Exemplar-based Worn Edges

C. Brachmann[1], B. Walther-Franks[1] and J. Loviscach[2]

[1]Fachbereich 03: Mathematik/Informatik, Universität Bremen, Bremen, Germany
[2]Fachbereich Elektrotechnik und Informatik, Hochschule Bremen, Bremen, Germany

Abstract

To counteract the sterile look of computer-generated worlds, we introduce a procedural model for worn edges that can be seen in buildings and stones: edges that have suffered minor damage at multiple points, resulting in material being chipped off. Employing data gathered from photographs, the method adds minute detail along exposed edges. The deformed geometry may be simulated with normal maps to also support real-time applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric algorithms

1. Introduction

In most 3D software the artist can fully control every detail of a model—or rather *must* fully control every detail, apart from basic randomization. However, the intricacy of real-world objects is hard to achieve by hand. Typical approaches to this problem concern aging and weathering effects through deep fractures, shallow cracks or changes in texture. The contribution of this paper is a method that focuses on the geometry of edges.

With the proposed method, the defects are not modeled through physical simulation, which typically is slow to compute and hard to steer. Rather, we rely on exemplar silhouettes gained from photographs or drawn freely. The basic idea of the method is to map the exemplars along "exposed" edges of the mesh, use them to cut it, and then fill the holes smoothly. Since such a geometric model often leads to a considerable number of polygons, we propose a further processing step for real-time applications: the chipped mesh is used to create a normal map for the original mesh.

This paper is structured as follows: Section 2 gives an overview of related work. Section 3 discusses the extraction of curves, Section 4 their mapping to the mesh. Section 5 describes the construction of the chip's inner surface. Section 6 details the results with normal mapping. The implementation is covered in Section 7. Section 8 concludes the paper.

2. Related work

Research into the graphical simulation of object aging has dealt with a variety of physical effects such as corrosion, dust accumulation, erosion, and crack formation. A comprehensive overview can be found in Lu et al. [LGG*07].

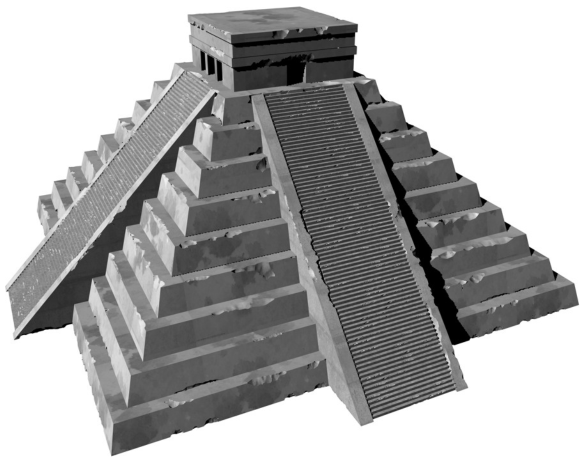

Figure 1: *The Mayan temple demonstrates worn edges added with our technique to a clean geometrical model.*

Close to the actual physics, Dorsey et al. [DEJ*99] employ voxels and Even and Gobron [EG05] use cellular automata to simulate erosion. These techniques yield smooth surfaces at the cost of rendering time or complex geometric and algorithmic structures. On top of that, they cannot easily reproduce the hard edges created by chipping.

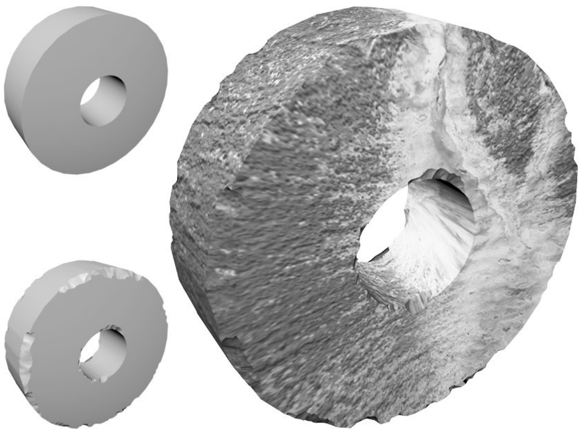

Figure 2: *A perfect ring shape (top left) is turned into a believable millstone (bottom left; right: with color texture).*

Crack formation methods come close to our objective, even though they address an object's faces rather than its edges. Many such methods are based on cellular automata [GC01], partially augmented by effects such as the shrinking of fragments [VPL06] or by more detailed physics [HTK00]. In a more heuristic fashion, Iben and O'Brien [IO06] create cracks using only a standard 2D mesh. Desbenoit et al. [DGA05] employ user-defined curves to generate the depth profiles of cracks.

Fracture simulations may be close to physics such as the finite-element approaches of O'Brien and Hodgins [OH99] or Bao et al. [BHTF07]. Pauli et al. [PKA*05] present a physically grounded meshless method to fracture a 3D object. Fracture effects may also be created from pre-formed fragments: Martinet et al. [MGDA04] slice through 3D objects with fracture masks. Socha and Wan [SW06] mark plausible fragments of an explosion in a preprocessing step.

Several works address the use of normal maps to change the look of edges: Cignoni et al. [CST02] emphasize sharp edges; Bahnassi and Bahnassi [BB07] simulate filleted edges with normal maps. In a similar vein, Softimage XSI offers a shader called "Rounded Corners."

3. Curve extraction

Our method is based on a simplified description of chips that occur along an edge between two approximately planar surfaces: each of the two surfaces contains a line that forms the corresponding boundary, see Figure 3.

To turn this observation into a modeling method, we collect exemplars of chips, each represented through two planar polylines that meet at their ends. One could employ a procedural approach or design such curves manually. Striving for a realistic look, we traced photographs, see Figure 4. To undo the perspective distortion, the following process can be employed: the photograph is placed as a backdrop in standard 3D software; auxiliary planes are matched by eye to the walls in the photograph; the boundary lines are traced by hand, using the auxiliary planes as 3D drawing surfaces.

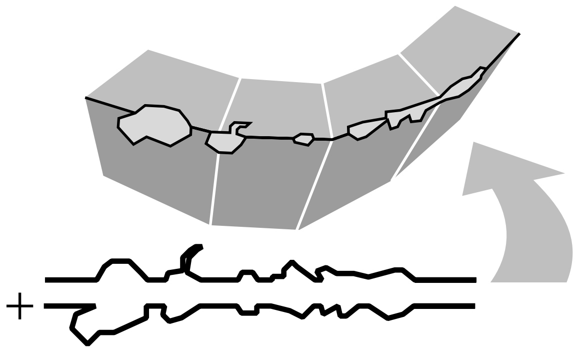

Figure 3: *The chips along a chain of edges are described by two curves contained in nearly planar adjacent surfaces.*

4. Curve mapping

Typically, only those edges of a mesh that are "exposed" to a certain degree will suffer a damage. In our examples, edges were treated as exposed if the normals of the adjacent faces bend away from each other by more than 60°. The terraces of Figure 1 are layered, separate blocks to allow defects along the seemingly concave edges as well, which looks more plausible for this building. Exposed edges are grouped into chains, as long as their direction does not change too abruptly. We used an angle of 60° for this threshold, too. Each of the chains is attacked as a whole. This helps to keep the results independent of the mesh's resolution.

For a given number of times, the method chooses the next chain of exposed edges and the next exemplar of the collection of curve pairs, starting from the first exemplar if it arrives at the end of the list. The exemplars are applied one after the other, each time operating on the mesh resulting from the previous step. Thus, a later cut may carve into an earlier one. However, to reproduce the look of the given exemplars, such overlaps have to be prevented. Thus, the exposed

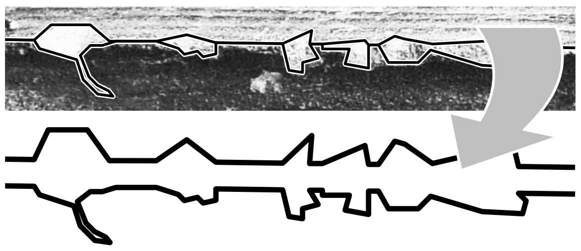

Figure 4: *Defect shapes may be collected from photographs.*

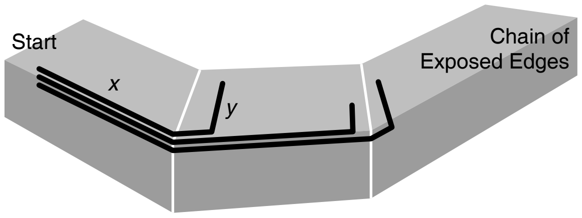

Figure 5: *The x and y of an exemplar's vertex are traced across the mesh to find the position the vertex is mapped to.*

edges are marked on the initial mesh and tracked during the split and removal operations. After a given chain of exposed edges has been attacked, the chipped part is removed from the list, leaving the remainder of the chain open to further attacks.

For each exemplar to be applied, the algorithm chooses a starting point on the selected chain. It identifies the straight centerline of the exemplar with the chain of exposed edges through the path length measured from the starting point. No exemplar is used for a chain that is shorter than the exemplar itself.

The two polylines forming the exemplar are mapped onto the mesh, using the centerline for alignment. Every vertex (x, y) of both polylines is placed as follows: the length x is traced along the chain of exposed edges from the starting point; from the point thus found, a path of length y is traversed across the mesh, approximately perpendicular to the chain. The path's endpoint forms the mapped position of the vertex. To improve the continuity along curved chains, the "y" paths do not start perfectly perpendicular to the chain, but in linearly interpolated directions, see Figure 5.

Such a path traversed in y direction across the surface may run through a number of triangles. The vertices found at the ends of all these paths are connected into a polyline. For the most part, this polyline already is the mapping of the exemplar polyline onto the mesh. The polyline may, however, not be completely contained in the mesh surface, as it may jump over or under edges crossing it. To correct this, additional points along these edges are added to the polyline. On top of that, this polyline may fold over itself if the chain of exposed edges is strongly curved, the vertices' distance from the centerline is large, and/or the density of vertices is large. As a remedy, overlaps in the polyline are removed.

5. Chip surfaces

To create the chips, the polygons are spliced along the polylines mapped onto the mesh. The polygons enclosed by the chip outline are removed from the resulting mesh. The hole thus created is filled with new triangles, leveraging the corresponding modeling command of Cinema 4D, the host software we're using. The mesh forming the defects' surface is

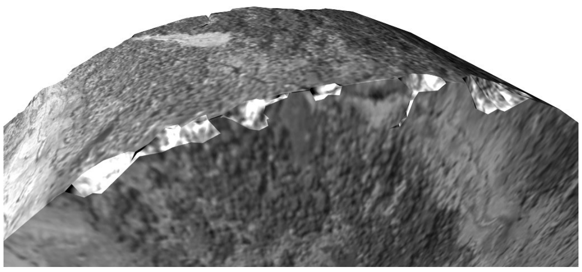

Figure 6: *The defects' surfaces are smoothed and may carry a special material.*

subdivided. Its inner part can be translated along the inverted vertex normals to push it toward the inside.

The user can choose whether or not the outer edges and/or the inner edges of the chip should break Phong interpolation. The standard setting is to break Phong interpolation for the outer edges but keep it for the inner ones, so that the surface of the defect appears smooth. With the help of an automatically created polygon selection, the material of this surface can be set differently from that of the regular surface. This helps to create the look of a coated surface, the base material of which is revealed through the damages, see Figure 6.

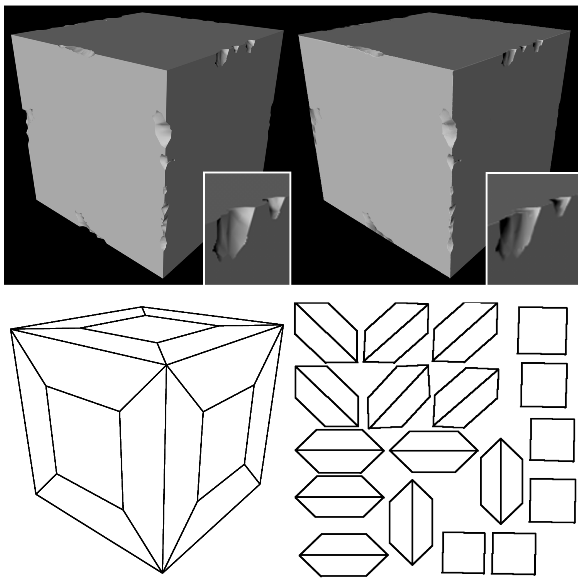

Figure 7: *Actual deformation (top left) can be simulated through a normal map (top right). The automatic generation of this map requires an uncommon tessellation (bottom left) with uv continuity across the edges (bottom right: uv space).*

Object	# of Defects	Time	# of Triangles Before	# of Triangles After
Millstone	17	0.4 s	256	5,476
Temple w/o stairs	254	32.3 s	1,300	89k
Temple with stairs	917	94.7 s	5,236	251k

Table 1: *On a 1.3 GHz Intel Pentium Mobile processor, the unoptimized algorithm requires about 0.1 s per defect.*

6. Simulation through normal maps

To simulate bump-like defects through a normal map has become standard in real-time applications. In our case, the idea is to create a normal map from the deformed version of the mesh and then apply this normal map to the much simpler original mesh. Only an astute observer will notice the difference from actual geometric distortion if the chips are sufficiently small, see the insets in the upper half of Figure 7.

The normal map is created by "baking" the normal vectors of the deformed mesh into an image file, employing the *uv* coordinates of the vertices as positions on the image. To yield a clean mapping of the defects, the *uv* texture coordinates have to be continuous across the edges of the original mesh. This may require an unusual tessellation, see Figure 7.

7. Implementation

The prototype has been implemented as a plug-in for Maxon Cinema 4D release 10. The user may adjust parameters such as the number of attacked edges, the polygonal resolution and the global scale of the exemplars. This scale may vary randomly by a given amount. The depth of the centerline of the chips may be shallower or deeper. The chips may be scattered evenly along the lengths of the chains of exposed edges or they may occur close to the centers of those chains.

Table 1 contains benchmark data for the objects of Figure 1 and 2. Cinema 4D handles arbitrary n-sided polygons, the numbers of which are hard to compare. Thus, the table specifies the numbers of triangles after triangulation.

8. Conclusion

We have presented a method to add natural geometric details to the edges of 3D objects. It is steerable through outline curves that may be designed freely or may be traced from photographs. With the help of a normal map, the method's results can be applied in real-time settings, too. Future work may address the temporal evolution of damage and may incorporate mesh smoothing to simulate micro-scale erosion. One may evaluate spatially averaged curvature to also find and attack beveled edges. To chip off corners, one may specify one curve for each face adjacent to the corner or map a single closed outline onto all faces meeting at a corner.

References

[BB07] BAHNASSI H., BAHNASSI W.: Micro-beveled edges. In *ShaderX^5: Advanced Rendering Techniques*, Engel W., (Ed.). Charles River Media, 2007, pp. 23–30.

[BHTF07] BAO Z., HONG J.-M., TERAN J., FEDKIW R.: Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 370–378.

[CST02] CIGNONI P., SCOPIGNO R., TARINI M.: Normal enhancement for interactive non-photorealistic rendering. In *Eurographics 2002 Short Presentations* (2002), pp. 95–103.

[DEJ*99] DORSEY J., EDELMAN A., JENSEN H. W., LEGAKIS J., PEDERSEN H. K.: Modeling and rendering of weathered stone. In *Proc. of SIGGRAPH '99* (1999), pp. 225–234.

[DGA05] DESBENOIT B., GALIN E., AKKOUCHE S.: Modeling cracks and fractures. *Visual Computer 21*, 8-10 (2005), 717–726.

[EG05] EVEN P., GOBRON S.: Interactive three-dimensional reconstruction and weathering simulations on buildings. In *Proc. of CIPA 2005* (2005), pp. 796–801.

[GC01] GOBRON S., CHIBA N.: Crack pattern simulation based on 3D surface cellular automata. *Visual Computer 17*, 5 (2001), 287–309.

[HTK00] HIROTA K., TANOUE Y., KANEKO T.: Simulation of three-dimensional cracks. *Visual Computer 16*, 7 (2000), 371–378.

[IO06] IBEN H. N., O'BRIEN J. F.: Generating surfaces crack patterns. In *Proc. of SCA 2006* (2006), pp. 177–185.

[LGG*07] LU J., GEORGHIADES A. S., GLASER A., WU H., WEI L.-Y., GUO B., DORSEY J., RUSHMEIER H.: Context-aware textures. *ACM TOG 26*, 1 (2007), to appear.

[MGDA04] MARTINET A., GALIN E., DESBENOIT B., AKKOUCHE S.: Procedural modeling of cracks and fractures. In *Proc. of SMI* (2004), pp. 346–349.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH '99* (1999), pp. 137–146.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ; P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM TOG 24*, 3 (2005), 957–964.

[SW06] SOCHA J., WAN J. W. L.: Graph-based crack formation algorithms for rigid body explosions. In *Proc. of CASA '06* (2006), pp. 159–168.

[VPL06] VALETTE G., PRÉVOST S., LUCAS L.: A generalized cracks simulation on 3D-meshes. In *Eurographics Workshop on Natural Phenomena 2006* (2006), pp. 7–14.