CEIG - Spanish Computer Graphics Conference (2024)
J. Marco and G. Patow (Editors)

Simulation of Mechanical Weathering for Modeling Rocky Terrains

D. Mateos, L. Carranza, A. Susin, O. Argudo

Universitat Politècnica de Catalunya, Spain

Abstract

Synthetic terrains play a vital role in various applications, including entertainment, training, and simulation. This work focuses on rocky terrains akin to those found in alpine environments, which contain many complex features such as sharp ridges, loose blocks, or overhangs that are often inadequately represented by standard 2D elevation maps. We propose a novel method based on a simplified simulation of mechanical erosion processes commonly observed in high-altitude terrains, in particular the weathering due to freeze-thaw cycles. The ultimate objective is to generate plausible rocky geometry from existing 3D models, as well as account for the temporal evolution due to these weathering processes. Additionally, we have developed an artist-friendly tool integrated as an add-on into Blender.

CCS Concepts

• ***Computer Graphics*** → *Shape modeling; Physical simulation;*

1. Introduction

Synthetically generated or enhanced terrains are a fundamental part of virtual scenes, which are widely used across industries like entertainment, training or simulation. Researchers have made progress in creating synthetic terrains using three primary strategies: procedural modeling, physical simulations, and synthesis from examples [GGP*19].

Most terrain generation methods, however, still rely on 2.5D height field representations, and so are the majority of publicly available data sets from cartographic providers, which are as elevation grids. The result is terrains lacking intricate details on steep areas, which are enhanced through the application of noise textures, displacement maps, or the overlaying of meshes. Some recent approaches have leveraged implicit surfaces to model truly 3D features like overhangs, arches or caves [PGP*19, PPG*20, PGP*21]. These methods take inspiration from erosional processes and weathering to obtain geologically-based models, but their direct output - before applying noise operators or displacements - tends to be too smooth due to the use of functions with continuous gradients.

In this paper, we introduce a novel geometric approach to model sharp 3D terrain features and simulate the temporal evolution of the scene due to mechanical weathering processes. In particular, we take inspiration in frost wedging due to freeze-thaw cycles: water entering through small cracks in the rock freezes, expands its volume and exerts an outwards pressure widening the fracture. This phenomenon is common, but not limited to, alpine rocky environments like high-altitude ridges (Figure 1). Our contributions can be summarized as follows: 1) a graph-based representation of rock shards and their bonds, 2) a mechanical erosion simulation efficiently leveraging on these graphs, 3) a publicly available Blender plugin with our implementation, which facilitates its integration into artistic pipelines. Finally, our method also offers a high degree of control both on the shard generation and the simulation stages.

Figure 1: *Sample images of rocky formations found in mountainous landscapes. The detailed blocks of rocks and steep walls are poorly represented in regular elevation maps. Also notice how some of the cuts are extremely straight leading to very polyhedral shaped rocks.*

DOI: 10.2312/ceig.20241139

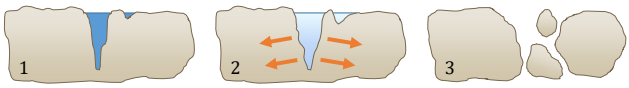

Figure 2: *Mechanical erosion produced by freeze-thaw cycles. Water infiltrates into fractures of the rock (1). When it freezes, its volume expands and pressures on the fracture walls, enlarging the fracture (2). Rising temperatures will melt the ice, repeating the cycle and eroding the rock (3).*

2. Previous Work

Mechanical weathering is a geological process in which rocks and minerals are broken down into smaller particles or fragments through physical forces, as opposed to chemical weathering which can alter the composition of the rocks (see Chapter 7 in [Hug11]). Examples of mechanical weathering processes are frost wedging, abrasion, exfoliation, and some forms of biological activity (e.g. tree roots growing between rock joints).

2.1. Mechanical weathering in applied sciences

A comprehensive review of advances in computational fracture mechanics of rock is provided in [MLC*21]. It deeply delves into fracture mechanics and their effects at different micro to macro scales, and provides a classification of computational methods. Typically, micro- and meso-scale studies target the understanding of how cracks form due to heterogeneities in the rock and what are the corresponding failure mechanisms. On the other hand, macro-scale analysis models cracks expanding several decimeters but the internal structure of the material is not taken into account. Since we are interested in very large scale, where fractures can extend over a few meters, it seems logical to consider chunks of rock (shards) as having homogeneous properties and ignore their minerals microstructure.

The same survey [MLC*21] also classifies numerical methods into continuum and discontinuum. Continuum methods include Finite Elements, Finite Differences and Smooth Particle Hydrodynamics, among others. These methods treat the material as a homogeneous continuum, with smoothly varying properties and physical quantities, and elements are viewed as samples of this media. On the other hand, discontinuum formulations divide the material into distinct individual components, like rigid bodies or spheres, that can move, slip, rotate and interact between them through forces. Cracks can form along their boundaries when stress exceeds a certain threshold. Our method, based on a decomposition into Voronoi cells, is related to this family of discontinuum formulations. However, since our goal is to enhance terrains interactively with plausible formations instead of predicting the actual evolution of a landscape, we will avoid computationally demanding simulations and propose a much simpler physical model.

Several works specifically focus on the breakdown of rocks by freeze-thaw cycles, explained in Figure 2, as frost action has been considered to be of the major agents in the weathering of periglacial landscapes. A recent review about these processes can be found in [DDKDSC20]. A case study of long-term large-scale frost weathering and rockwall erosion in the Alps can be seen in [Mat08], which details several observations between 1994 and 2006. Another extensive rock slope monitoring campaign was done in [HGB12] at the Swiss Alps. Based on empirical findings, the authors identify that most erosion occurs during summer due to water melted in clefts containing perennial ice. Additionally, it also shows how the thawing related processes can affect stability within hours or weeks, basically, short-term stability minima may activate rock masses subject to the slow changes and lead to acceleration and failure. In our simulation, we want to mimic this behavior and potentially detach large masses of rock due to cracks between shards.

2.2. Mechanical weathering in Computer Graphics

In Computer Graphics, weathering - sometimes also called aging - is seen as a gradual process that modifies the appearance of an object, for example by altering its geometry or materials, with the aim of improving its perceived realism [MG08]. Erosion and fracturing are, thus, types of weathering phenomena. Many works have already addressed the generation of fractures on models, see for example the review in [MBP14]. However, most of the time, these methods aim to simulate brittle fractures due to impacts [MCK13, HW16, FCK22]. In contrast, we aim to simulate the gradual fracturing of terrain scenarios due to weathering processes like frost wedging.

Erosion and weathering of rocky terrains have also been addressed from the Computer Graphics literature. For example, [JFBB10] generates concave rock formations by estimating curvatures from a voxel grid and simulating spheroidal weathering (smoothing of sharp corners) and cavernous weathering (pit formation). Wind abrasion is modeled in [KHM*20] using an interactive particle based simulation.

In [PGGM09], the authors propose a layered grid structure to represent stacks of various materials including air, and they can model 3D features like overhangs and arches. This layered grid is used to simulate material stabilization, and then a coupled implicit representation of the surface is used to provide sculpting tools and to reconstructing the terrain model. This work was later extended in [PGP*19] by proposing a volumetric terrain representation based only on implicit surfaces. Through construction trees, they model both the terrain and the underlying geological structure of faults and rock layers. Erosion is performed through the insertion of negative sphere primitives into the tree. The evolution of caves and tunnels carved by water on karst landscapes are obtained through an invasion-percolation procedure that determines how these negative spheres advance from the surface. In a recent work, [PGP*21] directly addressed the modeling of cave networks by computing anisotropic shortest paths between endpoints in a volumetric representation of the geological properties. Our technique also simulates how water infiltrates from the exterior into the terrain, but instead of directly carving it we reduce the strength of the bonds between rock shards until they fracture. Moreover, the implicit representations used in their works lead to rounded smooth surfaces, whereas we target sharp corners like the ones found in fractured rock walls and ridges.

Our method is based on a connectivity graph between neighboring convex shards, upon which an iterative simulation removes

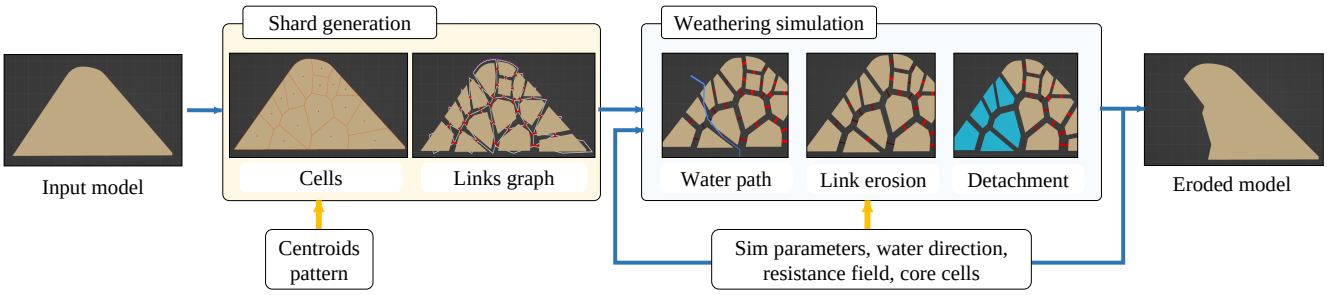

Figure 3: *Overview of our pipeline and its two main stages: the fracture layout generation and the iterative freeze-thaw weathering simulation. User controls are possible through the pattern of centroids in the first stage, and by tuning simulation parameters and mechanical properties during the latter.*

links, generating fractures and eroding away shard groups that get disconnected from the main model. This idea of link removal was already proposed in [IFMC03], but from links between voxels and their six neighbors. A distribution of 3D oriented discs is then used to remove intersected links, resulting in connected components of voxels as the shards. Gravity and friction forces are then used to remove unstable shards that should fall or slide apart. This method, however, presents the usual drawbacks of voxelizations: blockiness and memory consumption. Similarly, discs are also used to remove links in the work by [PPG*20], but in this case they start from a graph connecting randomly sampled points with their neighbors. They use implicit functions to approximate each resulting convex block, and then add micro-surface relief details with a displacement operator. The obtained 3D model can then be tiled on top of steep terrain surfaces to enhance the appearance. However, the changes to the terrain are limited to the size of this tiled block, and they do not simulate any temporal evolution.

3. Overview and notations

Conceptually, our method can be divided into shard generation (Section 4) and weathering simulation (Section 5), as illustrated in Figure 3. The former involves computing the geometry of some initial shards of material and their properties. The latter uses all this information to perform an iterative erosion simulation inspired by percolation and freeze-thaw weathering, which changes the shape of the original model by detaching shards over time. The user has control over both processes through a series of parameters, and also decides when to proceed to the erosion simulation and when to end it.

The shards represent chunks of a continuous material kept together through virtual bonds yet to be fractured by the simulation. Based in the freeze-thaw cycle, we simulate how trapped water between the shards pushes their faces in opposite directions and causes internal fractures between them. The state of this internal separation is modeled with virtual links, which technically model the bonding connection between shards instead of the physical gap. This means that the gap between two shards is inversely proportional to the sanity of their bonding link. Therefore, when a link is broken, it indicates that the separation has reached certain threshold and we consider the shards to be no longer attached.

We denote the set of cells as $C = \{c_i\}$ and the set of links as $L = \{l_i\}$. Each cell can be in one of three possible states: solid, air or core. Solid is considered the default rock type. Air cells are used to mark the shards that have already been eroded away, and core cells are selected by the user as shards that are never removed. Links can be interior or exterior, so $L = L^{\text{int}} \cup L^{\text{ext}}$. An interior link l_i connects two cells c_a and c_b. Sometimes we want to specify these cells, so we will denote the link as $l_i = l_{a,b} \in L^{\text{int}}$. Exterior links connect a surface cell with the outside of the model, and as the erosion progresses interior links can become exterior. Note that two convex cells can only share one interior link, however one cell can have many exterior links.

Finally, we build two undirected graphs: the cells graph G_C, which represents the connectivity of adjacent shards (cells) through bonds (links), and the links graph G_L, which represents the connectivity between a link $l_i \in L$ and its neighbors.

4. Shard generation

Given an input model, the first step consists in generating the set of potential fractures by decomposing the model into a set of shards, which we represent as convex cells. The boundaries of these shards, i.e. the faces between adjacent cells, will be the future fracture locations so we create virtual links to represent the state of the bond between them. Additionally, we also precompute during this stage the cell adjacency graph G_C and the links graph G_L.

4.1. Cell decomposition

In our implementation, we decided to use Voronoi cells because they result in a more natural appearance than tessellations based on tetrahedra or voxels. However, any decomposition yielding cells and their connectivity could be used. Therefore, and without loss of generality, we will explain our Voronoi-based solution in this section.

Given a set of n unique points $P = \{p_i\}$, the Voronoi cell for p_i is defined as:

$$V(p_i) = \{\, x \in \mathbb{R}^3 \mid d(x,p_i) \leq d(x,p_j), \forall p_i, p_j \in P,\, p_i \neq p_j \,\}$$

To compute the Voronoi cells, we have used the implementation from the library *Voro++* [Ryc09]. This algorithm computes

the cells independently as the intersection of the half-spaces defined by the bisecting planes between the cell's centroid and all other points. Formally, if $H_{i,j} = \{ x \in \mathbb{R}^3 \mid d(x,p_i) \leq d(x,p_j) \}$, then the Voronoi cell is $V(p_i) = \bigcap_{j \neq i} H_{i,j}$. This approach has the benefit of being easily parallelizable, and it also allows to introduce additional planar restrictions which the library calls *walls*. In particular, we add the faces of the original model as walls in order to preserve the original surface as much as possible. Finally, we define our set of cells as $C = \{ V(p_i) \mid V(p_i) \neq \emptyset \}$.

Users can control the output of this stage through the point distribution. Figure 4 shows some examples with different patterns like random point sampling, uniform and hexagonal grids with a small amount of jittering, and a gradually increasing radius for a Poisson disc distribution.

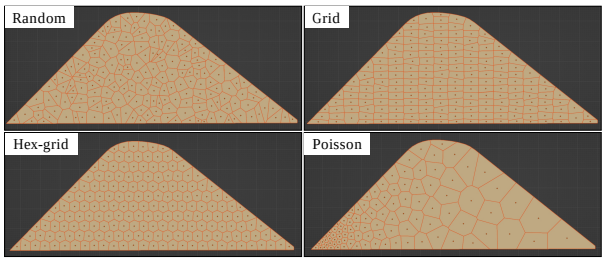

Figure 4: *Shards obtained with different centroid distributions.*

4.2. Links graph

Links are our abstract representation of the bond between two cells that are adjacent through a face, but also for the interface between a surface cell face and the exterior.

Following the construction algorithm presented above, cell faces are either generated from a bisecting plane between two centroids or from a wall. In the first case, we create an interior link $l_i \in L^{\text{int}}$ that represents the two faces of the adjacent cells, identical except for small numerical errors and with opposite normals. In the second case, we create an exterior link $l_i \in L^{\text{ext}}$ representing the surface face.

Let us denote as $F(l_i)$ the set of faces represented by a link $l_i \in L$. The cardinality of F is two for interior links and one for exterior links. Given a set of faces F, we can also find the set $\text{Adj}(F)$ containing all the faces that are geometrically adjacent to those in F, i.e. the cell faces that share an edge with a face in F. Then, the set of neighboring links of l_i is defined as:

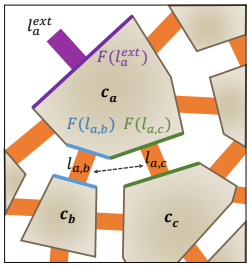

Figure 5: *Neighbor links.*

$$\text{Neigh}(l_i) = \{ l_j \mid F(l_j) \cap \text{Adj}(F(l_i)) \neq \emptyset \}$$

so links l_i and l_j are neighboring links if one of their faces is adjacent to one of the other link's faces. Figure 5 illustrates an example.

Finally, we construct two undirected graphs upon which the simulation is computed (Figure 6). The first one, which we call the cells graph, represents the bonds between shards and is used to detect when parts of the model detach after a link breakage. Thus, it has a vertex for each cell and the interior links as edges:

$$G_C = (C, L^{\text{int}})$$

The second graph, called the links graph, is used for water infiltration and propagation. Each node of the graph corresponds to a link, and each edge represents a neighboring pair of links:

$$G_L = (L, \{(l_i, l_j) \in L \times L \mid l_j \in \text{Neigh}(l_i),\ l_i \neq l_j\})$$

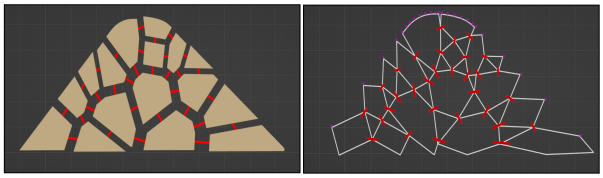

Figure 6: *Left: cells graph, with generated cells connected by interior links (red lines). Right: links graph, with links (red interior, purple exterior) connected to neighboring links (white lines). Note that curved areas, with many faces in the original model, originate several exterior links.*

5. Weathering simulation

The simulation process models the freeze-thaw cycles at link level using the graph G_L and utilizes G_C to determine when rock shards detach from the model. Every iteration starts with water infiltrating the model through an external shard face, then continues to propagate towards the interior, running through the spaces between the shards. We call *path* the route the body of water follows throughout a percolation. Inevitably, part of the water is trapped along the path, which will produce weathering due to ice expansion once it freezes. In our model, we translate this erosion to a reduction in the sanity of the bonding links between cells. After a series of simulated water infiltrations, the sanity of certain links decreases enough to be considered broken. When this happens, we analyze the remaining structure of shards and remove the disconnected groups from the model, thus eroding them away.

5.1. Water propagation

We model paths using a stochastic flow method in which there is a single linear trajectory of the water at a time, there is no water flow division (Figure 7 illustrates an example). Since the internal propagation depends only on the current state of G_L, it can be interpreted as a Markov random walk process.

A path starts with some volume w^{in} of water reaching the model from an incoming direction $\vec{\omega}_w$ and being captured by an external link in G_L. The direction $\vec{\omega}_w$ is sampled from a user-specified distribution. This allows us to obtain different degrees of erosion on different orientations (Figure 8), for example because they are more

humid or more exposed to prevailing rainfall directions. The starting link is randomly chosen among all external links l_i^{ext} based on the following infiltration probability:

$$p_{\text{inf}}(l_i^{\text{ext}}) \propto a_i \cdot \max(\vec{n}_i \cdot \vec{\omega}_w, 0)$$

where a and $\vec{n}$ represent the area and normal of the link's face. Large faces oriented toward the incoming water direction will have more probability of being the starting point of the path.

The probability of water propagating from a current link l_i to one of its neighbors $l_j \in \text{Neigh}(l_i)$ is defined as:

$$p_{\text{flow}}(l_i, l_j) \propto f_g(l_i, l_j) \cdot (1 - f_r(l_j))$$

where f_g is a term that accounts for the alignment between gravity and flow direction, and f_r the resistance that the neighboring link opposes to the flow. Although links represent an entire face, we approximate the flow direction between two links as the vector between the centroids $\mathbf{x}$ of their faces. Thus, given a gravity direction $\vec{g}$, we compute the alignment term as:

$$f_g(l_i, l_j) = \max(\vec{g} \cdot \frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_j - \mathbf{x}_i||}, 0)$$

The resistance that water might encounter when percolating through the face surface of l_j depends on the link's integrity and on the underlying material at the link's face. The integrity of a link l_i, which we also call the link's *life*, is a value $\lambda_i \in [0,1]$ such that when it reaches zero the bond between the two faces is considered broken and the fracture opened. Consequently, flow resistance should be proportional to λ_j. To account for different underlying materials with different resistances, like layers or pockets of different rock types, we consider the model is embedded in a resistance field $\rho : \mathbb{R}^3 \to \mathbb{R}$. In our framework, ρ is defined either from analytic functions or through 3D textures (Figure 9). From these ideas, we model the resistance term as:

$$f_r(l_j) = \lambda_j \cdot \rho(\mathbf{x}_j)$$

Along its path, the initial water volume w^{in} is gradually absorbed by the traversed links, leading to their subsequent damage as we will explain in the next subsection. The termination of the water path occurs when one of the following two conditions is met: either there are no neighboring links with $p_{\text{flow}} > 0$, as may occur, for instance, at the bottom of the model; or all the water has been absorbed.

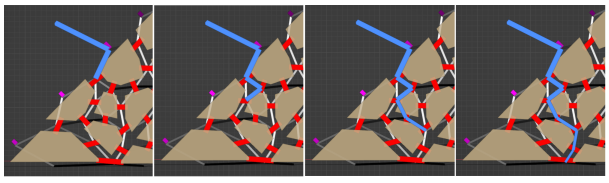

Figure 7: *Example of a water propagation path through G_L. The width of the blue line represents the remaining fraction of water. Neighboring link relationships are depicted with gray lines, with brighter shades indicating a higher alignment with gravity f_g.*

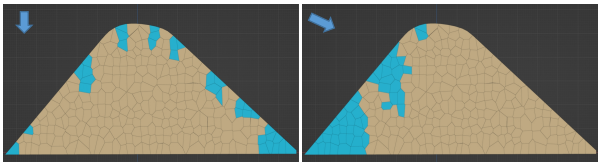

Figure 8: *Incoming water direction can have a large impact on the erosion of the model. Left: $\vec{\omega}_w = (0,-1)$. Right $\vec{\omega}_w = (1,-0.5)$. Both cases show the state after 500 water paths.*

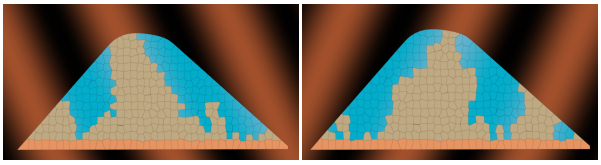

Figure 9: *Effect of the resistance field term on the erosion after 300 paths. Resistance field, modeled from an sine function, is depicted as the background; brighter means higher resistance. The bottom row of* core *cells prevents the model from splitting.*

5.2. Water absorption and link erosion

Water is absorbed during the propagation path, and the specific volume lost depends on the link being traversed. We model and combine two kinds of water absorption: one derived from the link resistance and another based on the friction over the face surface area, independent of the link opposition. Given that the resistance of a link partly depends on its life, broken links will let water run through them quite freely. Thus, we define different absorption coefficients for solid and broken links. The absorbed water quantity is computed at l_i as:

$$w_{\text{abs}}(l_i) = \begin{cases} a \cdot k_{\text{solid}} + f_r(l_i) \cdot k_{\text{res}} & \text{if } \lambda_i > 0 \\ a \cdot k_{\text{air}} & \text{otherwise} \end{cases}$$

where k_{solid}, k_{air} and k_{res} are the absorption coefficients for solid links, broken links and due to flow resistance, respectively. If link l_i received an amount of water w_i, the neigh link l_j along the path will receive $w_j = w_i - w_{\text{abs}}(l_i)$. In our experiments, we have used the following parameters: $k_{\text{solid}} = 0.1/\bar{a}$, $k_{\text{air}} = 0.05/\bar{a}$ and $k_{\text{res}} = 0.25$, where $\bar{a}$ denotes the average cell face area.

In addition to this, we added a random event that triggers a complete absorption of the remaining water volume w. Its probability is proportional to the relative lost volume: $1 - \frac{w}{w^{\text{in}}}$. This stochastic event contributes to modeling natural irregularities present in the material, while also acting as an early termination mechanism for paths that would not cause significant damage to subsequent links. The result is a performance improvement without noticeable link erosion differences.

The water absorbed by the material proportionally erodes each solid link when traversing it. Broken links do not register more erosion. The damage inflicted to a link is distributed over its surface

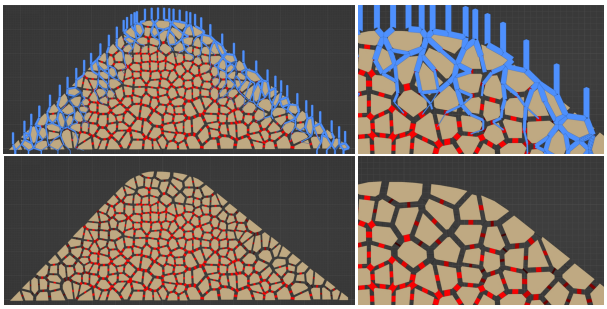

Figure 10: *Top: visualization of 200 simulated water paths, line width represents percentage of initial water volume. Bottom: darker and thinner lines represent more damaged links.*

area, and weighted by a user-defined parameter k_{dmg}:

$$\lambda' = \lambda_i - k_{\text{dmg}} \frac{w_{\text{abs}}(l_i)}{a_i}$$

When the life of a link reaches zero, a breakage is produced and a detachment process is started. We used $k_{\text{dmg}} = 0.5\bar{a}$.

Analogue to the complete absorption random event, there is also a stochastic event forcing link breakage. It can happen when the link's life λ_i drops below a threshold parameter k_{break}, with a probability proportional to: $1 - \frac{\lambda_i}{k_{\text{break}}}$. Note that these low-life links targeted by this event oppose almost no resistance at all and therefore cause very little water absorption through resistance. This, in turn, contributes to a lower link erosion causing some low life links to resist breakage for longer. Forcing some of these links to be broken a bit in advance speeds up the simulation and detachment while only reducing a small amount of remaining life for random links. We have used $k_{\text{break}} = 0.4$ and the maximum probability scaled to 0.9. Figure 10 illustrates the aggregated effect of several paths on the links' life.

5.3. Cells detachment

As mentioned earlier, a cell detachment computation is triggered every time a link is broken. This process relies on the concept of connected components of the cells' connectivity graph G_C. Essentially, breaking a link l_i involves removing one edge from the cells graph: $G'_C = (C,\ L^{\text{int}} - \{l_i\})$, and removing one node and its adjacent edges from the links graph: $G'_L = (L - \{l_i\},\ \{(l_j, l_k) \in L \times L \mid l_j, l_k \neq l_i\})$.

This operation may result in a split of a connected component of G_C (an example is shown in Figure 11). In such case, we obtain two smaller connected components of cells $C_a, C_b \subset G_C$. If neither component contains cells marked as core, the cells in the smallest one are marked as air. If only C_a contains core cells, C_b is marked as air, and vice versa. If both components contain core cells, the cells remain as solid. The cells marked as air are conceptually removed parts of the model (in fact, we could even delete them) so G_C and G_L need to be updated accordingly.

Finally, when cells are marked as air (detached), we need to recalculate the exterior links. From the previous set of exterior links

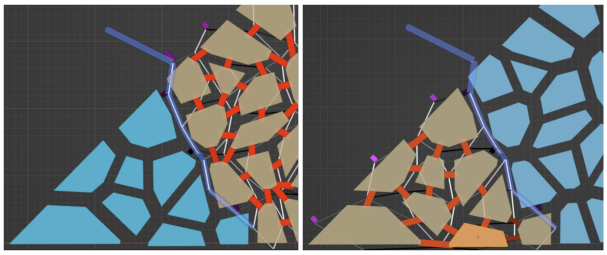

Figure 11: *After a link rupture that disconnects G_C, the smallest of the two components is eroded away (left) unless it contains a core cell (right). Air cells are depicted in blue and the core cell in orange.*

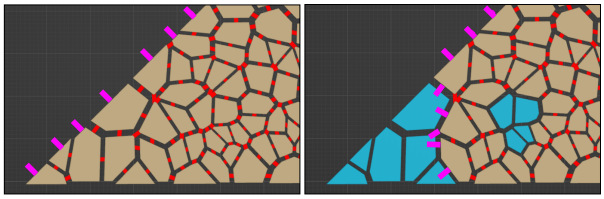

Figure 12: *Exterior links, depicted in magenta, are recalculated when cells are detached.*

L^{ext}, we eliminate those that became adjacent only to air cells. Then, to identify the updated set, we perform a traversal on G_L using a Breadth First Search (BFS) starting from these remaining exterior links. To improve performance, the traversal is restricted to nodes of G_L that correspond to links between a solid cell and an air cell. The rationale behind performing this BFS instead of directly listing all solid-air links as current exterior links is to account for potential *air pockets* created inside the model, for which we do not want to consider their links as exterior (see Figure 12).

6. Results

We have implemented our algorithm as a Python plugin for Blender. This facilitates its integration into existing artistic modeling and rendering pipelines, but also benefits from the many built-in functionalities of this software. We have tested and distributed our implementation within a portable version of Blender 3.4, available in the following repository: `https://github.com/dimateos/UPC-MIRI-TFM-erosion`.

During shard generation, centroids can be specified from a (random) subset of vertices from a mesh, or using Blender's particle systems, which can be defined on surfaces or the interior of volumes. For the simulation part, the basic controls are the incoming water direction and the resistance field, which can be defined through a Python function (we plan to extend the plugin with support for 3D textures). All the parameters have been exposed through the plugin interface, as well as several logging, inspection and visualization utilities in order to follow the execution of the algorithm.

Cells	$\lvert L^{\text{int}}\rvert$	$\lvert L^{\text{ext}}\rvert$	T_{gen}	$\overline{T}_{\text{path}}$
100	504	123	0.015	0.245
500	2360	319	0.078	0.388
1000	5236	433	0.141	0.461
5000	18201	1407	1.281	1.265
10000	32535	2493	2.469	2.202

Table 1: *Statistics on the* Hill3D *model depending on the number of generated cells: internal links, external links, time to generate the Voronoi cells and links graph (in s), and average time to simulate one water infiltration path (in ms).*

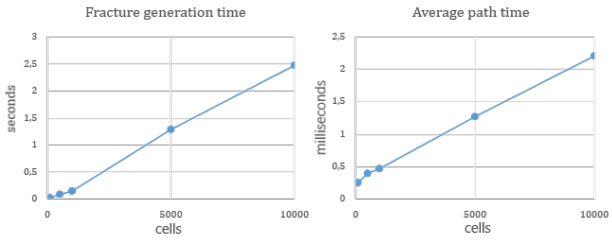

Figure 13: *Generation and simulation time plots.*

6.1. Performance

One of the goals of our method was to offer an interactive editing tool for artists, so we measured the performance of our plugin. We ran our experiments on a laptop computer equipped with an Intel Core i7-12650H processor (2.3 GHz), 16 GB of RAM, and an NVIDIA GeForce RTX 3060 laptop GPU.

Table 1 summarizes statistics about the number of cells, links in the graph and timings. We used the *Hill3D* model (see Figure 14) and a random distribution of centroids inside its volume to generate the cells. Upon the resulting fractured model, we simulated 1000 water paths, measuring the total computation time, and report an average path time. The measured path time includes water propagation and cell detachment, but omits time spent on UI updates and debug logs. Moreover, since these graph algorithms depend on the number of links, time per path decreases as the simulation advances and more cells are detached. This is especially relevant in small models, in which an early detachment of a large component can significantly reduce the timings. To mitigate this potential bias, we measured five times the execution of 1000 paths, starting each time from the same graph produced by the generation algorithm.

Both fracture generation time and path infiltration time are linearly proportional to the number of cells (Figure 13). This second relationship might seem counter-intuitive, since the traversal of the path only depends on the neighbors of each link, which is small and quite uniform across different links. Since our tested absorption factor k_{solid} is normalized by average area, absorption in our experiments is scale independent so having a model fractured into more cells does not produce longer traversals with more visited links. The linear relationship we observe comes from the detachment computation, which recomputes the connected components of cells in G_C every time a link breaks. This suggests us that the largest cost of our simulation is this connectivity test, although we have not tested this hypothesis further.

6.2. Qualitative results

We now present use case scenarios of simulations obtained using our add-on, then rendered directly within Blender. Please see the accompanying videos for the complete animations.

Figure 14 shows an erosion sequence on the *Hill3D* model. The model resembles a cone and was fractured into 2908 cells using a jittered distribution of centroids inside the volume, which led to irregularly shaped convex shards. The number of internal links is 18560. Exterior cells were marked as core, except for the ones facing front in the picture, in order to keep the overall shape and create a hole progressing towards the interior, forming a cave. Incoming water direction was mostly oriented towards the hole, with a small vertical component. Given the large number of cells and links, many path iterations are needed before the cave starts to carve. The simulation time for the 70k paths was 209 seconds.

Some specific patterns can also be enforced. For example, Figure 15 shows an erosion sequence on a wall made of *columnar basalt*, which are hexagonal columns formed when a lava flow cools down and contracts. To create this specific fracture pattern, we placed the centroids in a hexagonal grid pattern on a plane parallel to the floor. A small amount of jittering to their position reduces the regularity of hexagons. The generation step created 300 cells with 1497 internal links. The last row of columns was marked as core cells, so the erosion can evolve to disconnected column groups, as seen in the last two images. In this smaller example, a few thousand paths are enough to substantially modify the initial geometry. In this small example, simulating 4000 paths took 0.6 seconds.

Placing the centroids in a rectilinear grid pattern creates rectangular Voronoi cells. Figure 16 demonstrates a use case of this grid to erode a rectangular box into *slabs*, like strata of different materials. We adjusted the depth of each strata by modifying the distances along the vertical axis between consecutive pairs of 2D rectilinear grids. Finally, we tilted the grid to generate a shard pattern like the ones found in inclined strata. The generated fracture pattern contains 108 cells and 449 links. For this simulation, we wanted to obtain an erosion progression mimicking planar slabs detaching, from the top-left corner towards the bottom-right. Thus, we defined the incoming water direction as horizontally coming towards the left side of the prism. Setting the gravity upwards then creates L-shaped paths that progressively erode the model in the desired way, although intermediate overhanging structures appear (second image in Figure 16). While this is unrealistic from a physical point of view, this flexibility in the parameters serves as artistic purpose. We also doubled the strength of the absorption coefficients, to generate shorter paths with more erosion and remove outer layers of shards first.

Finally, Figure 17 shows an example of mechanical erosion applied on a 3D model extracted from a Digital Elevation Map of a ridge in the Pyrenees. The Voronoi cells were created using a jittered distribution of points inside the bounding box of the model, and then all cells that did not intersect the terrain were directly

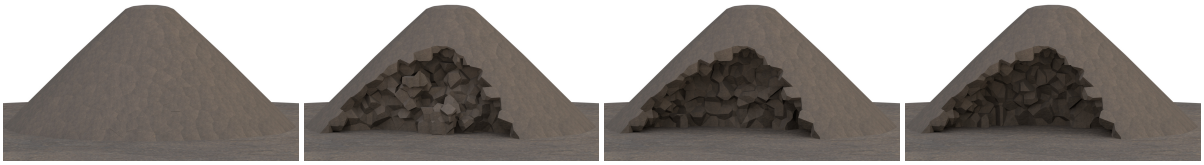

Figure 14: Hill3D *erosion sequence. Initial state, 23k, 46k and 70k water paths.*

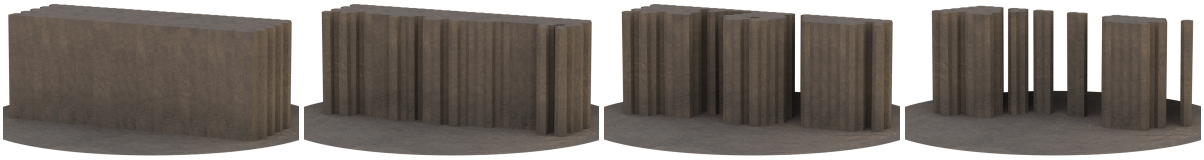

Figure 15: Columnar basalt *erosion sequence. Initial state, 1200, 2000 and 4000 water paths.*

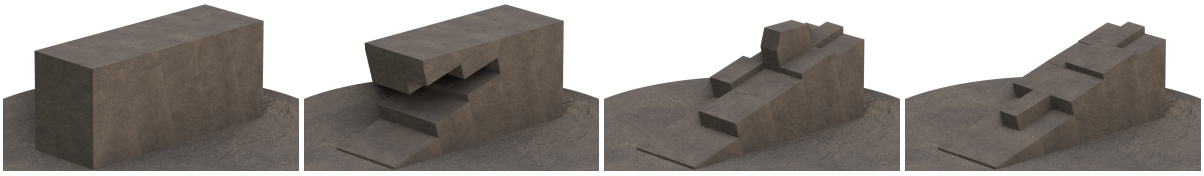

Figure 16: Slabs *erosion sequence. Initial state, 500, 1500 and 3000 water paths.*

marked as air cells. We then ran the simulation of 15k infiltration paths from a vertical direction slightly tilted towards the ridge wall. As a post-process step, we perform a boolean intersection operation between the remaining solid cells from the simulation and the original terrain volume. This ensures that the obtained terrain has only been modified by carving.

6.3. Discussion

Precomputed fracture locations from Voronoi cells have already been used before [Rag02, MCK13], but mainly in the context of brittle fracture after an impact. Instead, since we are representing a progressive weathering process, we precompute this pattern in order to efficiently simulate water propagation along the faces and how frost wedging pulls them apart.

This notion of weathering, or time progression, is also one major novelty of our work with respect to, for example, the implicit rock blocks by Paris et al. [PPG*20]. They also use a graph between neighboring points randomly placed inside the volume. In their case, however, the graph edges are removed at once given a set of discs, and a shard is generated for each connected component of points. Since shards are not removed, most of the initial model volume is present in their output model, they only generate a pattern of cracks. By replacing our shard generation with their method, and adding the links between adjacent implicit blocks, our simulation algorithm could extend their results with a temporal progression. Another improvement in our method is that, even though our individual cells are convex, connected components of cells can be concave. In contrast, their blocks are always represented as the convex intersection of half-spaces.

The method presented by Ito et al. [IFMC03] is, to the best of our knowledge, the closest work to our own in the sense that they do take into account progressive weathering. However, as mentioned in the previous work section, they rely on a costly voxelization and do not offer artistic controls apart from the distribution of random discs to disconnect neighboring voxels. On the other hand, they compute the stability of fractured blocks given gravity and friction forces, so they can model blocks that remain in place after breaking from the rest. This remains as future work in our case.

Lastly, another key difference between our method and those of [IFMC03] and [PPG*20] is that our erosion advances from the outermost layers towards the inner layers, due to our infiltration and absorption mechanisms.

6.4. Limitations

Currently, our method has some limitations. Voronoi cells might not allow us to obtain a certain desired fracture pattern; for example, an interleaved layered structure like in a brick wall. However, the generation algorithm could be extended to support other structures as long as it generates the two graphs for the simulation.

On the simulation side, while the current parameters are intuitive and easy to tune, it is not always straightforward to guide it towards a desired sequential effect. Resistance fields only produce a limited effect, since this term is also aggregated with orientation and life terms to determine the propagation probability towards a neighboring link. We could explore strategies to allow the user to balance or even neglect the effect of the different terms. Lastly, since we do not compute the stability of resulting rock structures and we are

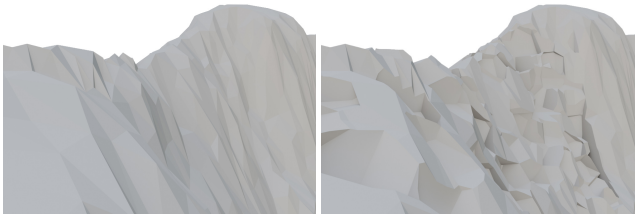

Figure 17: *Weathering computed from a Digital Elevation Map of Aneto peak, in the Pyrenees.*

doing a geometric approach for erosion (even if the propagation rules are physically inspired), we can obtain unnatural features like a large group of shards only joined to the main model by a tiny fragile bridge that would break in the real world. In the future, we want to take into account stress to solve these situations.

7. Conclusion

We have presented a method inspired by mechanical weathering processes that allows us to enhance existing models with a rocky appearance. The presented method is versatile and the controls and parameters are intuitive. Moreover, our implementation has been already integrated into Blender in the form of an add-on, so it can be directly tested in artistic pipelines.

There are possible directions to improve this work. For example, incorporating a computation of the forces acting on a cell and checking for its stability and potential breakage of links due to overstress. This could allows us to model immediate fracture propagation after a link breaks. Other possible research directions would be using a multi-resolution fracture approach, to generate consistent fractures at multiple scales, or avoiding the need for a prefracturing step and generate shards dynamically during simulation.

7.1. Acknowledgments

This work has been funded with Grant PID2021-122136OB-C21 funded by MICIU/AEI/10.13039/501100011033 and ERDF/EU, and with the support of the Department of Research and Universities of the Government of Catalonia (2021 SGR 01035). Luis Carranza was supported with Grant PRE2022-102912 funded by MICIU/AEI/10.13039/501100011033 and ESF+, and Oscar Argudo with a Maria Zambrano fellowship from Ministerio de Universidades and European Union—NextGenerationEU.

References

[DDKDSC20] DEPREZ M., DE KOCK T., DE SCHUTTER G., CNUDDE V.: A review on freeze-thaw action and weathering of rocks. *Earth-Science Reviews 203* (2020), 103143. doi:10.1016/j.earscirev.2020.103143. 2

[FCK22] FAN L., CHITALU F. M., KOMURA T.: Simulating brittle fracture with material points. *ACM Transactions on Graphics 41*, 5 (2022), 1–20. doi:10.1145/3522573. 2

[GGP*19] GALIN E., GUÉRIN E., PEYTAVIE A., CORDONNIER G., CANI M.-P., BENES B., GAIN J.: A review of digital terrain modeling. *Computer Graphics Forum 38*, 2 (2019), 553–577. doi:https://doi.org/10.1111/cgf.13657. 1

[HGB12] HASLER A., GRUBER S., BEUTEL J.: Kinematics of steep bedrock permafrost. *Journal of Geophysical Research: Earth Surface 117* (2012). doi:10.1029/2011JF001981. 2

[Hug11] HUGGETT R. J.: *Fundamentals of geomorphology*, 3 ed. Routledge fundamentals of physical geography series. Routledge, 2011. 2

[HW16] HAHN D., WOJTAN C.: Fast approximations for boundary element based brittle fracture simulation. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. doi:10.1145/2897824.2925902. 2

[IFMC03] ITO T., FUJIMOTO T., MURAOKA K., CHIBA N.: Modeling rocky scenery taking into account joints. In *Proceedings Computer Graphics International 2003* (2003), IEEE Comput. Soc, pp. 244–247. doi:10.1109/CGI.2003.1214475. 3, 8

[JFBB10] JONES M., FARLEY M., BUTLER J., BEARDALL M.: Directable weathering of concave rock using curvature estimation. *IEEE Transactions on Visualization and Computer Graphics 16*, 1 (2010), 81–94. doi:10.1109/TVCG.2009.39. 2

[KHM*20] KRS V., HÄDRICH T., MICHELS D. L., DEUSSEN O., PIRK S., BENES B.: Wind Erosion: Shape Modifications by Interactive Particle-based Erosion and Deposition. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters* (2020), Michels D. L., (Ed.), The Eurographics Association. doi:10.2312/sca.20201216. 2

[Mat08] MATSUOKA N.: Frost weathering and rockwall erosion in the southeastern swiss alps: Long-term (1994-2006) observations. *Geomorphology 99*, 1 (2008), 353–368. doi:10.1016/j.geomorph.2007.11.013. 2

[MBP14] MUGUERCIA L., BOSCH C., PATOW G.: Fracture modeling in computer graphics. *Computers & Graphics 45* (2014), 86–100. doi:10.1016/j.cag.2014.08.006. 2

[MCK13] MÜLLER M., CHENTANEZ N., KIM T.-Y.: Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Transactions on Graphics 32*, 4 (2013), 1–10. doi:10.1145/2461912.2461934. 2, 8

[MG08] MERILLOU S., GHAZANFARPOUR D.: A survey of aging and weathering phenomena in computer graphics. *Computers & Graphics 32*, 2 (2008), 159–174. doi:10.1016/j.cag.2008.01.003. 2

[MLC*21] MOHAMMADNEJAD M., LIU H., CHAN A., DEHKHODA S., FUKUDA D.: An overview on advances in computational fracture mechanics of rock. *Geosystem Engineering 24*, 4 (2021), 206–229. doi:10.1080/12269328.2018.1448006. 2

[PGGM09] PEYTAVIE A., GALIN E., GROSJEAN J., MERILLOU S.: Arches: a framework for modeling complex terrains. *Computer Graphics Forum 28*, 2 (2009), 457–467. doi:10.1111/j.1467-8659.2009.01385.x. 2

[PGP*19] PARIS A., GALIN E., PEYTAVIE A., GUERIN E., GAIN J.: Terrain amplification with implicit 3d features. *ACM Transactions on Graphics 38*, 5 (2019), 1–15. doi:10.1145/3342765. 1, 2

[PGP*21] PARIS A., GUERIN E., PEYTAVIE A., COLLON P., GALIN E.: Synthesizing geologically coherent cave networks. *Computer Graphics Forum 40*, 7 (2021), 277–287. doi:10.1111/cgf.14420. 1, 2

[PPG*20] PARIS A., PEYTAVIE A., GUERIN E., DISCHLER J.-M., GALIN E.: Modeling rocky scenery using implicit blocks. *The Visual Computer 36*, 10 (2020), 2251–2261. doi:10.1007/s00371-020-01905-6. 1, 3, 8

[Rag02] RAGHAVACHARY S.: Fracture generation on polygonal meshes using voronoi polygons. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (New York, NY, USA, 2002), SIGGRAPH '02, Association for Computing Machinery, p. 187. doi:10.1145/1242073.1242200. 8

[Ryc09] RYCROFT C. H.: *Voro++: A three-dimensional Voronoi cell library in C++*. Tech. rep., Lawrence Berkeley National Lab (LBNL), Berkeley, CA (United States), 2009. 3