2010-01-01

Directable Weathering of Concave Rock Using Curvature Estimation

Matthew Beardall
mbeardall@byu.edu

Joseph Butler

McKay Farley

Michael D. Jones
mike.jones@byu.edu

Original Publication Citation

M. Jones, M. Farlay, J. Butler and M. Beardall, "Directable Weathering of Concave Rock using Curvature Estimation", IEEE Transactions on Visualization and Computer Graphic, IEEE Press, Vol. 16(1), pp.7-8. January 21.

Directable Weathering of Concave Rock Using Curvature Estimation

Michael D. Jones, *Member*, *IEEE*, McKay Farley, Joseph Butler, and Matthew Beardall

Abstract—We address the problem of directable weathering of exposed concave rock for use in computer-generated animation or games. Previous weathering models that admit concave surfaces are computationally inefficient and difficult to control. In nature, the spheroidal and cavernous weathering rates depend on the surface curvature. Spheroidal weathering is fastest in areas with large positive mean curvature and cavernous weathering is fastest in areas with large negative mean curvature. We simulate both processes using an approximation of mean curvature on a voxel grid. Both weathering rates are also influenced by rock durability. The user controls rock durability by editing a durability graph before and during weathering simulation. Simulations of rockfall and colluvium deposition further improve realism. The profile of the final weathered rock matches the shape of the durability graph up to the effects of weathering and colluvium deposition. We demonstrate the top-down directability and visual plausibility of the resulting model through a series of screenshots and rendered images. The results include the weathering of a cube into a sphere and of a sheltered inside corner into a cavern as predicted by the underlying geomorphological models.

Index Terms—Physically based modeling, modeling packages.

1 Introduction

This work simplifies the creation of weathered rock for use in computer-generated animation or games. Weathered rock is difficult to create algorithmically because it almost always includes overhangs or concavities, which do not have a simple heightmap representation. Tools for creating weathered rock should also support the creative artistic process and fit into the existing animation production pipeline.

The prevalence of weathered rock as visual cue in film indicates that directors find weathered rock useful in telling a story. Sandstone rock, in particular, has been widely used to create a feeling of being somewhere remote, exotic, or unique. Sandstone terrain as a visual cue can be found in live-action film [1], numerous automobile commercials, documentary works [2], and in both hand-drawn [3] and computer-generated animation [4]. In one live-action film [5], sandstone rock formations were used as the setting for scenes on an alien planet.

Many algorithms have been published for terrain generation in computer graphics but none have adequately addressed the artistic creation of weathered rock. Most algorithms are difficult to control and do not admit concave surfaces. With a few exceptions, terrain generation algorithms operate on heightmaps in which a 2D array of elevations represents the height of the terrain at grid points. The heightmap representation is compact and easily processed but is poorly suited for concave surfaces in which each grid point may be associated with more than one elevation.

Three algorithms for concave terrain generation have been presented. Gamito and Musgrave present a purely procedural approach to concave terrain, which is limited to terrain described by analytically differentiable functions [6]. Both Ito et al. and Beneš et al. give algorithms on voxel grids. Ito et al. simulated rockfall using graph component analysis [7] and Beneš et al. simulated hydraulic erosion [8] using computational fluid dynamics. Both Ito et al. and Beneš et al. create convincing concave terrain, but neither address the problems of weathering and top-down control.

Algorithms for weathering exposed rock have also been presented but these algorithms are either too slow or do not admit concave surfaces. Musgrave et al. reduce steep slopes into more gradual slopes until the slope reaches the angle of repose [9] but do not include concave surfaces. Dorsey et al. convincingly weather exposed concave stone [10] but the process is slow and not designed for directability.

Commercial landscape modeling tools, such as [11], [12], do not support the manipulation of 3D shapes using physically based algorithms such as weathering or erosion. These tools do support physically based manipulation of height fields but do not apply to concave rock. Weathering and erosion could be performed manually using 3D modeling packages, such as [13], [14], but this would involve a tedious process of moving vertices and edges. Wang and Kaufman created a volumetric sculpting tool that allows a user to remove varying sizes of sections from a voxel representation of clay [15]. Such a tool could also be used to manually simulation weathering and erosion but at the cost of moving voxels rather than vertices and edges.

This paper builds on our preliminary work on spheroidal weathering of sandstone goblins (which are a kind of hoodoo), as reported in [16]. In this paper, we extend the original algorithm by including the sign of the surface curvature rather than just the magnitude. This allows us to simulate two kinds of weathering using curvature estimation.

- *M.D. Jones, M. Farley, and J. Butler are with the Department of Computer Science, Brigham Young University, 3328 TMCB, Provo, UT 84602. E-mail: jones@cs.byu.edu, macfarley5@gmail.com, joebutler@byu.net.*
- *M. Beardall is at Caselle, Inc.—Engineering, 1570 N Main St, Spanish Fork, UT 84660. E-mail: beardalls@gmail.com.*

Manuscript received 23 May 2008; revised 5 Dec. 2008; accepted 4 Mar. 2009; published online 24 Mar. 2009.
Recommended for acceptance by B. Guo.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2008-05-0068.
Digital Object Identifier no. 10.1109/TVCG.2009.39.

We also integrate the user interface with the simulation to improve directability, and add rockfall and colluvium deposition to improve realism. In our previous work, the user interface allowed durability to be set to constant values at a fixed number of fixed locations in the rock. In this paper, we present a user interface, which allows the user to vary rock durability during simulation at any altitude and for any number of vertical locations.

This work uses an estimation of the signed mean surface curvature and top-down interactive control to address the problem of directable weathering of concave rock. Efficient estimation of the signed curvature by counting voxels containing air or stone in bubbles centered on the rock surface allows us to simulate both spheroidal and cavernous weathering. The geological foundation [17], [18], [19] for this approach is that exposed rock with high positive curvature has more exposure to air and wind than less exposed rock. Protected rock, which has high negative curvature, retains moisture that accelerates decomposition of the cementation between particles in rock. Rockfall and colluvium deposition are added to enhance the look of weathered rock shapes.

Top-down interactive control is provided by a durability graph in which the user edits a collection of line segments, which represent the durability of each rock layer as a function of the altitude. The profile of the final rock shape matches the shape of the durability graph and the durability graph can be edited during the weathering simulation. This creates a feedback loop in which the user can control the weathering simulation by changing simulation inputs based on simulation outputs.

The user interface mitigates problems with the run-tweak-run cycle typically found in simulation for graphics as the artist tweaks the parameters and repeats the process. Our approach to top-down directability was inspired by Prusinkiewicz et al.'s use of curves for artistic control of plant generation [20] and is similar to Dobashi et al.'s use of profiles to suggest cloud shapes [21].

When the user is satisfied with the weathered rock shape, a triangular mesh is extracted, which can be imported into the production pipeline. Any of a number of third party modeling and rendering tools can then be used to tweak, texture, and render the object.

Examples of weathered rock shapes are given in Section 4. These results illustrate the control provided by the interface and the variety of shapes generated by the weathering simulation. In the next section, we review related work in terrain generation and weathering. We then discuss spheroidal and cavernous weathering, which are the geological foundations for our algorithm. The algorithm appears in Section 4 and we discuss the user interface in Section 5. We conclude in Section 7 and discuss ideas for future work in concave terrain generation.

2 Related Work

This work is most closely related to other work in terrain generation and weathering for computer graphics. Our algorithm is a geomorphological based approach to concave rock formation, which relies on a phenomenological approach to weathering simulation in that we do not simulate the chemistry of rock decomposition through exposure to the elements. The geomorphological theories of weathering on which our simulation is based are discussed separately in the next section.

2.1 Foundational Work

Our work is built on the foundation created by several key papers, which we highlight here.

Colluvium is deposited using an algorithm similar to Musgrave et al.'s thermal weathering simulation [9] and has been adapted to work on a voxel grid rather than a heightfield. Rockfall is simulated using a simplified version of Ito et al.'s rockfall simulation [7] in which we perform the component analysis to identify rocks that are not connected to the ground but we do not compute friction angles and perform a slippage analysis to move rocks that are connected to the ground but lie on an unstable slope.

Editing durability graphs in the user interface is a kind of top-down approach to artistic modeling, which was inspired by Prusinkiewicz et al.'s use of curves to describe position information for L-systems [20] and is similar to Dobashi et al.'s more recent work on controllable cloud generation [21]. As with Dobashi et al.'s work, the final shape may not match the desired profile due to the nature of the underlying natural phenomena. However, this is desirable because it creates a plausible shape, which matches the artist's intent.

We estimate curvature using a new application of a technique first used to model diffusion-limited aggregation [22]. Later, Pimienta et al. verified that the actual curvature of a circle is linearly related to the estimated curvature and applied the technique to sintering simulation [23]. Vicsek and Pimienta both estimated curvature in two dimensions. We independently reinvented the idea but in three dimensions.

2.2 Terrain Generation

Terrain generation has a long and productive history in computer graphics. Here, we discuss only the most relevant work. Terrain generation algorithms can be divided into phenomenological approaches and geomorphological approaches. Phenomenological approaches begin with the appearance of terrain and design algorithms that create shapes, which mimic that appearance. Geomorphological approaches begin with a geological process, which might have created the terrain, and design algorithms which simulate that process with the goal of generating the right appearance. Phenomenological approaches mimic the appearance but ignore the process while geomorphological approaches mimic the process to get the appearance.

It should be noted that all geomorphological approaches become phenomenological at some level. Otherwise, geomorphological approaches would require the simulation of physics down to the atomic or even subatomic level. As computational power has increased, geomorphological approaches have become more complex and realistic.

We briefly summarize key work in both approaches.

2.2.1 Phenomenological Approaches

Phenomenological approaches can be traced back to Mandelbrot's observation that certain mountain ranges resemble the trace of a particle moving in Brownian motion in 1D. Generalizing to 2D gives a striking approximation of

a mountain range [24]. Further generalization of fractional Brownian motion (fBm) gives terrain with varying degrees of ruggedness. Later, Fournier et al. simultaneously invented a more efficient approximation of fBm using midpoint displacement [25].

The primary drawback of fBm terrain is that the frequency distribution of the resulting terrain does not vary with altitude as it does in nature[1] and that it does not include rivers and river networks, which are an important part of mountain range appearance. Geomorphological approaches to rivers and erosion were then added to fundamentally fBm models (i.e., [26], [9], [27], [28]) and will be discussed below with other geomorphological approaches.

Other purely phenomenological approaches to improving fBm terrain were undertaken by Mandelbrot, Voss, and Prusinkiewicz. Mandelbrot used scaling in postprocessing as well as other displacement methods to differentiate valleys and mountains in fBm terrain [24]. Voss also scaled fBm terrain as a postprocessing step for a similar purpose [29]. Prusinkiewicz et al. used rewriting systems on fBm terrain to model rivers as squig-curve fractals [30]. The resulting terrain includes rivers, but the displacement of the terrain around river channels looks unrealistic in places.

Later, Gamito and Musgrave presented a new model of phenomenological terrain based on the displacement of vector fields. Vector field displacement describes a new kind of terrain, which allows overhangs [6]. The terrain images in [6] are not particularly realistic and require the surface to be described by a differentiable function, but the work is significant because it both presents a different approach to phenomenological terrain and allows overhangs. The method does generate realistic models of breaking ocean waves.

The compelling advantage of phenomenological terrain is that it can efficiently be generated and is viewable at many scales using a small amount of data. This can be done because the surface can procedurally be generated at arbitrary resolutions on-the-fly. This is useful for long tracking shots, which include viewing terrain from hundreds of kilometers above and from millimeters away from the terrain surface. These kinds of shots are unusual in film and are not addressed by geomorphological approaches including our work on weathering presented here.

The main problem with phenomenological terrain is that it is not always clear how the input to a phenomenological algorithm relates to the final terrain shape. In phenomenological approaches, designing terrain requires understanding seemingly unrelated mathematical objects. Our work is fundamentally geomorphological and avoids this problem by simulating a process. This, however, creates a new set of problems.

2.2.2 Geomorphological Approaches

Geomorphological approaches began in an effort to make phenomenological mountain ranges look more like mountain ranges by mimicking the effects of river systems. Kelley et al. gave the first model of river flow in a mountain range [26]. The results are visually plausible and even more impressive given the computational resources of the time. Musgrave et al. then presented an erosion and weathering algorithm for fractal terrain, which begins with a heightmap generated through fBm [9].

As computing power increased, it became possible to write increasingly accurate models of hydraulic erosion. Nagashima simulated erosion on geologically distinct rock layers in a flat plain [31]. The resulting terrain resembles entrenched meandering rivers in sandstone canyons. Later, Chiba et al. calculated velocity fields from particle flow simulations and used these fields to erode terrain [27].

Beneš et al. perform hydraulic erosion by simulating 3D fluid flow on voxel grids using the Navier-Stokes equations [8]. This model includes overhangs and other types of surfaces created through fluid flow. The results are extremely realistic and the process is fully general, but the simulation is tractable on terrains consisting of on the order of only $120 \times 32 \times 120$ voxels. Šťava et al. integrated several hydraulic erosion processes into a single algorithm that is implemented on the GPU [32]. This method can erode scenes consisting of $2{,}048 \times 1{,}024$ pixels at 20 frames per second, but is based on a heightmap rather than a voxel grid, and thus, cannot represent concave terrain.

Other work in geomorphological terrain generation has centered on weathering. Weathering was beyond the scope of Kelley's early work on geomorphological terrain generation but is included in Musgrave's work on fractal terrain. Musgrave et al. give an algorithm for simulating a process they call "thermal weathering." Thermal weathering models the process by which steep cliffs become sloping hills through the accumulation of talus at the base of the cliff. Musgrave's weathering model is effective but does not admit concave surfaces. We use Musgrave's model as the foundation of our colluvium deposition algorithm. Approaches to weathering for objects other than terrain features will be discussed separately in the next section.

Ito et al. modeled the geomorphological process of rockfall in jointed cliff faces [7]. The algorithm uses a search through an undirected graph representing connections between blocks of rock to find blocks, which form connected components. A simulation of gravity and slope is used to determine which component is most likely to fall next. The algorithm results in convincing images of cliff faces particularly those found by the sea and in the presence of prevailing winds and tides.

The collective advantage of geomorphological approaches is that they generate a wider variety of terrain, which is more visually plausible than most terrain generated using phenomenological approaches. Phenomenological approaches have had success generating mountainous terrain, which is often viewed from above at a distance. Geomorphological approaches can be used to create terrain features, such as rock columns, which look realistic from smaller viewing distances. This is particularly useful in film for scenes, which are set in or near terrain features.

A key limitation of geomorphological approaches is that they are highly specialized and can be difficult to control. Even Beneš' work, which simulates generic fluid flow, mimics only a single geomorphological process. Our work is specialized as well but we are able to generalize curvature estimation to mimic two weathering processes. We address

1. The accumulated effects of weathering and erosion reduce high-frequency variations in older rock which tend to lie at lower altitudes.

problems with controllability using a top-down interactive control during the simulation.

2.3 Weathering

Previous work in weathering simulation has focused on either convex terrain, as mentioned above, or man-made objects as will be discussed here.

Dorsey et al. designed a weathering algorithm based on a simulation of fluid flow beneath the surface of an object [10]. Slabs of voxels were aligned perpendicular to surface normals and used in the weathering simulation. Voxel slabs were used to create a detailed simulation grid near the surface of an object while ignoring the center. The resulting algorithm includes what Dorsey calls "corestone weathering," which is similar to spheroidal weathering. We use a single axis-aligned voxel grid rather than slabs of voxels aligned to surface normals.

Chen et al. simulate weathering by tracing the flow of γ-ton particles through a scene [33]. The main advantage of γ-ton tracing is that it can propagate weathering effects globally throughout a scene. In the simulation, γ-ton particles cause weathering or staining when they strike a surface. If a γ-ton ray causes a particle to separate from the surface, then subsequent interaction points between the γ-ton particle and the surface can deposit accumulated material. Emitters of γ-ton particles can be defined to simulate, for example, stains created by dripping water. Computing global weathering effects in a scene is much like computing global illumination in a scene and is computationally difficult. Chen's model required 90 minutes of computation time to weather a scene with 50,000 vertices.

Hsu and Wong also use a ray casting technique but do so to simulate dust accumulation on 3D objects [34]. They derive an equation, which is based on the angle between the surface normal and direction of the dust source and an approximation of the surface friction. We use a simpler method to calculate where eroded voxels fall and accumulate colluvium at the base of the rocks.

Our algorithm is less physically based than Dorsey's or even Chen's but can only simulate a single process, while Dorsey's and Chen's can mimic several processes including staining. Like Dorsey's and Chen's models, our algorithm allows the deposition of weathered material at the base of the object being weathered, which is similar to how Hsu and Wong simulate dust accumulation.

2.4 Estimating Curvature

We estimate curvature by computing the ratio of rock to air in a fixed volume. This approach to estimating curvature avoids the differentiation of functions, which describe the surface. Sarracino et al. give a series of differential equations, which describes weathering due to fluid flow through rock in three dimensions, but the resulting equations have been solved only for the two-dimensional case [35].

Vicsek first proposed estimating curvature at point x by counting discrete units of volume within a fixed distance from x [22]. Later, Pimienta et al. verified that the actual curvature of a sphere is linearly related to the estimated curvature and applied the technique to sintering simulation [23]. Like Sarracino, Vicsek and Pimienta estimated curvature in two dimensions.

Hsu and Wong use ray casting to determine exposure of surface points as part of their dust accumulation algorithm [34]. They cast random rays at each point on the surface to determine the surface exposure. The surface exposure is used to scale dust accumulation at each point. This is similar to our approach to concave weathering using negative curvature. This approach may produce better surface curvature estimates than our bubble-based method for points that lie inside thin-walled caverns.

Miller proposed a voxel-based method for determining accessibility of 3D objects for shading purposes [36]. His method involves calculating an offset surface in voxel space by placing a voxel sphere at each voxel. The accessibility at each point is radius of the smallest sphere that touches empty space in the offset representation. Wang and Kaufman use a similar approach for rendering [15]. They center spheres around each point in voxel space and use ray tracing to render a smooth surface representation of the voxel grid.

Our curvature estimates are based on the same concept as of Sarracino, Vicsek, and Pimienta, but are done in three dimensions. Miller also worked in three dimensions but did not use spheres to estimate curvature. We also modify curvature estimates by counting discrete units of volume in skewed volumes, rather than spheres, in order to exaggerate curvature in a given dimension and change the resulting shape of the weathered rock.

3 Weathering

In this section, we describe spheroidal and cavernous weathering from a geological perspective and discuss three other factors, which influence the weathering rate in natural rock. These factors are: durability variations between rock layers, durability variations within rock layers, and the ratio of exposed surface area to volume per unit volume of rock. Variations in the weathering rate are responsible for the visually interesting shapes of natural rock, and therefore, important elements in our weathering algorithm.

In weathered rock, a pile of weathered material called colluvium can be found at the base of the rock formation. Colluvium is soft and can quickly be transported by air or water and is important to the appearance of a rock formation.

The discussion of spheroidal weathering is based on Milligan's geological description of the rock formations in Goblin Valley State Park, Utah [19], and the discussion of cavernous weathering is based on Huinink et al.'s and Turkington's description of small cavern formation in a variety of sandstone formations [17], [18].

3.1 Spheroidal Weathering

Spheroidal weathering is a rock changing, or geomorphological, process by which sharp corners are reduced to rounded corners in exposed rock. When rock cracks, sharp angular edges may be formed. Spheroidal weathering gradually reduces those sharp edges to rounded corners. Milligan's theory is that spheroidal weathering occurs in certain sandstones because sharp angular edges have more exposed surface area per unit of volume than flat surfaces.

Fig. 1. Spheroidal weathering in soft sandstone. Spheroidal weathering will eventually reduce the sharp edges on the left to the rounded edges seen on the right.

Fig. 2. Cavernous weathering on a canyon wall.

It is called "spheroidal" weathering because it tends to produce sphere-like shapes.

The spheroidal weathering rate is proportional to the ratio of surface area to volume in a unit volume of rock. A unit volume of rock on a corner has more exposed area than a unit volume on an edge, which has more exposed area than a unit volume on a face. Because the unit volume on a corner has more exposure to air and other weathering elements, a unit volume on a corner will weather more quickly than a unit volume on an edge or face.

The ratio of surface area to volume at a given point is proportional to the positive surface curvature at that point. A sharp corner or edge has high positive curvature while a flat surface has zero curvature. In this framework, spheroidal weathering is a process that equalizes positive curvature across the surface of a rock.

The photograph in Fig. 1 highlights the effects of spheroidal weathering in soft sandstone near Cohab Canyon in Capitol Reef National Park, Utah. The sharp edges on the left are the result of recent rockfall and have not yet been rounded by spheroidal weathering. The rounded edges are the result of spheroidal weathering following past rockfall events.

3.2 Cavernous Weathering

Cavernous weathering is a geomorphological process by which caverns, or pits, form in the rock surface. Fig. 2 contains example of caverns formed through cavernous weathering in Forgotten Canyon, Utah. The rate of cavernous weathering depends on both material durability and surface curvature, as with spheroidal weathering. However, the cavernous weathering rate is proportional to negative curvature rather than positive curvature. Cavernous weathering equalizes negative curvature on the inside of a curve up to a limit.

Cavernous weathering increases with negative curvature because surface points with negative curvature tend to be protected from exposure by adjacent rock. These protected regions hold moisture longer than neighboring, exposed rock, and this leads to increased deposition of salt (when compared to exposed neighboring rock). Increased salt concentrations accelerate the breakdown of chemical cementation between adjacent rock particles and cause the protected rock to disintegrate more quickly.

The foregoing description of cavernous weathering follows the salt weathering theories advanced by Huinink and Turkington. Huinink explains that as water collects at the surface of the rock, this water is absorbed by the rock at a rate related to the porosity of the rock. When this wet layer reaches a certain saturation level, the transportation of salts dissolved by the water becomes possible. The folds and curvature of the rock are important because they allow for uneven drying across the rock surface. If parts of the surface dry at slower rates, they will remain above the saturation threshold where salt transportation is possible, thus allowing these areas to collect more salt from the surrounding rock.

Turkington also suggests that caverns are likely to begin on areas of the surface that are more shaded from the sun than neighboring areas. These shaded areas are more likely to retain the water for more time than areas that are not shaded, thus adding to the amount of salt deposited. Huinink states that depressions that exist in the rock create a difference in the airflow above the surface of the rock, which further slows the rate of evaporation. With higher levels of salt being deposited in these areas, the rate of salt weathering will be higher than surrounding areas, thus weakening and eventually creating and enlarging these depressions into caverns.

4 Algorithm

The weathering algorithm uses surface curvature estimation to simulate two weathering processes. The weathering rates depend on curvature as well as durability variations between and within rock layers. Weathering is simulated for surface voxels in a voxel grid in a time-based simulation. We first discuss the data structures used in the algorithm, then present the algorithm, and finally, give a detailed discussion of curvature estimation. Important implementation details are given at the end of the section.

The simulation also includes colluvium accumulation and rockfall. Rockfall is a reimplementation of Ito et al.'s rockfall algorithm [7] based on component analysis, but without friction, and is not discussed further.

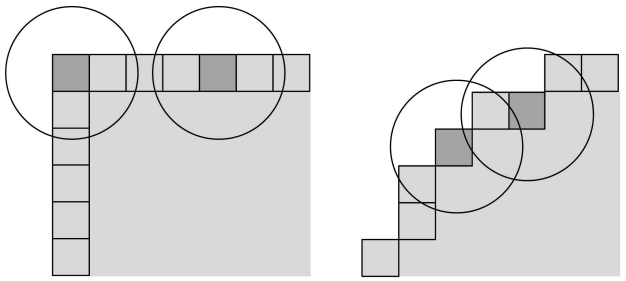

Fig. 3. Weathering rates are computed using curvature estimates based on the percentage of air contained in a bubble centered on each voxel on the rock surface.

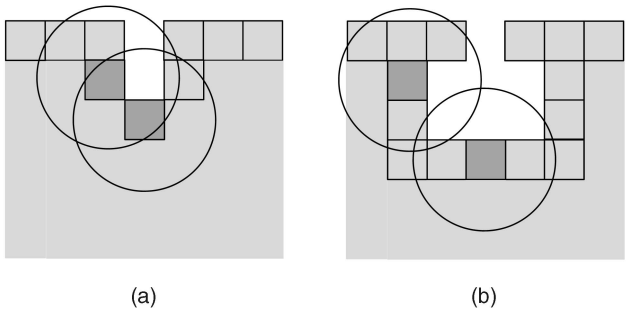

Fig. 4. Cavernous weathering equalizes the rate of change of negative curvature across the surface. The surface curvature is estimated to be negative if more than half of the bubble contains rock.

4.1 Data Structures

The weathering algorithm is defined on a uniform voxel grid in three dimensions. Each voxel in the grid represents a cubic unit of either air, rock, or colluvium.

Voxels containing air have no associated properties because air is modeled as a static object. The rock voxels are organized into layers by altitude as defined by the user. Layer boundaries are limited to horizontal planes. Each layer has a durability, which is defined using the durability graph. Perlin noise [37] can also be added to vary layer durability. Variations in layer durabilities cause variations in weathering rates.

Each rock and colluvium voxel is associated with a durability and a decimation. The durability is the sum of the durability of the enclosing layer and the value of a noise function evaluated at the voxel location.

Voxel decimation is a measure of the degree to which a voxel has weathered. Voxels are initialized with a decimation of 100. When decimation reaches 0, the voxel is fully decimated, removed from the surface, and added to a colluvium pile. Colluvium voxels fall and settle to an angle of repose using a process similar to Musgrave et al.'s thermal weathering algorithm [9].

4.2 Simulation Algorithm

The weathering simulation is based on computing the ratio of air to rock in a bubble centered on each voxel. The spheroidal weathering rate is proportional to this ratio and the cavernous weathering rate is inversely proportional. The geological justification for this approach to spheroidal weathering is that the weathering rate increases as the amount of exposed surface area per unit of volume increases. Fig. 3 illustrates the approach to spheroidal weathering in the two-dimensional case. Fig. 3a shows two bubbles centered on unweathered rock and Fig. 3b shows two bubbles centered on weathered rock. Initially, the bubble centered on the corner contains a greater percentage of air than the bubble centered on an edge. After weathering, the corner is rounded and the bubbles contain similar percentages of air. The final effect of spheroidal weathering is to equalize the rate of change of positive curvature across the rock surface.

Fig. 4 illustrates the process for cavernous weathering. Fig. 4a contains rock before cavernous weathering and Fig. 4b shows the same rock after weathering. The geological justification for this approach to cavernous weathering is that moisture accumulates in caverns as the surrounding rock, which is more exposed, dries. Cavernous weathering tends to equalize the rate of change of negative curvature on the surface by increasing the cavern size.

The simulation algorithm is a synchronized time-based simulation in which each simulation step represents a single atomic unit of time. This approach is simple but looses accuracy because all particles which erode in a given unit of time erode at the same instant regardless of when they actually eroded during that time unit.

The simulation can be designed as an asynchronous event-based process in which particles erode at any time. This can be done using a time-sorted event queue rather than a for-loop over time. We have implemented both methods and found that the time-based simulation, as described below, is more efficient and stable than the event-based method.

The pseudocode for the weathering and voxel erosion algorithms is shown in Fig. 5. The core of the algorithm is a loop (line 3), which traverses the surface voxels in a voxel grid V to compute the decimation experienced by each voxel during a single time step. The weathered surface is redisplayed after each simulation step and the user can terminate weathering and export a surface at any time.

For each voxel v at or near the surface, we estimate a weathering rate by computing the percentage of voxels, which contain air in a bubble b centered on v by counting the number of voxels containing air v_a inside b and dividing by the number of voxels $|b|$ in b (line 4):

$$v_{air} = \frac{v_a}{|b|}.$$

We then use this percentage to determine the decimation experienced by the voxel. Decimation due to spheroidal weathering is computed (line 5) using the durability at v, $v_{durability}$, and a function f (lines 6 and 8), which can scale or clamp the curvature estimate to produce different effects. The relationship between curvature and decimation rate is not established in the geology literature. The function f is a tunable parameter, which is set to produce plausible and controllable results. Decimation due to cavernous weathering is computed (line 7) using a similar process, but v_{air} is inverted to make the weathering inversely proportional to the ratio of air to stone in the bubble.

```
1   Algorithm Weathering (voxel grid V)
2   while not done
3     foreach voxel v on the surface of V
4       v.air = percentage of air in a bubble centered on v
5       v.decimation = v.decimation -
6                      (f(v.air)/v.layer.durability)
7       v.decimation = v.decimation -
8                      (f(1 - v.air)/v.layer.durability)
9       if (v.decimation ≤ 0) then v.erode()
10    extract and redisplay surface using marching cubes
11    if (user clicks done button) then done = true
12  invoke marching cubes to export a surface for rending
13
14  Algorithm voxel::erode()
15  if (this voxel is colluvium)
16    remove this voxel from voxel grid V
17    initialize newly exposed voxels in voxel grid V
18  else
19    c = new colluvium voxel at location of this voxel
20    remove this voxel from voxel grid V
21    initialize newly exposed voxels in voxel grid V
22    repeat
23      g = angle of steepest down slope at (c.x, c.y, c.z)
24      s = direction of steepest down slope at (c.x, c.y, c.z)
25      while (g > angle of repose and
26              the voxel in V at (c.x, c.y − 1, c.z) ≠ air)
27        (c.x, c.y, c.z) = move c in direction s
28        g = angle of steepest slope at (c.x, c.y, c.z)
29        s = direction of steepest slope at (c.x, c.y, c.z)
30      while (the voxel in V at (c.x, c.y − 1, c.z) = air)
31        c.y = c.y - 1
32    until (g ≤ angle of repose and
33           the voxel in V at (c.x, c.y − 1, c.z) ≠ air)
34    c.layer.durability = durability of layer containing c
35    add c to voxel grid V
```

Fig. 5. Weathering and erosion algorithms.

Bubbles are stored as a three-dimensional cube of Boolean values, which indicate whether or not the corresponding voxel is in or out of the bubble. In order to determine which voxels are in the bubble, we treat the bubble space as a coordinate space with the center of the bubble at the origin. We then calculate the distance to each point in bubble space from the origin. For sphere-shaped bubbles, the array value for a voxel is set to true if the distance from the center to the voxel is less than the radius. Bubbles with different shapes can be defined by modifying the criteria used to determine whether or not a voxel lies in the bubble.

The bubbles used to estimate curvature can also be defined as a mixture of primitive bubble types with the mixture controlled by the value of a Perlin noise function. As with durability variations within a layer, varying bubble types also produces irregular rock shapes. The bubble distribution is set by partitioning the range of a noise function into b equal intervals, where b is the number of bubble types to be mixed, then assigning the bubble type based on the range in which the noise value falls.

If the new decimation value $v_{decimation}$ is less than or equal to the minimum decimation value, then v is fully decimated at which point it erodes (line 9) and becomes colluvium. The colluvium then settles to a stable location and begins to weather. The colluvium voxels take on the characteristics of the voxels within the layer in which they settle. However, if v is already colluvium (line 15), then v is simply removed from the terrain and that voxel becomes air.

Musgrave's thermal weathering algorithm is the inspiration for our colluvium distribution algorithm. In Musgrave's algorithm, material on a steep slope is distributed to neighboring locations, which results in talus[2] slopes at the base of mountains. Our weathering simulation also mimics the accumulation of material but does so in three dimensions on a voxel grid rather than on a heightmap.

Colluvium distribution is a two-step process, which is repeated until the colluvium comes to rest in a stable location (lines 22-32). First, we allow the colluvium to slide (lines 25-29) down steep slopes until reaching the angle of repose. If the downward slope in the direction s of the steepest downward gradient g is steeper than the angle of repose, then the colluvium particle moves one unit in the direction s. Then, we allow the colluvium to fall by iterating toward the ground in the $-y$ direction until reaching a nonair voxel (lines 30 and 31). We repeat both steps until the colluvium particle lies on either rock or colluvium on a gradual slope at or below the angle of repose (line 32).

Gradient is estimated by counting the number of voxels containing air beginning at vs elevation continuing downward to the first nonair voxel. This is done in each of the four connected columns surrounding v. The direction of the steepest gradient s points toward the column adjacent to v, which contains the most air voxels below v. The steepest gradient g is compared with the angle of repose by estimating the slope angle using the number of voxels containing air between vs elevation and the first nonair voxel below. For example, if the angle of repose is 45 degrees and the steepest gradient lies in a column with one voxel containing air between vs elevation on the first nonair voxel, then colluvuium movement halts because $\arctan(1/1) = 45$.

When the user decides that the weathering is complete (line 11), the marching cubes algorithm [38] is implemented to generate a surface, which can be imported into third party rendering tools. The colluvium and rock are exported as different surfaces in the same file so that different textures can easily be applied to each. We duplicate geometry wherever the rock and colluvium are adjacent to prevent gaps between the colluvium and rock surface. Before the surfaces are split into rock and colluvium, the vertex positions are averaged with their neighbors' to smooth the surface boundaries.

4.3 Curvature Estimation

The algorithm estimates the mean curvature at surface voxel v by counting voxels in a fixed volume around v. Given a three-dimensional curved surface, the mean curvature H at some point p on the surface is

$$H = \frac{\kappa_1 + \kappa_2}{2},$$

2. Talus is a kind of colluvium which is composed of rock.

where κ_1 and κ_2 are the principle curvatures at p. The principle curvatures at p are the maximal and minimal normal curvatures at p. The sign of κ_1 or κ_2 is positive if the surface curves away from the surface normal and negative if the surface curves toward the surface normal. In general, points which lie in a concavity have negative curvature and points which lie on corners have positive curvature.

The algorithm estimates mean curvature rather than Gaussian curvature (which is the product $\kappa_1\kappa_2$) because the sign of the mean curvature is a better predictor of the ratio of exposed surface area to volume than the sign of the Gaussian curvature, and this ratio is a key factor in the underlying geomorphological process. For example, the Gaussian curvature is positive for points which lie in a concavity because both principle curvatures are negative.

The ratio of rock to air in a fixed volume around surface voxel v is a reasonable approximation of the mean curvature at v if the surface is smooth and thick in the neighborhood of v. The neighborhood of v is the area contained in the volume used to approximate curvature at v. A surface is smooth at v if the rate of change of the direction of the surface normal in the neighborhood of v is constant or small. A surface is thick at v if the neighborhood of v contains no surface points with surface normal equal to the negation of the surface normal at v.

Let r represent the voxels containing rock and a represent the voxels containing air within the neighborhood of v. If the surface is smooth and thick at v, then the mean curvature at v is proportional to $a - r$. If the surface is flat at v, then $a - r = 0$ and the mean curvature is 0. If the surface is concave at v, then $a - r < 0$, because more voxels contain rock than air and the mean curvature is also less than 0. Similarly, if the surface is convex at v, then $a - r > 0$, because more voxels contain air than rock and the mean curvature is also greater than 0.

There are several significant cases in which the estimated curvature is incorrect. A voxel, which lies on a saw-tooth surface, will have estimated curvature at or near 0 if the neighborhood is sufficiently large. A voxel which lies on a flat thin wall on the inside of a concavity will have estimated large positive curvature because air outside the wall will be included in the neighborhood.

We can skew the estimated curvature to exaggerate the contribution of the normal curvature to the mean curvature in a given direction by changing the shape of the bubble. For example, an ellipsoid bubble elongated in the vertical direction includes more voxels containing rock if the normal curvature in the vertical direction is negative. This, in turn, overestimates the contribution of vertical curvature in the estimate of the mean curvature.

4.4 Implementation Details

The implementation is designed to save space and time. We save space by instantiating only the voxels on or near the rock surface. A three-dimensional array of pointers to voxel objects represents the simulation space. This reduces space overhead by allowing us to store a pointer per voxel cell rather than an entire voxel object per voxel cell.

Fig. 6a identifies three groups in the voxel cell space which can be handled differently to save space. Air and unexposed rock can be represented by copies of the null pointer and copies of a pointer to a default rock object. Only surface voxels require representation by unique objects. Array entries for voxels on the surface point to the voxel object for that surface location. When default rock voxels become exposed to weathering, a new voxel object is created and the pointer in the array is updated. Colluvium voxels are handled like exposed rock voxels.

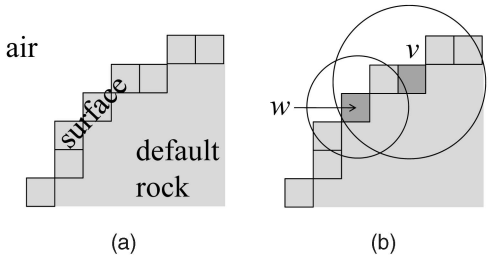

Fig. 6. (a) Voxels are instantiated only for voxels, which lie on the rock surface. Voxels containing air and unexposed rock are collectively represented by either the null pointer or a pointer to a default rock object. (b) Time is saved by storing rather than recomputing the weathering rate. However, this requires voxels to notify each other when the weathering rate may change. When voxel w becomes fully decimated, voxel v must update its weathering rate.

The weathering algorithm, as written, has a time bound that contains a cubic term in the bubble radius because the entire volume of each bubble must be visited in order to compute the percentage of the bubble, which contains air. If the bubble radius is R and the number of voxels within d units of the surface is V, then each round of simulation will require time on the order of

$$O(R^3V).$$

Although the bubble radius R is often small (on the order of four or five) compared to the number of voxels V (on the order of 10,000s to 100,000s), the operation is costly because it must be performed once per voxel per weathering step. We reduce the time bound on each weathering step by computing the percentage of air in a bubble once and storing information related to the weathering rate. However, that information must also be updated as the rock shape changes.

After the initial computation of the percentage of air in a bubble, the ratio is used along with the durability to compute a weathering rate and determine when the voxel will become fully decimated. If voxel v weathers at rate r_v, then r_v is stored with v and reused during subsequent simulation steps. The bubble is then ignored until v becomes fully decimated or until a voxel in the bubble for v becomes fully decimated.

If the bubble for voxel v contains a voxel w and w becomes fully decimated before v, then v must update its weathering rate based on the loss of w, as illustrated in Fig. 6b. When voxel w becomes fully decimated, voxel w notifies all neighbors within a distance of r from w on the rock surface. The update radius r is chosen so that all voxels which may contain w in their bubbles are notified.[3] When voxel v is notified that a voxel in its bubble has become fully decimated, voxel v increases its weathering rate r_v.

3. The update radius r is not just the radius of the bubble centered on w because neighboring voxels may have differently sized bubbles.

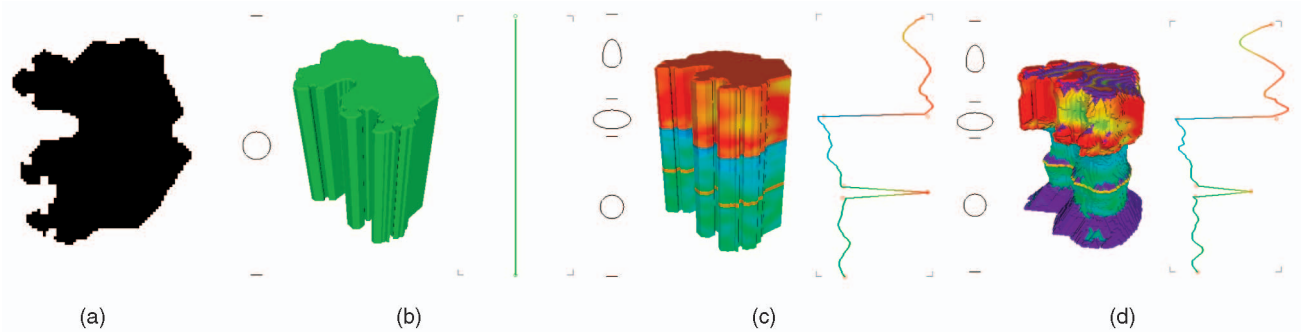

Fig. 7. Steps in making a weathered rock column.

If D voxels become fully decimated and N voxels are newly exposed behind the decimated voxels, then new time bound for a single simulation cycle is, on average,

$$O(R^3N + R^2D + RV), \quad (1)$$

because the weathering rate for the newly exposed voxels must be initialized, the neighbors of the fully decimated voxels must be notified, and the rest of the voxels checked to see if they are fully weathered.

5 User Interface

The user interface is designed to give the user top-down artistic control over the weathering process while allowing the simulation to manage the problem of creating the look of weathered rock based on that input.

User input is required to set the initial rock shape, the durability of rock layers, and to determine when the simulation is complete. User input is given by editing durability and bubble graphs. The durability graph allows the user to specify the target profile of the weathered rock. The final rock profile may not exactly match the durability graph profile, but results in a weathered rock with a similar shape.

Fig. 7 illustrates each step in the creation of a weathered rock column. First, the initial shape of the rock column is specified by a black and white image as shown in Fig. 7a. That image is used to create a rock column as shown in Fig. 7b.

The durability for each rock layer is set using the durability graph shown on the right side of Fig. 7b. In the durability graph, the vertical axis corresponds to altitude and the horizontal axis corresponds to durability with greater durability to the right. The user clicks on the line segment to create or select a vertex, which can then be dragged to alter the durability. Clicking the right mouse button on a durability graph line segment allows the user to increase or decrease the period or amplitude of the Perlin noise for that line segment. The color of the line segment and the surface of the rock are keyed to the durability with red hues representing durable rock and blue hues representing less durable rock. Later, in the simulation, purple will be used to indicate accumulated colluvium.

The bubble sizes and shapes are set using the bubble graph shown on the left side of Fig. 7b. As with the durability graph, clicking on the graph creates or selects the boundary between two bubble types. The bubble type includes a bubble collection type in which a set of bubbles is distributed across the surface and used to estimate the curvature.

Fig. 7c shows the final bubble and durability graphs before weathering. The bubble graph specifies three bubble shapes and sizes. In the durability graph, the segments between two layer endpoints are curved based on the amplitude and frequency of the noise added to the durability of each segment. The effects of the noise can also be seen in the coloration of the rock column.

Specifying durability by editing curves gives the user visual feedback on the approximate profile of the future appearance of the weathered rock. Durability can also be edited during the weathering process. This allows the user to interactively change the weathering effect during the weathering simulation. The process is reminiscent of wheel-thrown pottery in which the artist adjusts the shape of the piece during the creation of the piece.

Fig. 7d shows the rock column after 12 cycles of weathering simulation. The durability graph was edited during weathering to modify the Perlin noise and reduce the durability of the small layer in the middle of the base. The profile of the final rock is similar to the profile of the final durability graph. Parts of the rock that are covered by colluvium are shown in purple.

The user can then implement the marching cubes algorithm to export the rock and colluvium surfaces. The resulting surfaces can be imported into third party tools for rendering. The weathered shape in Fig. 7d was used to create the image shown in Fig. 8 using Vue-6 from e-On software.

Fig. 8. A scene containing the final weathered rock shape from Fig. 7d along with several other weathered rock shapes which were also created using the weathering simulation.

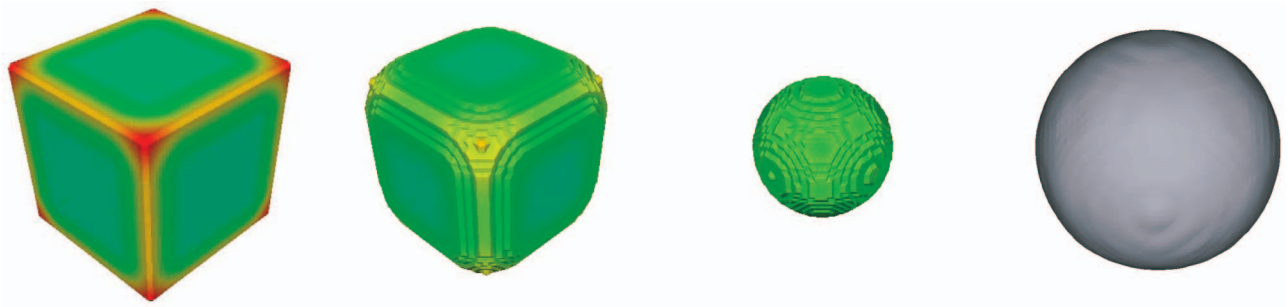

Fig. 9. Spheroidal weathering equalizes positive curvature across the rock surface. This process transforms a cube of uniform material into a sphere. The first three images depict that process and the final image shows the final sphere at a higher resolution. Surface color for the first three images is a function of the estimated curvature with red corresponding to estimated positive curvature.

6 Results

The results demonstrate the use of positive and negative curvature in weathering simulation, user control over the weathering process, and the variety of rock shapes which can be created using the algorithm.

6.1 Curvature in Weathering Simulation

Estimates of mean curvature are used to control the weathering rate. Surface points with large positive curvature experience spheroidal weathering while surface points with large negative curvature experience cavernous weathering.

Fig. 9 shows the estimated curvature on the surface of a cube. Red and orange colors correspond to large positive curvature while green corresponds to low or no curvature. After 13 rounds of spheroidal weathering, the cube weathers into a sphere as predicted by the underlying geomorphological theory. Curvature on the final sphere is more uniform than curvature on the original cube. The final image shows the surface of the sphere modeled at a higher resolution. Resolution in the user interface (from which the first three images were captured) is kept low to reduce the time spent extracting a surface from the internal voxel representation.

Fig. 10 contains a rock shape with a fold, which creates a region of negative curvature. Surfaces with large negative curvature are colored blue. Cavernous weathering equalizes negative curvature by creating spherical caverns inside the rock shape. The image on the right side of Fig. 10 contains a cavern in which the negative curvature is more uniform than in the original rock shape. This shape was generated by applying 20 rounds of cavernous weathering with a round of spheroidal weathering between the 17th and 18th rounds. Spheroidal weathering was applied to make the cavern visible by removing the outer cavern walls (which have high positive curvature). Rockfall simulation removed material left suspended by the weathering process. Colluvium deposition was also disabled. The rock durability was set to restrict the cavern to the middle of the rock by placing a less durable layer between two durable layers.

The curvature estimation overestimates positive curvature for voxels on flat faces near corners, edges, or caverns. Curvature is estimated by checking voxel contents in a bubble surrounding a voxel. The curvature estimate is incorrect near edges, corners, and caverns because these voxels are surrounded by more voxels, which contain air rather than rock. Similarly, curvature estimation underestimates negative curvature for voxels inside thin-walled caverns. Curvature estimation through ray casting, as in [34], may reduce this error.

Errors in the curvature estimation can be seen in Figs. 9 and 10. The cube in Fig. 9 includes regions of high estimated positive curvature near the corners. These regions are shown in red. Similarly, the weathered rock on the right side of Fig. 10 includes high estimated positive curvature on the rock face to the right of the cavern. In both cases, the actual surface has zero curvature on a flat rock face. In the case of caverns, the curvature estimate is more skewed because the volume behind the flat cavern face contains air rather than rock.

Skewed curvature estimates using asymmetric bubbles can be exploited to obtain different weathering effects. Fig. 11 shows the shapes created by weathering a cube using curvature estimated by, respectively, a sphere, a gumdrop, and a disc. Colluvium deposition was disabled during weathering simulation for these shapes. Fig. 12 shows the effects of colluvium deposition on weathering. The shape in Fig. 12 was created using the same process as

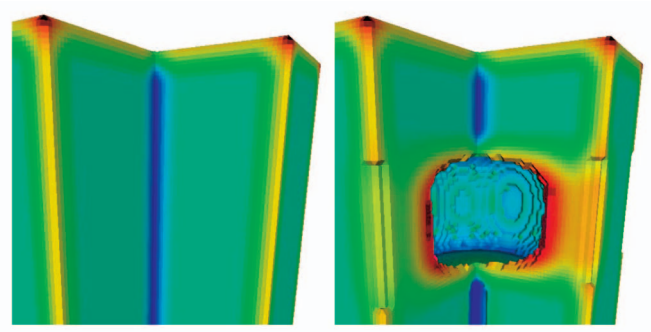

Fig. 10. Cavernous weathering equalizes negative curvature by creating spherical caverns inside the rock. Color is based on estimated curvature with blue corresponding to estimated negative curvature.

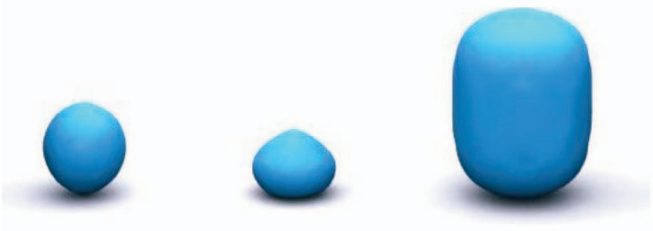

Fig. 11. A cube weathered into different shapes using different bubbles shapes to estimate curvature. The weathering shape resembles the bubble shape.

Fig. 12. A cube weathered using spheroidal weathering but with colluvium deposition enabled.

the spheres in Figs. 9 and 11 but with colluvium deposition enabled. Colluvium protects rock at the base, which causes the base to weathering more slowly.

Fig. 13 shows the effect of varying the function, which relates the curvature estimate to the decimation experienced by a voxel (this is the function f in lines 6 and 8 in Fig. 5). The rock column on the left was created with the decimation rate equal to the estimated curvature c and the rock column on the right was created with the decimation rate equal to $3c^4$. Setting the decimation rate per unit time equal to the fourth power of the curvature smooths the surface because protruding regions with large positive curvature experience more decimation per unit time than regions with low or no curvature. All other figures in this paper were created with decimation equal to $3c^3$.

6.2 User Control

The user controls the weathering process by setting durability within a layer, varying durability within a layer, and changing the way curvature is estimated. All three can be done during weathering simulation. The first two tools correspond to geomorphological processes while the third is a useful artifact of the weathering algorithm.

Fig. 14 contains a screenshot of an unweathered rock column as seen in our weathering tool. This rock column will be used throughout the section to illustrate the user control over different weathering effects.

The rock column contains 23,452 voxels either on the surface or one voxel below the surface. The time required

Fig. 13. Effect of varying the relationship between the decimation rate and the estimated curvature.

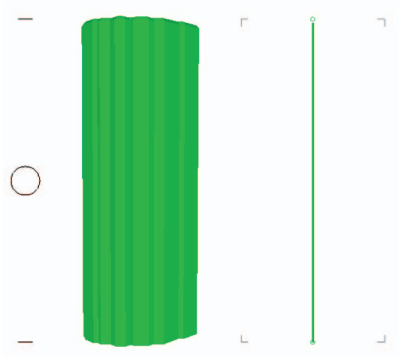

Fig. 14. A column of rock which will be used to illustrate different weathering effects created by varying the layer durability. In this image as well as in Figs. 15, 16, 17, 18, surface color corresponds to durability with green representing less durable and red representing more durable rock.

to complete one cycle of the weathering simulation for this column varies with the number of voxels being removed, instantiated, or ignored, and the bubble radius, as described by (1), but ranges from around 0.2 second (all times exclude extraction of the surface used in the display) for bubbles with radius 3 to around 3.0 seconds for bubbles with radius 7. The weathered shapes based on this column, except Fig. 18b, were each generated using between 10 and 15 seconds of weathering time. Fig. 18b requires more weathering simulation time because using a collection of bubbles in a layer adds an additional layer of indirection inside the critical loop of the weathering simulation.

The first tool for artistic control of the weathering process is the durability graph. The durability graph is used to set the durability within each rock layer. The spheroidal weathering simulation creates rock shapes with silhouettes that match the profile of the durability graph while still maintaining the appearance of weathered rock.

Fig. 15 contains two weathered rock columns with different layer durabilities. The bubble graph is omitted as the bubble graph, shown in Fig. 14, is used in both cases. The silhouette of the final weathered rock column resembles the shape of the durability graph with two exceptions that are related to weathering. First, the top layer of rock weathers more quickly than similarly durable rock lower in the column. This can be seen on the right side of Fig. 15 and

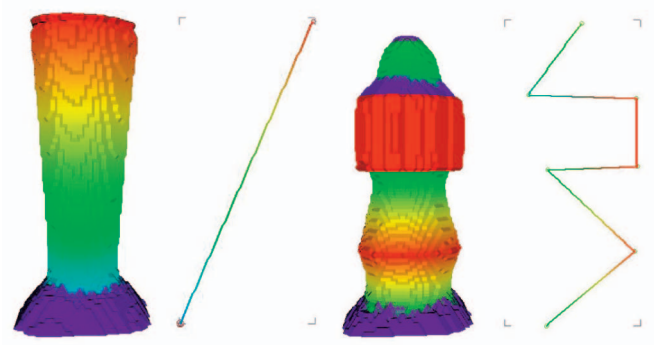

Fig. 15. The weathering simulation creates visually plausible models of weathered rock columns, which match the shape of the durability graph.

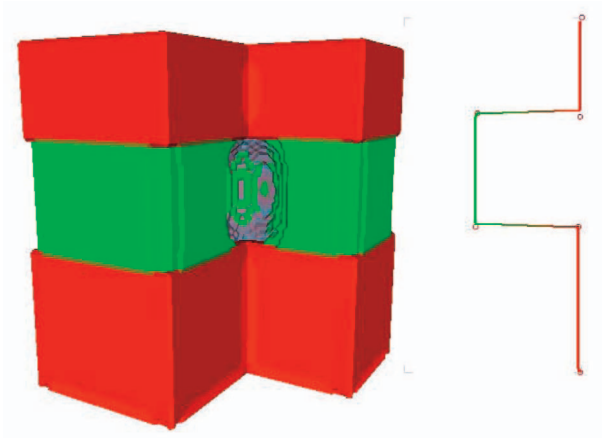

Fig. 16. The surface is colored bright red and purple to highlight the areas where cavernous weathering will produce decimation.

happens because the top layer of rock is more exposed to weathering elements, such as air, than the lower layers.

The second exception is due to the colluvium accumulation. Both rock columns in Fig. 15 contain accumulated colluvium at the base of the column. The column on the right side of the figure also contains accumulated colluvium on intermediate layers.

The relationship between the durability graph and the result of cavernous weathering is more difficult to see because cavernous weathering does not influence the final shape profile. To help the user see the effect of cavernous weathering before applying cavernous weathering, we set the surface color to a color between bright red and purple depending on the decimation to be generated by cavernous weathering. This is computed by taking the product of the negative curvature and the durability for each voxel on the rock surface. Fig. 16 highlights the locations of cavernous weathering for an intermediate step in the creation of the shape in Fig. 10.

Perlin noise is the second user tool for controlling the final rock shape. The user can add different kinds of Perlin noise with varying frequency and amplitude in order to create durability variations within layers. Adding durability variations within layers creates organic shapes, which more closely resemble natural rock. Durability variations within layers are created and displayed in the durability graph.

Fig. 17 demonstrates the effect of varying durability within rock layers. The same durability graph, bubble, and

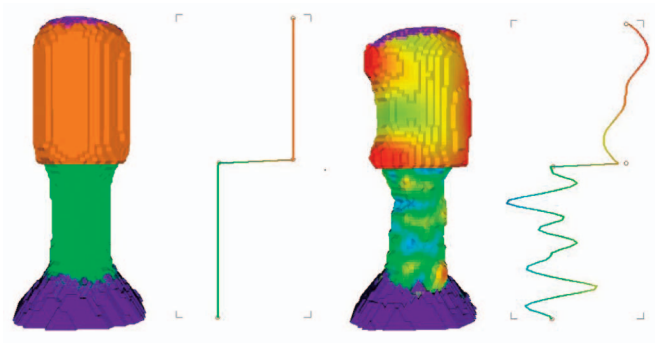

Fig. 17. Adding Perlin noise to layer durability creates variety within rock layers as found in natural rock.

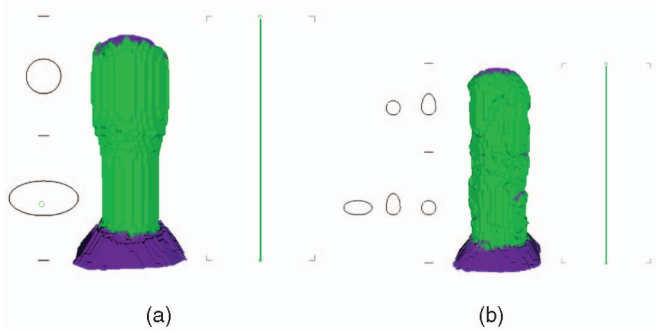

Fig. 18. (a) The disc-shaped bubble suppresses vertical curvature estimates in the weathering algorithm, which results in faster weathering compared to the sphere-shaped bubble. (b) Perlin noise can also be used to distribute bubble sizes and types.

simulation duration were used in both cases with two kinds of Perlin noise added to the rock column on the right. The top layer contains low frequency, low amplitude noise, as might be seen in more durable rock, and the bottom layer contains high frequency, high-amplitude noise, as might be seen in less durable rock.

The final user control of weathering is the size and shape of the bubble used to estimate curvature. Altering the bubble size and shape allows the user to exaggerate or suppress horizontal or vertical curvature. The effect of bubble size and shape is more subtle than the variations in durability.

Fig. 18 illustrates the use of bubbles to change weathered shape of the rock column from Fig. 14. In Fig. 18a, a sphere is used to estimate curvature in the top layer and a small disc is used to estimate curvature in the bottom layer. The disc-shaped bubble results in faster weathering because a disc-shaped bubble centered on a surface voxel contains a larger percentage of air than a sphere-shaped bubble at the same location. In this case, the influence of the vertical curvature reduces the weathering rate because the curvature is constant when traveling vertically on the local surface.

Fig. 18b shows the effect of using Perlin noise to distribute different bubble types and sizes within a layer. The resulting effect is similar to using Perlin noise to vary durability within a layer but is harder to control because it is more difficult to visualize and understand distributions of bubble types than it is to visualize and understand variations in durability.

6.3 Variety of Rock Shapes

Figs. 19 and 20 contain a rendering of weathered rock shapes generated by our algorithm. The shapes in these

Fig. 19. Scenes with rock shapes created using simulation of spheroidal and cavernous weathering.

Fig. 20. A variety of weathered rock shapes created through simulation of spheroidal and cavernous weathering.

images demonstrate the range of effects that can be achieved by varying the simulation settings using the durability and bubble graphs.

7 Conclusion and Future Work

Directable weathering of exposed rock with concave surfaces can be achieved through mean curvature estimation on voxel grids. Curvature estimation supports the simulation of spheroidal and cavernous weathering. The user interface allows the specification of a profile shape and the weathering simulation fills in the details to create the look of weathered rock with the given profile. The weathering simulation is not intended to be scientifically accurate, but the underlying geomorphological process provided a useful metaphor for both the algorithm and the user interface.

The weathering algorithm presented here may be more efficient if defined on a geometric representation other than a voxel grid. This would avoid the extraction of a surface mesh from the voxel grid and may simplify curvature estimation. For example, local surface curvature on a mesh representation can be computed directly from the mesh representation.

We have discussed weathering primarily in the context of sedimentary sandstone rock. More specifically, the layered rock model used here targets sedimentary rock with horizontal layers. However, both cavernous and spheroidal weathering occur in other rock types, such as granite [39], [40]. Future work might include extending the rock model to include characteristics of other rock types. The weathering algorithm based on curvature estimation could then be applied directly to the resulting rock models.

The algorithm may be parallelizable as the weathering behavior depends on localized interactions between surface points, which can be distributed across processing engines. The primary difficulty in parallelizing the algorithm lies in synchronizing colluvium deposition. Colluvium deposition allows particles to interact in ways that are not easily limited to one part of the rock surface, which makes it difficult to partition the work. Modifying the weathering algorithm for implementation on graphics processing units (GPUs) may allow the algorithm to be applied on larger scales.

This work, like other work in geomorphological terrain generation, focuses on a small set of specific processes in isolation. Further work to unify several geomorphological approaches, such as [27], [8] with the algorithm presented here, on a single data structure may result in a tool for creating a wider range of terrains.

References

[1] *Indiana Jones and the Temple of Doom,* Lucasfilm, Ltd., 1984.
[2] *The Mormons Part 2: Church and State,* Public Broadcasting Service, Inc., 2007.
[3] *Wile E. Coyote and Roadrunner: Zoom and Bored.* Warner Brothers, 1957.
[4] *Boundin.* Pixar, 2003.
[5] *Galaxy Quest.* Dreamworks SKG, 1999.
[6] M. Gamito and F.K. Musgrave, "Procedural Landscapes with Overhangs," *Proc. 10th Portuguese Computer Graphics Meeting,* pp. 33-42, 2001.
[7] T. Ito, T. Fujimoto, K. Moraoka, and N. Chiba, "Modeling Rocky Scenery Taking into Account Joints," *Proc. Computer Graphics Int'l Conf.,* pp. 244-247, 2003.
[8] B. Beneš, V. Těšínský, J. Hornyš, and S. Bhatia, "Hydraulic Erosion," *Computer Animation and Virtual Worlds,* vol. 17, no. 2, pp. 99-108, 2006.
[9] F.K. Musgrave, C.E. Kolb, and R.S. Mace, "The Synthesis and Rendering of Eroded Fractal Terrains," *Proc. ACM SIGGRAPH,* pp. 41-50, 1989.
[10] J. Dorsey, A. Edelman, H. Jensen, J. Legakis, and H. Pedersen, "Modeling and Rendering of Weathered Stone," *Proc. ACM SIGGRAPH,* pp. 225-234, 1999.
[11] *Bryce 6.0.* DAZ Productions, 2006.
[12] *Vue-6 infinite.* e-on Software, 2006.
[13] *Maya 2008.* Autodesk, 2008.
[14] *Blender 2.48a.* Stichting Blender Foundation, 2008.
[15] S.W. Wang and A.E. Kaufman, "Volume Sculpting," *Proc. Symp. Interactive 3D Graphics (SI3D '95),* pp. 151-156, 1995.

[16] M. Beardall, M. Farley, D. Ouderkirk, J. Smith, C. Rheimschussel, M. Jones, and P. Egbert, "Goblins by Spheroidal Weathering," *Proc. Eurographics Workshop Natural Phenomena,* pp. 1-8, 2007.
[17] H. Huinink, L. Pel, and K. Kopinga, "Simulating the Growth of Tafoni," *Earth Surface Processes and Landforms,* vol. 29, pp. 1225-1233, 2004.
[18] A.V. Turkington, "Cavernous Weathering in Sandtone: Lessons to be Learned from Natural Exposure," *Quarterly J. Eng. Geology,* vol. 31, pp. 375-383, 1998.
[19] M. Milligan, "Geology of Goblin Valley State Park, Utah," *Geology of Utah's Parks and Monuments,* Utah Geological Assoc. and Bryce Canyon Natural History Assoc., pp. 421-432, 2003.
[20] P. Prusinkiewicz, L. Mundermann, R. Karwowski, and B. Lane, "The Use of Positional Information in the Modeling of Plants," *Proc. ACM SIGGRAPH,* pp. 289-300, 2001.
[21] Y. Dobashi, K. Kusumoto, T. Nishita, and T. Yamamoto, "Feedback Control of Cumuliform Cloud Formation Based on Computational Fluid Dynamics," *Proc. ACM SIGGRAPH,* vol. 27, no. 3, 2008.
[22] T. Vicsek, "Pattern Formation in Diffusion-Limited Aggregation," *Physical Rev. Letters,* vol. 53, no. 24, pp. 2281-2284, Dec. 1984.
[23] P. Pimienta, W. Carter, and E. Garboczi, "Cellular Automaton Algorithm for Surface Mass Transport Due to Curvature Gradients: Simulation of Sintering," *Computational Materials Science,* vol. 1, pp. 63-77, 1992.
[24] B. Mandelbrot, *The Fractal Geometry of Nature.* W.H. Freeman and Co., 1982.
[25] A. Fournier, D. Fussell, and L. Carpenter, "Computer Rendering of Stochastic Models," *Comm. ACM,* vol. 25, no. 6, pp. 371-384, June 1982.
[26] A. Kelley, M. Malin, and G. Nielson, "Terrain Simulation Using a Model of Stream Erosion," *Proc. ACM SIGGRAPH,* pp. 263-268, 1988.
[27] N. Chiba, K. Muraoka, and K. Fujita, "An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain Scenery," *J. Visualization and Computer Animation,* vol. 9, no. 4, pp. 185-194, 1998.
[28] F. Belhadj and P. Audibert, "Modeling Landscapes with Ridges and Rivers: Bottom up Approach," *Proc. Third Int'l Conf. Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE '05),* pp. 447-450, 2005.
[29] *Fractals in Nature: From Characterization to Simulation,* H.-O. Peitgen and D. Saupe, eds., ch. 1, pp. 21-70. Springer-Verlag, 1988.
[30] P. Prusinkiewicz and M. Hammel, "A Fractal Model of Mountains with Rivers," *Proc. Graphics Interface Conf.,* pp. 174-180, 1993.
[31] K. Nagashima, "Computer Generation of Eroded Valley and Mountain Terrains," *The Visual Computer,* vol. 13, pp. 456-464, 1997.
[32] O. Šťava, B. Beneš, M. Brisbin, and J. Křivánek, "Interactive Terrain Modeling Using Hydraulic Erosion," *Proc. ACM Eurographics/Symp. Computer Animation,* 2008.
[33] Y. Chen, L. Xia, T.-T. Wong, X. Tong, H. Bao, B. Guo, and H.-Y. Shum, "Visual Simulation of Weathering by γ-Ton Tracing," *Proc. ACM SIGGRAPH,* pp. 1127-1133, 2005.
[34] S. Hsu and T. Wong, "Simulating Dust Accumulation," *IEEE Computer Graphics and Applications,* vol. 15, no. 1, pp. 18-22, Jan. 1995.
[35] R. Sarracino, G. Prasad, and M. Hoohlo, "A Mathematical Model of Spheroidal Weathering," *Math. Geology,* vol. 19, pp. 269-289, 1987.
[36] G. Miller, "Efficient Algorithms for Local and Global Accessibility Shading," *Proc. ACM SIGGRAPH,* pp. 319-326, 1994.
[37] K. Perlin, "An Image Synthesizer," *Computer Graphics,* vol. 19, no. 3, pp. 287-296, 1985.
[38] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH,* pp. 163-169, 1987.
[39] C.R. Twidale and J.R.V. Romani, *Landforms and Geology of Granite Terrains.* CRC Press, 2005.
[40] Y. Matsukura and Y. Tanaka, "Effect of Rock Hardness and Moisture Content on Tafoni Weathering in the Granite of Mount Doeg-Sung, Korea," *Geografiska Annaler,* vol. 82, pp. 59-67, 2000.

Michael D. Jones is an associate professor of computer science at Brigham Young University, where he directs the Laboratory for computer-generated natural phenomena. His research interests include animation and modeling of natural phenomena using algorithms based on physical models. He is a member of the IEEE.

McKay Farley received the BS degree in computer science from Brigham Young University, where he is currently working toward the MS degree in computer science. His research interests include graphics, modeling, rendering, and special effects.

Joseph Butler received the BS degree in computer science from Brigham Young University, Idaho, where he is currently working toward the MS degree in computer science. His research interests include graphics, computer vision, and biometrics.

Matthew Beardall received the BS degree in computer science from Brigham Young University. He is currently working as a software engineer at Caselle, Inc. His research interests include graphics and persistent languages.