke’s law may be applied to this formulation:

\[
F = -\left(\frac{EA}{L_0}\right)\Delta L
\]  
(4.3)

As mu-tons propagate through the scene they change the various material property attributes. Only the mutating springs and mu-tons are updated in the physically-based simulation. When constraints are very stiff they do not require numerical integration.
They may, however, fracture and break due to the stress and strain calculations. This produces a variety of local aging effects grounded in material science.

**Figure 4.16:** This figure illustrates the relationship between stress and strain, and where the fracture point occurs. [43]
Chapter 5

A $\mu$-ton Tutorial Example

In this section we present what steps are required to model and simulate decay phenomena listed in Chapter 3 (Taxonomy of Aging Phenomena) using the $\mu$-ton framework. This tutorial shows the detailed steps a user needs to get their desired effect. Technical directors can do what is needed to add a wide range of aging effects into a production scene. Also, this begins to validate our model’s use as a general framework since we can produce numerous effects.

5.1 Setting up an Aging Effect

To demonstrate the generality of the $\mu$-ton framework, we set up a simple wall and propagate the growth of mold, a scenario similar to the fruit decay work (Appendix A). We have built an Autodesk Maya plug-in to assist the user in creating the aging effect that they want in their custom scene. In Figure 5.1, the user sets up the $\mu$-ton emitters (here a hemispherical source), as well as the object to attach material $\mu$-tons (the prompt currently shown, here we just randomly scatter nutrients and water). A few volume material $\mu$-tons are scattered and the Voronoi regions are calculated. The constraint map is formed by then taking the dual. The hemispherical $\mu$-ton emitter sends decay $\mu$-tons into the scene with the following aging attribute values:
Algorithm 3 $\mu$-ton material and decay attributes

Decay $\mu$-ton

\{
  c_r = 1; c_g = 1; c_b = 0; c_a = 1;  \quad \text{// Yellow Color for visualization}
  p_x = 0; p_y = 0; p_z = 0;  \quad \text{// Initial position set uniquely when emitted}
  v_x = 0; v_y = 0; v_z = 0;  \quad \text{// Initial velocity set for each particle}
  f_x = 0; f_y = 0; f_z = 0;  \quad \text{// Initial force acting on the particle}
  size = 5;  \quad \text{// Sphere of influence the particle has}
  mold_name = black mold  \quad \text{// mold type being transported}
  \sigma_1 = 0.01;  \quad \text{// scaling parameter}
  \sigma_2 = 5.0;  \quad \text{// scaling parameter}
  \sigma_3 = 5.0;  \quad \text{// scaling parameter}
  u_0 = 1.0; v_0 = 1.0;  \quad \text{// active and inactive fungus}
  d_1 = 0.01; d_2 = 0.01;  \quad \text{// model parameters for diffusion}
  f_1 = 1.0;  \quad \text{// model parameters for active fungus}
  f_2 = 1.0;  \quad \text{// model parameters for inactive fungus}
  \theta = 0.05;  \quad \text{// constant spread and nutrient consumption rate}
  a_1 = 0.000417;  \quad \text{// model parameter for diffusion of bacteria}
  a_2 = 0.00833;  \quad \text{// model parameter for diffusion of bacteria}
  D_n = 0.000001;  \quad \text{// nutrient diffusion coefficient}
  mold = 2;  \quad \text{// Amount of mold being transported}
  c_r = 0.2; mc_g = 0.1; mc_b = 0; mc_a = 0.1;  \quad \text{// mold color}
  c_r = 0.2; bump = 0.1;  \quad \text{// mold bump map parameter (for rendering)}
  water = 0;  \quad \text{// Amount of water transported}
  nutrients = 0;  \quad \text{// Amount of water transported}
  energy = 3  \quad \text{// impacts before settling}
\}

Material $\mu$-ton

\{
  \text{nutrients} = 0.5  \quad \text{// current amount of nutrients}
  \text{water} = 2  \quad \text{// current amount of water}
  \text{mold}[] = 0,0...  \quad \text{// current vector of mold pointers}
\}
Figure 5.1: This figure shows our Maya plug-in in action: a hemispherical $\mu$-ton source, decay $\mu$-tons visualized in yellow, and the plane to age selected in green.

These effects can be authored directly in code by extending the classes in Chapter 4 or processed into the system through a decay.muton and material.muton file reader via the plug-in. The phenomena is transported from the decay $\mu$-ton to the material $\mu$-ton with the following transport rule when the decay particle finally settles onto the surface:

$$Material \mu ton.mold[0] = Material \mu ton.mold[0] + 0.05 \times decay \mu ton.mold; \quad (5.1)$$

Then at every time-step when the decay $\mu$-ton has settled onto the object, we spawn new colonies of mold. The first step is to search the k-d trees for the 5 nearest (model parameter above) material particles and set their transport rules. The bottom row of Figure 5.2 shows the first decay $\mu$-ton hits. We are simulating 3 types of mold. We calculate the reaction diffusion step which follows the following equations:

$$\frac{\delta u}{\delta t} = \nabla \cdot (D_c \nabla u) + \theta f(u, n) - a(u, n)u \quad (5.2)$$

$$\frac{\delta v}{\delta t} = a(u, n)u \quad (5.3)$$
Then the following equations use the kinetic function described in detail in Appendix A to model the reaction of the nutrient map and the active mold growth. Here the nutrient concentration is given by \( n \). \((u + v)\) is the total density of the active \( u \) and inactive \( v \) fungus. \( D_c \) is the diffusion coefficient. \( d_1, d_2 > 0 \) are model parameters and \( \sigma_1, \sigma_2 \) and \( \sigma_3 \) are scaling parameters.

\[
D_c = s \cdot \sigma'_1 \cdot d_1 u \left( 1 - \frac{u}{d_2} \right) \cdot n \tag{5.4}
\]

\[
\sigma'_1 = \sigma_1 (1 + \delta), \quad -0.5 < \delta < 0.5 \tag{5.5}
\]

\[
f(u, n) = \sigma_2 \left( \frac{f_1 n}{1 + f_2 n} \right) \cdot u \tag{5.6}
\]

\[
a(u, n) = \sigma_3 \cdot \frac{1}{\left( 1 + \frac{u}{\sigma_1} \right) \cdot \left( 1 + \frac{n}{\sigma_2} \right)} \tag{5.7}
\]

\( \sigma'_1 \) is a random value to produce different growth patterns around the nuclei growth site. Here \( s = \text{porosity} + \text{temp} + \text{humidity} + \text{waterContent} \) produces the growth rate. More details for setting up these organic equations for mold and fungus can be found in Appendix A and in Kider et al. [39].

The next step is to make whatever geometry changes that are required by the aging effect. We search the volume k-d tree and set the material properties for the volume material \( \mu \)-tons. The geometry is re-tesselated as needed, and if the strain-stress threshold is reached, the geometry will break its constraints.

When creating a new aging effect, the user would specify their own growth patterns if they want the effect to propagate on impact of a decay particle. This does not need to happen for every particle. For example, the water \( \mu \)-tons only transport water around and in this example do not actually spread the amount of water to neighboring material \( \mu \)-tons.

The user also specifies how the decay \( \mu \)-tons move in a scene. In the mold example, particles move with an RK4 integrator. The motion may be defined in a variety of ways.
For example, $\gamma$-ton tracing shoots particles in a straight line from their emitters. This process can be captured with our system as well. An application might call for more flow movement, and a user can change the particle propagation class in the plug-in to follow a smooth-particle hydrodynamic approach to simulate fluid flow.

The last parametrization determines how quickly the $\mu$-tons will settle into the scene. This is the energy parameter in the decay $\mu$-ton example above. This number determines how many bounces the $\mu$-ton can make before settling in our system. $\mu$-tons, in general, will randomly settle over time. This parameter effects the probability of how quickly settling will happen. The longer a particle moves around a scene, the greater the chance that the aging will be transported to an area with low surface accessibility.

In this example, we showed that the $\mu$-ton framework effectively generates a varied set of aging effects. This process does not require a great amount of user effort, since they only author decay.muton and material.muton files, create and select the geometry in Maya, and use our plug-in to create new effects.
Figure 5.2: This figure shows the various stages of the mold / wall example. On the top left is the nutrient map which was created with 2 passes of a noise function, and the top right image is the distribution map calculated from the $\mu$-ton hits. The second row shows the various stages of the $\mu$-ton hits. The left image shows the intersection maps show the first time $\mu$-ton decay particles interact with the wall, the right image shows the beginning of growth from the reaction-diffusion process, and the left bottom image is the final growth pattern. Finally, the lower right image is the final result rendered onto the wall.
Chapter 6

Inorganic Aging

In this chapter, we describe the technique for aging inorganic materials. We illustrate how the $\mu$-ton framework is able to mutate geometry over time due to chemical or physical changes. We provide some examples of changing the strength bonds on the constraint lattice to simulate different materials. Finally, we set up and demonstrate concrete crumbling and pitted corrosion example to illustrate the technique on inorganic materials.

6.1 Geometric Decay

A major transformational process in inorganic materials is the mutation of the underlying material structure. From oxidation which forms patina on cooper surfaces, to concrete crumbling from stress, all materials change their structure and shape. The $\mu$-ton framework captures many of these chemical and physical phenomena. As $\mu$-tons spread in the scene they bring inorganic aging effect properties with them. These properties spread to the underlying material $\mu$-tons and produce change. When the lattice is weakened between neighboring Voronoi cells, pieces begin to crumble off (then the separated pieces can be sent to an underlying rigid body system to simulate the falling debris).

Figure 6.1 shows a street curb scene in the $\mu$-ton framework. Here we demonstrate multiple geometric mutation effects at once. The decay $\mu$-tons first hit the curb. The curb
Figure 6.1: This figure shows a more complex street scene from an alley way. Here there are multiple geometry mutations happening at once. The curb is breaking off, pieces are falling to the ground, and the paint on the road is chipping and peeling away.

then crumbles after prolonged exposure to the aging elements. These pieces fall to the ground. As time proceeds forward paint on the ground also starts to peel and chip away (Figure 6.2). The paint forms a thin rigid film layer on top of the street. To understand the process of chipping geometry better Figure 6.3 shows a toy example of a cube with 10 Voronoi cells. After every decay $\mu$-ton impact (red point) a piece breaks its lattice and is removed. This process would slowly start to erode the geometry over time. The $\mu$-ton system can subdivide the geometry as shown in the toy example in Figure 4.12. This allows for more localized chipping in areas of higher stress. Pieces are finally broken off when the stress-strain threshold has been reached as scene in Figure 4.16.
6.2 Simulating Different Materials

The $\mu$-ton framework allows for the simulation of different material properties. This is achieved by changing the underlying parameters for the Young’s modulus of elasticity. As this parameter is varied across the material, the simulation produces a variety of different results. This allows artists to achieve the effect they desire or some semblance of a realistic material - if the values are derived from material science literature.

Figure 6.2: This figure shows a more detailed view of the street scene from an alley way. Here there are multiple geometry mutations happening at once. The curb is breaking off, pieces are falling to the ground, and the paint on the road is chipping and peeling away.

Figure 6.3: This figure shows a very simple cube with only 10 voronoi regions. The geometry starts to erode away from multiple decay $\mu$-ton hits.
Figure 6.4: This figure shows different runs of the µ-ton framework on a cube (top left). The constraint strength parameters are changed in the various simulations which lead to a wide range of mutation effects.

Figure 6.4 demonstrates different material property settings in the µ-ton framework. The figure shows a test cube in the upper left corner, and five runs of the same simulation setup with only the constraint values changing based on different strength properties of the Young’s elastic modulus for different material types. (In this example we just arbitrarily changed the constants to show the range of effects that can be produced by the µ-ton system). All the simulations ran for the same time period, and had the same decay µ-ton cast. Only the second image on the top row had a second emitter to try to reproduce a decay effect of a softer wall crumbling in the middle.

6.3 Corrosion Example

Corrosion is a chemical reaction that plays a key part in metallurgical failure. Corrosion is a very deep and studied subject, and this section only breaks the surface of types of corrosion simulations that are possible with the µ-ton framework. In this section, we attempt to simulate one particular type of corrosion. Pitted corrosion is a localized attack that forms
holes in the underlying metal. The pit grows based on the environmental conditions, material properties, and object stress. A reference example image of this aging effect is shown in Figure 6.5. This image shows the extremely localized attacks that occur. Pitting corrosion initially starts due to a variety of reasons: flaws, damages, dents, dislocations, and underlying atomic structure differences. All these effects share a lack of uniformity in the underlying local material structure.

![Figure 6.5: This figure shows a photograph of pitted corrosion (JEI Metal Inc).](image)

The simulation of pitted corrosion employs a two phase metal as described by Svintradze and Pidaparti [69], where the surface has a film (f) and metal (M). We analytically determine the pit radii growth based on the following two equations derived from two partial integrations found in [69] to find the corroded volume $V_f$ and $V_M$:

$$V_f = \sum_{i}^{n} \left( \frac{1}{k_{f} N_{i,f}} \right)^{i+1} f^{(i)}(t) e^{k_{f} N_{i,f} t}$$  \hspace{1cm} (6.1)

$$V_M = \sum_{i}^{n} \left( \frac{1}{k_{M} N_{i,M}} \right)^{i+1} M^{(i)}(t) e^{k_{M} N_{i,M} t}$$  \hspace{1cm} (6.2)

Similar to the organic and fruit decay (Appendix A), we need to define a reaction equation that governs the volume change. We define $k$ by the Collins-Kimball expression [36] as follows:
\[ k = \frac{k_r k_D}{k_r + k_D} \]  

In Equation 6.3 \( k_r \) is the kinetic rate of the reaction, \( r \) is the distance, and \( k_d \) is a rate constant that is defined by \( k_d = 4\pi r D \). We next define the diffusion coefficient \( D \) as follows:

\[ D = \frac{RT\sigma}{Z^2 F^2 C} \]  

In Equation 6.4 \( C \) is the concentration of reacting atoms, \( Z \) is the ion valence, \( F \) the Faraday constant, \( R \) the gas constant, and \( T \) is temperature which remains a constant.

This leaves the following two equations for the solvent and impurity solution which provide the \( f(t) \) and \( M(t) \) values in Equation 6.1 and Equations 6.2 respectively:

\[
f(t) = \frac{[\varepsilon_f^M + \varepsilon_0^M + (\varepsilon_m + \varepsilon_f) (3\ln(1-e^{-\theta/T}) - D(\theta/T))]}{P_f k_f N_f,0 N_{i,f}} - \frac{[\mu_0(P,T) + \sum_i (n_i T/N) \ln(n_i/eN) + \Sigma_i n_i \psi_i(P,T)]}{P_f k_f N_f,0 N_{i,f}} \tag{6.5}
\]

\[
M(t) = \frac{[\varepsilon_d^M + \varepsilon_0^M + (\varepsilon_m + \varepsilon_f) (3\ln(1-e^{-\theta/T}) - D(\theta/T))]}{P_M k_M N_{i,M}} - \frac{[\mu_0(P,T) + \sum_i (n_i T/N) \ln(n_i/eN) + \Sigma_i n_i \psi_i(P,T)]}{P_M k_M N_{i,M}} \tag{6.6}
\]

Here we used constants from Ternes et al. [70] to set the following variables: \( \varepsilon_f^M = \varepsilon_0^M = 5.1 \times 10^{-22} J \). We attempted different values for the range of \( M(t) \) however stayed within \( M(t) \rightarrow (0, 10^{-25}) \) with \( k_M N_{i,M} \rightarrow 10^{-8} \). This gave us the following decay growth Figure 6.7. As the pitted corrosion grew larger, the constraints on the lattice were weakened and finally broken to simulate the effect.
6.4 Inorganic Results

In Figure 6.6 we show the various levels of geometry that have mutated due to the $\mu$-ton framework. The figure illustrates the paint chipping off, and small areas of pitted corrosion forming where the surface has been damaged. Here we provide some final results of inorganic decay in Figure 6.7 and Figure 6.8.

Figure 6.6: This figure shows the result of inorganic decaying over time of paint chipping, rust, and pitted corrosion. The left top image is the original geometry in Maya. The next two on the top row show the paint chipping away, and the pitted corrosion forming. The bottom left image is the distribution map of the rust flowing down and away from the holes in the paint, then all the geometry together, and then final rendered frame.
**Figure 6.7:** This figure shows the result of two rendered frames of the inorganic aging process.

**Figure 6.8:** This figure shows the result of the street scene with paint chipping away and the pavement crumbling.
Chapter 7

Organic Aging

In this chapter we describe the simulation of organic decay in the $\mu$-ton framework. We present a novel physically-based approach for simulating the biological aging and decay process that is grounded in scientific research. We apply the $\mu$-ton framework to a log to grow moss. For further specific details of our fruit decay simulation pipeline and framework, please see Appendix A. The same reaction-diffusion growth approach was utilized in the mold, moss, and fruit simulations in this dissertation.

7.1 Implementation Overview

We utilize numerical reaction-diffusion equations to form fungal and bacterial colonies, and a transpiration model to drive the volume collapse. These calculations affect the underlying maps and substrate meshes as the fruit decays. Since the soft rot, nutrient, and fungal maps interact, we simulate all three processes, the internal deformable body, skin, and output the maps simultaneously.
The following actions are performed at each time step:

**Step 1.** We spawn new colonies in the *fungal* and *soft rot maps* to simulate the effect of individual cells spreading from the existing mold or bacterial colonies and taking root in nearby areas of the fruit’s surface. For each existing colony, we create a seed point in a randomly selected location within some radius $r$ with a probability given by $1/(10 - \text{propagationRate})$, with $\text{propagationRate}$’s range between $[0 ... 9]$.  

**Step 2.** We form additional new colonies in the *fungal map* in areas of the surface affected by bacterial soft rot. The *soft rot map* serves as a probability map, with more concentrated areas of bacteria containing higher probabilities of spawning a new colony.

**Step 3.** We calculate the reaction diffusion step. The *fungal* and *bacterial maps* follow similar reaction diffusion processes, with differences in equation parameters, while the nutrient map undergoes a separate nutrient diffusion process.

   a) The fungal and bacterial concentrations at each cell (corresponding to a pixel on the map) are calculated from known concentrations at the previous time step and based on Equation 7.1, where $u$ is the active fungal density at the cell in the previous time step.

   b) The diffusion term is determined as an average of the immediate neighboring cells, each with a contribution given by Equation 7.3 and offset by a stochastic element determined by Equation 7.4 to represent random movement.

   c) The reaction term is computed in Equation 7.6 based on density at the corresponding cell in the nutrient map during the previous time step.

   d) We calculate the nutrient reaction-diffusion process, which is defined in Equation 7.8.

   e) We update the maps by writing the new concentration values at each *fungal, rot, and nutrient* cells.

**Step 4.** We mutate the geometry and update the constrain lattice.
7.2 Algorithm Details

**Fungal, Moss, and Bacterial Growth.** Multiple researchers have modeled fungal proliferation and morphological colony formation based on a non-linear reaction-diffusion process [55, 27, 35]. In our model, the fungus is separates into two states, active and inactive. As active fungi grow and proliferate, they deplete the underlying nutrient concentration defined in our *nutrient map*. As values in the *nutrient map* approach 0, the fungi at that point become inactive and cease to spread. We calculate the concentration of both active and inactive fungi by the following reaction-diffusion equations describing the fungal movement:

\[
\frac{\delta u}{\delta t} = \nabla \cdot (D_c \nabla u) + \theta f(u, n) - a(u, n)u \tag{7.1}
\]

\[
\frac{\delta v}{\delta t} = a(u, n)u \tag{7.2}
\]

Equation 7.1 provides the diffusion term which governs the non-linear growth at time \( t \). Equation 7.2 is the reaction term which depends on the nutrient concentration \( n \), fungal density, and environment parameters. \((u+v)\) is the total density of the active \( u \) and inactive \( v \) fungus. The motility of active fungus is expressed by the diffusion coefficient \( D_c \), which is defined by the logistic Equation 7.3. \( d_1, d_2 > 0 \) are model parameters and \( \sigma_1 \) is a scaling parameter. An additional scaling parameter produces the growth rate difference between the soft rot and fungal colonies, and between different species of fungus. This parameter is determined by the user through the interface tool. A stochastic element, \( \sigma'_1 \), adds random movement to the diffusion coefficient in Equation 7.4.

\[
D_c = s \cdot \sigma'_1 \cdot d_1 u \left(1 - \frac{u}{d_2}\right) \cdot n \tag{7.3}
\]

\[
\sigma'_1 = \sigma_1 (1 + \delta), \quad -0.5 < \delta < 0.5 \tag{7.4}
\]

Fungal growth depends on the physical environment. Relative humidity \( \phi \) and temperature \( T \) dampen \( D_{\text{damp}} = \text{norm}\left(T + \phi\right) \) the diffusion coefficient \( D_c \) expressed in Equation 7.5.

65
We then use a Michaelis-Menten kinetic function (Equation 7.6) to model the reaction kinetics of enzymes. In Equation 7.1, \( f(n, u) \) describes the consumption of nutrients by the active fungus \( u \). Here \( \theta \) is the constant rate in which fungal proliferation connects to nutrient consumption. \( f_1, f_2 \) are model parameters and \( \sigma_2 \) is a scaling parameter.

\[
f(u, n) = \sigma_2 \left( \frac{f_1 n}{1 + f_2 n} \right) \cdot u
\] (7.6)

Equation 7.1 transforms active fungus \( u \) to inactive fungus \( v \). Here we used Mimura’s formula (Equation 7.7). Active fungus normally diffuses along the border while inactive fungus resides in the center. Once again, \( a_1, a_2 \) are model parameters and \( \sigma_3 \) is a scaling parameter.

\[
a(u, n) = \sigma_3 \cdot \frac{1}{(1 + \frac{u}{a_1}) \cdot (1 + \frac{n}{a_2})}
\] (7.7)

We model the spread of bacterial colonies using the same formula. The bacterial soft rot map guides the propagation of fungal colonies. The fungi cannot propagate into areas unaffected by bacteria, thus reflecting the notion that the growth of certain organisms on the surface depends on existing infection.

Nutrients exist in a non-uniform random distribution on the surface and substrate of the fruit. This is the main factor for fungal growth in our model. Nutrients deplete and do not allow fungus to remain active indefinitely. Nutrient transport is defined by Equation 7.8. The nutrients are diffused by the term \( D_n \nabla^2 n \) and consumed by \( f(u, n) \). \( D_n \) is the nutrient diffusion coefficient.

\[
\frac{\delta n}{\delta t} = D_n \nabla^2 n - f(u, n)
\] (7.8)

Table 7.1 provides the initial model parameters of our simulation. These parameters change the growth rate and nutrient depletion. This process is further affected by water vapor pressure, humidity and temperature. Our system provides default parameters for the most popular types of mold growth. However, our simulation method below can
generalize to different mold types by manipulating the input parameters.

<table>
<thead>
<tr>
<th>Model Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1 = 0.01$, $d_1 = d_2 = 0.01$, $n_0 = 1.0$</td>
</tr>
<tr>
<td>$\sigma_2 = 5.0$, $f_1 = f_2 = 1.0$, $\theta = 0.05$</td>
</tr>
<tr>
<td>$\sigma_3 = 5.0$, $a_1 = 1/2400$, $a_2 = 1/120$</td>
</tr>
<tr>
<td>$u_0 = 1.0$, $v_0 = 0.0$, $D_n = 0.000001$</td>
</tr>
</tbody>
</table>

Table 7.1: This table provides the biologically inspired parameters [35] for our reaction-diffusion mathematical model.

7.3 Organic Decay Results

Here we provide some frames of the results of moss growth on a log in Figure 7.1.

Figure 7.1: This figure shows the result of an organic aging processes in the $\mu$-ton framework. The left is a plane log, and the next two images in the growth of moss over time.
Chapter 8

Evaluation and Analysis

This chapter provides implementation details and performance testing of the $\mu$-ton framework. We compare our work to $\gamma$-ton tracing [16], since this framework is the most closely related work to the $\mu$-ton framework’s design and implementation. We additionally ran a short user study to further validate the perceptual realism of the simulations. Finally, we end the chapter with a short discussion about the framework.

Figure 8.1: This figure show images of an OpenGL (left) implementation and a Maya plug-in (right) of the framework.
8.1 Implementation

We provide two different implementations of the \(\mu\)-ton framework: an OpenGL/C++ tool (Figure 8.1 left), and a Maya implementation (Figure 8.1 right). This shows our tool is easily reproducible in different application environments. To calculate the Voronoi-cells and constraint lattice we used the Voro++ software library to carry out the Voronoi tessellations. This is an optimized library authored by Chris Rycroft. This library was modified to work in our OpenGL and Maya implementations to suit our needs. The k-d tree implementation is a custom implementation that was adapted from a previous rendering project to allow the K-nearest neighbors to be returned so \(\mu\)-tons could have a sphere of influence. In general, we tried to avoid native Maya boolean operations and instead used CARVE. Our system used a heavily customized version of CARVE, which itself is implemented as our own Maya-plug-in, but was useful for the C++/OpenGL implementation. Our OpenGL implementation used standard libraries such as GLEW, FreeGLUT, and GLM. Our particle system followed standard conventions and was implemented in CUDA on the GPU for efficiency. This was a simple port, and no fancy tricks were used to optimize the code directly for the GPU. When we simulated the crumbling effects this was done by making the current Voronoi cell an active rigid body in the Bullet Physics Library [1]. Bullet is an open source physics engine that is integrated into Maya.

8.1.1 User Interface Tool

We have created a tool that integrates the \(\mu\)-ton framework into a plug-in inside Maya. The tool first asks the user to specify their emitters and amount of decay particles they wish to cast into the scene. At this point they can attach and change properties for these decay particles to capture the aging effects that the user wishes to produce in the scene. The next step requires the user to select the objects to decay, the amount of material particles, and their material properties. The user codes any growth or decay patterns using the factory design described in Chapter 4 and Chapter 5. This allows specialized growth
processes, such as fungus and mold to occur. Figure 8.1 shows a screen-shot of our tool, and an illustration of the tool working in Maya. These material and growth equations are the only user-programmed components needed within the μ-ton framework. The rest are parameters, influence choices, and model geometry and attributes set by the interface tool.

Figure 8.2: A screen shot of our user interface tool in Autodesk’s Maya. Our plug-in tool allows users to choose presets and parameterize the simulation to generalize output. The user imported her own geometry and is currently displaying the nutrient map on the object.

We have designed and developed a customized authoring tool that integrates our fruit simulation work into a plug-in to Maya 2010 and updated the tool for Maya 2012 to produce more simulations for Appendix B. Our user-interface is scripted in MEL and uses the Maya plug-in API to develop features to allow the user to create and parameterize a simulation. Figure 8.2 shows the various features our tool provides the user. The designer selects meshes to use, and customizes the simulation. The user may also visualize the various fungal, soft rot, and nutrient maps on the mesh using the button on the top of the
tool. Additionally, the user may select pressure points that allow new fungus to begin to grow and the fruit to begin collapse on directed input.

The two main advantages to these plug-in tools is Maya provides a standard accepted framework that is commonly used in industry, and Maya allows us to leverage better rendering systems (V-ray and Mental Ray) to produce production quality results. Full resolution images can be viewed in Appendix B. This shows off the practicality of our work by allowing users to age their own art assets.

Rendering took place in Mental-Ray and V-Ray. This was the most expensive part of the simulation since many of the images and movies were rendered in 1080p with multiple shaders, light sources, and subsurface scattering. Rendering a frame took anywhere from 1 minute to 10 minutes per frame depending on scene complexity.

8.2 Performance Testing

The μ-ton framework was evaluated on numerous models. The three most expensive parts of the simulation are the calculation of the Voronoi-regions (namely the boolean operation part) and the constraint lattice, finding the mesh intersections in the particle simulation, and finding the nearest material μ-tons in the scene. To speed up the Voronoi calculation, we calculate the Voronoi regions on the object’s bounding box and perform a boolean intersection. This works well for concave meshes. For a convex mesh, cutting planes can be specified and used to calculate the Voronoi cells directly on the object, without needed an intersection operation.

The simulations were run on a Pentium I7, 2.5Ghz computer with 6GB of RAM. Figure 8.3 shows the running time for different numbers of Voronoi calculations on 4 different meshes from 10 to 1,000,000 Voronoi tessellations (1 million cells is very extreme, and normal subdivision steps usually do not scatter this many particles). For every particle that was scattered we timed the Voronoi cell computation and boolean operation. This is the slowest part of the code when we profiled it. Figure 8.4 shows the running times for for
building the k-d trees. This is the second slowest part. This acceleration structure greatly speeds up computation for decay particle intersections which take around 0.053 ms per particle to look up nearest neighbors. These graphs scale well as the object face counter increases to handle many games and simple production assets. Different approaches are needed to age an entire city scene at once.

![Figure 8.3: Performance testing of the major bottlenecks for the µ-ton framework. Here we tested 4 objects: Object 1 (8 vertices and 12 faces), Object 2 (382 vertices and 760 faces), Object 3 (1562 vertices and 3120 faces), and Object 4 (6242 vertices and 12,480 faces). The graph plots number of particles scattered vs time to compute on a log scale. For every particle scattered the Voronoi cell computation and boolean operation were timed.](image)

8.2.1 Memory

In the OpenGL implementation, memory is not a great concern. As the simulation scales up to handle a large number of subdivisions and pieces, the memory footprint remains around 400-500MB for almost a half million Voronoi-cells. The bottleneck is draw calls. In contrast, memory is a big issue in the Maya plug-ins. Around 50,000 Voronoi cells easily hog around 5 GB of memory. This is more of a design “choice” of Maya’s implementation. For our implementation we make a new MFnMesh for every cell that is constructed. The cells undergo a great amount of change from the beginning to the end
Figure 8.4: Performance testing of the second slowest bottleneck for the $\mu$-ton framework. Here we tested 4 objects: Object 1 (8 vertices and 12 faces), Object 2 (382 vertices and 760 faces), Object 3 (1562 vertices and 3120 faces), and Object 4 (6242 vertices and 12,480 faces). The graph plots number of particles scattered vs time to compute on a log scale. This calculates the time to build the k-d tree acceleration structures to help speed up the collision detection and finding the nearest neighbors.

of the simulation, and this change is captured in Maya’s Directed Acyclic Graph (DAG). Maya is not a “database”. Data flows between frames. If given the chance to re-implement this framework, we would stay away from many of the internal Maya constructs as much as possible since the objects tend to take up a lot of memory since their construction history and transforms are stored in the DAG.

8.3 Comparison with $\gamma$-ton tracing

We compared our work to $\gamma$-ton tracing [16] since this work was a key source of inspiration for the $\mu$-ton system, and is the closest competitor to what we have presented. $\gamma$-ton tracing also casts particles into the scene, but a key difference is it more strictly follows photon tracing conventions than our work. For example, $\gamma$-tons are cast in straight lines from their emitting source, then have 4 choices on an intersection: bounce, flow, reflect, or settle. This process produces nice images and allows aging effects to move and flow around a scene. This was some of the first work to simulate multiple effects and transportation of
Figure 8.5: This figure illustrates the differences and advantages of the μ-ton tracing framework. (Left) γ-ton tracing [16] framework. (Right) μ-ton tracing framework. On the right the μ-ton tracing framework updates and mutates the underlying geometry. The framework also accounts for more complex emission sources.

The key difference in our system is that most aging effects have local growth and are not merely transported around. This fact makes our framework much more general than the γ-ton approach. We can capture a greater range of aging effects. This is due to two main features of the μ-ton framework. The first is ability to capture growth processes (this allows aging effects to add substance in a scene). This difference

Figure 8.6: Rendered log with moss. On the left is the result of γ-ton tracing. γ-ton tracing transported moss into the cracks of the log and blended the textures together. The next two images are frames from the μ-ton framework where the moss grows over time from reaction-diffusion. Here the simulation adds substance to the scene.
is captured in Figure 8.6. Here we present two different simulations of a log with moss growth. On the left, γ-ton tracing produces moss growth in the grooves of the log and uses a multi-texturing approach to blend the moss and the log texture together to get a final result. The next two images show our results from the μ-ton framework run on the same geometry. The difference is how the moss grows on our log after the μ-tons hit the object. γ-ton tracing primarily transports aging effects around the scene and blends the result. The μ-ton framework adds and grows substances in a scene to more realistically capture a wide range of aging phenomena, such as mold, moss, fungus, and rot which would be difficult to reproduce with γ-ton tracing.

Figure 8.7: This figure shows a rendered sphere with geometry changes. On the top is the result of γ-ton tracing, which uses displacement maps. The bottom image is the result of the μ-ton framework. Here the actually geometry is changed, and pieces break off the object over time and gather near the sphere. This is not possible with γ-ton tracing.
The second feature that differentiates the two approaches is the mutation of underlying geometry. $\gamma$-ton tracing uses a displacement map approach to fake changes in the geometry. This is shown on the sphere example in Figure 8.7 (top). The $\mu$-ton framework alters the underlying geometry at every frame, and spreads material $\mu$-tons inside the geometry to capture local aging effects shown in Figure 8.7 (bottom). These features are not possible under the current published work of $\gamma$-ton tracing.

The main advantage to $\gamma$-ton tracing only using displacement maps is displacement maps take less memory than changing the geometry at every frame. Our system does not have to mutate the geometry at every frame if the user rather works with only displacement mapping and could utilize the same approach. In other words, $\gamma$-ton displacement maps can be simulated within the $\mu$-ton framework. $\gamma$-ton tracing still does not capture the local growth and mutation changes we have demonstrated in the $\mu$-ton framework.

8.4 Comparison with a Simple Erosion system

Figure 8.8: This figure shows a rendered sphere with geometry changes. Here we simulated a simple erosion system, and compared the results to the $\mu$-ton system. Our simulation allows for more localized effects, and less tessellation then subdividing the entire mesh at once in the beginning.
We also compared our work to simple erosion models. This is demonstrated in Figure 8.8. Here the object is initially broken into a large number of Voronoi cells. The images in Figure 8.8 (left) initially had 10,000 cells. Every time a decay particle hit a cell it was deleted. This approach would slowly eat away at a substance and produced a decent - but limited - result. The two problems with this approach are the initial Voronoi calculation is very expensive and it does not capture strength and strain features. The \( \mu \)-ton framework only adds geometric detail where it is needed. This is achieved by only emitting the \( \mu \)-tons where there are a lot of local impacts or growth processes. Additionally, our system does not easily chip geometry away. The simulation has to work hard to remove cells based on the material properties of the object. Stronger bonds on the lattice between Voronoi cells take more decay particle impacts to break off. This better captures different material properties.

Figure 8.9: Sample screen shots from one of the videos shown. The top row is the reference time-lapse video of a decaying tomato. The bottom row is a comparison of a simulated decaying tomato.

### 8.5 Perceptual User-Study

A limited perceptual study was done to determine whether the \( \mu \)-ton framework produced perceptually realistic and convincing results. Two 15 second movies were captured from
the simulation at a fixed viewpoint for the tomato and orange. These videos were selected since we have reference time-lapse videos directly associated with them at similar viewpoints. Videos were encoded using the MP4 format at a resolution of 640x480. Four different sets of before-and-after still images were chosen and rendered at 1080p. Screen-shots of these videos and images are shown in Figure 8.9 and Figure 8.10.

![Figure 8.9](image1.png)

**Figure 8.10:** An example set of images shown in the user-study. A simulated wall of concrete, simulated mold growth, and a real photograph of mold growth [Basement Dehumidifiers].

Fifteen subjects were recruited online, none were computer graphics programmers. Each subject was shown one simulation video and one ground truth video, and the before-and-after images. Subjects could view the video once and images for around 20 seconds. After each viewing of an aging process, subjects were asked to fill out a brief survey. The questionnaire determined how realistic and convincing the simulation was viewed.

The results from the user-study is shown in Figure 8.11 and Figure 8.12. This result shows that our system was very convincing on a perceptual level to subjects. Subjects found that the images were very close to reality, and many could not guess whether they were a photograph or simulation. For each evaluation, the subjects were asked to score on
a five-point Likert scale whether the simulation appeared realistic (1 for “Strongly Disagree”, 5 for “Strongly Agree”). Subjects found the aging processes convincing. Subjects also were asked to rate “how gross and disgusting” the simulations were, though this may sound like a “fun” question, the aim was to indirectly see what kind of reaction the images invoked.

Figure 8.11: Perceptual study responses plotted on a five-point Likert scale for the perceived realism of the simulation videos an images. Higher numbers are better.

Figure 8.12: Perceptual study responses plotted on a five-point Likert scale for how disgusting are the simulation videos an images. Higher numbers are better.
8.6 Discussion

The ability to simulate and render a variety of aging effects is a very useful tool for the graphics community. Technical directors can easily edit more decay effects and simulate what they need for their production requirements. The hardest part of this work is true validation. Many aging processes, such as corrosion, have to simulate at the atomic level to truly capture the material science correctly. Biological and chemical processes are extremely difficult to fully understand, however our $\mu$-ton framework provides a good approximation of the general form of the aging process. There is still a ways to go to truly have a scientifically accurate simulation and verifiable simulation. This was not the intent of this framework. We deliberately targeted our algorithm towards a system that capture the essence of aging effects and provided an easy framework where the effects can interact and mutate geometry. In the future, we plan to further obtain and study the scientific details of aging processes to better simulate them. It would be exciting for us to apply the mu-ton framework to real problems in the material science domain.
Chapter 9

Summary of Contributions

The main contribution of this thesis work is a general aging system that simulates various aging classes that address two fundamental problems that prevent aging simulation from being actively used in industry. The first problem is that it is difficult to simulate multiple interacting aging phenomena. The second problem is that it is particularly challenging to simulate processes that affect both the visual appearance and geometric structure. The $\mu$-ton approach we present, produce aging and decomposition effects of better quality than previous approaches. We show that the $\mu$-ton pipeline is appropriate and effective for a variety of off-line simulation applications. Some example applications include providing a time slider to simulate the level of decay in a virtual world and to simulate level tiles in games. Many existing decay algorithms can benefit from this work, because we provide a general framework that exploits a mass-point and constraint system, which is a well studied approach in discretizing simulations in computer animation.

The $\mu$-ton approach concentrates on the more general problem of global decay simulation with multiple interacting phenomena. We reduce the decay simulation to a point-mass and constraint particle simulation problem. This is a well-studied problem in physically based animation, and we leverage many accepted constructs and practices. The user can build the simulation to handle a variety of decay types. To our knowledge, our work is the first research effort that addresses the problem of automatically simulating a variety
of decay effects grounded in scientific research to produce more realistic decay effects that really mutate the geometry on a fundamental level. The mutability of the \( \mu \)-ton between particle and tetrahedral mass is a useful technique for bridging multiple interacting phenomena across geometry and texture spaces.

The essence of the system was first tested by aging and rotting fruit (Appendix A). This approach focuses on simulating fruit senescence and decay. To our knowledge this is the first work that simulating the fruit decomposition process which simulated multiple interactions simultaneously. The decomposition simulated decay effects both on the surface and the underlying substrate. The fruit decay pipeline is a novel physically-based approach that simulates various organic growth patterns grounded in scientific research. This was the first application of most of the features used in the \( \mu \)-ton framework. Additionally, we provide a custom fruit decay tool in Maya that allows an artist to direct the simulation. This generalizes the simulation to a variety of fruits, real or imaginary, and even other geometry, allowing artists to produce decay simulations based on their production requirements. This work was published in Computer Graphics Forum (Proceedings of Eurographics 2011).

We feel the \( \mu \)-ton framework discussed in this dissertation succeeded in presenting

- Simulating multiple interacting aging phenomena: The \( \mu \)-ton system stores information that allows multiple effects
- Simulating geometry and texture changes: The \( \mu \)-ton framework simulate processes that affect both the surface and the substrate. This is a vital part to any aging effect.
- Provides a general framework: In Chapter 5 (Tutorial) we provide a tutorial how to apply an again process from Chapter 3 (Taxonomy). The \( \mu \)-ton system can easily be extended by users to novel decay effects.
- Ease of implementation: The system is reasonably straightforward to implement and leverages many production pipeline tools: Maya, mass-spring-constraint systems, texture mapping, etc. So the work can be easily recreated.
• Tools: We provided 2 different implementations of the work: An OpenGL framework and 2 Maya Plug-ins that allow the tool to be easily used to create new examples.

• Fruit Decay: We provided the first work on simulating the fruit aging process (published in Eurographics 2011).

9.1 Future directions

The framework presented in this dissertation opens novel avenues of research that would not otherwise be available. As noted in Appendix A.2, we are not simulating everything that occurs, and only scratched the surface with these examples. An interesting avenue of future work would be to spend time simulating very complex systems of interaction and trying to provide insights into the real world. For example, simulating different conditions of fungus growth on fruit and rotting of fruit could help shippers test different storage and shipping conditions. Simulating barnacles and rust on boats could provide an answer when the ship needs to be repainted.

9.1.1 Practical Use

The $\mu$-ton framework has not been production stress-tested yet. In an academic setting, the aged models might be complex but they do not reach the level of industry’s complexity due to the time and resources we have available. It is our hope that research developers at production studios will use this technique in their production pipelines. One way to facilitate this would be to generalize the $\mu$-ton system further. One drawback to the current implementation is the processes only work on polygonal models. Many production studios rather use subdivision surfaces to represent their geometry. A future goal would be to allow numerous geometric representations into the $\mu$-ton framework to allow production studios to use their assets in the $\mu$-ton system.
Appendix A

Fruit Senescence and Decay Simulation

In this appendix, we describe the technique for fruit senescence and decay simulation. The fruit decay is grounded in the relevant biology to produce more accurate results. This method is the first physically-based approach to simulate interactions between multiple interacting processes simultaneously. This chapter presents the detailed biological background for correctly simulating fruit decay, and describes the implementation.

A.1 Introduction

Food scientists and biologists recognize fruit senescence and decay as a serious problem in the shipping and storage of produce. The short ripening and post-harvest life-cycle of fruit influences their value. Once one piece of fruit begins to rot, other fruit become more susceptible to the mold spores. Animation studios realize rotting fruit make scenes appear more realistic in movies such as Ratatouille and Toy Story 3. These movies feature numerous actions shots in dumpsters and trash heaps. Hero characters pick up and interact with trash in both the main story line and in the background.

While there has been some significant work in the animation industry revolving around the creation of realistic computer generated food, there has been little emphasis on the reproduction of natural decay processes. Animation studios create food for a particular shot,
and are not concerned with simulating the aging process. Artists hand fashion textures and
displacement maps onto carefully crafted models to achieve the look they desire on a shot
to shot basis. Traditional surface texture imperfection and aging techniques are visually
satisfying, but they do not take into account the effect of mold and other microorganisms
on environments in a way that is biologically accurate. These methods neither directly
include, nor can they be readily extended to, the complex biological processes associated
with decomposition. An artist needs to hand fashion textures and displacement maps at ev-
every stage of the decay process. This texture based method is not tractable for realistically
simulating time-varying decay, nor tractable for a large number of fruit.

The senescence and decay model accounts for their entire decomposition process.
We do not rely on tediously hand fashioning textures, or nave procedural approaches.
This method changes both the overall shape and the appearance properties of the fruit’s
anatomy. First, the fungus and bacteria are grown on the fruit’s exocarp, or outer skin.
This process is directly affected by moisture, temperature, and nutrient concentration at
each point on the surface. Multiple types of fungus, bacteria, and rot grow on the fruit’s
exocarp concurrently. These decay processes only grow when sustained by nutrients from
the fruit. Second, we address the gradual collapse and shrinking of the internal structure.
Because water accounts for a large percentage of a fruit’s mesocarp (interior), water tran-
spiration is a primary factor in the way a fruit decays and loses internal mass. Additionally,
rot makes areas of the substrate soft and more prone to collapse. This effect accounts for
the structure of the fruit, and the interaction of the multiple decomposition processes. To
the best of our knowledge, this is the first approach to describe a visual simulation of fruit
decay.

The simulation is based on user-defined parameters for skin thickness and porosity,
water content, flesh rigidity, ambient temperature, humidity, and proximity to other sur-
faces. These parameters allow direct control of the simulation and a broad range of results.
We demonstrate visually accurate simulations on three fruits: a tomato, apple and orange.
We provide a variety of fruit presets and parameters to automatically produce proper fungal growth and rot for the selected fruit.

Additionally, we provide a tool in Maya that allows an artist to direct the simulation. This generalizes our simulation to a variety of fruits, real or imaginary, and even other geometry, allowing artists to produce decay simulations based on their production requirements. To demonstrate our method’s generalization to any user-defined object, we imported a unique mesh into our user interface and simulated the decay processes based on user parameterization.

A.2 Biological Background of Fruit Decay

Decay is an essential step in the reproductive process of fruiting plants - the decomposition of flesh frees and feeds the seeds of the fruit to grow into a new plant. Decomposition is a natural part of fruits’ lifecycle. While disease can strike a fruit at any stage of life, fruit naturally becomes more vulnerable to disease over time. This work exclusively investigates post-harvest rot in a typical home environment.

Prasanna et al. [64] and Pech et al. [61] studied factors in the fruit ripening process. Numerical approaches were developed to predict morphological patterns of bacterial colonies [47, 58]. Post-harvest disease is typically initiated when a point on the fruit’s surface is compromised, either due to injury or structural vulnerabilities. One major wound is the stem puncture where the fruit was picked. Once an initial infectious agent like bacteria has attacked, it further weakens the surface and leaves it susceptible to infection by other organisms. This is the reason that mold often forms in areas already infiltrated by bacteria. The soft rot serves as an entry point for fungal spores.

**Fungal Growth:** Various reaction-diffusion processes have been proposed to better predict fungal growth based on a variety of parameters [55, 27, 35]. Bacteria and fungi thrive on the nutrient-rich surface of a ripened fruit. However, nutrient levels deplete over time as
colonies use up resources, slowing down the growth process. Environmental factors such as temperature and humidity greatly affect mold and bacterial growth. Warmer temperatures lead to more rapid proliferation, while colder temperatures retard growth. Our model incorporates the interaction between these different factors to determine the concentration of bacteria and fungi at each point on the fruit surface.

**Mold:** Various types of mold affect fruit. Andersen and Frisvad [3] showed Penicillium and Rhizopus were two species present in moldy tomatoes. Rhizopus is a white thread-like Mucoralean mold that infects tomatoes. Erwinia carotovora is a type of soft rot that affects the internal substrate, making the fruit collapse in the infected regions. The species of mold that most commonly attacks apples is Penicillium expansum. Apples are also often infected by the organism Neofabraea, which causes a soft spot. This infected region appears as a growing and darkening brown spot on the surface. Penicillium digitatum primarily affects oranges. These colonies have a gray-blue center and are white along the edges.

Figure A.1 provides three photographed examples of Penicillium digitatum, Penicillium expansum, and Rhizopus mold on an orange, apple and tomato. Additionally, Figure A.1 shows three time lapse images of a tomato photographed from the bottom through a piece of glass. Mold starts to form at the points of contact with the glass. This occurs because the flesh weakens under the pressure of the weight of the fruit, providing entry.
points for pathological agents.

Table A.1: This table provides the weight, water content, and the transpiration coefficient rates ($K$) for several fruits. [4, 15].

<table>
<thead>
<tr>
<th>Fruit</th>
<th>Food Weight (g)</th>
<th>Water Weight (g)</th>
<th>% Water</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>138</td>
<td>116</td>
<td>84</td>
<td>0.17</td>
</tr>
<tr>
<td>Orange</td>
<td>140</td>
<td>122</td>
<td>87</td>
<td>8.48</td>
</tr>
<tr>
<td>Peach</td>
<td>87</td>
<td>76</td>
<td>88</td>
<td>14.1</td>
</tr>
<tr>
<td>Pear</td>
<td>166</td>
<td>139</td>
<td>84</td>
<td>0.68</td>
</tr>
<tr>
<td>Tomato</td>
<td>123</td>
<td>115</td>
<td>94</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Transpiration. Fruits have high water content in proportion to their weight [4, 33, 63]. Table A.1 relates the weight of the fruit to the water content. Moisture loss due to transpiration is a major factor in fruit decay and is one of the main factors that drive changes in the internal structure [14, 65, 15, 44]. As water continuously transpires from a fruit, it loses mass and shrinks. The rate of this process largely depends on the structure and the properties of the fruit skin. Certain fruit skins are porous, while others are less permeable. A fruit exhibits a hydrostatic pressure as it sits. Transpiration also depends on the physical environment. Temperature and water vapor pressure, as well as the size and shape of the fruit, are accounted for when calculating transpiration rate. Our model utilizes transpiration as the main factor for internal volume loss.

Factors not discussed. Fruit decay is a complex natural phenomenon that is driven by innumerable conditions in addition to those we have discussed. We do not consider elements such as nutrient deficiency or poor fertilization while growing. The model does not consider if the internal structure is weakened by insects. Our mathematical model can be generalized to different types of fruits and infectious agents we do not discuss based on the input parameterization framework the system does provide.
A.3 Fruit Senescence and Decay

In this section, we provide details about our model structure and a set of parameters to generalize the decay process. Figure A.2 shows the pipeline of our simulation using the plug-in tool we created. First, the user sets both the skin and internal substrate meshes. Next, the user either selects fruit, mold, and environmental presets, or sets custom values for each parameter. These input parameters drive the reaction-diffusion and transpiration mathematical models described in detail below. We use a cloth-like simulation as the external skin model. Then, the user selects contact points, which will serve as initial seed points for the bacterial soft rot map. Finally, the user runs the simulation. The program output includes a series of maps representing distributions of fungus, bacteria, and nutrients for each frame of the simulation. The output also includes a vertex cache which is automatically applied to the internal substrate mesh. The alpha maps shade the different layers of the fruit-fresh surface, bacterial soft rot, and mold. We will describe our particular rendering approach later.

A.3.1 Distribution Maps

The results of the reaction diffusion processes that drive the surface component of our simulation are visualized as a sequence of time-varying maps. Our maps range $X$ between $[0 \ldots 1]$, with white representing a maximum density of 1, and black representing a density of 0. We use three different map sequences: the nutrient map, the fungal map, and the soft
**Figure A.3:** Results of the final maps: Values range between \( \{0 \ldots 1\} \) for the phenomenon concentration values. These processes change the physical properties of the surface and internal volume substrate. (a) Shows nutrient depletion, (b) Fungal colony growth, (c) Soft rot spread, (d) Final rendered frames.

**rot map.** The nutrient map represents the distribution of nutrients over the surface of the fruit, and serves as a distribution map for fungal growth. We show in Figure A.3 (a) how the nutrients deplete over time as the active fungus grows. Next, the fungal maps track the areas of fungal colony formation. These grow over time based on our reaction-diffusion model. The soft rot map represents the areas of bacterial growth that form on fruit. This map serves as a probability map for the propagation of new fungal colonies. Mold grows primarily on areas that have already been infected by the bacteria. A new set of maps is generated and stored at each frame. For our implementation, the maps were sized at
512 × 512 pixels. In the μ-ton framework, decay μ-tones would spawn the new fungal and mold growth.

Figure A.4: A composite image of a cross-section view of our fruit’s anatomy. (Right) shows a final rendered frame. (Left) shows the skin and deformable body of our substrate.

A.3.2 Internal Volume

The volume of the fruit is modeled as a deformable object. These springs provide the constraints between the μ-ton material particles for the fruit. The shape and volume of this mesocarp changes based on the forces of rot and transpiration derived from our mathematical model. We use a mass-spring system that utilizes a set of point masses connected by elastic links that form the constrain lattice. Figure A.4 illustrates this setup and provides a cross-section view of the underlying anatomical simulation body. Each mass node corresponds to a specific vertex of the object. Initial spring constants are determined by the user-defined firmness parameter (\( K_s = \text{firmness} \)), which varies across different fruits. For example, tomatoes have high water content and are softer than apples. The spring constants and resting lengths (\( rLen \)) are governed by a pressure map, which may change the material μ-ton at every time-step.

As the soft rot map changes the outer appearance from the reaction-diffusion growth, spring parameters at the material μ-tones at the same locations in the pressure map decrease.
in stiffness and length, affecting the internal volume. This represents areas of soft bacterial rot where the fruit is softer. Therefore certain areas may collapse faster based on the level of infection. At each time step of the simulation, spring constants and resting lengths are decreased at a rate proportional to water content and the calculated transpiration rate.

Transpiration coefficients for various common fruit types, including those that we used in our simulation, are given in Table A.1. The volume is proportional to the cube of the spring resting lengths. Since we know the proportion of the fruit’s volume, which is water (waterContent), and the rate at which water volume is lost (tRate), the percent total volume we would want the fruit to lose at each time-step is: \( tRate \cdot waterContent \cdot volume \). Therefore, change in resting length at each time step is given in equation A.1:

\[
\Delta rLen = -c \cdot rLen \cdot \sqrt[3]{tRate \cdot waterContent} \tag{A.1}
\]

Where \( c \) is a weighting factor. In addition, we wanted the springs to lose tightness over time to reflect the softening that occurs due to water loss and decrease in structural integrity of the internal fibers. The change in \( K_s \) at each time-step is given by:

\[
\Delta k_s = c \cdot waterContent \cdot tRate \cdot K_s, \text{ where } c \text{ is a weighting factor.}
\]

### A.3.3 External Skin

We found that the dynamic properties of fruit skin can be modeled in much the same way as cloth by spreading a series of \( \mu \)-ton surface particles. First, we bound the skin to the internal volume mesh with point constraints. As the internal mesh started to collapse, it naturally produced the effects that occur during decay, such as wrinkling and shriveling. The simulation parameters of the cloth were determined by user-defined thickness (\( S_t \)). See Table A.2 for our settings. The fruit skin layer was simulated using the nCloth cloth solver in Maya.
Table A.2: This table provides the input parameters for the different fruits in our results. The temperature was set to 72 degrees and humidity to 50%.

<table>
<thead>
<tr>
<th></th>
<th>Apple</th>
<th>Orange</th>
<th>Tomato</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porosity</td>
<td>25</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>Thickness ($S_t$)</td>
<td>30</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>Firmness</td>
<td>80</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>Growth Rate</td>
<td>20</td>
<td>50</td>
<td>75</td>
</tr>
<tr>
<td>Propagation rate</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

A.3.4 Presets and Parameters

We provide a set of fruit, mold, and environmental presets and parameters to effect the decay of fruit in our simulation. Fruit presets include tomato, orange and apple. Environment presets include tropical, cold desert, hot desert, and temperate climates.

To generalize the simulation, our custom parameters allow a variety of effects to be created. Surface parameters include the thickness and porosity of the exocarp. Internal parameters include water content, transpiration coefficient and firmness. Table A.1 provided values for only fruit types found in our examples. Many researchers have conducted studies to accurately determine these values for other types of fruit [4, 15]. Mold parameters, such as growth rate, propagation rate, and color allow for different mold growth patterns. Ito and Mizuno [35] specify other growth settings for more mold types. Environmental factors, such as temperature and humidity, also influence the simulation. Lastly, we allow the user to specify initial areas of rot by selecting vertices on the input mesh. By default, the simulation begins to rot at the points of contact with other surfaces and at the stem. However, a user can specify new wound points on the pressure map as desired, to get “directable” simulations.

A.3.5 Rendering

In order to give our fruits a realistic look and feel, we modeled and rendered our results in Autodesk Maya 2010. The color and polygonal structure of each fruit were empirically derived by observing and photographing real fruit (Figure A.1). However, a more complex
capture process [32] can also be used. We used sub-surface scattering shaders to create the effect of translucency in the fruit flesh, and modified specularity and color to match the captured image data. We used a layered shading system, applying the map sequence acquired from our simulation as alpha maps for the decay layers. The fruit color changes based on the maps. We colored the soft rot layer brown and decreased specularity, and used the fungal map as a length and baldness map for Maya Fur to simulate hair-like mold fibers. Our results were rendered with Mental Ray for Maya.

### A.4 Simulation Framework

We simulate both fungal colony and mold growth on the exocarp, and soft rot formation and transpiration on the mesocarp. We present mathematical models that tie together these time-varying biological processes on the multiple layers.

#### A.4.1 Implementation Overview

We utilize numerical reaction-diffusion equations to form fungal and bacterial colonies, and a transpiration model to drive the volume collapse. These details are found in Chapter 7.

**Transpiration.** The transpiration rate of fruit plays a considerable role in the water balance and weight during decay. Equation A.2 expresses the typical transpiration rate \( m \). Here \( K \) is the overall mass transfer coefficient. Table A.1 provided values for the various fruit types found in our examples. (Depending on various characteristics of each individual fruit, this coefficient varies slightly sample by sample.) \( A \) is the surface area of the fruit. \( P_s \) is the water vapor pressure at the evaporating area surface. \( P_a \) is the water vapor pressure of the surrounding atmosphere.

\[
m = KA(P_s - P_a)
\]  

(A.2)

94
The water vapor pressure $P_s$ is calculated by $V_p \cdot P_{sat}(T_s)$ [14]. Here $P_{sat}$ is the saturation vapor pressure of water, $T_s$ is the surface temperature and $V_p$ is the vapor pressure set as either 0.98 or 0.99. $T$ is the atmospheric temperature. $P_s$ is $P_{sat}$’s steam point pressure. A good estimate of the saturation of water vapor pressure at a given temperature can be acquired using the Goff-Gratch equation.

$$\log P_{sat} = -7.90298 \cdot \left(\frac{T_s}{T} - 1\right) + 5.03808 \cdot \log\left(\frac{T_s}{T}\right) - 1.3816 \cdot 10^{-7} \cdot \left(10^{11.344(1-T/T_s)} - 1\right) + 0.0081328 \cdot \left(10^{-3.49(T_s/T-1)} - 1\right) + \log P_s$$ (A.3)

A.5 Results

We simulated our results on a 3.2GHz quad core Xeon processor with 4 GB of memory. The deformable body and fungal growth codes were written in C++. Each time-step averaged around 10 seconds to compute the reaction-diffusion, transpiration, and deformable collapse, and to write the texture maps. Applying the vertex cache data generated by the deformable body simulation to the user-defined mesh took 1.5 seconds per frame. Rendering time was approximately 10 minutes per frame with our scene lighting.

We provide final renders for three types of fruit: an apple, an orange, and a tomato. We demonstrate that we can simulate a variety of fruits and fungi. Figure A.5 shows results of a tomato and a bowl of oranges. Figure A.7 shows a final rendered sequence of multiple orange and a tomato. Figure A.3 displays the apple sequence.

A.5.1 Multiple Fruit

We have demonstrated our simulation on a group of oranges. Figure A.5 shows a final rendered result. In this example, there are more contact points between the multiple pieces of fruit. These contact points provide more pressure to the fruit skin causing wounds and
locally high humidity. When a fruit becomes damaged, fungus begins to appear at these locations due to the increase in nutrient concentration. Since our simulation is parameterized on pressure points, it was straightforward to show the results of mutual contacts among the grouped fruits, demonstrating the full power of our method.

A.5.2 True Life Comparison

In Figure A.6 we compare our final rendered frames to real photographs. We demonstrate that our method produces similar effects. A key feature of our system is that the biologically derived models simulate perceptually equivalent and indistinguishable results. Every individual piece of fruit has its own unique decay since it will have some random variation and growth. However, our simulation produces similar decay patterns to the real data.

A.6 Drawbacks

Though our model attempts to create a wide range of effects, it still does not demonstrate every single process occurring in fruit decay. We do not consider factors such as calcium deficiency, infection while on the vine, and poor fertilization. Nor do we consider if the fruit was sliced, and we also do not take into account any liquid phase losses from the fruit.
Figure A.6: A comparison of real photographs (right), and our simulation results (left) of a tomato and orange.

due to punctures. Certain fruits, when wounded, tend to leak volume from a tear in the surface as they decay. Lastly, we do not consider the effects of macroscopic organisms such as worms or flies. These creatures eat the internal substrate, produce new wounds, and change the underlying nutrient content.

A.7 Discussion

In this chapter, we present an approach for simulating fruit senescence and decay. Our method is perceptually similar to real photographs and time-lapse videos. The system takes advantage of a biologically-derived reaction-diffusion model to create growth patterns for areas of fungal and bacterial infection. We simulate interactions between multiple phenomena acting on both the surface and internal substrate. Additionally, we provide presets and parameters that allow a user to simulate a variety of novel effects. We
Figure A.7: Simulation results of a tomato and an orange under three different parameter settings set by three users from our user interface. Simulation (a) and (b) set two different pressure points and produce different results. (c) changes parameters in the mold growth to simulate different growth rates.

demonstrate our method on three fruits: an apple, orange, and tomato. To demonstrate our method’s generalization to any user-defined object, we imported a unique mesh into our user interface and simulated the decay processes based on user parameterization. Our model simulates a complex class of time-varying aging effects. In utilizing common production tools to render our images, we show that our final product has straightforward industry applications in movie and production pipelines that require images or animations of decomposing fruit.
Appendix B

Detailed Result Images

This appendix provides high resolution images of the results of using the simulation that have been presented in this dissertation. This appendix shows off the robustness of the $\mu$-ton system by simulating 10 different scenes and 17 different decay processes.

**Figure B.1:** This is a demonstration of a tomato and bowl fruit aging on a table scene.

**Figure B.2:** This is a demonstration of a tomato being aged.
Figure B.3: This is a demonstration of a second tomato being aged.

Figure B.4: This is a demonstration of a third tomato being aged.

Figure B.5: This is a demonstration of a final apple being aged.
Figure B.6: This is a demonstration of a second apple being aged.

Figure B.7: This is a demonstration of an orange being aged.

Figure B.8: This is a demonstration of a second orange being aged.
Figure B.9: This figure shows a log with moss growth.

Figure B.10: This figure shows a log with more moss growth.
Figure B.11: This figure shows a log with more moss growth.

Figure B.12: This figure is a blank wall with mold growth and propagation.
Figure B.13: This figure is a concrete wall with mold growth and propagation over time.

Figure B.14: This figure is a 2nd run of the mold growing on a wall.
Figure B.15: This figure shows the bowl after the aging process and mold has grown inside of it.

Figure B.16: This figure shows a close up image of the mold rendered on the bowl.
Figure B.17: This figure shows paint chipping away, rust, and pitted corrosion.

Figure B.18: This figure shows paint chipping away, rust, and pitted corrosion.
Figure B.19: This figure shows small a test scene of the curb crumbling.

Figure B.20: This figure shows our street scene of a curb crumbling and paint chipping away.
Figure B.21: A rendered alley scene with all the previous effects simulated.
Bibliography


[55] Paul Molin, Patrick Gervais, and Jean Pierre Lemi`ere. A computer model based on
reaction-diffusion equations for the growth of filamentous fungi on solid substrate.

[56] F. K. Musgrave, C. E. Kolb, and R. S. Mace. 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,
1989. ACM.

[57] Yuuji Ogasawara, Kazunobu Muraoka, and Norisige Chiba. Visual simulation of
moss taking into account local environment on temperature and humidity. The Journal

[58] Masahiro Ohgiwari, Mitsugu Matsushita, and Tohey Matsuyama. Morphological
changes in growth phenomena of bacterial colony patterns. Journal of the Physical

[59] Eric Paquette, Pierre Poulin, and George Drettakis. Surface aging by impacts. In

[60] Eric Paquette, Pierre Poulin, and George Drettakis. The simulation of paint cracking
and peeling. In Graphics Interface, pages 59–68. Canadian Information Processing

[61] J.C. Pech, M. Bouzayen, and A. Latch. Climacteric fruit ripening: Ethylene-
dependent and independent regulation of ripening pathways in melon fruit. Plant


