Surface Aging by Impacts

Eric Paquette, Pierre Poulin, George Drettakis

Surface Aging by Impacts

Eric Paquette[a,b] Pierre Poulin[a] George Drettakis[b]

[a]LIGUM
DIRO, Université de Montréal, Canada

[b]*i*MAGIS-GRAVIR, Inria
Sophia Antipolis, France [1]

Abstract

We present a novel aging technique that simulates the deformation of an object caused by repetitive impacts over long periods of time. Our semi-automatic system deteriorates the surface of an object by hitting it with another object. An empirical simulation modifies the object surface to represent the small depressions caused by each impact. This is done by updating the vertices of an adaptively refined object mesh. The simulation is efficient, applying hundreds of impacts in a few seconds. The user controls the simulation through intuitive parameters. Because the simulation is rapid, the user can easily adjust the parameters and see the effect of impacts interactively. The models processed by our system exhibit the cumulative aging effects of repetitive impacts, significantly increasing their realism.

Key words: Realism, simulation, aging, deterioration, imperfection, impact, surface, compaction.

Figure 1: Examples of aged objects.

1 Introduction

The quest for improved realism has always been an important goal for computer graphics. One major difficulty in this process is the extreme complexity of real objects, which exhibit many subtle variations over their entire surface. In contrast, synthetic objects often look "too perfect"; to achieve realistic synthetic images, these subtle variations need to be simulated.

One such important effect is the cumulative aging that affects all real objects. Detailed synthetic objects and environments rarely reflect the fact that people, objects, chemicals, etc., affect them over the course of time, increasing their complexity. Aging processes are numerous: abrasion, stains, pealing, dust, scratches, compaction, oxidation, etc. Examples of objects that cannot be realistically modeled without considering aging by impacts are presented in Figure 1.

Manually modeling the effects of complex aging processes is time-consuming: effective tools are needed to speed-up the design process. Textures are commonly used to add details to models. Unfortunately, they require lengthy, skilled user interaction and are not always suited to the representation of geometry modification. Texture parameterization can also be a daunting task, and may introduce unwanted distortions and discontinuities.

Many object-modeling techniques have been introduced to ease and accelerate the tasks of design and editing. Although well trained users can perform some of these tasks efficiently, the repetitive nature of the aging processes makes manual editing inconvenient. To ease the design of aged models, researchers have introduced different semi-automatic aging techniques. Most of those techniques cover only a specific aging effect, sometimes restricted to particular materials and objects. Many real aging processes result from *impacts* of moving objects onto other static objects (*e.g.*, dropping or throwing objects, moving furniture). Aging caused by repetitive impacts over long periods of time is one such process that has not yet received much attention.

In this paper we present a novel method that deals with long-term effects from moving objects that repetitively hit a static object. Such aging by compaction often modifies only the region close to the surface, adding small de-

[1] *i*MAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

pressions. This allows us to consider only surface-level impacts. Thus, we do not need a strict volume definition or representation. This is simple, does not restrict the type of input model, and permits efficient simulation.

Our semi-automatic technique uses empirical simulation to modify the object so that its surface reflects the many small depressions caused by individual impacts. The simulation first intersects the object with a user controlled tool. The model mesh is then adaptively tessellated so that it can represent the impact compaction. The surface geometry is finally updated to reflect the impact effect. This process is repeated for each generated impact. The simulation algorithm is efficient and simple to control, allowing the user to intuitively affect an object with impacts and interactively see the results. The aged objects can be rendered interactively or with a higher quality, but slower, renderer. The realism of the resulting aged models is significantly increased, as can be seen in the comparisons with real-world objects (see Figure 7).

2 Previous Work

We now present previous work designed for, or related to, surface details for aging purposes.

Texture mapping is commonly used to add details to models. By mapping an image onto the surface [5], one can modify its colors, the orientation of its normals [4], shading parameters, or any combination of these [7]. Displacement mapping can also be used to displace the surface along its normal. Creating textures manually (*e.g.,* painting) or selecting procedural texture [13] parameters is time-consuming and requires both artistic and technical skills. To speed up the design process, one can begin with photographs of the desired effect [27] or ready made textures from libraries which include textures of specific materials aged by common processes. Nevertheless, to get realistic results, the user must modify and combine many textures, controlling different shading parameters.

Geometry modification, in the context of modeling, is such a common task that many techniques specifically address this problem. We present those most related to surface details.

Sculpting is an intuitive technique to edit a model. Many sculpting systems use a volume representation [1, 15, 26]. These systems suffer from artifacts caused by the discrete and regular subdivision, and permit interactive editing of fine details only at the expense of high memory requirements (512 Mb or more[2]). BSP [23] and CSG [22] sculpting systems allow exact geometry modification and representation as opposed to the sampled approximation of volume sculpting. However, this advantage is also a drawback for repetitive modifications since the model representation grows without bounds. In most sculpting systems, the tool can freely penetrate the object to any depth as if the material offered no resistance. These systems often do not take into account the presence of the object when computing tool positions. Therefore, it is harder for the user to achieve realistic results.

Surgery simulation systems are designed for fast and accurate response to pressure [8] and cuts [3]. They work well for a few interactions, but they are not adapted for the many small interventions of repetitive processes.

Multiresolution mesh editing [19, 29] permits coarse to fine scale modification of the model. In these systems, coarse scale modifications are just as easy to perform as modifications at a fine scale. Nonetheless, fine scale modifications are as hard as in other manual modeling systems, and thus inconvenient for repetitive effects.

Reproducing aging effects is a difficult and time-consuming task. To get realistic results with less effort, semi-automatic aging methods have been introduced. Here we concentrate on surface aging techniques. There are two main approaches towards developing aging techniques, physically based and empirical.

Physically based techniques try to mimic the actual physical process by using laws of physics adapted to specific effects. Flow simulation [12] uses gravity and differential equations to approximate the interaction of rain on an object. The flow of water washes and soils different areas, giving realistic results on stone and concrete. The effects of water on stone [9] have been further developed with more sophisticated differential equations controlling the change of rock properties. Other physically based techniques simulate cracks [16] and fractures [24].

Physically based systems often result in impressive, realistic images. They also give, to some extent, physically accurate results, which may be useful for some applications. But this accuracy often results in complex and time-consuming systems, which are difficult to control.

Empirical techniques try to mimic the appearance of the effect, not the physical process that is taking place. These techniques often lead to simpler systems with more intuitive control. Accessibility shading [21] approximates the accumulation of dirt in corners and areas difficult to reach by considering local geometric factors. The accumulation of dust [18] has been specifically addressed using a mixture of accessibility shading and simple user controlled dust sources. To ease the design of metallic patinas textures, Dorsey and Hanrahan [10] brought together a collection of tools to design, combine, and render layered textures with variable thickness. Tracks left in soft materials [25] have been approximated by a simple compression and redistribution technique applied to a height field.

[2] From promotional material of the FreeForm system by SensAble.

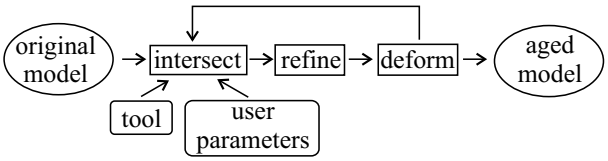

Figure 2: Simulation cycle of our system.

Some research has also been done on the integration of several aging processes into a single consistent aging system. Becket and Badler [2] present a text-based user interface for easy interaction. Wong *et al.* [28] use simple tendency sources to control the tendency of the surface to be affected by a particular aging process. For both methods, the effect is computed from the user data employing a predetermined set of standard or effect specific techniques and parameters. The overall generality of these systems is limited since they rely on many specialized algorithms, each dealing with one aging effect.

Synthetic object aging [11] is a general problem that has many applications in the computer graphics industry [14, 27]. Some commercial systems and tools were inspired by published research: *Dirty Reyes*[3] and *DirtMap*[4] are both based on accessibility shading, and *MultiPatinae*[4] is similar to the work by Dorsey and Hanrahan [10].

3 Compaction Simulation

We present an empirical aging technique that simulates surface compaction caused by repetitive impacts. In our system, dynamic objects (the *tools*) hit a static object (simply the *object*), repetitively compacting it. Our method simulates the deformation of the object surface caused by series of impacts. Each impact corresponds to a simulation step in which the system computes the surface deformation caused by the user controlled tool.

3.1 Overview

A simulation step starts with the selection of the tool, its properties and its trajectory from user-specified values or statistical distributions. The tool is intersected with the object surface to find the area which will be potentially modified. The object surface, expressed as a mesh, is adaptively refined in regions where it is hit by important tool features. This refinement proceeds until the impact effect can be represented on the mesh. The object surface is updated by modifying the adapted mesh geometry. This is done by moving the object mesh vertices. This whole process, shown in Figure 2, is repeated for each impact.

When the simulation is completed, the model is rendered in our 3D viewer or by a more sophisticated renderer. Since the geometry has been modified, normals need to be recomputed. These must be consistent with both the original normals and the adaptive refinement incurred by the simulation. Information about the normals is extracted in a preprocess and appropriate normals are assigned before rendering.

As in many other techniques (*e.g.,* [15, 22, 23, 25]), we ignore the aging effects on the tool for simplicity.[5] This restriction is not severe, since in most cases we see the aging effects on the affected object (*e.g.*, we see a scratch on the floor, but not the shoe that caused it) rather than the objects which cause the impacts (*e.g.*, the mugs, pens or boxes hitting a desk).

The following sections explain the different aspects of our method in more detail.

3.2 User Interface

For simple and intuitive user interaction, we restrict tool motion to linear trajectories only, which we will refer to as the tool *paths*. A linear tool path is computed from a point (source or target) and a direction. The point and direction can be specified or derived from the camera and cursor in the 3D viewer. For example, a ray can be shot in a zone around the mouse location to find the path target point, and the path direction can be computed from a cone of directions defined by the ray.

When interactively specifying paths with the 3D viewer, the user can specify how many paths are computed on a single mouse click. The computed paths can be executed immediately and/or recorded and stored in a file. This is useful when re-applying an effect after the original model has been edited.

The user also has control over the tool shape and size, and how the object will be modified in response to the impact. The response is specified as a *compaction volume*. The compaction volume is in modeling units and the user can specify a different one for each impact. In a simulation step, the object surface is moved until the swept volume is equal to the compaction volume specified by the user (see Section 3.4 and Figure 6(a)).

The values (scalars, directions, and points) specified for the path, tool size, and compaction volume can be used directly or as input to statistical distributions (*e.g.*, uniform, gaussian, poisson, turbulence, 3D field).

The new object geometry is approximated using an adaptively refined object mesh. The user selects a *feature size* that controls the resolution of the object mesh and the selection of important tool features. To capture finer details a smaller feature size is used, at the cost of a

[3] by *Reyes Infographica* `www.reyes-infografica.com`
[4] by *Phoenix Tools* `www.phoenixtools.com`

[5] This restriction could be removed at the expense of a higher computational cost which we think is not needed in most cases.

larger number of faces in the resulting mesh and slower simulation.

3.3 Modeling

A mesh is used to represent surface-level details. We chose meshes since they are widely used, and since other representations, such as implicit or parametric, can be converted to meshes. The common polygon "soups"[6] can also be converted, at least partially, into a coherent mesh. We use triangle meshes for simplicity, to avoid non-planar polygons and because fast intersection routines are available.

To compute the adaptive object mesh refinement, we must take the tool path and geometry into account. A straightforward approach would be to directly move the tool along the path, thus performing complex face/edge intersections with the object mesh. Instead, we simply project points along the path, from the tool onto the object faces. Thus the tool is represented as a mesh and has a collection of *feature points*. These points capture important features of the tool, and are used to adaptively refine the object. The feature points should represent the tool edges, and should accord more importance to those that are sharp and perpendicular to the path. In our implementation, we take into account the angle between the edge and the path (dot product between the edge and path direction vectors), curvature across the edge (dot product between the triangles normals), and the feature size to determine how many feature points will be assigned to an edge of the tool. The feature points are placed at evenly spaced locations with jittering to reduce the effect of regular patterns.

The object is a mesh augmented by simulation and rendering properties. The simulation property is a per vertex *deformation direction*. When the mesh is updated to represent the impact effect, the deformation could lead to self-intersections and distortions. To reduce these problems, we force the object vertices to move along a deformation direction. The direction for each vertex is computed in a preprocess, by finding all the faces sharing the vertex, computing the weighted average of their normals, normalizing the result, and assigning the opposite vector to the vertex deformation direction. The weight used is the angle subtended by the face at the vertex. In our tests, the deformation direction alone is enough to prevent self-intersections since we deal with small surface details. To strictly prevent self-intersections, the idea of simplification envelopes [6] could be used to forbid vertex movement beyond a distance which guarantees that no self-intersections can occur.

[6] List of polygons with no adjacency information, possibly affected by intersecting/overlapping parts, inconsistent front/back for adjacent polygons, and edges shared by more than two polygons.

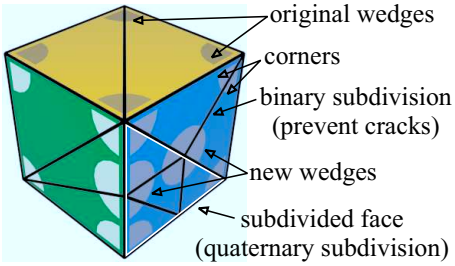

Figure 3: A triangulated cube with one face refined using quaternary subdivision. Adjacent faces are split to avoid T-vertices. Around each new vertex, one or two new wedges are inserted depending on the normal continuity across the edge.

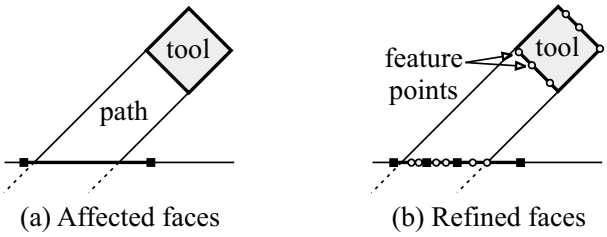

Figure 4: The first two stages of a simulation step: finding and refining the affected faces.

The object also stores rendering information. The original object mesh can have one different normal for each vertex/triangle *corner* (see Figure 3). To be able to recompute appropriate normals for the corners after the simulation, corners adjacent to each other and sharing the same normal are grouped into a *wedge* structure [17].

3.4 Simulation

A simulation step starts by generating the tool path, the tool size, and the compaction volume from the user-specified parameters. The simulation step proceeds through different stages shown in Figures 4 and 6. The set of faces that intersect the tool path are identified (Figure 4(a)). These faces are adaptively refined by projecting the tool feature points along the path (Figure 4(b)). Since we use linear tool paths, the projection can be computed by simple ray casting. If the number of projected feature points on a face is larger than a threshold, the face is subdivided. This is done recursively as long as the number of feature points projecting on a face is higher than the threshold, and the longest edge of the face is longer than the specified feature size. We use a quaternary subdivision of the triangles, with standard restriction and anchoring to avoid T-vertices (see Figure 3). To reduce the

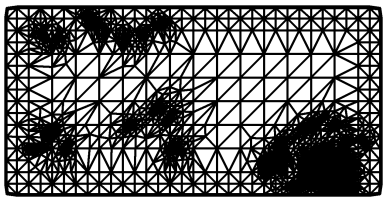

Figure 5: Adaptively refined mesh.

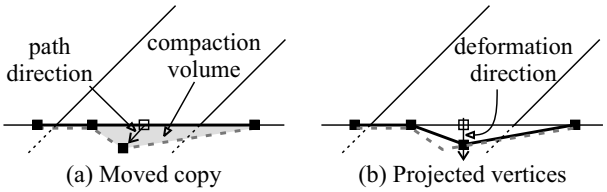

Figure 6: Stages 3 and 4 of a simulation step: moving the copy vertices and projecting the current vertices on the copy faces.

regularity of the quaternary subdivision process, we jitter the position of the new vertices along the edge. This jittering is typically done in a range of 40 percent of the size of the edge and can be controlled by the user. The deformation direction of a new vertex is obtained by linearly interpolating the values of the vertices defining the split edge. An adaptively refined mesh is shown in Figure 5.

When all the faces needed to represent the impact are available, their displacement is computed. Recall that vertex motion is restricted to the deformation direction to reduce self-intersections and distortions. We thus use a copy of the current faces and vertices to compute the movement in the path direction. The deformation process is computed in two stages: in the first stage we move the copy of the faces in the path direction (Figure 6(a)), and in the second stage we project the current vertices along their deformation direction onto the copied faces (Figure 6(b)). In the first stage, as the tool penetrates the object, the intersected copied vertices move in the path direction. More copied vertices are considered as the tool hits them. In this process, the corresponding copied faces also move. Their swept volume is recorded and the process stops when it is equal to the compaction volume, thus potentially leaving some copied vertices unprocessed. In the second stage, the movement of the current faces is computed by projecting the current vertices along their deformation direction on the copied faces. This approximation is close enough to the desired effect since we deal with small repetitive impacts on an approximating mesh.

3.5 Rendering

Since the simulation modifies the geometry of the model, normals must be recomputed for proper display. The simulation process does not require the new normals, so we compute them in a postprocess stage. The face normal alone is insufficient since it often introduces undesirable normal discontinuities along the face edges. In most cases normals must be specified per wedge, and Gouraud interpolation is used to get realistic results. The new normals must be consistent with the movement of vertices and the adaptive mesh refinement. Edges split by the simulation process are checked to determine if the normal across the edge is continuous or not, creating respectively one or two wedges associated with the new vertex and faces (see Figure 3). Similarly to the treatment of the deformation direction, all the faces sharing the same wedge are used to compute the weighted average of their normals which will be associated with the wedge.

After the normals are computed, the user can visualize the aged model using *OpenGL* in our 3D viewer or can save the model and use a standard renderer.

4 Results and Discussion

As with any method trying to reproduce natural phenomena, it is very instructive to compare our results with real images. In Figure 7(a)–(e), we show several real objects aged by impacts which we use as goals for our tests. We used the 3D viewer to interactively position and adjust impacts on the models. Large impacts were added one at a time, while others were added in groups by specifying the affected area and the number of impacts. We used various tools such as spheres, chisel-like "V" shapes, and simple squares. This entire interactive process of trial and error selection and adjustment of tools and parameters took between 30 minutes and 2 hours of user time per aged object. We believe that this compares favourably to the time required to manually add each individual compaction effect in order to get aged objects that look like the real objects. Also recall that once the user is satisfied with the aging, the whole list of impacts parameters can be saved in a file and re-applied whenever the original model is edited.

For rendering, the synthetic models were saved and rendered with *Maya 3.0* on a SGI Onyx (4 x R4400, 200 MHz processors, 512 Mb memory). We can see a great increase in realism between the synthetic images before (Figure 7(k)–(o)) and after (Figure 7(f)–(j)) aging effects are applied. The real and synthetic images show similar compaction aging effects. It is obvious that these effects could not be achieved only with noise functions. The system needs to be instructed by the user about the characteristics of the impacts. Note also that the corners of the

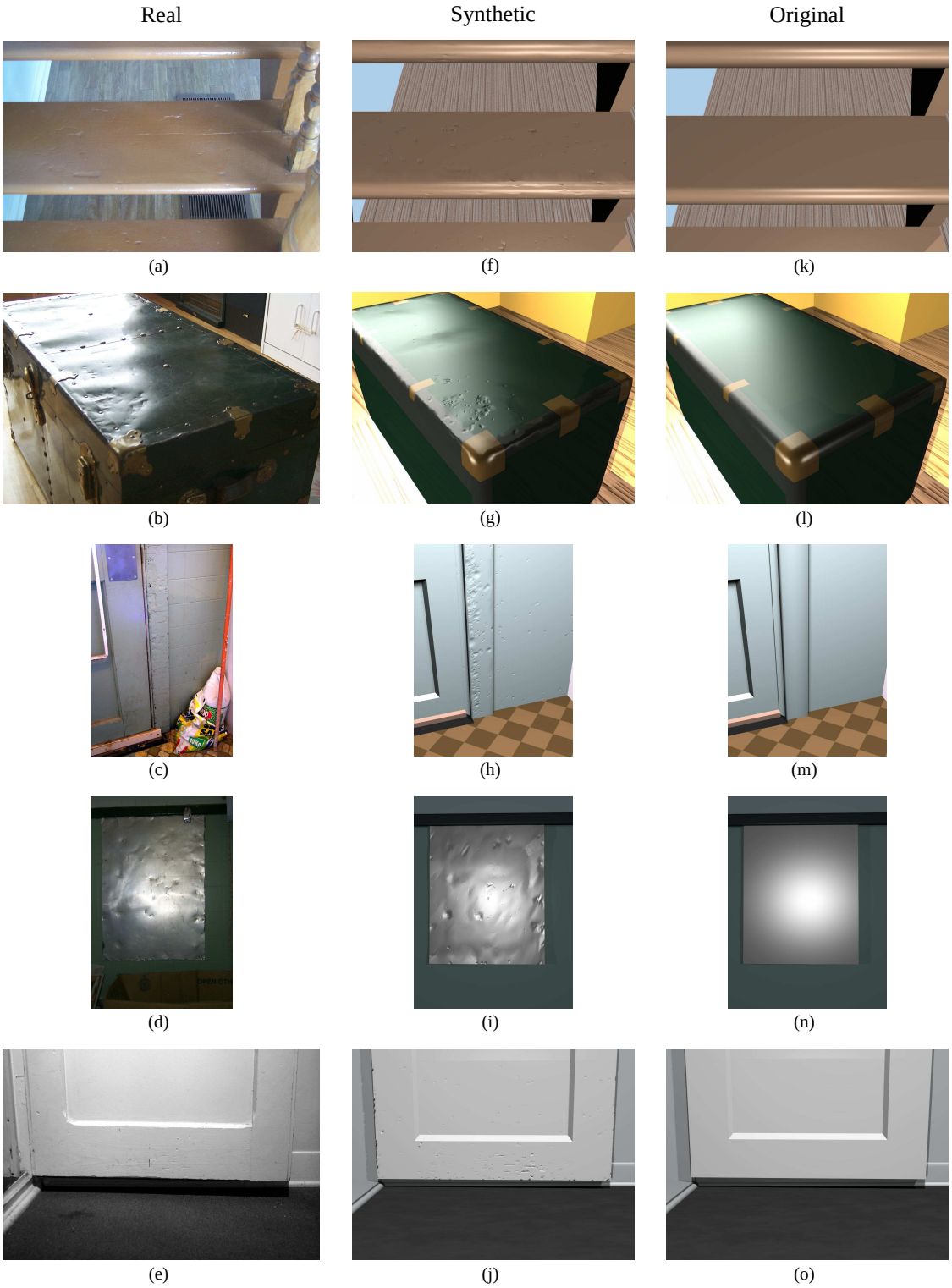

Figure 7: Real and synthetic images. (a) – (e) show photographs of real objects affected by compaction aging: stairs, a trunk, a door frame, a metal plate fixed on a wall, and the lower part of a door. (f) – (j) show synthetic versions aged by our system. (k) – (o) show the synthetic scenes before aging is applied.

	Stairs	Trunk	Frame	Plate	Door
Nb imp	128	149	300	234	276
Total	4.13s	4.78s	7.59s	6.17s	8.54s
Avg	32ms	32ms	25ms	26ms	31ms
Original	6k	2k	352	2	984
Final	22k	18k	23k	20k	25k
Regular	232M	80M	11M	16M	32M

Table 1: Simulation statistics: number of impacts, total simulation time in seconds, average step time in milliseconds, number of triangles in the original and final mesh, number of triangles if we used regular subdivision (m= 10^{-3}, k= 10^3, M= 10^6).

	Stairs	Trunk	Frame	Plate	Door
Affected	54%	40%	50%	40%	52%
Feature	1%	13%	3%	13%	2%
Adaptive	42%	34%	40%	33%	43%
Others	3%	14%	7%	14%	4%

Table 2: Time required for the different parts of the simulation: detect the affected faces, project the feature points, adaptive mesh refinement, and the other parts of the simulation.

trunk, the door frame and the door, show aging effects that would be harder to represent using a height field or a displacement map, because of the non-trivial parameterization that must take place across the edge to prevent discontinuities and self-intersection of the surface.

The aging simulations were ran on a Linux workstation (AMD Athlon 600 MHz processor, 256 Mb memory). Statistics are presented in Table 1. We can see that the simulation is quite efficient, applying hundreds of compaction effects in only a few seconds. With an average time per simulation step below one-tenth of a second, the user can interactively age the object. This is an important advantage of our empirical method over physically based approaches: the user has intuitive control over the system and can interactively see the resulting aging effects.

We can see from Table 2 that the most time-consuming parts are finding the affected faces and adaptively refining the model. Since the tool is usually small relative to the object, we took great care to quickly identify and keep track of the small set of affected faces and vertices that go through each stage of a simulation step. This significantly reduces the computations, except for the adaptive refinement which is quite involved, since we keep a consistent mesh representation while updating it with respect to the tool features.

Extracting the wedge information and computing the direction of projection in the preprocess phase take a few seconds, and the postprocess phase of computing appropriate wedge normals takes about one second. High quality images are rendered at a resolution of 1024x768 in 5 to 15 minutes. The increase in rendering time from the original to the aged objects is 20 percent on average, but varies between 0 and 80 percent depending on the object.

Another way to evaluate our method is to compare the maximal mesh size, defined by the user-specified feature size, to the adaptive refinement which minimizes unneeded mesh refinement. Table 1 shows that the increase in the number of triangles from the original model is far less than the increase if we used regular subdivision. Adaptive refinement is also useful compared to height fields or displacement maps in that its resolution locally adapts to the features that have to be represented. It also needs no parameterization and it is less prone to distortions. The refinement also shows no unwanted normal discontinuities while it can maintain sharp normal discontinuities, and naturally handles corners and curved areas. Its main drawback is that it cannot be filtered easily and may result in a large number of triangles.

5 Conclusion

We have presented a new aging technique that increases the realism of objects affected by repetitive compaction of their surface. In our empirical simulation, tools repetitively hit the object along user-specified paths. Each tool impact modifies the object mesh by adaptively refining it and moving its vertices. The adaptive refinement concentrates on important areas, and does not require any parameterization. With the vertex deformation direction, our method handles different features (*e.g.*, corners, curved and flat areas) in a unified manner which also reduces distortions and self-intersections. The simulation is efficient, computing hundreds of impacts in a few seconds.

The user has intuitive control over the modification by specifying a deformation volume and a feature size. Our technique can be easily used by an artist with no knowledge of physics, and since impacts can be generated at interactive rates, trial and error adjustments do not require lengthy simulations. Since our technique can take as input a model and an aging session file, it can be integrated in a production or rendering pipeline with limited effort.

6 Future Work

The realism of our aged models could be improved by using appropriate reflection models (currently only Phong reflection is used), and adding other aging effects such as pealing, abrasion, scratches, and dirt accumulation. Our method is not restricted to any specific refinement method. We think that using a proper multiresolution

mesh framework, while maintaining suitable adaptive subdivision, could enable appropriate mesh simplification and even allow easier conversion to displacement or bump maps. Since the tool is mainly represented by feature points, it should be possible to use any curved geometry. The interface could be improved with a mechanism similar to the Design Galeries [20] to present the user with effects of different tools and parameters.

Acknowledgements

Our work was supported by grants and scholarships from FCAR, NSERC, and MRI-MEQ. We thank *Alias|Wavefront* for the *Maya* system, and Jocelyn Houle, the reviewers, and everyone who gave us good advice from *i*MAGIS and LIGUM.

References

[1] A. Baerentzen. Octree-based volume sculpting. In *Proceedings of Late Breaking Hot Topics, IEEE Visualization '98*, pages 9–12. IEEE, 1998.

[2] W. Becket and N. I. Badler. Imperfection for realistic image synthesis. *Journal of Visualization and Computer Animation*, 1(1):26–32, August 1990.

[3] D. Bielser, V. A. Maiwald, and M. H. Gross. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum*, 18(3):31–38, 1999.

[4] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, volume 12, pages 286–292, August 1978.

[5] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of CS, U. of Utah, December 1974.

[6] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, Jr. F. P. Brooks, and W. Wright. Simplification envelopes. *Proceedings of SIGGRAPH 96*, pages 119–128, August 1996.

[7] R. L. Cook. Shade trees. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, volume 18, pages 223–231, July 1984.

[8] S. A. Cover, N. F. Ezquerra, and J. F. O'Brien. Interactively deformable models for surgery simulation. *IEEE Computer Graphics and Applications*, 13(6):68–75, November 1993.

[9] J. Dorsey, A. Edelman, H. Jensen, J. Legakis, and H. Pedersen. Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH 99*, pages 225–234, August 1999.

[10] J. Dorsey and P. Hanrahan. Modeling and rendering of metallic patinas. In *Proceedings of SIGGRAPH 96*, pages 387–396, 1996.

[11] J. Dorsey and P. Hanrahan. Digital materials and virtual weathering. *Scientific American*, pages 46–53, February 2000.

[12] J. Dorsey, H. K. Pedersen, and P. Hanrahan. Flow and changes in appearance. In *Proceedings of SIGGRAPH 96*, pages 411–420, August 1996.

[13] D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, October 1994.

[14] B. Fleming. *3D Photorealism Toolkit*. Wiley Computer Publishing, 1998.

[15] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, volume 25, pages 267–274, July 1991.

[16] K. Hirota, Y. Tanoue, and T. Kaneko. Generation of crack patterns with a physical model. *The Visual Computer*, 14(3):126–137, 1998.

[17] H. Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, February 1998.

[18] S. C. Hsu and T. T. Wong. Simulating dust accumulation. *IEEE Computer Graphics and Applications*, 15(1):18–25, January 1995.

[19] L. Kobbelt, S. Campagna, J. Vorsatz, and H. P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998.

[20] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of SIGGRAPH 97*, pages 389–400, August 1997.

[21] G. Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of SIGGRAPH 94*, pages 319–326, July 1994.

[22] S. Mizuno, M. Okada, and J. Toriwaki. An interactive designing system with virtual sculpting and virtual woodcut printing. *Computer Graphics Forum*, 18(3):183–193, 1999.

[23] B. Naylor. SCULPT an interactive solid modeling tool. In *Proceedings of Graphics Interface '90*, pages 138–148, May 1990.

[24] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99*, pages 137–146, August 1999.

[25] R. W. Sumner, J. F. O'Brien, and J. K. Hodgins. Animating sand, mud, and snow. In *Proceedings of Graphics Interface '98*, pages 17–21, June 1998.

[26] S. W. Wang and A. E. Kaufman. Volume sculpting. In *1995 Symposium on Interactive 3D Graphics*, pages 151–156. ACM SIGGRAPH, April 1995.

[27] R. Warniers. Dirty pictures. *Computer Graphics World*, pages 50–60, June 1998.

[28] T. T. Wong, W. Y. Ng, and P. A. Heng. A geometry dependent texture generation framework for simulating surface imperfections. In *Eurographics Rendering Workshop 1997*, pages 139–150. Springer Wien, 1997.

[29] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997.