EUROGRAPHICS 2019
A. Giachetti and H. Rushmeier
(Guest Editors)

*Volume 38 (2019), Number 2*
*STAR – State of The Art Report*

# A Review of Digital Terrain Modeling

Eric Galin[1], Eric Guérin[1], Adrien Peytavie[1], Guillaume Cordonnier[2], Marie-Paule Cani[2], Bedrich Benes[3], and James Gain[4]

[1]Université de Lyon [2]Ecole Polytechnique [3]Purdue University [4]University of Cape Town

**Abstract**

*Terrains are a crucial component of three-dimensional scenes and are present in many Computer Graphics applications. Terrain modeling methods focus on capturing landforms in all their intricate detail, including eroded valleys arising from the interplay of varied phenomena, dendritic mountain ranges, and complex river networks. Set against this visual complexity is the need for user control over terrain features, without which designers are unable to adequately express their artistic intent. This article provides an overview of current terrain modeling and authoring techniques, organized according to three categories: procedural modeling, physically-based simulation of erosion and land formation processes, and example-based methods driven by scanned terrain data. We compare and contrast these techniques according to several criteria, specifically: the variety of achievable landforms; realism from both a perceptual and geomorphological perspective; issues of scale in terms of terrain extent and sampling precision; the different interaction metaphors and attendant forms of user-control, and computation and memory performance. We conclude with an in-depth discussion of possible research directions and outstanding technical and scientific challenges.*

Categories and Subject Descriptors (according to ACM CCS): Shape Modeling [Computer Graphics]: —

## 1. Introduction

Terrain plays a fundamental part in many virtual scenes across a broad range of applications, including games, films, training, and simulation.

For example, open-world computer games often include large-scale natural environments for players to explore, films sometimes require scenic locations that either do not exist or are difficult or dangerous to film, particularly in the science fiction and fantasy genres, and training applications, such as combat, driving and flight simulators, rely on realistic environments to be effective. In such applications a bare earth terrain is usually the first part of the authoring pipeline to be subsequently augmented with props that represent rocks, trees, plants and buildings.

The shape of real-world terrains is complex and varied. Diverse landforms ranging from featureless plains to mountain ranges crumpled by tectonics and scored by erosion can co-exist within a single scene. Terrain formation is a combination of long-term influences, such as gradual erosive forces, and occasional catastrophic events, such as landslides and lightning strikes on bedrock. Furthermore, specific features appear at different scales, spanning tens to hundreds of kilometers in the case of tectonics and glaciation, whereas only evident over a few meters for certain forms of hydraulic and aeolian erosion. It is no wonder then that, despite more than four decades of research, there remain many unsolved challenges in terrain modeling.

A particular challenge is that Computer Graphics applications invariably require iterative authoring of the final terrain shape. Real-world terrains emerge from complex geomorphological processes for which simulation seems like a natural fit. Unfortunately, a direct simulation with only boundary and initial conditions is often at odds with user intent and rarely suffices.

In this paper, we present an overview and critical comparison of terrain generation methods in Computer Graphics. We begin with a consideration of the different terrain representations (including elevation and volumetric models), followed by coverage of terrain generation under three broad headings: *procedural generation* methods, which rely on algorithms (noise, faulting, subdivision, and the like) that do not directly emulate erosive processes; *simulation* techniques, which iteratively apply computer simulations of geomorphological processes, and *example-based* methods, which extract and combine scanned data of real-world terrains.

Finally, we compare existing methods on the basis of a number of practical considerations: the range of different landforms that can be represented (variety); the perceptual or geomorphological accuracy with which they are reproduced (realism); specific limits on feature extent and detail (scale); the types of user control available (authoring), and memory and computation overheads (efficiency). If a generated terrain is used in an application where rendering speed is important, there is a choice among several level-of-detail methods to improve performance, as described in the survey of [PG07]. These criteria serve two intended purposes: first, to

help developers select a terrain modeling method that best fits their particular application.

For instance, efficiency is vital for terrains generated on the fly (as opposed to pre-designed), while authoring is secondary, since there is no opportunity for direct designer involvement. Another consideration is whether the terrain is to be modeled entirely from scratch, dropped into a cut-out region in a broader terrain, or up-sampled from low-resolution input. The most appropriate technique will depend in part on such requirements.

Second, these criteria could be used to guide future research by identifying failings exhibited by current approaches. For ease of reference, Tables 2 and 3 provide a comparison of terrain modeling methods on the basis of achievable landforms variety, underlying method and model, and the supported authoring, scale, and efficiency.

## 2. Terrain representation

In this section, we present an overview of the underlying models used to represent and synthesize terrains. We focus on definitions of the terrain surface, and not on volumetric representations of the underlying geology. An overview of models for representing terrains and their subsurface geology can be found in [NLP*13] and a survey of cellular data representations for the entire globe (the digital Earth) in [MAAS15].

### 2.1. Elevation models

The terrain elevation can be defined as a function $h : \mathbb{R}^2 \to \mathbb{R}$, which is at least $C^0$ and represents the altitude at any point in $\mathbb{R}^2$. Such a definition restricts modeling to terrains without overhangs, arches, or caves. Let $\mathcal{T}$ denote the terrain defined over a domain $\Omega \subset \mathbb{R}^2$. The surface area of $\Omega$, expressed in $m^2$ or $km^2$ depending on the size of the domain, will be referred to as the *extent*. In general, and unless stated otherwise, $\Omega$ will be a rectangular domain $\mathcal{R}(\mathbf{a},\mathbf{b})$ where $\mathbf{a}$ and $\mathbf{b}$ are points in the plane representing opposite corners of $\Omega$. Let $h : \mathbb{R}^2 \to \mathbb{R}$ define the elevation of $\mathcal{T}$. The slope is defined as the norm of the gradient:

$$s(\mathbf{p}) = \|\nabla h(\mathbf{p})\|$$

The normal at a given point on the terrain is derived from the gradient of the elevation function $\nabla h$ by projecting it to $\mathbb{R}^3$ as $(-\nabla h(\mathbf{p}), 1)$, and then normalizing it:

$$\mathbf{n}(\mathbf{p}) = (-\nabla h(\mathbf{p}), 1)/\|(\nabla h(\mathbf{p}), 1)\|$$

**Function representation** The elevation can be represented by a procedural or closed-form expression of the function $h$. While this representation is compact in terms of memory and has infinite precision, but the evaluation of $h(\mathbf{p})$ at a given point may be computationally demanding. While in theory any suitable function $h$ can be used for representing elevation, the construction of a function for representing large terrains with realistic landforms and characteristics is a complex task (Section 3).

**Discrete heightfields** defined over regular grids are undoubtedly the most common representation for terrains. Heightfields consist of a collection of altitudes arranged on a regular 2D grid. They can be captured by remote sensing and are often referred to as Digital Elevation Models (DEMs). A regular square grid is defined by its rectangular region $\mathcal{R}(\mathbf{a},\mathbf{b}) \subset \mathbb{R}^2$ with $n$ subdivisions. Without loss of generality, we consider square grids of side length $a$ with $(n+1)^2$ elevation samples. In this representation, only the altitudes $z_{ij}$ are stored at the vertices of the grid $\mathbf{p}_{ij} = \mathbf{a} + (\mathbf{b}-\mathbf{a}) \bullet (i/n, j/n)$, with $(i,j) \in [0,n]^2$ and where $\bullet$ denotes the pairwise product of elements (Figure 1).

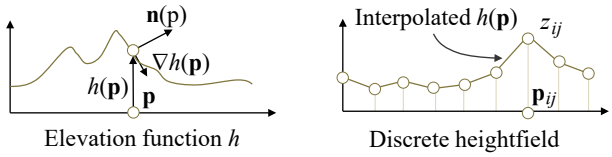


**Figure 1:** *Elevation can be represented by an analytic or procedurally defined function, or by discrete heightfield data, in which case the elevation at any point is reconstructed by interpolation.*

The accuracy or precision of the heightfield in the domain $\Omega$ is limited to $a/n$ and a continuous surface for the terrain needs to be reconstructed by interpolation of the elevation of grid points $z_{ij}$. A simple way to reconstruct the surface is to use two linear interpolations for the two triangles inside a cell, which matches the triangle polygonization of the surface of the terrain. Bi-linear interpolation produces elevation of class $C^1$ inside the cells, but is only $C^0$ at their borders. Higher order interpolations, *e.g.* , bi-cubic, define $C^k$, $k \geq 1$ continuous functions, but may generate elevations outside the range of the inputs $z_{ij}$. They are more computationally demanding, and require retrieving the values $z_{ij}$ on a larger neighborhood to compute the elevation $h(\mathbf{p})$ of a point $\mathbf{p}$ inside a grid cell.

In contrast to functions, this data-oriented representation supports different forms of elevated terrain at the expense of increased memory overheads (Figure 1, right), as it requires $O(n^2)$ storage. Reducing the accuracy of elevation data by quantization (in general using 8-bit or 16-bit) is one means of reducing memory costs. However, 8-bit quantization does not provide sufficient accuracy for procedural generation and simulation algorithms, and introduces visible step patterns when displayed. Heightfields are the most common data representation for digital terrains in the industry. They are used in many applications such as GIS, authoring tools, erosion simulations and video games. In the film industry, they are also used as bare terrains and usually completed with representations that allow true 3D features. Because heightfields are represented as grids, they also lend themselves to texture synthesis and machine learning methods.

**Layered representations** encode the different material layers of the terrain as a collection of ordered functions describing thicknesses in a pre-established layer layout. Material layers were proposed in [MKM89] as a way of modeling different sediments, and developed in [BF01, CGG*17] to represent different layers of granular materials, such as sand or rocks on top of the bedrock. The layered data representation has been widely used in erosion simulation ever since (Section 4). Layers can be represented as a set

of functions [GGP*15], where the bottom layer represents the elevation of bedrock $h_{\mathcal{B}}$, and each subsequent layer represents the thickness of other materials, including rocks $h_{\mathcal{R}}$, sand $h_{\mathcal{S}}$, or water $h_{\mathcal{W}}$ (Figure 2, left).

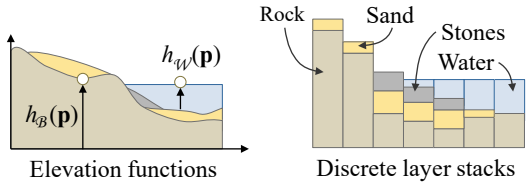


**Figure 2:** *Layered models represent different types of materials organized in a predefined sorting order (bedrock, then sand and rocks, followed by water).*

A discrete layer representation consists of an ordered collection of discrete grids (Figure 2, right) and the data structure reduces to a stack of heightfields. This data-structure is a compromise between single-layer heightfields and voxels, for which every layer is further regularly divided. Moreover, the vertical resolution of layered heightfields is potentially infinite, whereas voxels are grid limited. Alternatively, the ordering of the layers can be explicitly defined at every grid position, at the expense of more complex dynamic stack management of the different materials.

### 2.2. Volumetric models

Elevation alone cannot manifest internal structure. Instead, volumetric representations must be used to capture the folds and faults of different geological strata allowing terrains with overhangs, caves, and arches.

Volumetric models are defined by a function $\mu : \mathbb{R}^3 \to \mathcal{M}$, where $\mathcal{M} \subset \mathbb{N}$ denotes the index of the material at any point in space. Single material models use only one type of solid material, *i.e.* $\mathcal{M} = \{0, 1\}$, where 0 refers to air and 1 to bedrock.

**Function representations** define the terrain as an implicit surface extracted from a field function $f$ and defined as:

$$\mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) = 0\}$$

A given elevation function $h$ can be easily converted to a corresponding implicit representation by defining: $f(\mathbf{p}) = h(\mathbf{p}_{xy}) - \mathbf{p}_z$, where $\mathbf{p}_{xy}$ denotes the projection of the point $\mathbf{p}$ onto the plane, and $\mathbf{p}_z$ denotes its elevation. The material function $\mu$ can be derived from the field function $f$ as $\mu(\mathbf{p}) = 1$ if $f(\mathbf{p}) > 0$, *i.e.* inside the bedrock, and $\mu(\mathbf{p}) = 0$, otherwise. The field function $f : \mathbb{R}^3 \to \mathbb{R}$ should be at least of class $C^0$, although functions with higher continuity are often preferred since then implicit surface algorithms that require $C^1$ can be applied. Although this representation can, in theory, represent various complex landforms with concavities, it is seldom used in practice likely because of the complexity of authoring implicit surfaces and additional costs incurred in reconstructing the surface of the terrain through polygonization (Section 3.2.3).

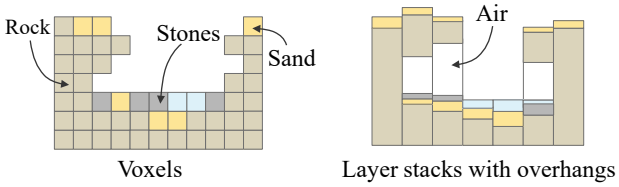


**Figure 3:** *Voxel representations allow the modeling of arches, caves or overhangs, but are limited by their discrete nature.*

**Voxels** offer a way to describe volumetric terrains, but at a high memory cost because of their explicit spatial enumeration. Space is partitioned into a 3D regular grid, and each cell is assigned a material index (Figure 3, left). The data structure can be optimized using compression techniques such as a Sparse Voxel Octrees [LK11] in order to reduce the memory cost. Further compression can be achieved by generalizing the tree structure to a directed acyclic graph [KSA13] and introducing symmetry transforms [VMG16].

Voxels are often used for representing terrains involving complex simulation processes, such as erosion (Section 4) or for representing complex and detailed landforms such as arches and caves (Section 3). Although the material function $\mu(\mathbf{p})$ can be evaluated efficiently, in addition to the disadvantage of their memory cost, their discrete nature does not lend itself to modeling continuous features such as gentle slopes.

**Hybrid representations** are inspired by layered-material representations, and exploit vertical run-length encoding of the different layer stacks to compress layers into intervals of constant material [PGMG09a].

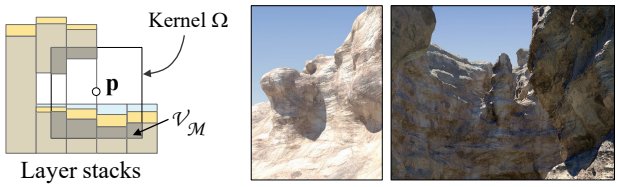


**Figure 4:** *Overview of the computation of the convolution $\mu \circledast k(\mathbf{p})$, defined as the volume of material $\mathcal{V}_{\mathcal{M}}(\mathbf{p})$ inside the compact support $\mathcal{V}_\Omega$ of the kernel function. Convolution produces a smooth implicit surface from the discrete layer stacks.*

The uppermost terrain is defined as a continuous implicit surface whose field function $f$ is derived from the discrete layer stacks $\mu$ by computing a convolution, denoted by $\circledast$, between a characteristic function of the layer stack and a box filter $k$. In spirit, this convolution plays the same smoothing role as interpolation does in heightfields. The field function $f : \mathbb{R}^3 \to [-1, 1]$ is defined as:

$$f(\mathbf{p}) = 2\mu \circledast k(\mathbf{p}) - 1 \qquad \mu \circledast k(\mathbf{p}) = \frac{\mathcal{V}_{\mathcal{M}}(\mathbf{p})}{\mathcal{V}_\Omega}$$

Here, the term $\mathcal{V}_\Omega = \sigma^3$ denotes the volume of the cubic compact support $\Omega$, which is a box of side length $\sigma$ and $\mathcal{V}_{\mathcal{M}}(\mathbf{p})$ is the volume of material (Figure 4). Unlike voxels, which are by definition a

discrete structure, the implicit surface representation derived from the layered representation generates a smooth continuous surface, at the expense of computationally demanding field functions.

## 3. Procedural generation

In this review, we call procedural any technique that is not directly related to a physical simulation and that does not use real data as exemplars. Instead of simulating the physical processes that sculpt the terrain such as hydraulic erosion, procedural approaches are phenomenological and aim at directly reproducing the effects of the phenomena. They often rely on the properties of terrains, such as the fractal characteristics, and construct terrain from the observations of the real world without taking real data as input.

Procedural generation methods can be classified into two categories. *Large scale terrain generation* methods synthesize a terrain over the entire plane $\mathbb{R}^2$ or a large domain $\Omega \subset \mathbb{R}^2$, focus on the fractal and self similar properties of the relief at different scales, and in general provide only little indirect control over the generated features (Section 3.1). In contrast, *procedural landform methods* target the synthesis of a specific landforms such as rivers, canyons, or hills, operate at a smaller scale and provide several direct or indirect control parameters to tune the resulting shapes (Section 3.2).

Both large scale and landform methods often rely on fractional Brownian motion [MVN68] (fBm), which is one of the most commonly used procedural models due to its simplicity. It can be synthesized in many ways, including random walk, subdivision schemes or additive synthesis of functions [EMP*98].

### 3.1. Large scale terrain generation

Those methods build on the observation that some terrains such as eroded mountain ranges or coastlines show dendritic structures that have fractal properties, *i.e.* self-similarities at several scales. Early works on terrain modeling often rely on fractal models; therefore most large scale terrain generation algorithms are *a-dimensional*, *i.e.* the fractal nature of the underlying structures allows the creation of an infinity of details that look similar at every scale.

#### 3.1.1. Subdivision schemes

Subdivision schemes iteratively refine an input terrain in order to introduce more and more details. They are used to produce fractal features, although not every subdivision algorithm follows the statistical fractional Brownian motion process. The recursive midpoint displacement subdivision algorithm [FFC82a, FFC82b] progressively refines an input grid by adding fractal details. Let $k$ the current level of the grid, new points $\mathbf{p}_{k+1}$ are introduced by averaging existing neighbor points $\mathbf{p}_k$ and by displacing them with a random value. As the iterations progress, the standard deviation of the random value distribution decreases, and the decreasing factor is directly related to the fractal dimension of the result.

Authors have proposed several modified subdivision and averaging schemes (including, triangle, square, and diamond-square) in order to limit visual directional artifacts [Mil86, Lew87, Man88]. The most popular is the diamond-square [FFC82b], even though it suffers from local extrema artifacts visible at the first subdivisions locations (Figure 5).The square-square [Mil86] scheme solves the problem of local extrema by using a more balanced subdivision. Different types of polygons have also been used in subdivisions with mixed types [DKW94], which necessitates maintaining both topology and geometric consistency in the subdivision process.

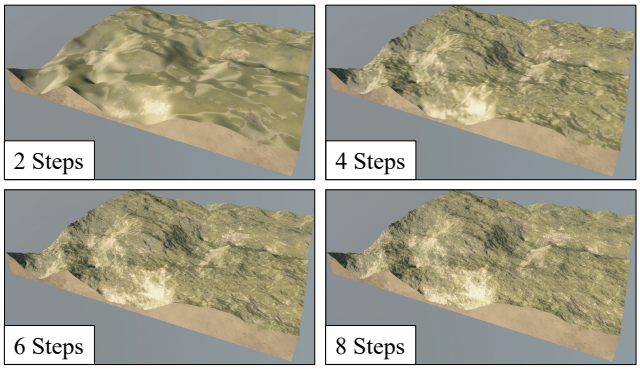


**Figure 5:** *Several iterations of the diamond square algorithm.*

#### 3.1.2. Faulting

The faulting algorithm was introduced in [Man82] and developed in [Vos91, EMP*98]. Starting from a flat terrain, the algorithm iteratively generates random vertical faults as lines: point on either side are displaced upwards or downwards according to the distance to the fault (Figure 6). Let $d(\mathbf{p},\phi_i)$ denote the distance to the faulting line $\phi_i$, and $g$ a smooth step function parametrized by a radius of influence $R$, such as the following quartic:

$$g(r) = \left(1-(r/R)^2\right)^2 \text{ if } r < R \qquad g(r) = 0 \text{ otherwise}$$

The elevation function can be defined by summing the influence of the faults as:

$$f(\mathbf{p}) = \sum_{i=0}^{i<n} f_i(\mathbf{p}) \qquad f_i(\mathbf{p}) = a_i\, g \circ d(\mathbf{p},\phi_i)$$

The coefficients $a_i$ represent the random vertical displacement applied at every iteration, and are in general decreasing in the same way as recursive subdivision with the Hurst exponent $(1/2^H, 0 < H < 1)$ and then $D = 3-H$ is the fractal dimension. While the line cuts the surface in an arbitrary position and the divided parts will be different in size, it is important that in average the random faults select each side with the same probability, otherwise the terrain will either sink or bloat.

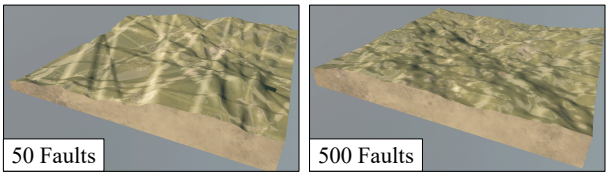


**Figure 6:** *The faulting algorithm generates random faults as lines and raises or lowers the terrain on either side of the fault line.*

The method can be generalized to other faulting geometries such as curves or circles, and scales to any size and can be extended to generate fractal planets [FFC82b] when applied to a sphere. [KSU07] proposed a variation of the faulting algorithm to control the shape of procedurally generated terrain by prescribing control parameters such as the height and extent of the base region at a given location.

#### 3.1.3. Noise

Noise functions [Per85, Per89, Wor96, LLDD09] have been widely studied and used for a variety of natural phenomena, including terrain modeling. A complete description of noise functions is beyond the scope of this paper, we refer the reader to [LSC*10] for an overview of procedural noise functions.

Noise functions have been proposed as basis functions for representing infinite terrains, *i.e.* defined over $\mathbb{R}^2$, as a set of scaled and warped noise functions [MKM89, EMP*98]. By adding several noises at different scales and amplitudes, it is possible to build a function that locally resembles a real terrain. Fractional Brownian motion, also sometimes referred to as turbulence and denoted as $t$, can be implemented as a combination of multiple steps of noise each with a different frequency and amplitude. In the context of procedural generation, the variation in frequency from a step to the next is called *lacunarity*, whereas the variation in amplitude from a step to the next is called *gain* or *persistence*.

Let $n$ denote a smooth noise function that maps from $\mathbb{R}^2$ to $[-1,1]$ interval; without loss of generality we consider that this fundamental function has a primary frequency of 1; which means that it interpolates values or gradients defined at every integer position. The turbulence function $t$ is defined by summing the contributions of noises with varying frequencies and amplitudes:

$$t(\mathbf{p}) = \sum_{i=0}^{i=o-1} a_i\, n(\varphi_i \mathbf{p})$$

where $a_i$ refer to the different amplitudes, $\varphi_i$ to the different frequencies. The number of terms $o$ is often referred as the number of octaves, even if the frequencies are not multiples of 2. In general, the amplitudes and frequencies are defined as a geometric series: $a_i = a_0\, p^i$ and $\varphi_i = \varphi_0\, l^i$ where $a_0$ and $\varphi_0$ are the base amplitude and frequency, $l \in [0,1]$ is the *lacunarity*, and $p \in [0,1]$ the *persistence*. The persistence defines how the amplitude decreases in the successive octaves. Values $p \approx 1$ produce very jagged terrains, whereas values $p \approx 0$ drastically limit the impact of the successive frequencies.

Exponential noise was proposed in [Par15] to better reproduce the slope distribution observed in real terrains. The exponential slope distribution spectrum is obtained by analyzing real terrain data, and used to constrain the generation process using gradient noise whose gradient samples match the multi-fractal spectrum [vLJ95, Par14, Par15].

The smoothness of the noise function $n$ prevents the creation of crests and ridge lines that can be found in mountainous terrains (Figure 7 left). *Ridge noise* was therefore introduced with a view to generating sharp features such as crests or ridges, and simply defined as $r(\mathbf{p}) = 1 - |n(\mathbf{p})|$. Although the absolute value was already present in the definition of the turbulence function in [Per85], the idea of turning it upside-down to produce crests on the top of mountain ranges was introduced in [EMP*98].

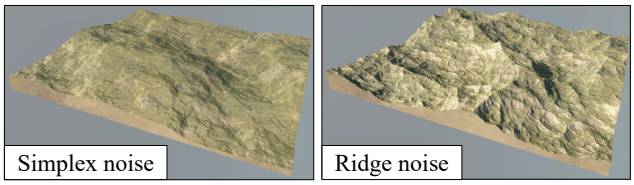


**Figure 7:** *A sum of simplex noise n (left) and ridge noise r (right) with 8 octaves: ridge noise generates crest lines suitable for young mountains, whereas simplex noise is more suitable to modeling hills.*

The fractal property of these noise functions is uniform, *i.e.* they generate mono-fractals, which does not conform to real landscapes. Multi-fractal terrains can be obtained by modifying the sum of octaves so that the amplitude of higher frequencies $a_{k+1}$ should be weighted by a function $\alpha$ according to value of the previously computed octaves [EMP*98]. This can be obtained by using the following recurrence relation:

$$t_{k+1}(\mathbf{p}) = \alpha(t_k(\mathbf{p}))\, a_{k+1}\, n(\varphi_i \mathbf{p}) + t_k(\mathbf{p}) \qquad t_0(\mathbf{p}) = a_0\, n(\varphi_0 \mathbf{p})$$

Lower elevations $t_k(\mathbf{p})$, computed at step $k$, scale down higher frequencies in order to smooth valleys, whereas higher values will boost high frequencies to enhance the mountains peaks with small details. Multi-fractal generate terrains that are not uniform featuring smooth areas in plains and ridged landforms in mountains (Figure 8 left).

Summing the same scaled noise function $n$ can lead to grid artifacts that can be avoided by applying warping functions made of rotations and translations. This concept is easily formalized by the means of an affine transformation applied to $\mathbf{p}$, *i.e.* using $n(\mathbf{R}_i\,\mathbf{p} + \mathbf{v}_i)$ where $\mathbf{R}_i$ is a $2 \times 2$ random rotation matrix and $\mathbf{v}_i$ is the translation vector.

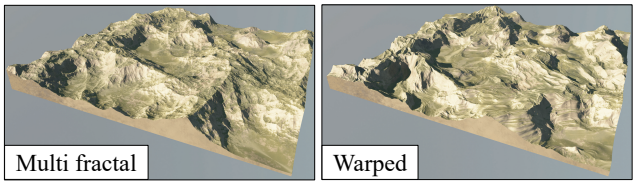


**Figure 8:** *A multi-fractal sum of 8 octaves of ridged noises (left) and its warped version (right).*

**Warping** Generally, warping can be used to deform the domain and break the monotonicity and regularity of noise. Formally, a continuous warping function may defined as: $\omega : \mathbb{R}^2 \to \mathbb{R}^2$; and warped terrains are computed by evaluating $n \circ \omega(\mathbf{p})$ instead of $n(\mathbf{p})$. The warping function $\omega$ may be defined as a sum of scaled displacement functions created from noise. Low frequency vector offsets are commonly used for this purpose (Figure 8 right). Limited erosion effects can be approximated [dCB09] by using warping functions based on scaled noise, oriented in the direction of the

gradient of the underlying terrain. Domain warping still only provides a coarse approximation of erosion, and simulation techniques are needed to obtain more realistic effects (Section 4).

#### 3.1.4. Analysis

Subdivision-based and noise-based approaches are commonly used to generate large scale synthetic terrains that have strong fractal properties. Unfortunately, these techniques do not capture the high-level structure of real-world terrains, such as the branching patterns evident in mountain ranges.

Noise-based terrain modeling is a powerful tool, but it can be difficult to control, requiring both a technical understanding of the many parameters and strong artistic skills. Subdivision is usually easier to control because it approximates a coarse terrain, which naturally lends itself initial specification by the user. In contrast, noise-based functions provide more flexibility and allow a variety of effects when combined with warping and other functions such as clamping or linear interpolation and blending. It is worth noting that noise-based terrain modeling is widely used in industrial software, while subdivision algorithms are almost absent. The popularity of noise functions comes from the independent evaluation of $n(\mathbf{p})$, which allows for an efficient parallel implementation on graphics hardware, allowing designers to brush the terrain with procedural functions.

However, this independence of the elevation $f(\mathbf{p})$ for a given input point $\mathbf{p}$ from its neightbours is simultaneously a significant disadvantage, because such function-based methods do not capture complex erosion phenomena, most importantly *material transport*. The very nature of terrains is not strictly fractal: many phenomena do not yield fractal landforms, therefore the realism of purely based fractal methods is limited to a restricted subset of scales, for example small-scale features, or to specific landforms such as some fractal coastlines.

The lack of local control exhibited by these procedural techniques has driven researchers to focus on generating rather specific landforms for applications where the speed of procedural function-based methods is paramount.

### 3.2. Landform generation

Local procedural techniques aim at shaping specific landforms such as rivers, cliffs or canyons without relying on simulations or synthesis from exemplars. They operate by the means of geometric control parameters such as features curves or anchor points. Contrary to large scale generation approaches that are often dimensionless, local methods introduce the notion of spatial dimension, which is crucial for reproducing specific structures and patterns.

#### 3.2.1. Controlled subdivisions

Because fractal subdivisions only provide a limited control to the user, several authors have tried to prescribe features such as rivers, crests, or any landforms in the generation process.

River networks with consistent watersheds were created by constraining the mid-point displacement algorithm [KMN88] to generate rivers. Subdivision rules were applied directly to river trajectories [PH93] and later extended to entire planets to generate large scale watersheds [DGGK11].

Another way to control the generation process is to constrain the fractal reconstruction [Bel07, BA05a, BA05b] with a subdivision algorithm to respect important features of the terrains like crests or river trajectories (Figure 9). Recently, these methods were adapted and implemented on graphics hardware [TB18].

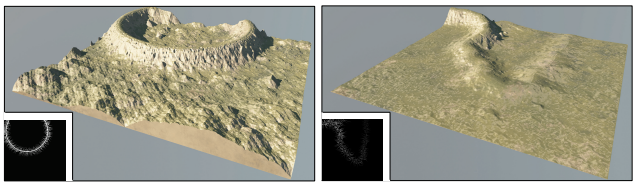

**Figure 9:** *Examples of controlled subdivisions obtained by [BA05a] (left) and [Bel07] (right), and their corresponding input constraints (insets).*

Hnaidi [HGA10a] proposed to break the systematic nature of fractal subdivision rules of a Projected Iterated Function System by inserting details at each step. Artists can thus combine the ease of use of subdivision schemes and the controllability of free forms. Ariyan *et al.* [AM15] used a formalism based on subdivision of planar networks using a path planning algorithm. These planar curves are then assigned heights using altitude profiles and the final heights are computed using interpolations.

#### 3.2.2. Feature-based construction

Feature-based construction methods come in two categories: *curve-based* terrain generation techniques which rely on generating feature curves to diffuse terrain characteristics and synthesize a variety of landforms, such as ridge lines, coastlines or rivers, and *primitive based* construction trees that take their inspiration from implicit surface models and combine specific landforms primitives by blending or warping operators.

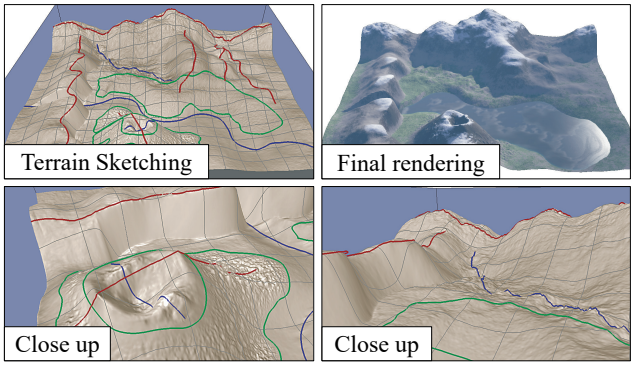


**Figure 10:** *Terrain sketching can produce a variety of landscape features. Peaks, cliffs, a volcanic cone, hills in different orientations, and a river canyon are sketched in a single terrain, with close-ups and a final rendering (from [GMS09]).*

**Curve-based models** The work of Gain *et al.* [GMS09] represents one of the earliest sketch-based interfaces for interactive modeling of terrains from control curves (Figure 10). The user specifies landforms, such as mountains and valleys, through a variety of sketched curves, including: baselines (representing the projection of a crest onto the ground plane), elevations (capturing the vertical shape of a crest), and boundaries (limiting how far the landform extends to either side of the baseline). The elevations can be sketched as silhouettes from a particular viewpoint, with additional controls to indicate how they fit relative to existing landforms. Each set of baseline, elevation and boundary curves is submitted to a multi-resolution deformation process which ensures that the terrain is constrained to fit the curves. One unusual aspect is that the noise attributes of the sketched elevation curve are analyzed and used to add corresponding wavelet noise to the local terrain being deformed. This means that a jagged mountainous silhouette will induce similar characteristics in the surrounding terrain. The technique can also be used to build terrains from scratch or modify existing landscapes.

Rusnell *et al.* [RME09] used a technique to blend feature curves defined by the combination of an elevation curve (defining crests) and a monotonically decreasing cross-sectional profile (defining slope on either side of the crest). The influence of features can be varied between a simple normalized sum and a weighting that favors feature curves locally. The key aspect that differentiates this from similar Euclidean distance-based schemes is that distance is calculated efficiently on the grid using a shortest-path algorithm. Every grid-node is connected edge-wise to its eight neighbors and the shortest path is used to determine distance to a feature curve, whose grid-vertexes serve as generator nodes for the algorithm.

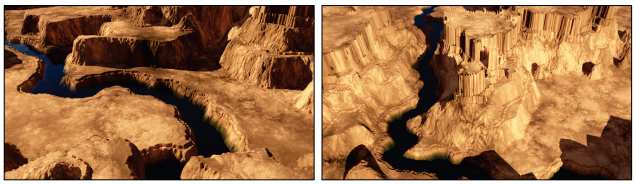

**Figure 11:** *Different canyon landscapes generated by procedural noise (from [DCPSB14]).*

A specific approach for modeling and generating canyon landscapes was introduced in [DCPSB14]. Given an input heightfield generated by using a sum of procedural noises, the authors layer the terrain into terraces by applying clamping functions (Figure 11). Mesas are isolated by creating paths as curves between valleys to create non connected mountain ranges. The trajectory of the river is then generated by using a shortest path algorithm between control points minimizing the elevation range between control points. Finally, the foothills are smoothed.

Hnaidi *et al.* [HGA*10b] created terrains from prescribed *feature curves* representing the skeleton of crests, rivers and major stream landforms (Figure 12). Feature curves are 2D splines with additional information attached along the curve, describing the elevation and the slopes on each side of the curve. This resulting vector-based model is compact in terms of memory ($\approx 10$ kB). The surface is reconstructed using a modified diffusion equation taking into account noise parameters and the slope. Noise parameters and gradient constraints are propagated across the cells by a standard diffusion equation. The obtained constraints maps are then used in a second pass in order to guide the diffusion in particular areas with slopes. Details are generated using a noise map synthesized using the diffused noise parameters, and terrain is obtained by adding details to the smooth diffused terrain. The multi-grid implementation of the diffusion process on graphics hardware allows for interactive generation rates. While the maximum resolution of the synthesized terrain was constrained by the available memory at that time, *i.e.* $2048 \times 2048$, it could probably be higher with recent hardware.


**Figure 12:** *Feature curves allow to author different types of terrains by prescribing the trajectories of rivers of the shape of crest lines (from [HGA*10b]).*

**Construction trees** The general approach consists in locally modifying the elevation of the terrain by using different kinds of models. This can be achieved by using compactly supported primitive function $f_i$ representing specific landforms over their domain $\Omega_i$, and combining them together to construct complex terrains [GGP*15], using either procedural generation approaches [GGG*13] or example-based sparse synthesis [GDGP16] (Section 5).

Génevaux *et al.* [GGP*15] proposed a hierarchical construction tree combining primitives representing a variety of landforms (Figure 13 right). The primitives at the leaves of the tree represent landforms at different scales such as hills, mountains or mountain ranges, valleys, riverbeds, and are implemented as elevation functions $h_i(\mathbf{p})$ associated to a compactly supported weighting function $\alpha_i(\mathbf{p})$ over their domain $\Omega_i$. The primitives can be hierarchically combined using different operators such as carving, blending or warping to apply deformation effects. The major contribution is a hierarchical model that allows to model and control the placement of landforms features.

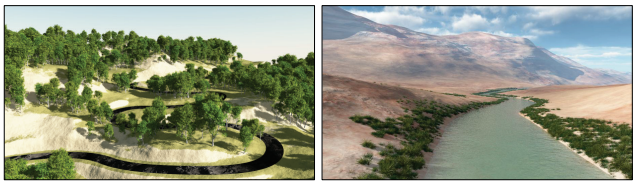

**Figure 13:** *Terrains produced by blending disc and curve primitives, from [GGP*15] (left) and [GGG*13] (right).*

Génevaux *et al.* [GGG*13] proposed to fully generate a terrain from its river-network (Figure 13 left). The river network grows over the input terrain by using a grammar that satisfies

Horton-Strahler properties. The Horton-Strahler number quantifies the branching complexity of the geometric graph representing the drainage network [Hor45]. The Holto-Strahler number is computed on every edge of the graph by using the maximum uphill edges values and by incrementing in cases of equality. Leaves of the graph (river sources) are initialized to one. Each river of the network is labeled with respect to the empirical Rosgen classification of watercourses [Ros94] so that is embeds useful geometrical information. The elevation of the terrain is derived from the propagation of the elevation along the trajectories of the rivers and are computed using a hierarchical terrain construction tree.

#### 3.2.3. Volumetric procedural terrains

Volumetric procedural methods are not very present in the literature. Gamito *et al.* [GM01] introduced implicitly-modeled terrains by defining a field function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ and by warping space along the vertical axis to generate overhangs. Recall that $f$ is derived from $h$ as $f(\mathbf{p}) = h(\mathbf{p}_{xy}) - \mathbf{p}_z$. Instead of defining the elevation $h$ as a function of points in the domain $\Omega \subset \mathbb{R}^2$, the warped surface of the terrain is therefore implicitly defined as:

$$\mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3 \mid f \circ \omega^{-1}(\mathbf{p}) = 0\}$$

The warping function $\omega^{-1} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ deforms space horizontally and transforms steep parts of the elevated terrain into a concave surface resembling cliffs with overhangs. In practice, the warping function can be defined as a procedural displacement function based on a sum of 3D noise functions, smoothly clamped to a given region of influence in space. Let $\alpha : \mathbb{R}^2 \rightarrow [0,1]$ denote the compactly supported function defining the region of influence, we have:

$$\omega^{-1}(\mathbf{p}) = \mathbf{p} + \alpha(\mathbf{p}) \sum_{i=0}^{i=n-1} \mathbf{n}(\varphi_i \mathbf{p})$$

Peytavie *et al.* [PGMG09a] allow the modeling of 3D features like overhangs by using *void* as a material layer (Figure 2). The corresponding implicit representation can be obtained from a layer stack model to smooth the surface (Figure 14).


**Figure 14:** *Arches and overhangs with different materials (bedrock and sand) generated by the hybrid layer-stack implicit surface representation (from [PGMG09a]).*

The complexity of implicitly-defined 3D terrains makes it harder to visualize than standard heightfields, requiring a computationally demanding polygonization step. Specific methods [SBD13] were proposed to speed up the generation of the mesh by using an adaptive level of detail algorithm based on a tetrahedral cell refinement process.

Close to the formalism introduced in [HGA*10b], Becher *et al.* [BKRE17, BKRE18] used 3D feature curves to generate volumetric terrains (Figure 15). Feature curves are augmented with control parameters that define the two tangent surfaces on both sides of the curve, together with noise constraints. The model is embedded in a voxel grid and the constraints are set into that grid, leading to a sparse constraints representation that prescribes normals and noise parameters. These constraints are then diffused over the voxel grid to obtain a dense representation.

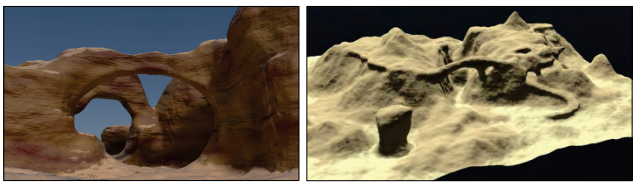

**Figure 15:** *Example of volumetric terrains featuring arches and overhangs produced by 3D curves (from [BKRE17, BKRE18]).*

The material function $\mu(\mathbf{p})$ defined over the voxel grid is reconstructed by propagating material from the initial positions of the curves by the means of the diffused normal map combined with a distance field. The surface of the terrain is obtained by polygonizing the corresponding implicit surface. The diffused noise parameters are used to displace the final mesh and improve the appearance of the surface of the terrain. The method can be implemented on graphics hardware to speed-up the computationally demanding diffusion and surface generation steps, therefore allowing for interactive authoring, with the current limitation that the resolution of the voxel grid should be limited to $128^3$.

#### 3.2.4. Analysis

To avoid the systematic fractal aspect of recursive subdivisions, constraint generation methods rely on user-provided constraints to adapt the subdivision process locally to create the desired landforms. User-controlled perturbations in the subdivision process were also proposed as a means to obtain terrains with different fractal characteristics.

In contrast, local methods and more specifically feature-based modeling provide a more intuitive way of authoring terrains: a complete discussion will be presented in Section 6.4. The main challenge of terrain modeling from features stems from the difficulty to propagate a sparse information, *i.e.* , specific landforms such as peaks or rivers, over an entire terrain, *e.g.* , generating hills between mountain ranges and plains. Several processes such as thermal diffusion, or shortest path computation have been proposed to address this problem. In those approaches, the global structure of the terrain often results from the authoring process during the creation. Because the placement of many features can be a tedious task for the user, research has also focused on the automatic creation of structuring features such as rivers.

## 4. Simulation

While *procedural approaches* are phenomenological, i.e., focus on the generation of a particular phenomenon, *simulation techniques*

model the causes and the effects that result from a simulation process. Simulations can be thought of as complex systems where the landform patterns and structures emerge from the interaction of the simulated elements.

Most of the simulations for terrain modeling are based on erosion. *Erosion* is the action of surface processes that remove material from one location and then *transport* it to another location, possibly outside the domain. Note that erosion is different from *weathering*, which involves no movement, but is rather a modification of physical properties and surface aspect.

Erosion can be described as a three-step process (Figure 16): the material is eroded and *detached* from the underlying terrain, then *transported* by the agent, and eventually *deposited* at a different location. Erosion processes include, but are not limited to, rainfall and surface runoff, river and stream erosion, coastal and sea erosion, glacial erosion, wind erosion, and mass movement.

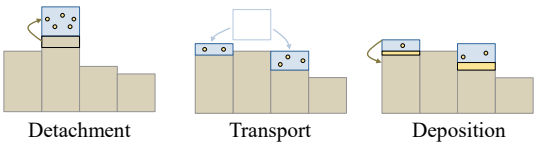


**Figure 16:** *Erosion processes detach some material that is then transported by the erosion agent and eventually deposited at a different location.*

Simulation methods compute the evolution of the boundary between the terrain and the erosion agent over time. They define the variation $\partial h/\partial t(\mathbf{p})$ as a function of the characteristics of the terrain such as the slope $s(\mathbf{p})$, the elevation (if the heightfield is being considered) $h(\mathbf{p})$ or the laplacian $\Delta h(\mathbf{p})$ , and the environment which includes the resistance of the bedrock $\rho(\mathbf{p})$, the tectonic uplift $u(\mathbf{p})$, or the amount of precipitation or wetness.

Different types of erosion can then be modeled by equations that describe various phenomena (for example in geomorphology [TH10]) such as the Stream Power equation [WT99], h ill slope erosion [BS97], debris flow equation [SD03], and other phenomenologically inspired equations. We classify the different simulation techniques according to the erosion agent that influences both the detachment and transportation phenomena.

### 4.1. Thermal erosion

Thermal erosion combines thermal weathering and mass movement, representing the downward movement of rocks and sediments on slopes, mainly due to the force of gravity. It is caused by water present inside cracks and small intrusions of the material boundary. As the temperature changes, the different thermal expansion of water and the material causes the material to break and fall.

#### 4.1.1. Heightfield thermal erosion

Thermal erosion was introduced in [MKM89] as a proxy for a group of various erosional processes other than hydraulic erosion. Note that thermal weathering is one of the many causes of rock breakage, large-scale landslides and the accretion of granular material, all of which results in a perceived regularity in the slope angle of many mountain and hill slopes (Figure 17).

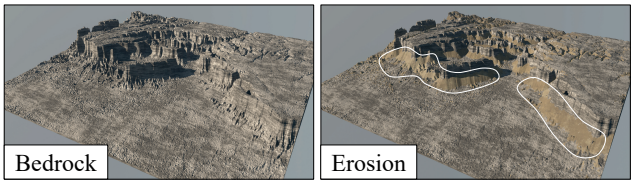


**Figure 17:** *Thermal erosion: (left) bare bedrock, and (right) talus deposits and scree at the base of crags and cliffs produced by thermal erosion.*

The transportation is caused by gravity and relies on the concept that the deposited granular material has an inner friction that stops the movement when a so-called talus angle has been reached. Let $\theta$ denote the repose angle, also referred to as the talus angle, of the deposited (granular) material. The equation is:

$$\frac{\partial h}{\partial t}(\mathbf{p}) = \begin{cases} -k(s(\mathbf{p}) - \tan\theta) & \text{if } s(\mathbf{p}) > \tan\theta \\ 0 & \text{otherwise.} \end{cases}$$

The talus angle can be used as a parameter to control the slope of the screes and cones that form at the bottom of the eroded landforms. The talus angle varies within $30-45^\circ$ degrees interval for earth forms.

Thermal erosion can be simulated as a relaxation process: at each time step the slope of the terrain $s(\mathbf{p}_{ij})$ is computed: if it is lower than $\tan\theta$ no erosion occurs, otherwise a certain amount of material proportional to $\tan\theta - s(\mathbf{p})$ is eroded and removed from the bedrock layer and distributed to the neighboring cells (Figure 18). When operating with a layered model, the amount of removed bedrock is converted to silt. The algorithm stops when there is no more material to move.

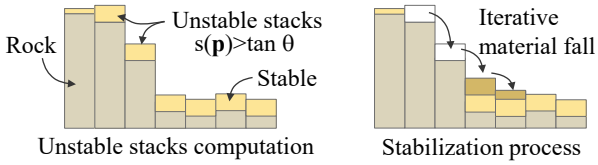


**Figure 18:** *Overview of erosion: erosion occurs on piles whose slope is greater than the talus angle.*

Roudier *et al.* [RPP93] introduced bedrock resistance for both hydraulic and thermal erosion. The concept consists in approximating the underlying geology of the terrain by defining a volumetric function $\rho : \mathbb{R}^3 \rightarrow \mathbb{R}$ representing the resistance of the different materials. The erosion equation is then modified as:

$$\frac{\partial h}{\partial t}(\mathbf{p}) = -\rho(\mathbf{p})\,(s(\mathbf{p}) - \tan\theta)$$

In general, the resistance is constant along a vertical line which allows it to be defined by a simpler function $\rho : \mathbb{R}^2 \rightarrow \mathbb{R}$.

#### 4.1.2. Specific landforms

Thermal erosion was used to simulate table mountains (mesas) in [BA05c]. The thermal erosion breaks rimrock (bedrock) into falling rocks that behave like granular material forming the typical accretion cones on the hillside of table mountains.

A variant of thermal erosion was proposed for small-scale volumetric simulation of cliff retreat in [IFMC03]. This algorithm procedurally generates faults around voxel groups that represent rock blocks and perform a discrete simulation of the detachment and falling of blocks. A set of blocks at the surface is chosen depending on sliding conditions, and removed from the system. Although not considering a time step, or a stochastic time-dependent probability of block breakage, this approach relies on a repose angle that generates talus, and therefore is a variant of thermal erosion.

#### 4.1.3. Cliffs and overhangs

The previously described thermal erosion algorithms operate on heightfield data-structures and do not allow for rock detaching and falling from cliffs, which would lead the formation of concave shapes, such overhangs or even caves. Peytavie *et al.* [PGMG09a] presented an approximation of 3D thermal erosion by using a hybrid implicit-surface and layerstack representation.

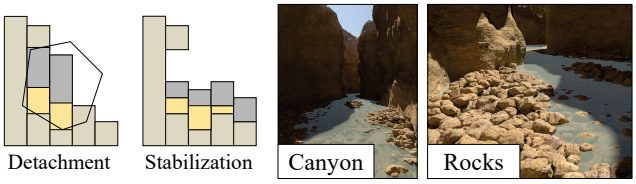


**Figure 19:** *Cliffs and overhangs created by 3D thermal erosion (from [PGMG09b]).*

Blocks detaching from vertical cliffs or steep slopes are computed using a Voronoi decomposition of space to define their boundary (Figure 19). The bedrock material is then converted into sand and rock material layers, which are stabilized into accretion piles. Piles of rocks and stones can also be generated from the granular material layers, as described in [PGMG09b].

While this approach allows for generating complex caves and cliffs with sharply sculpted overhangs, the volumetric model is limited to $\approx 1 \times 1 \times 1\,\mathrm{km}^3$ scenes.

#### 4.1.4. Analysis

Thermal erosion has been widely adopted in Computer Graphics both because of the simplicity of the governing equation, which allows for a straightforward implementation, even on graphics hardware, and because of its intuitive parametrization using the talus angle. It correctly creates accretion cones of fallen rocks, sand, and granular materials in both screes and accretion areas. Furthermore, it is an important complementary step to hydraulic erosion methods, which carve deep channels between high crests. In this context, thermal erosion gives a measure of the spacing between erosive features.

Thermal erosion has several limitations however. First, it us usually implemented as an iterating scheme that approximates forward Euler time stepping and this is physically accurate only for small time steps. Although this is of lesser importance when thermal erosion is applied as a post-processing step, this issue prevents its use when simulating large scale erosional features [CBC*16].

Second, fallen rocks do not only provide a fundamental visual improvement when represented at their rest state at the bottom of cliffs, they also have a non negligible erosive effect during their fall. Existing algorithms regard the fallen material as granular. However, larger blocks of material can fall, and subsequently break into rocks, which would end up making the simulation significantly more complicated and even harder to control. This effect can be approximated with more general approaches to *debris flow* erosion, as used in geology [SD03]. In addition to the breakage of rock, these model the erosive effect of falling rocks along their trajectory. Third, is the issue of perceived regularity: even when the talus angle is set with random perturbations, slopes look unnaturally similar. Geologists prefer *hillslope erosion* that averages a variety of lower scale processes into a diffusion equation:

$$\frac{\partial h}{\partial t}(\mathbf{p}) = -k\Delta h(\mathbf{p})$$

While this equation does not take into account the *material layers* introduced by thermal erosion, it constructs more visually irregular transitions between slopes, especially when combined with fluvial erosion [BS97].

### 4.2. Tectonic

A lot of attention in Geology and Geomorphology has been focussed on the simulation of crust deformation driven by the compression of tectonic plates [McC92]. This process results in the continuous vertical raising of mountain ranges – a phenomenon called uplift, which is countered by various forms of erosion that shape mountains [TH10]. In contrast, the simulation of tectonics for terrain modeling has received less attention in the Computer Graphics community. This can be explained by the complexity of geological phenomena described by differential equations that are computationally demanding. Moreover, it is difficult to set the parameters of the simulation so as to control the terrain generation process and follow the intent of the user.

#### 4.2.1. Tectonics and stream power erosion

Michel *et al.* [MEC15] procedurally generate the topography of a mountain range based from a user-defined sketch of peaks and rivers. This input is used to build a procedural approximation of tectonic plates whose speeds wrap a noise based fold map. Terrain elevation is obtained by combining the fold map with a watershed obtained by diffusing a slope value away from the user-sketched rivers. The relief is then eroded to generate the mountains by using a hydraulic erosion method.

Large scale terrain generation from tectonic uplift and fluvial erosion was addressed in [CBC*16]. This paper introduces the Stream Power erosion law derived from geomorphology [WT99] to computer graphics, which relates the erosion rate at a given point $\mathbf{p}$ to the drainage area $A(\mathbf{p})$ (which approximates the water flux by

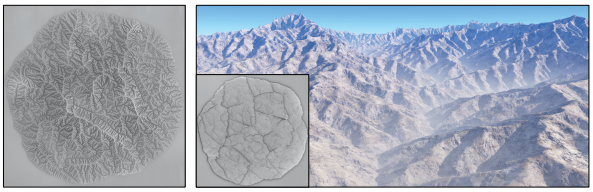

**Figure 20:** *Example of a large scale ($100 \times 100$ km) terrain generated by the method described in [CBC*16] combining a procedurally generated tectonic uplift map and fluvial erosion.*

integrating the precipitation rate over the upstream catchment domain), the local slope $s(\mathbf{p})$ and the tectonic uplift $u(\mathbf{p})$:

$$\frac{\partial h}{\partial t}(\mathbf{p}) = -kA(\mathbf{p})^m s(\mathbf{p})^n + u(\mathbf{p})$$

The mountain elevation results from the combination of a progressive uplift and the subsequent erosion (Figure 20). The uplift is used as an input in the form of a map of the rate of change of elevation, and the simulation is solved implicitly in linear time.

This work was extended in [CCB*18] with a geologically-coherent uplift derived from the relative movement of the tectonic plates prescribed by the user. The global mountain range uplift is computed by considering the crust as an incompressible viscous material. Folds are added procedurally and their wavelength is approximated by considering the crust as a stack of layered sheets of rocks with discontinuous physical properties. The terrain is finally eroded using an improved version of the algorithm presented in [CBC*16] that takes into account the characteristics of the different bedrock strata. The method generates a layered $2\frac{1}{2}$D model with folds and faults.

#### 4.2.2. Analysis

Tectonic-based simulations attempt to reproduce large scale erosion effects, taking into account the uplift of the bedrock balanced by different types of erosion described in geology. Similar to other mesh or grid-based techniques, those approaches are limited in scale range. Existing techniques produced realistic mountain ranges with dendritic river networks covering over $\approx 100$ km-wide domains, at a precision of $\approx 100$ m. Although the simulation can be controlled by interactively moving the tectonic plates, controlling the generation of the small-scale features remains difficult.

### 4.3. Hydraulic Erosion

Hydraulic erosion occurs when the motion of water against the bedrock surface produces mechanical detachment. Hydraulic erosion encompasses a number of mechanical erosional processes such as abrasion, corrasion (not to be confused with corrosion) and saltation. Chemical erosion (more often called chemical weathering) is distinct from mechanical erosion but is also part and parcel of hydraulic erosion. It changes the composition of rocks, transforming them when water interacts with minerals to create various chemical reactions.

The existing hydraulic erosion algorithms in Computer Graphics vary in the way they describe the fluid simulation (which in general also relates to the underlying models and data structures) and time and space scales.

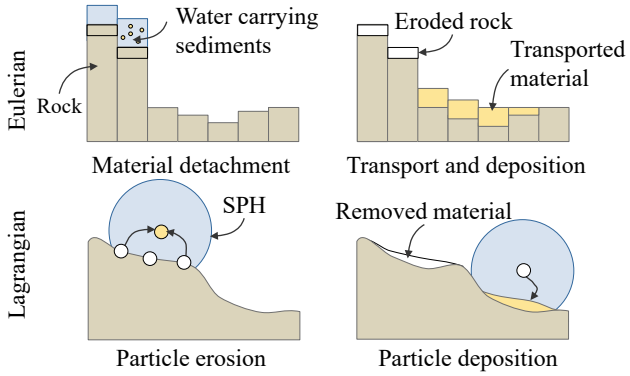


**Figure 21:** *Comparison of Eulerian and Lagrangian approaches to hydraulic erosion.*

Water movement acting to detach, transport and deposit material can be described by the Navier-Stokes equations [CF88] that have been studied in Computer Graphics for a long time [Bri08]. Fluid movement can be computed either by Eulerian approaches or Lagrangian methods (Figure 21). Eulerian approaches discretize space into a grid and simulate the amount of fluid in each cell, whereas Lagrangian approaches simulate the movement of the fluid particles themselves.

#### 4.3.1. Eulerian approaches

These methods rely on a discrete grid-based model that represents the input scene as well as the fluid. The simulation calculates the pressure, the velocity, and the amount of fluid in each discrete cell. Musgrave *et al.* [MKM89] presented a complete hydraulic simulation model by simulating material detachment, transport and deposition between heightfield cells. The erosive power of a given amount of water in a cell is a function of its volume and the amount of sediment already carried in the water. When the sediment capacity of fluid is reached, the material is deposited on the ground (Figure 22).

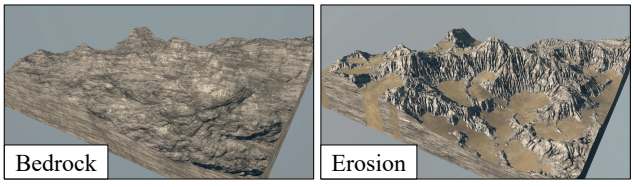


**Figure 22:** *Example of hydraulic erosion applied to a fractal procedural terrain.*

A geological representation for modeling the strata of the bedrock was introduced in [RPP93] that considers the characteristics of the different materials during the erosion process: the hydraulic erosion is the more intense as the rock is the softer.

These approaches were extended in [BF02] by introducing an algorithm for hydraulic erosion in which water dissolves soil, transfers it, and deposits at different locations by gravitational settling. This approach generates layers of smooth material deposited in floor bed and in pools of water after drying.

Nagashima *et al.* [Nag98] proposed a modified and improved physically-inspired erosion model and focused on the generation of valleys. The method combines a layer-based geological representation of the different strata of the bedrock with thermal erosion and a specific stream erosion process.

Neidhold *et al.* [NWD05] combined fluid simulation with erosion and analyzes the acceleration or deceleration of the fluid to erode the bedrock or deposit sediments. They also use the sediment capacity [MKM89] to deposit material in still water. The authors report fast simulation allowing this approach to be used at interactive speeds. A parallel implementation was later proposed in [ASA07].

Benes *et al.* [BTHB06] combined a 3D fluid simulation with sediment transportation and deposition to simulate hydraulic erosion. The underlying data structure for representing the terrain was a voxel grid, with data representing the variable amount of material effectively stored in every cell. The movement of the fluid is computed on a grid, and the force of the fluid applied to the material boundary defines the importance of erosion. When the force is larger than the material resistance, some material is eroded and transported by the moving fluid. When the fluid slows down, it deposits the material in the corresponding voxel cell.

Hydraulic erosion has been also used for generating rivers, and an approach inspired by self-organizing systems was proposed in [PM13]. Here, the simulation relies on self-organization and emergent phenomena to synthesize coastlines and terrains with the fractal features characteristic of hydraulic erosion.

Benes *et al.* [Ben07] combined shallow water simulation with hydraulic erosion. This approach allows the user to place water sources, control the flow of water, and edit the underlying terrain. The shallow water simulation, however, operates on a layered heightfield representation and does not allow for the simulation of volumetric erosion phenomena such as cliff overhangs produced by river erosion.

**Performance challenges** Fluid simulation itself is computationally expensive and becomes even more demanding when combined with erosion. Several approaches propose technical implementation improvements that speed up computations, in general by exploiting graphics hardware as most fluid simulation computations can be performed in parallel. Mei *et al.* [MDH07] demonstrated the effectiveness of parallel methods and achieved interactive feedback. Šťava *et al.* [ŠBBK08] further improved the method by using a multi-layered representation and increasing the discretization of the domain beyond $1024^2$ while preserving interactive feedback. Subsequently, Vanek *et al.* [VBHŠ11] addressed memory limitations by tiling the terrain and offloading the erosion into rectangular blocks that are swapped to the main memory of the computer.

A more general stochastic simulation of different events that impacts terrain at shorter geological timescales ($\approx$ 1 000 years) was introduced in [CGG*17]. The method combines the joint effects of erosion and vegetation by considering atomic events such as water drop (hydraulic erosion), rock fall, lightning strikes, and vegetation related events such as plant seeding, growth and death, as well as forest fires (Figure 23). An event is randomly chosen, and follows a simple cell by cell path (no loop, no interruptions, no branches) progressively transforming the multiple layers of materials that compose the terrain: bedrock, granular material (sand, humus, rock), vegetation densities and resources such as moisture or illumination.

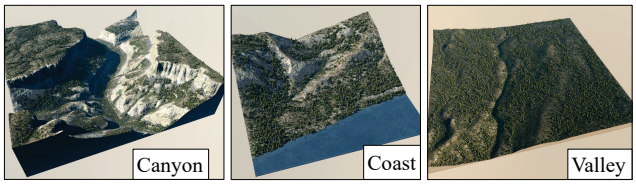


**Figure 23:** *Example of terrain erosion combined with ecosystem simulation (from [CGG*17]).*

### 4.3.2. Lagrangian approaches

Particle-based approaches for computational fluid dynamics model the flow as a collection of particles that move under the influence of hydrodynamic and gravitational forces. Eulerian methods obtain the solution relative to a fixed grid, whereas the Lagrangian framework define the flow in terms of the concentration of advected particles.

General terrain erosion methods distribute particles across the domain to approximate rainfall, and then simulate their movement and effect on the terrain by eroding bedrock into sediments and then lifting, transporting, and eventually depositing these sediments.

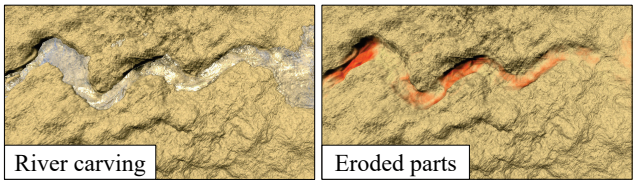


**Figure 24:** *Lagrangian erosion simulation [KBKŠ09]: the water particles form a river that erode the terrain and transport sediments that are deposited downstream.*

Water particles were introduced in [CMF98] to compute the erosive forces exerted by the flowing water down the terrain to simulate erosion. They introduced the concept of velocity fields that are calculated from motion of water and used it to erode the underlying terrains. Smoothed Particle Hydrodynamics were used in [KBKŠ09]. The fluid particles move and when they hit the terrain they erode it (Figure 24). Each moving fluid particle then transports material that changes its mass; when the speed of the particles is slow, it deposits the material.

**Specific landforms** The automatic generation of meandering rivers was addressed in [Kur12, Kur13]. It is an approximation in-

spired by erosion that uses particles that carry sediment and generate meandering riverbeds. The method aims at generating meanders and carves the riverbed in the ground. The resolution depends on the type of the river, and as for most simulations the system is limited to small parts of rivers.

#### 4.3.3. Analysis

Although erosion simulations are often considered the most realistic way of eroding terrains because the underlying physics capture emerging phenomena such as channels or accretion cones, several fundamental problems remain.

**Scale range** Many hydraulic erosion algorithms, some Eulerian methods simulating the dynamics of fluids and almost all Lagrangian methods, perform the simulations at a small time and space scale, and then implicitly scale up the results to approximate large scale terrains. This is intrinsically flawed. While generating visually convincing effects, these approaches are not realistic as the equations of the corresponding phenomena are not linear. The scale of the simulated features heavily depends on the underlying terrain representation. Such simulations can efficiently model scales of a certain size, but using the same approach to simulate much larger or much smaller phenomena is either impossible or computationally intractable.

Thus, depending on the simulated phenomena, simulation can become very time consuming, which makes it ill-suited to the detailed and precise modeling and generation of terrains with large extent.

**Control** A second problem is the control of these methods. While erosion simulations are more physically or geologically *accurate* than procedural approaches, they are notoriously difficult to control. The user is usually left with defining the initial conditions of the simulation and waiting for the generated results. If the result is not satisfactory, the initial conditions need to be changed. Moreover, they do not scale well and designing a large-scale simulation is a very difficult task.

It is common to use hydraulic and thermal erosion as a beautification processes to enhance procedurally generated terrain with sedimentary valleys and erosion landmarks, such as gorges and ravines. However, this neglects a crucial element: mountain ranges result from continuous uplift countered by various forms of erosion. The form of the initial input terrain thus plays a crucial role in the realism of the output. Ideally, the input prior to erosion should be geomeorphologically sound and encode the effects of uplift.

### 4.4. Other erosion phenomena

Compared to the large amount of work that dealt with hydraulic and thermal erosion, other phenomena have not received much attention by authors despite their potentially big impact on landforms generation. Phenomena that have seldom or never been addressed in computer graphics include aeolian, coastal, lightning, glacial, and karst erosion. Glacial erosion is a phenomenon that applies at smaller time scales than fluvial erosion and that induces the creation of U-shape valleys and hanging valleys. Coastal erosion and karst are phenomena that have a volumetric impact on the terrain and that are highly related to the subsurface geology.

**Aeolian erosion** is caused by winds that erode, transport, and deposit materials; it is more intense where vegetation is sparse since sediments are unconsolidated. Aeolian processes encompass the effects of wind that erode terrains by *deflation* (the removal of loose, fine-grained particles by the turbulent action of the wind) and by *abrasion* (the erosion of surfaces by the grinding and sandblasting action of sand particles).

While simulating the formation of *sand dunes* has received a lot of attention in physics [MMW01, NZRC09, Pel09], it has not been a focus in Computer Graphics. Onoue [ON00] borrowed the saltation equation and adapted simulations from physics into a procedural method to generate sand ripples. This approach was further extended in [BR04] by taking into account the collision of sand with obstacles. Those methods rely on discrete layered heightfields and therefore have a limited extent.

**Figure 25:** *Aeolian erosion structures obtained by spheroidal erosion forming Goblins (from [BFO*07, JFBB10, TJ10]).*

*Spheroidal erosion* on voxel grids was proposed in [BFO*07, JFBB10] for approximating the effects of wind erosion on rocks. The local curvature and accessibility of the surface indicates its exposure to the environment and causes this to be eroded faster. Positive curvature erodes edges and creases, whereas negative curvatures causes protrusions and holes. This approach simulates efficiently so called Goblin structures (Figure 25) commonly carved in sandstones such as in Goblin Valley State Park, Utah. An extended version of spheroidal erosion was adapted for eroding triangular meshes and generating Goblins but also concavities and overhangs [TJ10]. This method, however, does not allow the creation of arches or caves.

**Lightning** may have an erosive effect on exposed terrains parts [KG14], where a single strike can destroy a large amount of bedrock and project rocks within several meters. Cordonnier *et al.* [CGG*17] included the effects of lightning as an erosive agent: strikes are scattered randomly on the terrain, with a higher probability at higher locations with negative terrain curvatures. Bedrock is destroyed, and material is added in a rock layer in the grid cells near the impact, similarly to transport of the outcome of thermal erosion.

## 5. Example-based

Instead of starting from a blank or unformed state and gradually constructing a terrain through procedural or simulation methods, example-based approaches borrow from and combine existing terrains. The exemplar terrains can be sourced from anywhere

(even procedural and simulation outputs), but most often rely on scanned heightfield terrains in the form of Digital Elevation Models (DEMs), such as those provided by the U.S. Geological Survey [GOG*02], because these provide real-world fidelity.

A typical example-based terrain generation pipeline has the following form. First, a database of one or more terrain exemplars $\mathcal{E}$ is constructed. Then a synthesis step is used to select, warp and arrange terrain fragments $\mathcal{P}_k \subset \mathcal{E}$, with a corresponding elevation function $h_k(\mathbf{p})$ from the database. Finally, the disparate fragments are blended together into a seamless output terrain $\mathcal{T}$. Under this general framework there is, of course, considerable variation between individual methods. For instance, synthesis can involve square patches [ZSTR07, TGM12], individual pixels [GMM15], or circular kernels [AAC*17], while blending can range from relatively straightforward linear blending [AAC*17] to graph cuts with Shepard interpolation of the gradient field [TGM12].

There are some significant advantages to an example-based strategy. In a sense the entirety of its geological history is encoded in an input DEM leading to highly realistic outcomes. Furthermore, it is possible to leverage recent advances in texture synthesis and machine learning to provide interactive performance and effective user control. While such approaches can be fast, realistic and controllable, there are some specific limitations. The problem with data-driven approaches such as these is that they stand or fall on the quality of the input data. Any artefacts or errors resulting from capturing real-world elevations will likely appear (or possibly be magnified) in the output. Sampling resolution is also limited to that of the source scans. Publicly available sources are currently mostly in the range of a few meters per pixel. The SRTM program covers nearly the whole earth at 30m resolution, while the National Elevation Dataset of USGS covers almost all the United States of America at 10m. In rare cases, data can be found as fine as $0.5-1$m per pixel, especially in easily floodable or densely urbanised area (e.g., AHN program in the Netherlands), but is, unfortunately, not available for steeper regions. However, this situation is likely to improve over time.

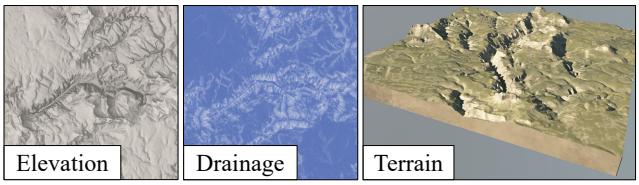


**Figure 26:** *A failure case for example-based synthesis. While locally plausible, the global drainage pattern of a real canyon are not respected as shown by the numerous disconnected white branching structures.*

Furthermore, features that do not exist in the source data cannot be reliably reproduced in the outputs. There are certain large-scale properties, such as globally consistent drainage patterns, that are often not respected in the output, even if it is locally plausible (Figure 26). Finally, these techniques are invariably limited to heightfields because of the structure of the input data.

### 5.1. Texture Synthesis

Texture synthesis takes a set of input images and derives an output with similar characteristics, usually by reassembling contiguous regions from the input in some fashion (Figure 27 for a terrain-specific example). This problem has been studied extensively in the context of colour images at both pixel and patch level [WLKT09]. If a heightfield terrain is treated as a grid of height values it shares some of the same characteristics as a greyscale image and thus texture synthesis can be applied with suitable modifications. The key differences to consider include: transferring absolute (rather than relative) height values from source to destination can restrict the variety of outputs, the nature of user control is fundamentally different since terrains are more geometric in structure, and maintaining the coherence of ridge, valley and erosion lines is crucial.

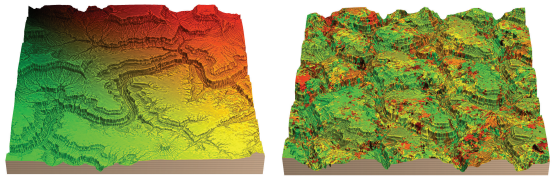

**Figure 27:** *Texture-based terrain synthesis can be viewed metaphorically as cutting and restitching contiguous regions from an input exemplar (left) into an output (right), even if the technique is pixel-based, as in this case (from [GMM15]).*

In general, developments in texture-based terrain synthesis build on and extend earlier work in image-based synthesis. For instance, Dachsbacher *et al.* [DMS05] provide an early proof-of-concept application of texture synthesis to the task of filling holes in DEM models and joining disparate terrains together by synthesizing pixels in empty regions. An early pixel-based non-parametric sampling technique [EL99] is employed. This attempts to match neighbourhoods around a single pixel in the destination to similar neighbourhoods in the source. The matching algorithm is modified from the usual colour-space metric to instead take terrain characteristics into account, in this case the absolute elevation $h(\mathbf{p})$ and the slope $s(p) = \|\nabla h\|$. The synthesized terrain is then blurred to remove high-frequency artefacts – an approach that is not advisable in general because it also removes salient detail. This technique is rather slow and does not produce particularly convincing results, but it represented a promising foray into the area.

Zhou *et al.* [ZSTR07] addressed user-control, by incorporating a 2D sketch map as an additional constraint to guide placement of ridges and valleys. They use a patch-based approach, in which exemplar DEMs are split into patches and reassembled in the output by overlapping patches and cutting them along seams. Standard patch-based texture synthesis is modified in several ways: a feature map of the curvilinear elements of the terrain is extracted (mountain ridges and valley lines) and this allows a user to guide the synthesis process using their top-down sketch (Figure 28); also, patch placement is reordered to follow and grow outwards from features. Patches are selected for placement based on how well they fit the user's sketch map and the already placed patches. The patches themselves can be warped horizontally for a better fit. Overlapping

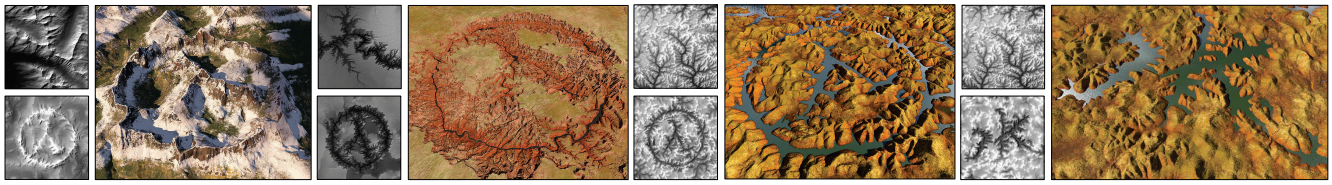

**Figure 28:** *Texture-based terrain synthesis [ZSTR07] allows the creation of different terrains using the same input sketch map (a lambda letter) and different exemplar terrains (insets).*

patches are then cut along a seam using a graph cut algorithm and the areas on either side are adjusted using a Poisson equation to hide the seam. Some tuning is needed in selection of patch sizes since only a limited range of sizes work well in practice and this depends on the sampling resolution of the terrain. Although the results are convincing, the approach is not interactive (it takes several minutes to generate a terrain) and the sketched features are planar, with users unable to control the elevation along sketched curves.

Some of these issues are addressed by Tasse *et al.* [TGM12]. First, while patch placement and merging are order dependent in the work of Zhou *et al.*, it is possible to process patch selection in parallel using graphics hardware. This leads to a sixfold speedup in patch matching, although the overall algorithm still remains non-interactive. Second, seam merging by Poisson smoothing can be replaced by a Shepard gradient interpolation process. This considers both position and gradients along seams and hence produces less noticeable artefacts, as confirmed by a user experiment. Finally, better control is offered by constraining elevation to match sketched curves using post-synthesis deformation. However, if the deformations are significant this can impact realism because the geomorphology is not respected.

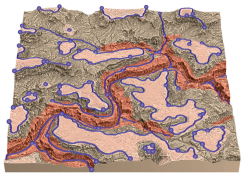

**Figure 29:** *Texture-based terrain synthesis: user controls employed in the creation of a canyon terrain include point constraints, curve constraints and painted terrain types (from [GMM15]).*

Gain *et al.* [GMM15] further address issues of efficiency and control by switching from patch-based to parallel pixel-based terrain synthesis. They adapt the stacked multiresolution pixel synthesis scheme of Lefebvre and Hoppe [LH05], which is well suited to implementation on graphics hardware. This enables a real-time synthesis process, with $5-30$ syntheses per second depending on terrain size. This technique is likely the fastest example-based method, which is important because it allows rapid cycles of fine-tuning in the design process. Indeed, a major focus of this work is on incorporating various user controls (some of which appear in Figure 29), including: localised point and curve constraints, with additional controls over slope and area of influence; paintbrush tools for specifying terrain character, and region-based cut and paste operations.

**Analysis** It is possible to achieve a trifecta of realism, control and real-time performance with these methods. However, as with all example-based methods, the output quality is heavily reliant on the input data. In particular, if a specific terrain feature, such as a Mesa, is desired and it is not found anywhere in the input data it cannot be reliably reproduced. It is possible to reposition, reorient and alter a landform in various ways but only if it appears as a source somewhere in the input data. The other issue is that large-scale geomorphological constraints are not always respected. Drainage basins are a case in point: it is quite possible to have several local minima appear on the terrain without outflows. Fixing such flaws is left in the hands of the user, whereas it would be better to have such geomorphological issues corrected automatically. Fortunately, there is nothing inherent to these techniques that prevents such additional constraints from being incorporated in future.

### 5.2. Altering Existing Terrains

Sometimes a particular source terrain is a reasonable match to the users desired end goal, barring some particular adjustments. There is thus a place for techniques that deform or otherwise alter a terrain provided as input by the user.

Dos Passos and Igarashi [dPI13] restructure a terrain based on a first-person silhouette sketching interface, where mountain ridges are sketched from the perspective of a person standing at a particular location, much as an artist would sketch the outlines of a real scene. Given a source heightfield and user sketch, their system proceeds as follows: sample viewpoints are placed radially around the source and strong silhouettes are extracted automatically. Recursively, ever shorter segments of this collection of silhouette contours are matched against the user sketch until a matching quality threshold is met. Once a match is obtained a wedge shaped section is cut from the source terrain and placed into the destination. These wedges are then blended using a simple sum, weighted according to distance from the center line of each wedge. When there is some choice among possible matching contours, the user is allowed to select among them using a suggestive interface. There are some definite weaknesses to this approach: it only considers strong silhouettes, which limits the types of landforms that can be sketched; artifacts appear due to the blending process, which means that it is only visually plausible near the sketched viewpoint; and the choice of source terrain is quite constrained.

Tasse *et al.* [TEC*14] start from a similar initial premise in that

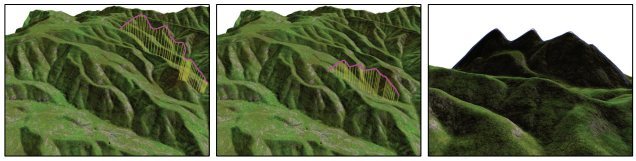

**Figure 30:** *Altering existing terrains: a terrain edited from a first person viewpoint using silhouette sketching (from [TEC*14]).*

they match first-person silhouettes using existing sample data (Figure 30), but in their system silhouettes can be in a much more complex arrangement, with internal silhouettes and t-junctions representing occlusions. The method is fully automated and deforms the source terrain rather than applying cuts and joins, which leads to more plausible results. First, crest lines in the source are automatically extracted as possible candidates for matching. Then a branch-and-bound algorithm pairs silhouette fragments with crest lines based on deformation cost while respecting relative depth. Crest lines are then deformed vertically to match the silhouettes using iterative diffusion. The original terrain may have features that block a user's silhouette and these are lowered by deformation. Although more natural than Dos Passos and Igarashi's method, the geomorphological characteristics of the terrain can still be broken, particularly if the required deformation of crest line onto silhouette is significant. In short, this works best for smaller modifications of the existing terrain silhouette.

Ketabchi *et al.* [KRS15] and later Samavati *et al.* [SR16] support sketch-based modification of a low-resolution DEM and co-located orthophoto to adjust terrain elevation, locate bodies of water, and demarcate regions for placing 3D vegetation models. This compensates for artefacts in the source DEM, such as uneven lake shores and road surfaces. The interface uses cross-sectioned curves: a base curve sketched onto the surface, followed by elevation offsets to this curve, and finally orthogonal curves placed at intervals to define the slope. The initial DEM is then deformed to fit the positional constraint of the curve, with an energy optimization term that minimizes the curvature of the surrounding deformation. The cross-section curves also provide constraints but are accorded less weight. Finally, since the DEM may have insufficient resolution, it is locally subdivided as required. This technique considers not only bare-earth terrain, but landscapes more generally, by allowing users to sketch closed loops to identify water and plant regions. The orthophoto is used as a final texture for the terrain but also acts as a guide during the sketching process. The primary focus here is on correcting errors in DEMs and it should not be considered as a general terrain modeling technique.

Emilien *et al.* [EPCV15] provide an interactive editor focused on laying out rivers and waterfalls with attendant sculpting of the underlying base terrain (Figure 31). The user employs a set of vector sketching tools to specify segments of a river network graph, both those in contact with the terrain and falling free. Given this network, consistent water flow information is computed, and the terrain is automatically deformed to adjust slope and create river beds, banks and basins that are consistent with a user's intent. Additional mechanisms are provided to enhance the realism of results, such as optionally improving river trajectories with more meanders in flat regions, or modeling overhangs under waterfalls.

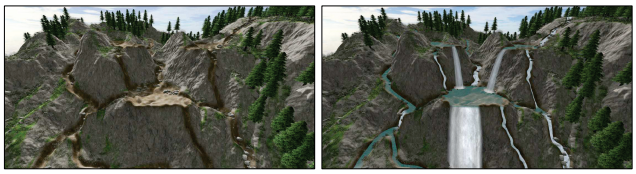

**Figure 31:** *Altering an existing terrain to carve out rivers and waterfalls (from [EPCV15]).*

**Analysis** The need to alter an existing terrain to remove artefacts or better fit an envisaged layout is not uncommon, but care must be taken. Unlike the texture synthesis, machine learning or sparse modeling approaches of other example-based techniques, large scale deformations do not respect underlying geomorphology and can break terrain realism. This is why attempts to minimize distortion, such as used in Tasse *et al.* [TEC*14], are so necessary.

### 5.3. Machine Learning

One promising and under-explored avenue is the application of machine learning to terrain synthesis. Guérin *et al.* [GDG*17] take this tack by training a conditional generative adversarial network (CGAN) on a set of annotated DEM exemplars. In this context, a CGAN trains two competing neural networks: a generator, which creates new example terrains, and a discriminator, which differentiates between real and synthetic terrains. Through competition both the generator and discriminator improve simultaneously. DEM samples are marked up in a variety of ways: for instance, by detecting crest and valley lines and points of interest such as mountain peaks. A CGAN then learns the relationship between the annotations and the source DEMs, so that later a user can provide the annotations as sketches and have a corresponding terrain generated. The method does rely on the users providing a detailed sketch (sparse regions can sometimes lead to repetition artefacts) but it is realistic and interactive. Figure 32 shows an example of the generated terrain from a sketch consisting of a few crest strokes.

**Analysis** The true virtue of a machine learning approach lies in its versatility. It can be used to fill in gaps from missing data, or to increase resolution using a kind of erosion filter (learnt from erosion simulations applied to DEM samples). Furthermore, it is possible to quickly and easily create different control mechanisms by supplying different forms of annotation during training. This is in contrast to texture synthesis methods, where new interaction mechanisms require a deep understanding of the underlying method. Given such versatility, this area is ripe for further research.

### 5.4. Sparse Modeling

Sparse modeling borrows from compressed sensing theory to encode signals compactly in a lower dimensional space. This has a variety of implications for terrain synthesis as explored by Guérin *et al.* [GDGP16], who represent terrains using a sparse construction

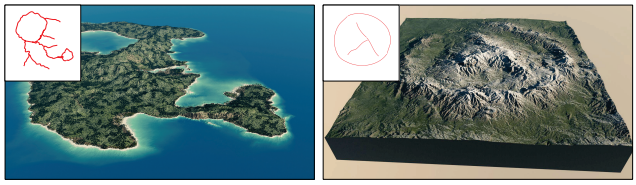

**Figure 32:** *Machine learning: An island generated from a simple sketch and the application of a CGAN network (from [GDG*17]).*

tree. The leaves of such a tree consist of atoms, which are circular terrain patches extracted from real-world source exemplars, and the internal nodes are combiners that fuse overlapping atoms. Atoms are arranged on a regular grid so that their domains overlap without leaving any of the terrain uncovered and they are combined using a weighted sum that depends on the distance from an atom center. The sparsity aspect is expressed by building a dictionary of atoms from a set of source exemplars in such a way that the reconstruction error is bounded while using the least number of atoms.

Compression is one possible application. The sparse construction tree is relatively compact when compared to a heightfield terrain, since it consists of a collection of atoms encoded as indexes into the atom dictionary with implicit locations on a grid and a few additional parameters. Of course, reconstruction cannot be achieved without storing the dictionary of atoms, but this too can be kept compact by replacing real-world patches with procedural equivalents, although quality tends to suffer. The most important application, however, is terrain amplification. This involves beautifying a previously defined coarse terrain by augmenting its resolution and adding consistent details. This is achieved by pairing low-resolution atoms with higher-resolution versions that replace them, increasing resolution by a factor of 4 to 8 with each replacement step (Figure 33).

This augmentation of database atoms is extended by Argudo *et al.* [AAC*17] to fuse multiple layers, such as base terrain, moisture, soil type and vegetation density. Not only does this allow enriched landscapes, featuring ecosystems and bodies of water, but it also improves the matching to preserve global coherence, such as drainage patterns.

**Analysis** While the focus of sparse modeling is on compression and various forms of amplification, it also supports the design of novel terrains. However, this has been somewhat de-emphasized in the literature and the interaction mechanisms lag behind other example-based methods.

## 6. Discussion

There are several criteria on which the effectiveness of a given terrain synthesis method can be judged. We base our in-depth discussion around the following: *variety* (how wide is the range of achievable landforms?), *scale* (what is the size of the area that can be modeled and in what detail?), *realism* (how plausible is a synthetic terrain when compared against the real world?), *authoring* (what mechanisms are provided for user control of terrain shape?), and *efficiency* (what are the computation and memory costs?). Tables 2 and 3 provide a comparison of terrain methods according to these different criteria.

### 6.1. Variety of landforms

There exists a wide range of different landforms on Earth, including hills, mountains, plateaus, canyons, and valleys. They may either be created by a dominant natural event or phenomenon, or result from the intricate interaction of many different factors over large periods of time. Landforms can be categorized according to their characteristics such as elevation, slope, orientation, stratification, bedrock exposure, and soil type. A complete description of the geomorphological landforms and the underlying physical processes can be found in specialized books [AA10, Har12]. In this section, we present a list of landforms that are predominant in natural landscapes and need to be addressed by Computer Graphics approaches.

- *Fluvial* and *hydraulic erosion* cause dendritic valleys, fractal distribution of rivers, footslope deposition and hydrological features: such as waterfalls, meanders, and oxbows.
- *Thermal erosion* decreases the slope angles down to a critical one. This terminology is not well admitted in geomorphology where landslides and debris flow erosion are preferred.
- *Hill slope erosion* averages many erosional effects in a diffusion equation which smooth mountain edges and round valley flanks.
- *Glacial erosion* gives rise to U-shaped valleys, hanging valleys, arêtes, pyramidal peaks (or horns), corries (or cirques, armchair shaped depression where glacier starts), lakes, moraines, drumlins (hills oriented in the direction of the glacier).
- *Karsts* result from underground erosion and form networks of underground streams and caves, canyons and chasms. If the sedimentary sedimentary strata is exhumed by erosion, they becomes visible.
- *Coastal erosion* or shoreline retreat is (a sudden) displacement of large blocks of soil caused by waves, tide, or ice. It may be slower on rocky coasts.
- *Wind erosion* produces dunes and volumetric shapes such as arches or Goblins.

Generating realistic landforms is a difficult task. The challenge is twofold: first creating an algorithm that could reproduce the fundamental geomorphological characteristics and, at the same time, providing a set of parameters that would allow for (interactive) control for authoring. Moreover, terrains features have different sizes and may vary from very large extent (hundreds of kilometer for rivers or mountain ranges) to small extent (tens of meters for some specific erosion features such as Goblins, or waterfalls). Maintaining coherent detail over such a wide range of scales is difficult.

Physically-based erosion simulations are in general ill-suited for simultaneously generating a variety of coherent landforms, mostly because of the complexity and the intricate interactions of many different natural processes. In contrast, procedural approaches or exemplar-based methods can generate atlases of landforms defined over a limited support, that can be combined together to synthesize larger terrains as described in [GGP*15]. Still, combining primitives to produce large-scale terrains remains an open research area.


**Figure 33:** *A sparse construction tree model compactly represents large scale terrains at fine resolution. Sparse modeling is employed to amplify a coarse terrain from a resolution of* $\varepsilon = 1\text{km}$ *to* $\varepsilon = 4\text{m}$ *per pixel, for a total amplification factor of* 256*. Note that procedural texturing and vegetation are applied to outline ground details such as ridges and erosion patterns generated by the amplification process.*

Several recent algorithms attempt to *amplify* large scale terrains with fine-scale features [GDGP16, AAC*17]. This is a promising avenue, but it does rely on a reasonable intial coarse terrain to be effective.

### 6.2. Realism

The question of whether a particular synthesized terrain is realistic can be surprisingly hard to answer. One way to assess realism is according to viewer perception, while another way is to use geomorphological assessment. These perceptual and quantitative approaches do not always agree: certain real-world landforms, such as Goblins, can appear unrealistic to a viewer who has never encountered their like. Moreover, it is also commonly agreed that if an algorithm simulates correctly physics or geology, the results should be in agreement with reality. This can, however, be problematic, because only a certain class of phenomena is usually addressed.

Given these difficulties, authors have chosen to measure realism in several ways, all of which are problematic to some extent.

- Static renderings: it is common to simply provide rendered images of terrains. In some cases, a photographic image of a physical scene may also be provided for side-by-side comparison. However, such an approach is influenced by the choice of viewpoint, rendering style, texturing, presence of landmark features ( *e.g.* trees, roads and buildings), and this makes a comparison of realism across methods difficult.
- User studies: a less frequent form of evaluation is to conduct a user study, where participants might typically select between real and synthetics terrains that are rendered side-by-side. Given a sufficient number of participants the favoured terrain method can be determined statistically. There are two issues here: first, such users are not necessarily used to assessing bare-earth landscapes from a raised vantage point. Second, they may also have had their expectations biased, for example to video games.
- Expert assessment: consulting experts in geomorphology has distinct advantages. They are often able to go beyond identifying a generalized lack of realism and diagnose specific flaws relating to subsurface geology and erosive phenomena.
- Validation of physical consistency: finally, it is also possible, albeit challenging, to build automated tests of specific geomorphological attributes. In practice, this is usually limited to a test for consistent drainage by simulating water flow and examining the resulting river network.

For the purposes of our evaluation we show, in Figure 34, images of terrains produced by a wide variety of synthesis methods and rendered from a consistent viewpoint and in a consistent style that highlights detail. From this it can be seen that erosion- and example-based methods appear significantly more realistic than procedural approaches.

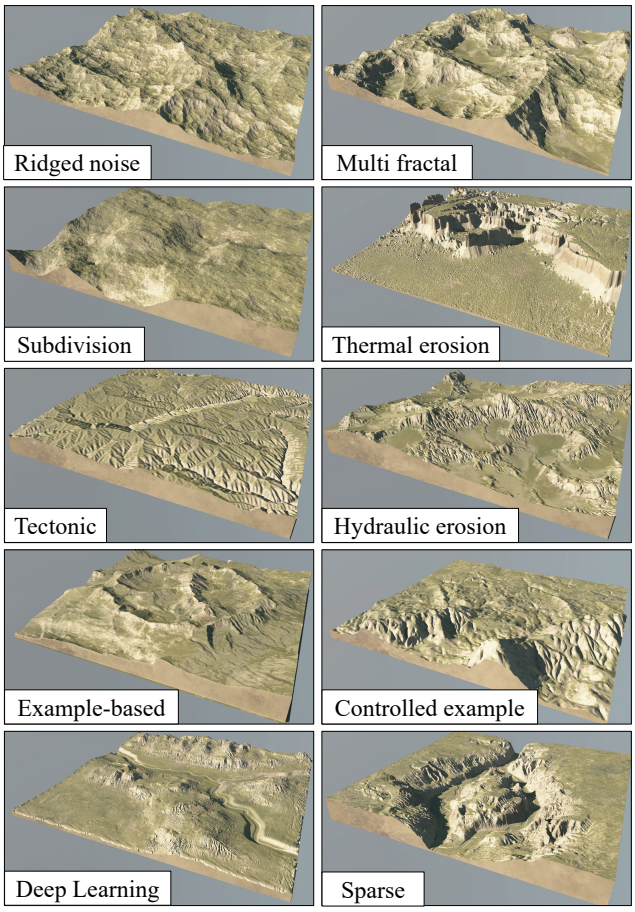


**Figure 34:** *An outlook of the state-of-the-art methods for generating digital terrains.*

In general, there are two major realism weaknesses in current methods: a poor depiction of landscapes with a combination of dis-

tinctive large scale and fine scale features (such as large riverbeds with surrounding fine-scale erosion), and difficulties in combining a variety of landscape features influenced by different formation processes within the same terrain ( *e.g.* rolling hills, plains, canyons, eroded mountains). This is because procedural and erosion techniques tend to be specialized to a particular effect. A canyon landscape with surrounding plains serves as a kind of stress test in this regard, which is why it often does not appear in papers.

### 6.3. Range of Scales

When it comes to evaluating terrain generation, the issue of scale is frequently glossed over. Nevertheless, questions of extent and precision – how large a terrain to generate and in what detail – are critical, because different applications have different requirements. A first-person computer game will require a much smaller land area but with greater detail than a flight simulator.

As a precursor to discussion, it is helpful to re-introduce some terminology, first defined in Section 2.1. While this is mostly presented in terms of discrete heightfields it can be generalized to other representations and dimensions. We define extent $a$ as the length of one side of the terrain in $m$ or $km$, resolution $n$ as the number of grid samples in that dimension, and precision $a/n$ as the distance between samples.

There are two major factors that affect the interplay of extent and precision. First, given that computation cost grows as $O(n^2)$, efficiency concerns often places a limit on resolution. All other things being equal, it is possible to trade off precision for extent, but a slow method is unlikely to be able to support both large extent and fine precision. Second, certain terrain synthesis techniques are scale bound. In theory, procedural methods, such as noise, subdivision, and faulting, can support infinite precision, but, in practice, they are only realistic across a specific range of medium scales. Likewise, simulation usually requires a precision that properly matches the frequency of the erosive features being captured.

Example-based methods, on the other hand, can accommodate a range of extents with a fixed resolution, simply by adapting the sampling precision of the exemplars. This does mean, of course, that exemplars of such precision must be available. In practice, limits on current scanning technology mean that accessible datasets offer a best precision of $1-2\,\mathrm{m}$. However, this does differentiate these methods from procedural and erosion methods, which often work best at specific scales.

The simultaneous need for large extent and high precision has yet to be adequately addressed. Many mountainous features, such as those caused by glacial erosion, are $50-100$ km in extent. At the same time localized hydraulic erosion can require a precision in the range of $1-2\,\mathrm{m}$. Very few current techniques can accommodate this range of high and low frequencies and the required resolution ($50,000-100,000$ samples per dimension).

### 6.4. Authoring

Another axis along which terrains generation methods should be analyzed is the degree and nature of control they afford the user. While the first procedural and simulation methods only offered indirect control through parameter tuning – requiring much trial and error before achieving the desired result, a lot of effort has recently been dedicated to user control, leading to a new generation of authoring tools for terrains. The key overarching problem, and an open research question in terrain modeling is: how can the user's intent be achieved quickly, intuitively, and without an overwhelming emphasis on domain-specific parameters? While geologically- or physically-inspired methods allow for simulation of existing phenomena, they fall short in terms of usability and often require domain-specific knowledge. In contrast, example-based techniques allow for rapid and intuitive transfer of terrain features, but may not always produce results that are geologically correct and can be limited by the source examplars.

To summarize these advances, we classify terrain authoring methods according to their underlying editing metaphor, namely: painting with brushes, sketching vector features, or real-time sculpting through local or global deformations. A few methods consider terrains as time-evolving phenomena, and these enable temporal control through a scenario expressed on a time-line. These interaction modes and their sub-categories are described in Table 1, enabling us to summarize the tools associated with each paper in the authoring column of Tables 2, 3, and 4. This enables us to cite only the most relevant papers below.

**Painting with brushes** Most authoring techniques aim at providing the designer with high level tools for modeling terrains, and removing the time consuming processes of editing every single detail by hand. Using a painting metaphor (eg., using brushes to highlight a region on a map, or painting simulation parameters directly onto the terrain) is a natural way to provide such coarse-scale interaction. Then the detailed terrain is generated either by procedural [dCB09], simulation [CGG*17] or example-based methods [ZSTR07, GDGP16].

**Sketching vector features** Providing a vector sketching interface is a second way to enable large-scale control followed by automatic synthesis of details. This differs from painting in that sketching allows for precise specification of vector features, from ridges to river beds. A detailed terrain matching these features can then be generated. Sketching vector features can be done either on a topdown 2D map or over a 3D terrain.

A first group of methods used feature curves for terrain ridges and valleys, sometimes resulting into overly smooth terrains were the surface was derived from these curves alone [HGA*10b]. Subsequent extensions allowed the specification of terrain silhouettes from a first person viewpoint, which is often more familiar and natural for a user with an artistic background. Surrounding details are then derived either through example-based methods [RME09] or deformation of an existing terrain model [dPI13, TEC*14]. The last category of methods based on sketching vector features are those that enable users to sketch detailed riverbeds, lakebeds or networks of rivers [GGG*13] and waterfalls [EPCV15], and generate consistent terrains matching these features. Using deep learning, both rivers and ridges can be encompassed [GDG*17].

**Interactive sculpting** Clay sculpting can also serve as inspiration for interactive terrain editing. These enable users to apply local

| Type | Description |
|---|---|
| Parameter tuning | Indirect control through parameter specification, usually in numeric form. |
| Painting | Use of brushes to outline regions and/or edit local parameters on maps or 3D terrains. |
| Vector sketching: | Vector drawing to specify feature points or curves on maps or 3D terrains: |
| Landforms | Vector landforms ( *e.g.* ridges, rivers, summits) on a 2D map. |
| Elevation | Sketching features with height values and additional controls ( *e.g.* influence region, bi-lateral gradients). |
| Silhouette | Feature curves drawn from a specific viewpoint to represent the outline of mountains. |
| Sculpting | Interactive, local or global deformation tools, enabling editing of the terrain as if it was clay. |
| Time-line control | Space-time control thanks to a scenario of events expressed on a user-controlled time-line. |

**Table 1:** *Different types of user control: a given authoring technique may support one or several of these controls.*

| Paper | Landforms $\mathcal{M}$ $\mathcal{R}$ $\mathcal{A}$ | Method | Model $\mathcal{G}$ $\mathcal{F}$ $\mathcal{V}$ | Control | Extent | Precision | Performance |
|---|---|---|---|---|---|---|---|
| [DMS05] | ● ○ ○ | Texture | ● ○ ○ | Parameters | $4-4000\,km$ | $1-1000\,m$ | Minutes |
| [ZSTR07] | ● ● ○ | | ● ○ ○ | Landforms | | | Minutes |
| [TGM12] | ● ● ○ | | ● ○ ○ | Elevation | | | Minutes |
| [GMM15] | ● ● ○ | | ● ○ ○ | Painting, Elevation | | | Real-time |
| [dPI13] | ● ○ ○ | Altering | ● ○ ○ | Silhouette | $1-100\,km$ | $1-100\,m$ | Minutes |
| [TEC*14] | ● ○ ○ | | ● ○ ○ | Silhouette | | | Interactive |
| [EPCV15] | ○ ● ○ | | ● ○ ○ | Elevation | | | Interactive |
| [KRS15] | ● ● ○ | | ● ○ ○ | Silhouette | | | Interactive |
| [GDG*17] | ● ● ○ | Learning | ● ○ ○ | Painting, Elevation | $1-1000\,km$ | $1-1000\,m$ | interactive |
| [GDGP16] | ● ● ○ | Sparse | ○ ● ○ | Landforms | $4-4000\,km$ | $1-1000\,m$ | Seconds |
| [AAC*17] | ● ● ○ | | ○ ● ○ | Landforms | | | |

**Table 2:** *Overview of example-based terrain synthesis methods. Under landforms: $\mathcal{M}$ = mountains, $\mathcal{R}$ = river networks, $\mathcal{A}$ = volumetric. Under model: $\mathcal{G}$ = grid , $\mathcal{F}$ = functions, $\mathcal{V}$ = Mesh. Control corresponds to the classification in Table 1.*

or global deformations to the terrain. For instance, sculpting can be done locally by interactively applying erosion tools [MDH07, ŠBBK08] or example-based manipulators [GMM15]. It also enables editing at the scale of entire mountain ranges, by pushing tectonic plates against each other [CCB*18] the earth crust is sculpted as if it was clay, with the guarantee that consistent mountain ranges (based on uplift and erosion) will be created. In both cases, this requires an underlying real-time simulation.

**Space-time editing along a time-line** Unlike most modeled shapes, terrains are not static but instead correspond to time-evolving phenomena. Therefore, in addition to standard shape-modeling metaphors, terrain can also be edited using space-time interaction tools, such as by specifying a particular event on a scenario time-line. This can be done for combining terrains with specific medium scale erosion events ( *e.g.* interaction with temperature, storms or vegetation) [CGG*17], and would be mandatory to capture phenomena such as glacial erosion, where terrains are shaped by the succession of glaciation and inter-glaciation periods.

In conclusion, interactive authoring techniques aim at providing the designer with high-level control of a modeled terrain, while allowing the system to add detail and, whenever possible, enforce consistency of the results. Most authoring techniques are inspired by expressive modeling metaphors such as painting, sketching or sculpting. Since a terrain is an evolving phenomena, a new trend is to define scenarios along a time line, enabling not only a final terrain but also its evolution over time. Whatever the authoring choices, providing the user with real time or interactive visual feedback is a challenge. Most of the authoring techniques are limited in some way by the speed of the associated terrain generation method. Some of them overcome this by providing a first quick rough approximation while refinement occurs subsequently in a separate thread. A few methods achieve real-time thanks to implementation on graphics hardware, or through machine learning (at the cost of longer preprocessing times).

### 6.5. Efficiency

It is worthwhile examining performance in terms of time and space efficiency, because they both impact the usefulness of terrain generation systems in practice.

**Time efficiency:** The computation cost of terrain synthesis directly effects how well a user designs terrains. Unfortunately, it is

| Paper | Target $\mathcal{M}\ \mathcal{R}\ \mathcal{A}$ | Method | Model $\mathcal{G}\ \mathcal{F}\ \mathcal{V}$ | Control | Extent | Precision | Performance |
|---|---|---|---|---|---|---|---|
| [MEC15] | ● ● ○ | Eulerian | ● ○ ○ | Landforms | 10 – 100 km | 10 – 100 m | Minutes |
| [CBC*16] | ● ● ○ | | ● ○ ○ | Painting | | | Seconds |
| [CCB*18] | ● ● ○ | | ● ○ ○ | Sculpting | | | Interactive |
| [MKM89, RPP93, BF02, PM13] | ● ● ○ | Eulerian | ● ○ ○ | Parameters | 1 – 10 km | 1 – 10 m | Seconds |
| [Nag98] | ● ● ○ | | ● ○ ○ | Parameters | | | Seconds |
| [NWD05, MDH07] | ● ● ○ | | ● ○ ○ | Sculpting | | | Interactive |
| [VBHŠ11] | ● ● ○ | | ● ○ ○ | Sculpting | | | Interactive |
| [CGG*17] | ● ● ○ | | ● ○ ○ | Painting, Time-line | | | Minutes |
| [BTHB06] | ● ● ○ | | ○ ○ ● | Parameters | | | Hours |
| [CMF98] | ● ● ○ | Lagrangian | ● ○ ○ | None | 1 – 10 km | 1 – 10 m | Seconds |
| [ŠBBK08, KBKŠ09] | ● ● ○ | | ● ○ ○ | Parameters, Sculpting | | | Interactive |
| [Kur12, Kur13] | ○ ● ○ | | ● ○ ○ | Parameters | | | Seconds |
| [BFO*07, JFBB10] | ○ ○ ● | Eulerian | ○ ○ ● | Sculpting | 100 m | 1 m | Minutes |
| [TJ10] | ○ ● ● | | ● ○ ○ | Silhouette | | | Seconds |

**Table 3:** *Overview of terrain simulation methods. Under landforms: $\mathcal{M}$ = mountains, $\mathcal{R}$ = river networks, $\mathcal{A}$ = volumetric. Under model: $\mathcal{G}$ = grid , $\mathcal{F}$ = functions, $\mathcal{V}$ = Mesh. Control corresponds to the classification in Table 1.*

difficult to accurately compare run-time performance short of profiling a fully-optimized implementation of every technique on the same hardware platform. Complexity analysis is not particularly helpful because most techniques are $O(n)$, where $n$ is the number of heightfield grid elements or points $\mathbf{p}_{ij}$. Reported run-times are also problematic, given changes in hardware performance over time and different effort applied to optimization.

Many erosion simulation techniques do not scale well with increased terrain precision, because they rely on successive iterations of erosion (in general hundreds), whose complexity is, at best, $O(n)$, where $n$ denotes the number of grid cells. A high resolution terrain, such as $16 \times 16\,\text{km}^2$ sampled at a $2^{15} \times 2^{15}$ resolution (with precision of $\approx 50$cm), has more than 1 billion grid cells. Combined thermal and hydraulic erosion typically takes several hours to complete on such large maps.

Fortunately, it is usually possible to place techniques into one of four general performance classes (see Tables 2, 3, and 4): real time ($\leq 1/3$s per update or 3Hz), interactive ($\leq 3$s per update), seconds ($> 3$s but less than a minute per update) and minutes (anything else). The distinction between real time and interactive performance is important because it impacts the type of interface that can be developed and the consequent user experience. For instance, a sculpting metaphor relies more strongly on direct manipulation and generally requires real-time response, whereas in vector sketching changes can be submitted periodically and it can therefore accommodate interactive updates (or sometimes seconds).

In broad terms the method categories can be ranked from highest to lowest performance as: procedural, example-based, then simulation. There is, of course, substantial variation within each category. The most efficient general technique in each category is as follows: procedural noise (real-time at $> 20$Hz), Stava *et al.*'s hydraulic erosion simulation [ŠBBK08] (interactive at $1 - 2$Hz), and Gain *et al.*'s texture synthesis [GMM15] (real-time at $5 - 20$ Hz). Unsurprisingly, most of these are amenable to implementation on graphics hardware.

It is evident from Tables 2 – 3 that there are not many techniques, outside of procedural methods and some example-based approaches, that are real time and this is an area that deserves more attention, since cycles of immediate feedback are very important for rapid and effective design.

**Space efficiency:** It is worth examining memory efficiency at two junctures: during the terrain generation process and in subsequent use of pre-generated terrain. During generation, the additional memory overheads incurred are generally low for procedural and simulation methods since these are algorithm driven. However, this is not the case for example-based methods, which are data driven and usually rely on a database of terrain exemplars. This is something to be aware of in resource constrained circumstances, such as with graphics hardware.

In subsequent use and at lower terrain resolutions ($512^2 - 1024^2$), memory requirements are in the range of $1 - 4$Mb and rarely a cause for concern. Furthermore, a regular heightfield grid usually exhibits strong local coherence and is therefore amenable to standard compression using schemes such as run-length encoding. It is only at larger resolutions that memory demands become an issue. In such cases, the concept of Kolmogorov compression can be exploited, in which a generating algorithm with real-time performance is paired with a set of input parameters to re-generate a terrain on the fly. Indeed this is arguably the largest single benefit of procedural methods, which exhibit a high degree of Kolmogorov compression.

| Paper | Landforms $\mathcal{M}\ \mathcal{R}\ \mathcal{A}$ | Method | Model $\mathcal{G}\ \mathcal{F}\ \mathcal{V}$ | Control | Extent | Precision | Performance |
|---|---|---|---|---|---|---|---|
| [MKM89, EMP*98] | ● ○ ○ | Noise | ○ ● ○ | Parameters | $\infty$ | $\infty$ | Real-time |
| [FFC82b, DKW94, vLJ95] | ● ○ ○ | Subdivision | ● ○ ○ | | | | |
| [Mil86, Lew87, Man88] | ● ○ ○ | Subdivision | ● ○ ○ | | | | |
| [Vos91, KSU07] | ● ○ ○ | Faulting | ○ ● ○ | | | | |
| [PH93] | ● ● ○ | Subdivision | ● ○ ○ | Parameters | $10-100$ km | $10-100$ m | Seconds |
| [KMN88] | ● ● ○ | | ● ○ ○ | Parameters | | | Minutes |
| [Bel07, BA05a, TB18] | ● ● ○ | | ● ○ ○ | Elevations | | | Real-time |
| [DGGK11] | ○ ● ○ | | ● ○ ○ | Parameters | $>10000$ km | $<1$ m | Real-time |
| [RME09] | ● ○ ○ | Shortest-path | ● ○ ○ | Elevations | $10-100$ km | $10-100$ m | Seconds |
| [GMS09] | ● ● ○ | Deformation | ● ○ ○ | Silhouettes | $1-10$ km | $1-10$ m | Interactive |
| [HGA*10b] | ● ● ○ | Diffusion | ● ○ ○ | Elevations | $1-10$ km | $1-10$ m | Interactive |
| [BKRE17, BKRE18] | ○ ○ ● | | ○ ○ ● | Elevations | 100 m | 1 m | |
| [BCA*14, Par15] | ● ○ ○ | Noise | ○ ● ○ | Painting | $\infty$ | $\infty$ | Real-time |
| [dCB09] | ● ○ ○ | Noise | ○ ● ○ | Painting | $1-10$ km | $\infty$ | Seconds |
| [DCPSB14] | ○ ● ○ | Noise | ● ○ ○ | Painting | $1-10$ km | $\infty$ | Minutes |
| [GGG*13, GGP*15] | ● ● ○ | Functions | ○ ● ○ | Primitives | $100-1000$ km | $1-10$ m | Seconds |
| [GM01] | ● ○ ● | Functions | ○ ● ○ | Parameters | $\infty$ | $\infty$ | Interactive |
| [IFMC03] | ● ○ ● | Joints | ○ ○ ● | Parameters | $100-1000$ m | $1-10$ m | Minutes |
| [PGMG09a] | ○ ○ ● | Functions | ○ ● ● | Sculpting | $100-1000$ m | $<1$ m | Interactive |

**Table 4:** *Overview of procedural terrain generation techniques. Under landforms: $\mathcal{M}$ = mountains, $\mathcal{R}$ = river networks, $\mathcal{A}$ = volumetric. Under model: $\mathcal{G}$ = grid , $\mathcal{F}$ = functions, $\mathcal{V}$ = Mesh. Control corresponds to the classification in Table 1.*

## 7. Conclusion

Terrain modeling has been an active field of research for more than four decades, and dozens of procedural, simulation, and example-based solutions have been proposed. Despite this, the sheer variety of terrain types, complexity of landforms, and diversity of patterns observed at different scales has inevitably left many unsolved challenges.

This forces designers to revert in part to manual terrain editing in order to realize their intentions and also prevents the generation of realistic terrain on a planetary scale.

Here, we briefly present the most important and difficult problems that remain.

**Landforms:** The variety of representable landforms remains limited. Existing systems simply do not provide artists with the level of landscape realism and variety required by the entertainment industry, which is why terrains are often still modeled by hand. Many techniques focus on a specific terrain feature, such as rivers [GGG*13, GGP*15] or canyons [DCPSB14], and this leads to relatively homogenous landscapes. Example-based methods do exhibit acceptable variety provided it is encapsulated in the source data, but they can have issues with global consistency. Recent work on the combined simulation of different forms of erosion [CGG*17] is a first step towards generating landscapes with more variety, but further improvement is required. In particular, we believe that generating large-scale terrains with a variety of landforms and geomorphological patterns, while maintaining overall consistency, remains an open challenge. Finally, some important processes are still missing, such as wind and glacial erosion. Specifically, geomorphology shows that many terrains have been shaped by glacial erosion, and exhibit characteristic features such as hanging valleys located at the top of U-shaped valleys.

**Volumetric models:** Effectively modeling 3D landforms (such as caves, arches, cliffs with overhangs, and Goblins) is another difficulty. A reliance on discrete data-structures, such as layer stacks [PGMG09a] or voxels [BFO*07, BKRE17, BKRE18] severely limits the achievable extent, in general, to less than a few hundreds meters due to the memory burden. To progress it will likely be necessary to dispense with discrete grids altogether. Fortunately, implicit surface representations offer promise because they exhibit a high degree of Kolmogorov compression.

**Large-scale terrains:** Another challenge is to break the current limits on terrain extent. Although some attempts have been made to generate planets with infinite detail [Vos91, EMP*98, DGGK11], these are overly reliant on procedural noise, which, despite theoretically infinite extent and precision, suffers from severe shortcomings in user control and realism. Therefore, generating entire planets while providing control over the position, extent and shape of landforms at different scales remains to be solved.

Multi-resolution methods operating at varying time and space

scales, combined with a careful analysis of the cause-effect relationships in geomorphology phenomena, including their mutual inter-influence, and temporal and spatial relationships, offer a possible route to generating large-scale terrains.

**Computational efficiency:** Rapid terrain generation is crucial to providing artists with the interactive or real-time feedback so necessary for intuitive authoring. Relatively few non-procedural methods presently achieve such update rates. Moreover, simulation approaches specifically, including thermal and hydraulic erosion, simply do not scale well to larger terrains. Simulation methods that require under a second on a standard $1\,\mathrm{km}^2$ terrain sampled at 1 m precision, typically take more than three hours to complete on $100 \times 100\,\mathrm{km}^2$ terrains at the same precision. Although implementations on graphics hardware do offer speedups, they do not change the overall complexity of the problem, and are only a temporary fix. One possibility is to begin by enhancing the realism of the more efficient methods, such as procedural generation. As an example, a new class of basis-functions for noise could be designed that directly generate the dendritic patterns observed in real terrain, and also approximate branching structures produced by erosion.

## Acknowledgments

This research was supported by the project PAPAYA P110720-2659260, funded by the Fonds National pour la Societé Numérique, the project HWD ANR-16-CE33-0001 supported by Agence Nationale de la Recherche, and funded in part by National Science Foundation grants #10001387, Functional Proceduralization of 3D Geometric Models.

## References

[AA10] ANDERSON R. S., ANDERSON S. P.: *Geomorphology: the mechanics and chemistry of landscapes*. Cambridge University Press, 2010. 17

[AAC*17] ARGUDO O., ANDUJAR C., CHICA A., GUÉRIN E., DIGNE J., PEYTAVIE A., GALIN E.: Coherent multi-layer landscape synthesis. *The Visual Computer 33*, 6 (2017), 1005–1015. 14, 17, 18, 20

[AM15] ARIYAN M., MOULD D.: Terrain synthesis using curve networks. In *Proceedings of Graphics Interface* (2015), pp. 9–16. 6

[ASA07] ANH N. H., SOURIN A., ASWANI P.: Physically based hydraulic erosion simulation on GPU. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (Perth, Australia, 2007), ACM, pp. 257–264. 12

[BA05a] BELHADJ F., AUDIBERT P.: Modeling landscapes with ridges and rivers. In *Proceedings of the Symposium on Virtual Reality Software and Technology* (Monterey, USA, 2005), ACM, pp. 151–154. 6, 22

[BA05b] BELHADJ F., AUDIBERT P.: Modeling landscapes with ridges and rivers: Bottom up approach. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (Dunedin, New Zealand, 2005), ACM, pp. 447–450. 6

[BA05c] BENES B., ARRIAGA X.: Table mountains by virtual erosion. In *Proceedings of the Eurographics Workshop on Natural Phenomena* (2005), Eurographics Association, pp. 33–39. 10

[BCA*14] BRADBURY G., CHOI I., AMATI C., MITCHELL K., WEYRICH T.: Frequency-based creation and editing of virtual terrain. In *Proceedings of European Conference on Visual Media Production* (London, UK, 2014). 22

[Bel07] BELHADJ F.: Terrain modeling: a constrained fractal model. In *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (Grahamstown, South Africa, 2007), ACM, pp. 197–204. 6, 22

[Ben07] BENES B.: Real-time erosion using shallow water simulation. In *Proceedings of Virtual Reality Interactions and Physical Simulations* (Dublin, Ireland, 2007), Eurographics Association. 12

[BF01] BENES B., FORSBACH R.: Layered data representation for visual simulation of terrain erosion. In *Proceedings of the Spring Conference on Computer Graphics* (2001), IEEE, pp. 80–85. 2

[BF02] BENES B., FORSBACH R.: Visual simulation of hydraulic erosion. *Journal of WSCG 10*, 1-3 (2002), 79–86. 12, 21

[BFO*07] BEARDALL M., FARLEY M., OUDERKIRK D., REIMSCHUSSEL C., SMITH J., JONES M., EGBERT P.: Goblins by spheroidal weathering. In *Proceedings of the Eurographics Workshop on Natural Phenomena* (2007), Eurographics Association, pp. 7–14. 13, 21, 22

[BKRE17] BECHER M., KRONE M., REINA G., ERTL T.: Feature-based volumetric terrain generation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2017), pp. 10:1–10:9. 8, 22

[BKRE18] BECHER M., KRONE M., REINA G., ERTL T.: Feature-based volumetric terrain generation and decoration. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 8, 22

[BR04] BENES B., ROA T.: Simulating desert scenery. *Journal of WSCG* (2004), 17–22. 13

[Bri08] BRIDSON R.: *Fluid Simulation*. A. K. Peters, Ltd., Natick, MA, USA, 2008. 11

[BS97] BRAUN J., SAMBRIDGE M.: Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. *Basin Research 9*, 1 (1997), 27–52. 9, 10

[BTHB06] BENES B., TĚŠÍNSKÝ V., HORNYŠ J., BHATIA S. K.: Hydraulic erosion. *Computer Animation and Virtual Worlds 17*, 2 (2006), 99–108. 12, 21

[CBC*16] CORDONNIER G., BRAUN J., CANI M.-P., BENES B., ÉRIC GALIN, PEYTAVIE A., ÉRIC GUÉRIN: Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum 35*, 2 (2016), 165–175. 10, 11, 21

[CCB*18] CORDONNIER G., CANI M.-P., BENES B., BRAUN J., GALIN E.: Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE Transactions on Visualization and Computer Graphics 24*, 5 (2018), 1756–1769. 11, 20, 21

[CF88] CONSTANTIN P., FOIAS C.: *Navier-stokes equations*. University of Chicago Press, 1988. 11

[CGG*17] CORDONNIER G., GALIN E., GAIN J., BENES B., GUÉRIN E., PEYTAVIE A., CANI M.-P.: Authoring Landscapes by Combining Ecosystem and Terrain Erosion Simulation. *ACM Transactions on Graphics 36*, 4 (2017), 134:1–134:12. 2, 12, 13, 19, 20, 21, 22

[CMF98] CHIBA N., MURAOKA K., FUJITA K.: An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation 9*, 4 (1998), 185–194. 12, 21

[dCB09] DE CARPENTIER G. J. P., BIDARRA R.: Interactive GPU-based procedural heightfield brushes. In *Proceedings of the International Conference on Foundations of Digital Games* (Orlando, USA, 2009), ACM, pp. 55–62. 5, 19, 22

[DCPSB14] DE CARLI D. M., POZZER C. T., SCHETINGER V., BEVILACQUA F.: Procedural generation of 3D canyons. In *Proceedings of the Conference on Graphics, Patterns and Images* (Rio de Janeiro, Brazil, 2014), IEEE, pp. 103–110. 7, 22

[DGGK11] DERZAPF E., GANSTER B., GUTHE M., KLEIN R.: River networks for instant procedural planets. *Computer Graphics Forum 30*, 7 (2011), 2031–2040. 6, 22

[DKW94] DIXON A. R., KIRBY G. H., WILLS D. P. M.: A data structure for artificial terrain generation. *Computer Graphics Forum 13*, 1 (1994), 37–48. 4, 22

[DMS05] DACHSBACHER C., MEYER M., STAMMINGER M.: Height-field synthesis by non-parametric sampling. In *Proceedings of Vision Modeling and Visualization* (Erlangen, Germany, 2005), pp. 297–302. 14, 20

[dPI13] DOS PASSOS V. A., IGARASHI T.: Landsketch: A first person point-of-view example-based terrain modeling approach. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2013), ACL, pp. 61–68. 15, 19, 20

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision* (Corfu, Greece, 1999), IEEE, pp. 1–6. 14

[EMP*98] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*, 3rd ed. The Morgan Kaufmann Series in Computer Graphics. Elsevier, 1998. 4, 5, 22

[EPCV15] EMILIEN A., POULIN P., CANI M.-P., VIMONT U.: Interactive procedural modelling of coherent waterfall scenes. *Computer Graphics Forum 34*, 6 (2015), 22–35. 16, 19, 20

[FFC82a] FOURNIER A., FUSSELL D., CARPENTER L.: Comment on computer rendering of stochastic models. *Communications of the ACM 25*, 8 (1982), 581–584. 4

[FFC82b] FOURNIER A., FUSSELL D., CARPENTER L.: Computer rendering of stochastic models. *Computer Graphics 25*, 6 (1982), 371–384. 4, 5, 22

[GDG*17] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Transactions on Graphics (Proceedings of Siggraph Asia 2017) 36*, 6 (2017). 16, 17, 19, 20

[GDGP16] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A.: Sparse representation of terrains for procedural modeling. *Computer Graphics Forum (Proceedings of Eurographics) 35*, 2 (2016), 177–187. 7, 16, 18, 19, 20

[GGG*13] GÉNEVAUX J.-D., GALIN É., GUÉRIN É., PEYTAVIE A., BENES B.: Terrain generation using procedural models based on hydrology. *ACM Transactions on Graphics (Proceedings of Siggraph) 32*, 4 (2013), 143:1–143:13. 7, 19, 22

[GGP*15] GÉNEVAUX J.-D., GALIN É., PEYTAVIE A., GUÉRIN É., BRIQUET C., GROSBELLET F., BENES B.: Terrain modeling from feature primitives. *Computer Graphics Forum* (2015), 198–210. 3, 7, 17, 22

[GM01] GAMITO M. N., MUSGRAVE K. F.: Procedural landscapes with overhangs. In *Proceedings of the Portuguese Computer Graphics Meeting* (Lisbon, Portugal, 2001). 8, 22

[GMM15] GAIN J., MERRY B., MARAIS P.: Parallel, realistic and controllable terrain synthesis. *Computer Graphics Forum 34*, 2 (2015), 105–116. 14, 15, 20, 21

[GMS09] GAIN J. E., MARAIS P., STRASSER W.: Terrain sketching. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (Boston, USA, 2009), ACM, pp. 31–38. 6, 7, 22

[GOG*02] GESCH D., OIMOEN M., GREENLEE S., NELSON C., STEUCK M., TYLER D.: The national elevation dataset. *Photogrammetric Engineering and Remote Sensing 68*, 1 (2002), 5–11. 14

[Har12] HARVEY A.: *Introducing Geomorphology: A Guide to Landforms and Processes*. Dunedin Academic Press, 2012. 17

[HGA10a] HNAIDI H., GUÉRIN E., AKKOUCHE S.: Multiresolution control of curves and surfaces with a self-similar model. *Fractals 18* (2010), 271–286. 6

[HGA*10b] HNAIDI H., GUÉRIN É., AKKOUCHE S., PEYTAVIE A., GALIN É.: Feature based terrain generation using diffusion equation. *Computer Graphics Forum 29*, 7 (2010), 2179–2186. 7, 8, 19, 22

[Hor45] HORTON R. E.: Erosional development of streams and their drainage basins: hydro-physical approach to quantitative morphology. *Geological Society of America Bulletin 56*, 3 (1945), 275–370. 8

[IFMC03] ITO T., FUJIMOTO T., MURAOKA K., CHIBA N.: Modeling rocky scenery taking into account joints. In *Proceedings of Computer Graphics International* (2003), IEEE, pp. 244–247. 10, 22

[JFBB10] JONES M. D., FARLEY M., BUTLER J., BEARDALL M.: Directable weathering of concave rock using curvature estimation. *Transactions on Visualization and Computer Graphics 16*, 1 (2010), 81–94. 13, 21

[KBKŠ09] KRIŠTOF P., BENES B., KŘIVÁNEK J., ŠŤAVA O.: Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum 28*, 2 (2009), 219–228. 12, 21

[KG14] KNIGHT J., GRAB S.: Lightning as a geomorphic agent on mountain summits: evidence from southern Africa. *Geomorphology 204* (2014), 61–70. 13

[KMN88] KELLEY A. D., MALIN M. C., NIELSON G. M.: Terrain simulation using a model of stream erosion. *Computer Graphics 22*, 4 (1988), 263–268. 6, 22

[KRS15] KETABCHI K., RUNIONS A., SAMAVATI F. F.: 3D Maquetter: Sketch-based 3D content modeling for digital Earth. In *2015 International Conference on Cyberworlds* (2015), pp. 98–106. 16, 20

[KSA13] KÄMPE V., SINTORN E., ASSARSSON U.: High resolution sparse voxel DAGs. *Transaction on Graphics 32*, 4 (2013), 101:1–101:13. 3

[KSU07] KAMAL K. R., SARWAR UDDIN Y.: Parametrically controlled terrain generation. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (Perth, Australia, 2007), ACM, pp. 17–23. 5, 22

[Kur12] KUROWSKI M.: Procedural generation of meandering rivers inspired by erosion. *Journal of WSCG* (2012), 79–86. 12, 21

[Kur13] KUROWSKI M.: Procedural generation of meandering rivers with oxbow lakes. In *Computer Graphics International – Short Paper* (2013), IEEE. 12, 21

[Lew87] LEWIS J.-P.: Generalized stochastic subdivision. *ACM Transaction on Graphics 6*, 3 (1987), 167–190. 4, 22

[LH05] LEFEBVRE S., HOPPE H.: Parallel controllable texture synthesis. *ACM Transactions on Graphics 24*, 3 (2005), 777–786. 15

[LK11] LAINE S., KARRAS T.: Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics 17*, 1 (2011), 1048–1059. 3

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse Gabor convolution. *Transaction on Graphics 28*, 3 (2009), 54:1–54:10. 5

[LSC*10] LAGAE A., SYVALIN L., COOK R. L., DEROSE T., DRETTAKIS G., EBERT D. S., LEWIS J. P., PERLIN K., M. Z.: A survey of procedural noise functions. *Computer Graphics Forum 29*, 8 (2010), 2579–2600. 5

[MAAS15] MAHDAVI-AMIRI A., ALDERSON T., SAMAVATI F.: A survey of digital earth. *Computer & Graphics 53*, B (2015), 95–117. 2

[Man82] MANDELBROT B. B.: *The Fractal Geometry of Nature*. W. H. Freeman & Co Ltd, 1982. 4

[Man88] MANDELBROT B. B.: *The Science of Fractal Images*, 1 ed. Springer, 1988, ch. Fractal Landscapes Without Creases and with Rivers, pp. 243–260. 4, 22

[McC92] MCCLAY K.: *Thrust Tectonics*. Springer, 1992. 10

[MDH07] MEI X., DECAUDIN P., HU B.: Fast hydraulic erosion simulation and visualization on GPU. In *Proceedings of the Pacific Conference on Computer Graphics and Applications* (Maui, USA, 2007), IEEE, pp. 47–56. 12, 20, 21

[MEC15] MICHEL E., EMILIEN A., CANI M.-P.: Generation of folded terrains from simple vector maps. In *Eurographics – Short Papers* (2015). 10, 21

[Mil86] MILLER G.: The definition and rendering of terrain maps. *Computer Graphics 20*, 4 (1986), 39–48. 4, 22

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The synthesis and rendering of eroded fractal terrains. *Computer Graphics 23*, 3 (1989), 41–50. 2, 5, 9, 11, 12, 21, 22

[MMW01] MIAO T.-D., MU Q.-S., WU S.-Z.: Computer simulation of aeolian sand ripples and dunes. *Physics Letters A 288*, 1 (2001), 16–22. 13

[MVN68] MANDELBROT B. B., VAN NESS J. W.: Fractional brownian motions, fractional noises and applications. *SIAM Review – SIREV 10*, 4 (1968), 422–437. 4

[Nag98] NAGASHIMA K.: Computer generation of eroded valley and mountain terrains. *The Visual Computer 13*, 9-10 (1998), 456–464. 12, 21

[NLP*13] NATALI M., LIDAL E. M., PARULEK J., VIOLA I., PATEL D.: Modeling terrains and subsurface geology. In *Proceedings of Eurographics – State of the Art* (2013), pp. 155–173. 2

[NWD05] NEIDHOLD B., WACKER M., DEUSSEN O.: Interactive physically based fluid and erosion simulation. In *Proceedings of the Eurographics Workshop on Natural Phenomena* (Dublin, Ireland, 2005), Eurographics Association, pp. 25–32. 12, 21

[NZRC09] NARTEAU C., ZHANG D., ROZIER O., CLAUDIN P.: Setting the length and time scales of a cellular automaton dune model from the analysis of superimposed bed forms. *Journal of Geophysical Research: Earth Surface 114*, F3 (2009). 13

[ON00] ONOUE K., NISHITA T.: A method for modeling and rendering dunes with wind-ripples. In *Proceedings of the Pacific Conference on Computer Graphics and Applications* (2000), IEEE, pp. 427–428. 13

[Par14] PARBERRY I.: Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques 3*, 1 (2014), 74–85. 5

[Par15] PARBERRY I.: Modeling real-world terrain with exponentially distributed noise. *Journal of Computer Graphics Techniques 4*, 2 (2015), 1–9. 5, 22

[Pel09] PELLETIER J. D.: Controls on the height and spacing of eolian ripples and transverse dunes: A numerical modeling investigation. *Geomorphology 105*, 3 (2009), 322–333. 13

[Per85] PERLIN K.: An image synthesizer. *Computer Graphics 19*, 3 (1985), 287–296. 5

[Per89] PERLIN K.: Hypertexture. *Computer Graphics 23*, 3 (1989), 253–262. 5

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer 23*, 8 (2007), 583–605. 1

[PGMG09a] PEYTAVIE A., GALIN É., MÉRILLOU S., GROSJEAN J.: Arches: a framework for modeling complex terrains. *Computer Graphics Forum 28*, 2 (2009), 457–467. 3, 8, 10, 22

[PGMG09b] PEYTAVIE A., GALIN É., MÉRILLOU S., GROSJEAN J.: Procedural generation of rock piles using aperiodic tiling. *Computer Graphics Forum 28*, 7 (2009), 1801–1810. 10

[PH93] PRUSINKIEWICZ P., HAMMEL M.: A fractal model of mountains with rivers. In *Proceedings of Graphics Interface* (Toronto, Canada, 1993), Canadian Information Processing Society, pp. 174–180. 6, 22

[PM13] PYTEL A., MANN S.: Self-organized approach to modeling hydraulic erosion features. *Computers & Graphics 37*, 4 (2013), 280–292. 12, 21

[RME09] RUSNELL B., MOULD D., ERAMIAN M. G.: Feature-rich distance-based terrain synthesis. *The Visual Computer 25*, 5-7 (2009), 573–579. 7, 19, 22

[Ros94] ROSGEN D. L.: A classification of natural rivers. *Catena 22*, 3 (1994), 169–199. 8

[RPP93] ROUDIER P., PEROCHE B., PERRIN M.: Landscapes synthesis achieved through erosion and deposition process simulation. *Computer Graphics Forum 12*, 3 (1993), 375–383. 9, 11, 21

[ŠBBK08] ŠŤAVA O., BENES B., BRISBIN M., KŘIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. In *Proceedings of Symposium on Computer Animation* (Dublin, Ireland, 2008), pp. 201–210. 12, 20, 21

[SBD13] SCHOLZ M., BENDER J., DACHSBACHER C.: Level of Detail for Real-Time Volumetric Terrain Rendering. In *Vision, Modeling & Visualization* (2013), Bronstein M., Favre J., Hormann K., (Eds.), Eurographics Association, pp. 211–218. 8

[SD03] STOCK J., DIETRICH W. E.: Valley incision by debris flows: Evidence of a topographic signature. *Water Resources Research 39*, 4 (2003). 9, 10

[SR16] SAMAVATI F., RUNIONS A.: Interactive 3D content modeling for digital earth. *The Visual Computer 32*, 10 (2016), 1293–1309. 16

[TB18] TALGORN F.-X., BELHADJ F.: Real-time sketch-based terrain generation. In *Proceedings of Computer Graphics International* (2018), CGI 2018, pp. 13–18. 6, 22

[TEC*14] TASSE F. P., EMILIEN A., CANI M.-P., HAHMANN S., DODGSON N.: Feature-based terrain editing from complex sketches. *Computers & Graphics 45* (2014), 101–115. 15, 16, 19, 20

[TGM12] TASSE F. P., GAIN J. E., MARAIS P.: Enhanced texture-based terrain synthesis on graphics hardware. *Computer Graphics Forum 31*, 6 (2012), 1959–1972. 14, 15, 20

[TH10] TUCKER G. E., HANCOCK G. R.: Modelling landscape evolution. *Earth Surface Processes and Landforms 35*, 1 (2010), 28–50. 9, 10

[TJ10] TYCHONIEVICH L. A., JONES M. D.: Delaunay deformable mesh for the weathering and erosion of 3D terrain. *The Visual Computer 26*, 12 (2010), 1485–1495. 13, 21

[VBHŠ11] VANEK J., BENES B., HEROUT A., ŠŤAVA O.: Large-scale physics-based terrain editing using adaptive tiles on the GPU. *Computer Graphics and Applications 31*, 6 (2011), 35–44. 12, 21

[vLJ95] VAN LAWICK J. V. P., JENSE H.: Dynamic terrain generation based on multifractal techniques. *High Performance Computing for Computer Graphics and Visualisation* (1995), 186–203. 5, 22

[VMG16] VILLANUEVA A. J., MARTON F., GOBBETTI E.: Ssvdags: Symmetry-aware sparse voxel dags. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2016), I3D'16, ACM, pp. 7–14. 3

[Vos91] VOSS R. F.: Random fractal forgeries. In *Fundamental Algorithms for Computer Graphics*, vol. 17. Springer, 1991, pp. 805–835. 4, 22

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Proceedings of Eurographics – State Of The Art* (2009), pp. 93–117. 14

[Wor96] WORLEY S.: A cellular texture basis function. In *Proceedings of SIGGRAPH* (New Orleans, USA, 1996), ACM, pp. 291–294. 5

[WT99] WHIPPLE K. X., TUCKER G. E.: Dynamics of the stream-power river incision model: Implications for height limits of mountain ranges, landscape response timescales, and research needs. *Journal of Geophysical Research: Solid Earth 104*, B8 (1999), 17661–17674. 9, 10

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain synthesis from digital elevation models. *Transactions on Visualization and Computer Graphics 13*, 4 (2007), 834–848. 14, 15, 19, 20