Terrain Amplification with Implicit 3D Features

Axel Paris, Eric Galin, Adrien Peytavie, Eric Guérin, James Gain

Terrain Amplification with Implicit 3D Features

AXEL PARIS, Université de Lyon, LIRIS, CNRS, UMR5205, France
ERIC GALIN, Université de Lyon, LIRIS, CNRS, UMR5205, France
ADRIEN PEYTAVIE, Université de Lyon, LIRIS, CNRS, UMR5205, France
ERIC GUÉRIN, Université de Lyon, LIRIS, CNRS, UMR5205, France
JAMES GAIN, University of Cape Town, South Africa

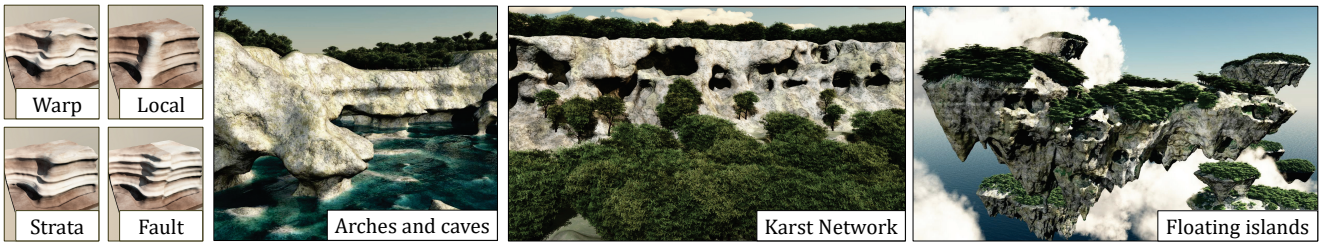


Fig. 1. From a 2D input height field, our method automatically generates an implicit model for representing the terrain, which is augmented with complex 3D landform features such as caves, overhangs, cliffs, arches or karsts. Our model can also represent dramatic and scenic science fiction landscapes such as floating islands, or giant rock spires.

While three-dimensional landforms, such as arches and overhangs, occupy a relatively small proportion of most computer generated landscapes, they are distinctive and dramatic and have an outsize visual impact. Unfortunately, the dominant heightfield representation of terrain precludes such features, and existing in-memory volumetric structures are too memory intensive to handle larger scenes.

In this paper, we present a novel memory-optimized paradigm for representing and generating volumetric terrain based on implicit surfaces. We encode feature shapes and terrain geology using construction trees that arrange and combine implicit primitives. The landform primitives themselves are positioned using Poisson sampling, built using open shape grammars guided by stratified erosion and invasion percolation processes, and, finally, queried during polygonization. Users can also interactively author landforms using high-level modeling tools to create or edit the underlying construction trees, with support for iterative cycles of editing and simulation.

We demonstrate that our framework is capable of importing existing large-scale heightfield terrains and amplifying them with such diverse structures as slot canyons, sea arches, stratified cliffs, fields of hoodoos, and complex karst cave networks.

CCS Concepts: • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: Procedural Modeling, Landscapes, Implicit Surfaces

Authors' addresses: Axel Paris, Université de Lyon, LIRIS, CNRS, UMR5205, France, eric.galin@liris.cnrs.fr; Eric Galin, Université de Lyon, LIRIS, CNRS, UMR5205, France, eric.galin@liris.cnrs.fr; Adrien Peytavie, Université de Lyon, LIRIS, CNRS, UMR5205, France, adrien.peytavie@liris.cnrs.fr; Eric Guérin, Université de Lyon, LIRIS, CNRS, UMR5205, France, eric.guerin@liris.cnrs.fr; James Gain, University of Cape Town, South Africa, jgain@cs.uct.ac.za.

ACM Reference Format:
Axel Paris, Eric Galin, Adrien Peytavie, Eric Guérin, and James Gain. 2010. Terrain Amplification with Implicit 3D Features. *ACM Trans. Graph.* 9, 4, Article 39 (March 2010), 15 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

Truly three-dimensional landscape features are some of the most visually arresting and memorable elements of real terrains. They are formed by different physical processes (including joint fracturing, percolation, and stratified erosion), take a variety of forms (from steep-walled canyons to underground cave complexes), and exhibit different scales (from mineral deposits, such as stalactites, less than a meter in diameter, to sea cliffs stretching for kilometers).

The sheer variety of shapes and scales of these landforms presents significant modelling challenges and, despite the wide application of digital terrain in games, film, and simulation, and extensive research in this area, effectively representing and generating 3D landforms remains an unsolved problem. This is because most solutions address only $2\frac{1}{2}$D heightfield terrains, and those that are truly 3D either focus on specific structural forms or are limited in the achievable resolution due to memory considerations. There is thus a need for compact, memory efficient models for representing large terrains featuring sparse and local volumetric landforms.

Existing methods fall into three categories: procedural generation, which applies phenomenologically-inspired algorithms, erosion simulation, where erosion patterns are carved into a base terrain, and example-based synthesis, in which samples from scans of real terrains are extracted and combined. These generally rely on heightfields, which locate scalar elevation values on a regular grid at a single sampling resolution. As a consequence, steep areas, such as cliffs, are generally undersampled, and overhanging features, simply cannot be represented. The alternative — an explicit volumetric representation — is also problematic as such structures are memory

intensive and, consequently, previous 3D terrain approaches either focus on smaller isolated landforms [Beardall et al. 2007; Ito et al. 2003; Jones et al. 2010] or represent larger landscapes at a lower sampling resolution [Becher et al. 2017, 2018; Peytavie et al. 2009].

Instead, we provide a conceptually simple solution to the problem of representing and authoring 3D terrain across a *range of scales*, achieved through a unified *implicit surface model*. This allows $2\frac{1}{2}$D elevation models to be augmented with compact, memory efficient sculpting primitives that encode volumetric landforms. Our approach can be integrated with existing modeling pipelines, captures a wide variety of landforms from underground cave complexes to coastal cliffs, incorporates geomorphological effects, such as stratified erosion and invasion percolation, and provides extensive user control (Figure 1).

Our implicit model allows the automatic enhancement of terrains with complex 3D landforms and generates visually appealing, although sparse, geological shapes, which are nonetheless essential for synthesizing dramatic and scenic landscapes. Furthermore, detail can be enhanced even where overhangs are not strictly present, such as on steep slopes and vertical sections. This is warranted because these often represent visually prominent landmarks.

At the heart of our method are various construction trees for blending implicit primitives that individually represent the input terrain, landform shape modifiers and geological structure, and collectively provide a full 3D volumetric terrain. In this we are inspired by and extend the notion of Implicit BlobTrees [Wyvill et al. 1999] and Feature Primitives [Génevaux et al. 2015]. Querying this new implicit terrain representation during surface extraction allows us to bypass an explicit and memory-intensive volumetric representation. Our pipeline imports a $2\frac{1}{2}$D input terrain and converts it into a coherent implicit surface, identifies 3D feature sites, and as specified by the user applies different generation algorithms, such as grammar-like production rules or erosion processes, to sculpt and augment the terrain with overhanging landforms. Finally, the sparse implicit representation is efficiently polygonized using a novel locally adaptative approach that generates a final mesh amplified with 3D terrain features.

More precisely, the main technical contributions of our work include: 1) a procedural model for representing the underlying geology of a terrain (Section 4) and guiding the generation processes (Section 6) in a memory-efficient fashion; 2) a coherent implicit surface-based sparse landform construction tree (Section 5) that supports the compact encoding of 3D terrains with local 3D landforms, such as arches and overhanging cliffs; 3) efficient 3D landform generators, which analyze the characteristics of the input terrain and assemble primitives to emulate erosion processes, such as stream or sea erosion, or incorporate specific landforms, such as goblins (Section 6); and 4) an efficient implicit surface polygonization algorithm (Section 7) adapted to the sparse amplified terrain data-structure of our volumetric terrains. Moreover, we demonstrate that our model supports both procedural landform shaping processes and interactive editing for the creation of complex terrain using high-level tools that bolster iterative refinement with seamless cycles of editing and simulation.

Our approach is the first capable of generating sparse volumetric landforms over terrains that exhibit both fine detail and large extent, as demonstrated in Figure 1. This work primarily benefit the entertainment industry, and could be implemented in middleware applications for handling scenic terrains with 3D landforms.

2 RELATED WORK

The field of synthetic terrain modeling, as surveyed by Natali *et al.* [2013] and more recently by Galin *et al.*[2019], can be separated into three classes of techniques: procedural generation, erosion simulation, and example-based synthesis. The overwhelming focus across these categories is on the creation of $2\frac{1}{2}$D height fields, with elevation specified for points on a regular grid.

Procedural generation exploits two characteristics of real terrains: the self-similarity of landforms across a range of scales and translations [Ebert et al. 1998] and the strong shaping influence of river networks and hydrological erosion. Generally, the first aspect is captured algorithmically through multi-frequency noise functions and the second through constrained procedural subdivision [Belhadj and Audibert 2005; Kelley et al. 1988; Prusinkiewicz and Hammel 1993], diffusion [Hnaidi et al. 2010; Tasse et al. 2014], or warping and blending [Gain et al. 2009; Génevaux et al. 2013; Génevaux et al. 2015; Rusnell et al. 2009], which can even extend to entire planets [Derzapf et al. 2011].

Although these core algorithms can efficiently generate near-infinite landscapes with unlimited precision, they only provide indirect global control and produce terrains without any underlying geomorphological structure. The paucity of user control can be corrected by allowing users to interactively specify constraints through sketching or painting [Gain et al. 2009; Hnaidi et al. 2010; Tasse et al. 2014, 2012], but the problem of geomorphological realism remains.

Erosion simulation [Musgrave et al. 1989] approximates the geological evolution of terrain through iterations of hydraulic erosion [Chiba et al. 1998; Nagashima 1998], subsurface tectonics [Cordonnier et al. 2018], or an amalgam of secondary erosion effects [Cordonnier et al. 2017] applied to a base terrain. Clearly subsurface strata play an important role in such simulations. Often this is encoded as a cell-based grid of layered stacks, with different thicknesses and material properties for the layers of each cell-specific stack [Cordonnier et al. 2018, 2017; Roudier et al. 1993]. Nevertheless, since the layers are solid and contiguous, this represents a layered extension of heightfields rather than a true 3D representation.

While simulation approaches can realistically capture an increasing variety of geological phenomena, they are difficult to control and computationally demanding. Even with GPU acceleration [Mei et al. 2007; Vanek et al. 2011] these methods cannot be used to author large-scale finely-sampled terrains that match a user's intent.

Example-based synthesis approaches borrow from texture synthesis and combine realism and high-level user-control by stitching new terrains from patches [Tasse et al. 2012; Zhou et al. 2007], pixels [Gain et al. 2015] or radial primitives [Guérin et al. 2016] extracted from exemplars. They are thus heavily reliant on sourcing high quality digital elevation models for the exemplar database. Early patch-based terrain synthesis driven by a user-painted 2D map [Zhou et al. 2007] has subsequently been improved in terms

of both computational efficiency and user control [Gain et al. 2015; Tasse et al. 2012]. Larger-scale alternatives include compiling a dictionary of feature-rich radial primitives that can later be extracted, sparsely placed and blended to form a terrain [Guérin et al. 2016] or using generative adversarial networks to learn and apply a correspondence between user sketch maps and scanned terrains [Guérin et al. 2017]. These methods are fast, controllable and locally realistic but can fail to respect large-scale geomorphological patterns, such as drainage networks.

In general, these techniques build on variants of a $2\frac{1}{2}$D heightfield encoding, which limits the resolution achievable on steep slopes and precludes truly three dimensional features such as overhangs, arches, and caves. There have been attempts to address this with alternative underlying representations: advected surfaces [Gamito and Musgrave 2001] borrow from fluid simulation by warping terrain according to a differentiable vector field. With suitable restrictions on the vector field, the 3D output surface can be guaranteed to be connected and non-self-intersecting. Alas, authoring is generally limited to a single global function and the surface extraction requires ray marching, which can be expensive.

Voxel structures are at the heart of several volumetric approaches. Ito *et al.* [2003] emulate fracturing along rock joints by linking neighbouring voxels and then selectively breaking these links in fracture zones. Stability analysis is then used to reposition or remove connected voxel structures. Although realistic, this strategy is costly, difficult to author and limited to a specific effect. Beardall *et al.* [2007] and later Jones *et al.* [2010] focus on modeling small-scale self-contained columnar structures, notably hoodoos and goblins. Users define initial conditions, such as a rough overall shape and the resistance of stratified voxels, and then spheroidal and cavernous erosion operators are iteratively applied. The curve diffusion method of Becher *et al.* [2018] is more general in scope. It extends the heightfield diffusion of feature curves [Hnaidi et al. 2010]. These curves embody prominent landforms, like ridges and river-beds, and constrain altitude along the curve, but also, crucially, the orthogonal slope. This is extended to 3D by writing feature, noise and diffusion attributes into a voxel grid, from which a surface can later be extracted. Nevertheless, because the voxel grid is encoded explicitly, available memory limits the achievable resolution.

In the Arches system, Peytavie *et al.* [2009] expand the concept of stacked layers, first introduced for erosion simulation, by incorporating water and air layers and thereby enabling caves and overhangs. A limited set of sphere and generalized cylinder tools are available for users to carve away and accrete material. Although more compact than voxels, the grid of material stacks is again stored explicitly and the stabilization process introduced to spread unstable materials to neigbouring grid cells prevents alternating strata of materials.

All these techniques suffer to a greater or less extent from memory and authoring issues and, with some exceptions [Becher et al. 2018; Peytavie et al. 2009], tend to be restricted in application to: a single global function [Gamito and Musgrave 2001], fracturing effects [Ito et al. 2003], or weathering of small isolated structures [Beardall et al. 2007; Jones et al. 2010].

In contrast, our implicit model efficiently combines $2\frac{1}{2}$D and volumetric information, which allows a bounded memory footprint and thus the ability to model far larger scenes. We also provide amplification processes and authoring tools at various levels of abstraction, enabling the creation of a wide variety of 3D landforms, including, but not restricted to, cave networks, hoodoos, canyons, stratified overhanging cliffs, and karsts.

3 OVERVIEW

This section provides an overview of the implicit construction trees that form the basis for the geology and implicit terrain models central to our technique. This is followed by a presentation of the workflow for generating 3D terrain features (see Figure 2).

3.1 Construction Tree Models

Two structures are central to our 3D amplification of terrains: a geology model $\mathcal{G}$ for compactly encoding the stratification characteristics of the bedrock, and an implicit terrain model $\mathcal{T}$, which defines the surface and captures complex volumetric landforms. Both are variants of hierarchical implicit construction trees with leaves that are implicit primitives and internal nodes that are combining operators. A depth-first walk of such a tree is equivalent to a function evaluation for a given 3D point in the domain. Crucially, these implicit construction trees enable a representation of volumetric data with a compact memory footprint.

In the case of geology, leaves of the construction tree are implicit skeletal primitives that define rock resistance for every point in space. The internal nodes are either binary operators for combining sub-trees or unary operators for reproducing folds and faults using various forms of warping. In effect, the geology tree provides a resistance function, denoted as ρ.

For terrain $\mathcal{T}$, the leaves are implicit shapes hierarchically combined to create specific geomorphological features (e.g., hoodoos, caves, and tunnels) and ultimately merged with the overall terrain using blending, carving and warping operators. The corresponding terrain field function is denoted as f.

3.2 Amplification Workflow

The stages of our amplification process are depicted in Figure 2. To begin with, we automatically convert the representation from a $2\frac{1}{2}$D heightfield $\mathcal{H}$, provided as input, to an implicit 3D terrain model $\mathcal{T}$ (Section 5). In this implicitization step care is taken to ensure that the implicit surface of $\mathcal{T}$ accurately embeds the surface of the initial heightfield $\mathcal{H}$. This is coupled with a geology construction tree $\mathcal{G}$, which defines bedrock resistance in the form of strata and fault lines.

This combined representation (Terrain and Geology) is amenable to various 3D modifications, such as blending and carving. Specifically, we augment $\mathcal{T}$ with 3D landforms encoded as sub-trees that are attached to and hence modify the construction tree of the terrain $\mathcal{T}$. Those landforms are generated at the most interesting locations, in our case where rock resistance $\rho(\mathbf{p})$ is low. During an authoring session, the user can choose from a library of geology and effect archetypes defined as pre-constructed or procedurally generated construction trees. Alternatively, they can manually edit the geology construction tree $\mathcal{G}$ by locally adjusting bedrock resistance,

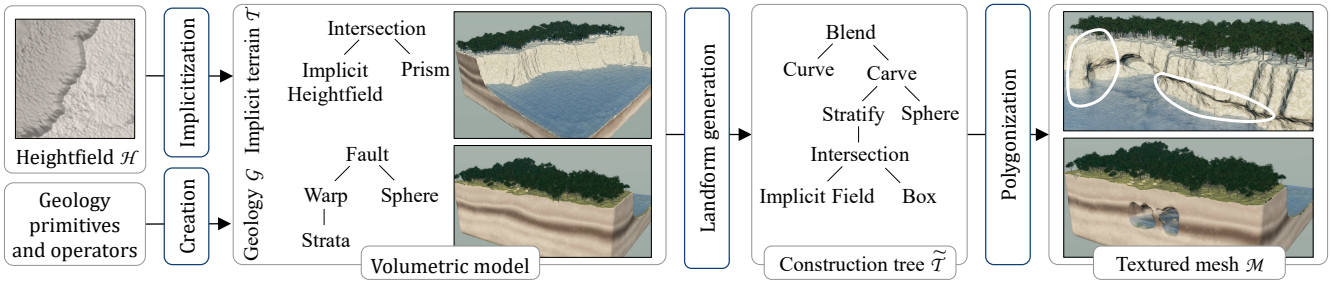


Fig. 2. Overview of our terrain amplification: Starting from a $2\frac{1}{2}$D heightfield $\mathcal{H}$, we first perform an implicitization process to create an implicit terrain model $\mathcal{T}$ suitable for 3D augmentation. At the same time, a model of the underlying geology $\mathcal{G}$ is created by the user. Next, a landform generation process converts $\mathcal{T}$ into an augmented construction tree $\tilde{\mathcal{T}}$ with sparse 3D features where required. Efficient polygonization is then used to extract a final mesh.

incorporating faults and folds, or re-weighting specific local erosion effects (see Section 6).

Ultimately, the amplified terrain is polygonized or rendered directly using ray-tracing. In this regard, we accelerated the polygonization of our volumetric implicit surface-based terrains by exploiting the compact support of volumetric landforms (Section 7).

Our framework incorporates multiple levels of user control: the geology and the parameters of the erosive agent (such as the sea level or the stream power) can be edited, the sampling process steered, and new features added. We also provide real-time authoring tools in the form of volumetric brushes that can be applied directly to the terrain.

As an illustration of a typical workflow (see Figure 2), a shore with undercut sea cliffs and caves could be created by deriving $\mathcal{T}$ from an input $2\frac{1}{2}$D elevation model, adding a stratified construction tree with layers of hard and soft rock for geology, and applying an erosion process according to a user-supplied control field $\mathcal{V}$ and the characteristics of the terrain. This process would probabilistically influence the inclusion of subtractive spheroidal erosion primitives in the terrain construction tree leading to overhangs and caves.

3.3 Notation

Throughout this paper, terrains and 3D landforms are created procedurally by building on atomic functions. We rely extensively on 3D simplex noise functions, denoted as $n : \mathbb{R}^3 \to [0, 1]$, and combine them into a fractal Brownian motion function t, defined as a sum of scaled simplex noise n over o octaves:

$$t(\mathbf{p}) = \sum_{k=0}^{k=o} \frac{1}{2^k} n(2^k\,\mathbf{p}).$$

We also use warping functions to distort the terrain surface by deforming space in a neighborhood. Such a warp ω is a homeomorphic mapping $\omega : \mathbb{R}^3 \to \mathbb{R}^3$. In implicit modeling, the field function of a warped surface is defined as $\widetilde{f} = f \circ \omega^{-1}$.

Finally, g represents a compactly supported C^2 continuous falloff function based on distance r to some geometry of interest, and parameterized by a radius of influence R:

$$g(r) = \begin{cases} \left(1 - (r/R)^2\right)^3 & \text{if } r < R, \\ 0 & \text{otherwise.} \end{cases}$$

4 GEOLOGY MODEL

In nature, landforms, such as karsts, cliffs and overhangs, are controlled not only by geomorphological processes, but also by the structure of the underlying geology. This defines the rock type of the different strata, and deformations, such as folds and faults. These bedrock characteristics lead to differentiated erosion rates, which can give rise to complex formations, such as arches and hoodoos.

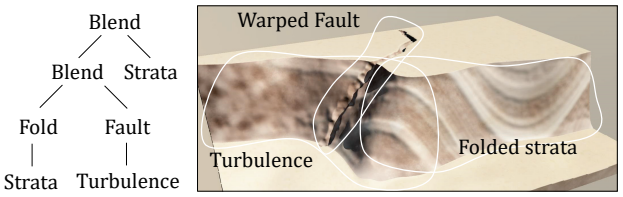


Fig. 3. An example of the hierarchical construction of a complex geological structure. Horizontal strata representing rock layers at different consistency are folded by warping (on the right) and this is separated off by a fault line from a turbulence function (on the left). Blend nodes combine the subtrees.

In our system, these geological characteristics are defined as a procedural field function $\rho : \mathbb{R}^3 \to [0, 1]$ that characterizes the strength with which the bedrock resists erosion at any point in space. The least and most resistant bedrock have resistance values of 0 and 1, respectively. Depending on requirements this function may be locally continuous (in the case of folds and warps) or discontinuous (in the case of faults).

We implement the resistance function as a hierarchical construction tree (Figure 3), with internal nodes that modify or combine resistance values spatially demarcated by the leaf node primitives. We have created several specific primitives and warping operators in order to effectively model bedrock strata.

4.1 Turbulence Primitives

Turbulence primitives are often used as a basis for more complex geology trees. Let λ_0 denotes the fundamental wavelength, the resistance is then defined as a function of elevation:

$$\rho(\mathbf{p}) = t(\mathbf{p}_z/\lambda_0).$$

This creates a set of horizontal strata whose resistances are defined by the turbulence function t. Figure 3 showcases two kinds of turbulence primitives: noise on the left and a strata obtained from a turbulence combined with a fold on the right.

4.2 Plane Primitives

Turbulence primitives allows us to create stratified geology easily but lack user control. We extend the geological model by introducing plane primitives, defined by the distance from a planar skeleton primitive, which acts as a central core (Figure 4 [*Strata*]). Let d denote the signed distance to the plane, g the falloff function and λ_0 the fundamental wavelength, then the resistance is:

$$\rho(\mathbf{p}) = g \circ d(\mathbf{p}) + t(\mathbf{p}/\lambda_0).$$

Some scaled turbulence t is added to the potential field to approximate irregularities, such as small fractures and joints that reduce rock durability.

In most of our scenes, the geology tree was first created by blending multiple plane primitives with a turbulence primitive. As in implicit modeling [Wyvill et al. 1999], blended resistance is defined as the sum of the resistance of the sub-trees: $\rho_{A+B} = \rho_A + \rho_B$

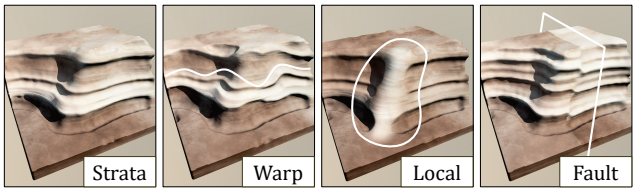


Fig. 4. Different forms of geology showcased on a simple cliff terrain : simple strata combined with noise, folds produced by a warping operator, a local increase in bedrock resistance produced by sphere primitives, and a fault line. Pale colors map to more resistant and darker to less resistant bedrock, respectively. Note that we applied a small erosion to the cliff to visually differentiate the strata.

In addition, we improve user control with skeletal primitives (spheres and curves blended with the construction tree) that locally modify bedrock resistance (Figure 4 [*Local*]). These are particularly useful for defining more resistant spatial regions that retard erosion and form promontories, or, in contrast, less durable regions leading to caves or arches (Section 6).

4.3 Fold and Deformation Operators

Folds and deformations (see Figure 4 [*Warp*]) contribute vital realism to geological strata patterns. They are defined as warping operators $\omega : \mathbb{R}^3 \to \mathbb{R}^3$ that deform space. Recall that, as in implicit modeling, the modified field function of a warped sub-tree is defined as: $\widetilde{f} = f \circ \omega^{-1}$, where $\omega^{-1}(\mathbf{p}) = \mathbf{p} + \delta(\mathbf{p})$ and δ denotes the displacement function.

In our system, random folds are introduced using a 3D turbulence function as displacement: $\delta(\mathbf{p}) = t(\mathbf{p}/\lambda_0)$, with λ_0 being the fundamental wavelength of the turbulence. Another useful deformation operator is tapering, which can be applied to locally compress strata.

4.4 Faulting Operators

Faults are generated by introducing discontinuities in the resistance function on the boundary of a given domain. Let $\Omega_{\mathcal{F}} \subset \mathbb{R}^3$ be such a domain and $\omega_{\mathcal{F}} : \mathbb{R}^3 \to \mathbb{R}^3$ an associated warping function. Given an input resistance function ρ, faults are created along the boundary of the domain $\partial\mathcal{F}$ by warping ρ strictly inside $\Omega_{\mathcal{F}}$:

$$\rho_{\mathcal{F}}(\mathbf{p}) = \begin{cases} \rho \circ \omega_{\mathcal{F}}^{-1}(\mathbf{p}) & \text{if } \mathbf{p} \in \Omega_{\mathcal{F}}, \\ \rho(\mathbf{p}) & \text{otherwise.} \end{cases}$$

Figure 4[*Fault*] shows an example of a fault created with a planar boundary and a translational warp; this results in a discontinuity in the resistance function, which in turn yields sheared strata.

5 IMPLICIT TERRAIN MODEL

Our terrain model $\mathcal{T}$ is based on the same underlying hierarchical construction tree as the geology model. The difference is that primitives and their subtree aggregations portray landforms with a compact volumetric support, such as hoodoos (Figure 5), rather than strata and bedrock density. Crucially, the model must also be amenable to the extraction of a final mesh surface. To achieve this, we associate a field function $f : \mathbb{R}^3 \to \mathbb{R}$ with $\mathcal{T}$ defining the intensity of a given position in space. The surface of the terrain is the set of points where the field function equals a user-defined threshold value T:

$$S = \{\mathbf{p} \in \mathbb{R}^3, f(\mathbf{p}) = T\}.$$

The value of f at a point $\mathbf{p}$ is computed by a depth-first traversal of the construction tree with evaluation of the potential field at each visited node.

Fig. 5. In this example the hoodoos were created by blending several perturbed sphere and cone primitives, and merging with the ground. Creases and cracks were added by a using warping operator.

5.1 Implicitization of Input Terrains

A heightfield is an unworkable format in the context of 3D landforms. Consequently, transforming input $2\frac{1}{2}$D terrains into our implicit 3D construction tree representation is a necessary precursor to any volumetric operations (Figure 6). The challenge is to derive a compactly supported function $f : \mathbb{R}^3 \to \mathbb{R}$ from $\mathcal{H}$, such that the implicit surface $\mathcal{T}$ taken at a particular field value T embeds $\mathcal{H}$. A bounding prism and a compact field function are required so that the resulting field function falls off to 0 at the border of the domain. This property guarantees that terrain primitives have no influence on the field function beyond a user-controlled distance threshold. This is particularly important when blending multiples terrains (see the case of the floating islands in Figure 21) or when carving caves or arches deep into the bedrock (see Figure 14).

This conversion is achieved by assembling a subtree $\mathcal{T}$ with three nodes: 1) a vertical prism primitive delimiting the extent of the terrain Ω both horizontally and vertically. It has a field function $f_{\mathcal{P}}$ that is equal to T within the prism and tails off to zero outside; 2) a surface primitive, with a field function $f_{\mathcal{T}}$ whose value is related to distance from the input heightfield and attains a value of T when this distance is zero; 3) an intersection node that combines $f_{\mathcal{P}}$ and $f_{\mathcal{T}}$ to produce f.

The shape of the input terrain is therefore preserved, but in an implicit form using a field function that combines three functions: 1) a bounding function $f_{\mathcal{P}}$, 2) a surface distance $f_{\mathcal{T}}$ function, and 3) a falloff function that guarantees that f should have a compact support. The definition of a prism-based region of influence extending beyond and surrounding the initial terrain $\mathcal{H}$ is essential for combining in subsequent landforms primitives, for instance when using the smooth blending operator. Using a domain Ω of arbitrary shape allows us to create sections of volumetric terrain with complex horizontal support (Figure 7), which can be combined as illustrated in Figure 21.

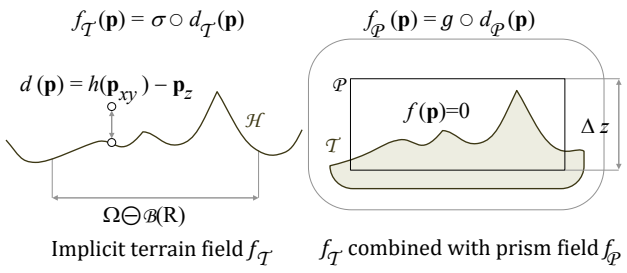


Fig. 6. Heightfield implicitization: given a C^1 elevation function h over a 2D domain $\Omega \subset \mathbb{R}^2$, we construct a 3D primitive with field function f such that the implicit surface $S = \{\mathbf{p} \mid f(\mathbf{p}) = T\}$ matches $\mathcal{H}$.

In more detail: for the prism, we define an enclosing domain as the Minkowski difference $\mathcal{P} = (\Omega \ominus \mathcal{B}(R)) \times \Delta z$, where Ω denotes the compact support of the heightfield elevation function h, $\mathcal{B}(R)$ is a disc centered at the origin with radius equal to the falloff R, and Δz denotes the range of elevations over Ω. The field function of the corresponding prism primitive is defined as: $f_{\mathcal{P}} = g \circ d(\mathbf{p}, \mathcal{P})$. The radius of influence R defines the extent of the potential field inside and outside the initial terrain surface skeleton $\mathcal{H}$.

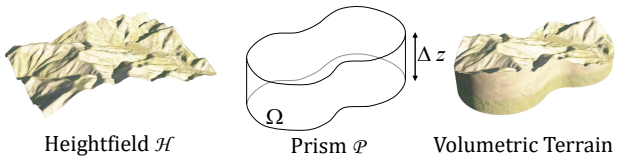


Fig. 7. Our implicitization process allows derivation of a consistent implicit field function representation from an input heightfield and a domain $\Omega \subset \mathbb{R}^2$.

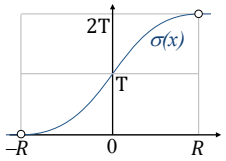
For the surface primitive, we seek to construct a field function $f_{\mathcal{T}}$ such that the extracted implicit surface $S = \{\mathbf{p} \mid f(\mathbf{p}) = T\}$ at field value T embeds $\mathcal{H}$. Let $h(\mathbf{p})$ denote the elevation function of the input terrain $\mathcal{H}$ over domain $\Omega \in \mathbb{R}^2$ and $v(\mathbf{p}) = h(\mathbf{p}_{xy}) - \mathbf{p}_z$ the signed vertical distance to the surface. We define $f_{\mathcal{T}}$ over domain $\Omega \times \mathbb{R}$ as:

$$f_{\mathcal{T}}(\mathbf{p}) = \sigma \circ v(\mathbf{p})$$

The function σ is a compactly supported sigmoid-like attenuation function that limits the range of $f_{\mathcal{T}}$ to $[0, 2T]$. We construct it as a piecewise odd cubic function (*i.e.*, $\sigma(-x) = -\sigma(x)$) satisfying the constraints: $\sigma(0) = 0$, $\sigma'(0) = 1$, $\sigma(R) = 2T$, $\sigma'(R) = 0$:

$$\sigma(x) = \begin{cases} T + x + \dfrac{3T - 2R}{R^2} x^2 + \dfrac{R - 2T}{R^3} x^3 & \text{if } 0 < x < R, \\ 2T & \text{otherwise.} \end{cases}$$

The function σ limits the range of the terrain field function $f_{\mathcal{T}}$ to $[0, 2T]$, with a minimum of 0 beyond the prescribed distance R, and a maximum of $2T$ at a distance of R or greater beneath the terrain surface. This is important to allow control when sculpting the implicit terrain with other primitives: the shape of the surface of the terrain no longer influences the field function beyond R. Finally, the terrain field function f is obtained by intersecting $f_{\mathcal{T}}$ and $f_{\mathcal{P}}$:



$$f = \min(f_{\mathcal{T}}, f_{\mathcal{P}}).$$

This subtree provides a C^1 compactly supported volumetric terrain primitive, such that $\mathcal{H} \subset \mathcal{T}$ over the domain $\Omega \ominus \mathcal{B}(R)$. Because such terrain subtrees are fully consistent with other volumetric primitives, they can be combined and sculpted just as readily. As an illustration, terrain implicitization makes it possible to fashion floating islands simply by intersecting two terrains, one of which is inverted with respect to the other (see Figure 1). Note that our representation is compatible with any type of $2\frac{1}{2}$D terrain, either in the form of DEM data or function-based models, such as procedurally generated terrains [Ebert et al. 1998] or construction trees [Génevaux et al. 2015].

5.2 Sculpting Primitives

In our system, we augment an implicit terrain model T with sculpting primitives to create diverse volumetric features, such as arches and caves. Sculpting primitives are controlled by a geometric skeleton $\mathcal{S}$ and a surrounding spatial density function f, which can be written as a composition $f = g \circ d$ of the falloff function g (with a radius parameter R controlling the extent of influence) and the Euclidean distance $d(\mathbf{p}, \mathcal{S})$ to the skeleton $\mathcal{S}$. For example, the simple and computationally efficient point-based primitive uses the Euclidean distance $d(\mathbf{p}) = \|\mathbf{c} - \mathbf{p}\|$ to its skeletal center point $\mathbf{c}$.

Skeletal primitives come in two broad classes: infinitely thin primitives, built around points, line segments, and curves [Wyvill et al. 1999], where the field tails off directly from the skeleton, and volumetric primitives, such as spheres, cones, boxes, ellipsoids, and more complex geometry [Barbier and Galin 2004], which have a constant field within their volume and only fall off from the boundary (see Figure 8). The former are helpful in delineating linear features and

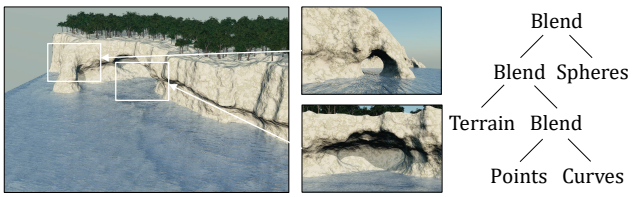


Fig. 8. We use sphere primitives with positive energy to sculpt the terrain and create arches. Negative skeletal primitives such as points are used to model caves and deep overhangs.

locations, such as stratification and point erosion, while the latter are effective for carving arches and karst structures, as described in Section 6.

One limitation of basing primitives on Euclidean distance is that it leads to smooth rounded shapes, which do not match the irregularities inherent in rocky surfaces. There is thus a need for suitably perturbed skeletal primitives. However, simply adding noise to f (i.e., placing a noise node at the root of the construction tree) often introduces unwanted holes and disconnected surface components, and removing such artefacts is computationally expensive [Gamito and Maddock 2008].

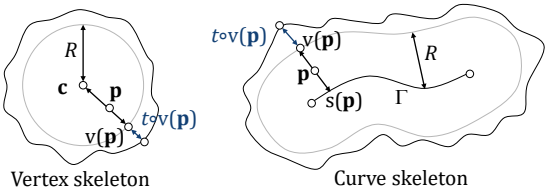


Fig. 9. Skeletal primitives with anisotropic star-shaped noise displacement for point $\mathbf{c}$ and curve Γ skeletons.

Rather, in the spirit of Crespin *et al.* [1996], we convert Euclidean distance into an anisotropic metric by deforming the area of influence parameter R with turbulence $t(\mathbf{p})$. Let $s : \mathbb{R}^3 \to \mathbb{R}^3$ denote the projection of $\mathbf{p}$ onto an arbitrary skeleton. We compute the normalized projection direction $u(\mathbf{p})$:

$$u(\mathbf{p}) = \frac{\mathbf{p} - s(\mathbf{p})}{\|\mathbf{p} - s(\mathbf{p})\|}.$$

The projection $v(\mathbf{p})$ of $\mathbf{p}$ onto the boundary of the primitive is then defined as: $v(\mathbf{p}) = s(\mathbf{p}) + R\,u(\mathbf{p})$. Finally, the modified anisotropic distance to the skeleton is:

$$\widetilde{d}(\mathbf{p}) = \frac{\|\mathbf{p} - s(\mathbf{p})\|}{R + t \circ v(\mathbf{p})}.$$

This method can be used to perturb the shape of any skeleton without artefacts, as illustrated in Figure 9 for the case of point-based and curve-based anisotropic star-convex primitives.

5.3 Operators

Operators are internal nodes that combine sub-trees. Our model implements blending and boolean operators as described by Wyvil *et al.* [1999]. Recall that blending, denoted as $B(\mathcal{N}_A, \mathcal{N}_B)$ defines intensity as the sum of the intensities of the two sub-trees: $f_{A+B} = f_A + f_B$, whereas union and intersection are defined by computing the minimum and maximum field values of the sub-trees.

Our model includes warping operators that distort the shape of the implicit surface by deforming the surrounding space. Such nodes are useful for generating fine details over extended regions of space.

To enhance control we localise deformations by limiting their influence to a compact domain $\mathcal{D}$. Let $f_{\mathcal{D}} : \mathbb{R}^3 \to [0,1]$ denote a compactly supported field function over $\mathcal{D}$, which in this case characterizes the magnitude of warping applied to a point $\mathbf{p}$. We incorporate this into the warping node as:

$$\widetilde{f}(\mathbf{p}) = f \circ \omega^{-1}(\mathbf{p}) \qquad \omega^{-1}(\mathbf{p}) = \mathbf{p} + \delta(\mathbf{p})\, f_{\mathcal{D}}(\mathbf{p}).$$

The term $\delta(\mathbf{p})$ is a turbulence-based displacement whose influence is weighted by the control field $f_{\mathcal{D}}(\mathbf{p})$. Note that such localising field functions have broader applicability.

In general, warping nodes are complementary to sculpting and landform generation (described in Section 6). As proposed by Gamito and Musgrave [2001] they can be used to augment terrain with coarse features, such as large overhangs or broad relief.

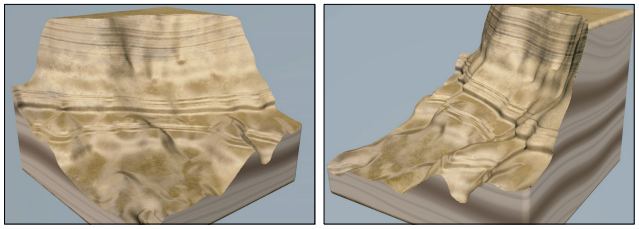


Fig. 10. A cliff amplified with an erosion operator over the terrain construction tree. Vertical cut planes show the geology tree $\mathcal{G}$ composed of warped high-frequency noise blended with three less resistant strata primitives. This operator allows us to represent precise stratification, creating overhangs and fine detail without resorting to thousands of primitives.

6 LANDFORM GENERATION

In order to account for widespread stratification of a scene without resorting to a huge number of primitives, we define an efficient erosion node over a controllable region $\mathcal{D}$ that draws on the geology model $\mathcal{G}$. Recall that the resistance function ρ of $\mathcal{G}$ generates values in the unit interval $[0,1]$ to reflect the durability of bedrock strata.

Our erosion node subtracts from the field function of a sub-tree f based on ρ, to produce:

$$\widetilde{f}(\mathbf{p}) = f(\mathbf{p}) - f_{\mathcal{D}}(1 - \rho(\mathbf{p})).$$

The cliff in Figure 10 as well as the strata of the floating islands in Figure 21 were created using this operator.

One notable limitation of such erosion nodes is that they relie on noise function evaluation in the geology tree, which can be difficult to configure and control. In the next section, we present more sophisticated approaches for building construction trees to amplify 3D detail and control the location of landforms.

A hierarchical implicit construction tree typically holds thousands of primitives, each of which in theory requires a costly field function evaluation involving cubic functions and turbulence. The

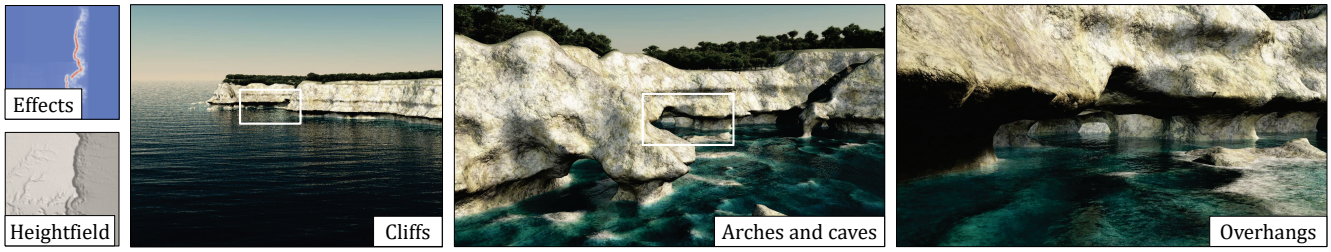


Fig. 11. An example of sea cliffs produced by our system. Starting from an input synthetic $2\frac{1}{2}$D heightfield and geology with horizontal strata, we incrementally applied 3 steps of sea erosion. Sea action was limited to an 8-meter range either side of average sea level, leading to strong overhangs at the base of the cliff. Less durable rock area was specified in the geology model, which automatically generated the arch and sea cave. The effects inset shows the repartition of volumetric features on the terrain from a top view perspective.

workaround is to impose a bounding volume hierarchy, which enables ≈ 1 million queries per second even with construction trees composed of several thousand primitives.

3D landforms are the result of complex erosion processes involving the shape of the terrain, its geology, and the action of environmental erosive agents. Simulating those phenomena would be computationally intensive and prevent interactive control. We thus avoid physically-based simulation and instead, our phenomenological approach augments a $2\frac{1}{2}$D input terrain with 3D landforms by using controllable and efficient procedural techniques.

Generally, our landforms generation algorithms proceed in two phases (Figure 12). First, given an input $2\frac{1}{2}$D terrain $\mathcal{H}$ and user-defined geology $\mathcal{G}$, we compute the intensity of the erosion over the terrain (as factors for stress and resistance) by taking into account terrain shape, geological structure and environment conditions. Second, we generate a construction tree $\tilde{\mathcal{T}}$ for the different features. Erosion processes spawn sphere primitives with a negative energy that are blended with the terrain. In contrast, hoodoo and goblin generation accretes spheres and cone primitives in an additive growth process.

Our method is capable of generating a vast array of landforms, such as sea cliffs produced by coastal erosion, overhangs caused by river stream erosion, or caves carved by water flowing into porous rock.

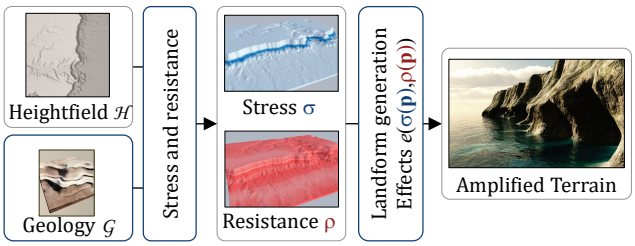
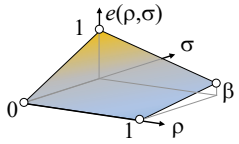


Fig. 12. An overview of our landforms generation pipeline.

6.1 Shallow Procedural Erosion

Shallow procedural erosion encompasses erosion processes that impact the terrain to a limited depth. Following the general template for landform generation, we proceed in two steps: we first perform Poisson-Sphere sampling in the erosion region to generate a set of points $\{\mathbf{p}_k\}$. Then, at every point $\mathbf{p}_k$, we locate sphere primitives with a negative energy derived from the erosion intensity at that location $e(\mathbf{p}_k)$. The user may specify the bounds on the erosion region through bounding volumes or an altitude range.



The effect intensity e at a point is determined by the geology $\mathcal{G}$, the shape of the terrain $\mathcal{H}$, and the erosion action. More precisely, we define a parameterized function $e : [0,1]^2 \rightarrow [0,1]$ that computes erosion according to the resistance of the rock ρ and the effect stress σ (such as sea elevation range, shown in Figure 12). In our implementation, e is a bi-linear interpolation of these quantities:

$$e(\rho,\sigma) = \sigma(1-\rho)(1-\beta) + \sigma\beta.$$

This obeys the constraint that $e(\rho,0) = 0$, namely that there can be no erosion effect without the rock being under stress. The parameter β controls the erosion intensity for the case where erosion is at a maximum and the material is highly resistant, *i.e.*, $e(1,1) = \beta$. This accords with the intuition that erosion will be stronger for less durable rock under high stress, whereas areas with little stress will not be eroded at all.

The energy of sphere primitives is proportional to $e(\rho(\mathbf{p}_k),\sigma(\mathbf{p}_k))$. Note that we discard samples with energy below a user-defined threshold, since the associated primitives would have negligible influence.

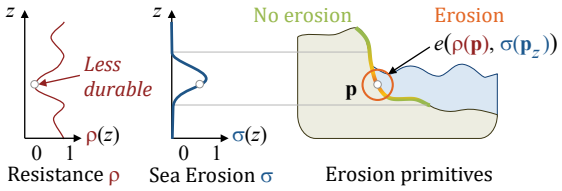


Fig. 13. Sea erosion impact e combines geology resistance ρ, sea erosion stress σ and accessibility (not illustrated). Sphere primitives are seeded over the surface with a negative energy derived from $e(\rho(\mathbf{p}),\sigma(\mathbf{p}_z))$.

Sea cliffs and arches. Formation of these features is dominated by the erosive action of the sea on coastal geology (refer to Figure 11).

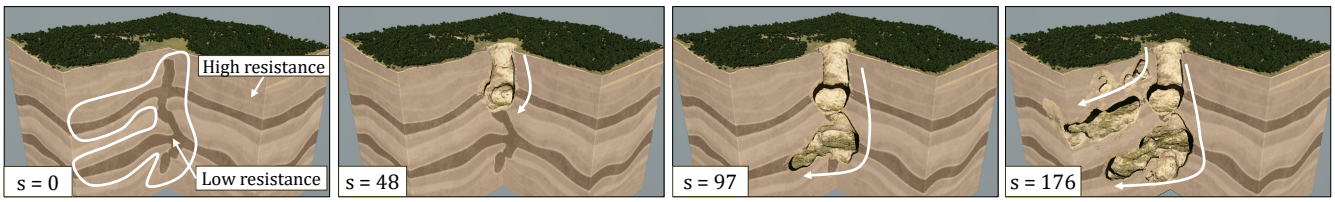


Fig. 14. Four steps in the generation of a cave system using a modified Invasion-Percolation algorithm. Starting from three sinks in the $2\frac{1}{2}$D heightfield, invasion percolation progressively carves the subsurface of the terrain by following the least resistant layers of bedrock. Poisson Sphere sampling allows the creation of multiple tunnels.

Sample points $\mathbf{p}_k$ are generated on the initial terrain around sea level (Figure 13). The stress of sea waves is approximated by combining a falloff distance from sea level w with local coastal accessibility α, as defined by Miller [1994]:

$$\sigma(\mathbf{p}) = w(\mathbf{p})\,\alpha(\mathbf{p}).$$

River canyons and gorges. The erosive action of strong rivers in narrow confines often creates scenic overhangs. One option for computing water impact is to run a fluid simulation on the 3D terrain and record the energy with which particles impact the canyon walls. This is a computationally costly prospect, so we approximate this flow impact by computing the stream power of the heightfield $\mathcal{H}$ [Cordonnier et al. 2016]. Let $A(\mathbf{p})$ denote the upstream area of a point $\mathbf{p}$ and $s(\mathbf{p})$ the average slope, then:

$$\sigma(\mathbf{p}) = A^{1/2}(\mathbf{p})\,s(\mathbf{p})$$

Our implementation uses the multiple flow model of Freeman [1991] to calculate drainage area in a manner that accounts for possibly divergent flow. Note that we apply a depression-filling algorithm [Barnes et al. 2014] beforehand to circumvent the possibility of local sinks in $\mathcal{H}$. We finally identify the riverbed region as the points on the terrain whose σ is greater than a user-prescribed threshold. We also use a small convolution to extend the influence of the riverbed to the banks, to account for the impact of flooding.

The erosion effect is again computed as $e(\rho(\mathbf{p}), \sigma(\mathbf{p}))$. Figure 15 shows a comparison between an original $2\frac{1}{2}$D terrain and the result of our amplification process.

6.2 Deep Procedural Erosion

Karst topography leads to caves and sinkholes through the dissolution of soluble rock, such as limestone and gypsum. Below ground they encompass complex drainage systems and networks with underground rivers and caves. On the surface, they are characterized by sinkholes and resurgence points.

We propose an original method for generating karsts, taking our inspiration from *invasion percolation* simulation [Wilkinson and Willemsen 1983]. This is a simplified physical model that simulates the pore-by-pore advancement of a fluid in a porous material when the flow is slow enough that viscosity effects can be neglected. Starting from a set of initial seed points, the algorithm updates a queue Q of candidates ordered by decreasing material resistance, and progressively advances in the direction of the least resistant material, adding new candidates to the queue as the fluid percolates into the material.

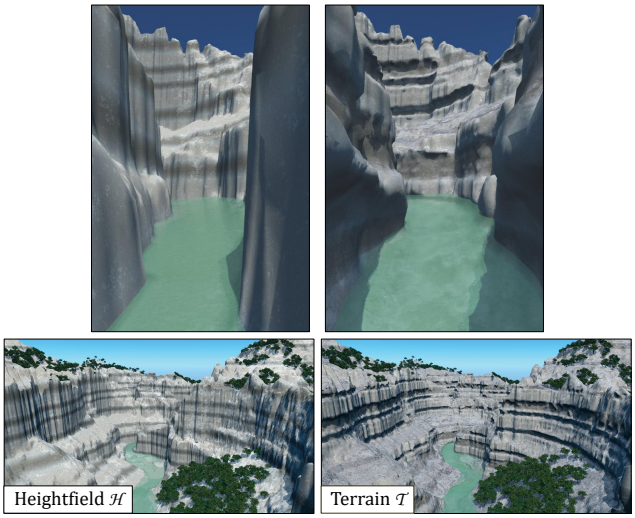


Fig. 15. A comparison between raw data (with a resolution of 1m per pixel) of The Mystery Canyon in the Zion National Park, Utah, and the outcome of our amplification process. Volumetric features occupy 20% of the scene's surface area.

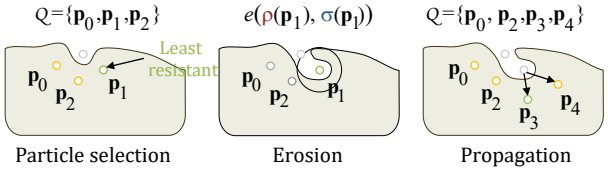


Fig. 16. Overview of our modified invasion percolation algorithm: after selecting the candidate point with least resistance ρ, the terrain is carved by generating a negative sphere primitive and new candidate points are added to the queue.

First, the queue of candidate points Q is initialized with the sinkholes of the input terrain. In our implementation, they can be found automatically by computing the sinks, also referred to as the pits [Barnes et al. 2014], of the drainage area, *i.e.* the cells in the grid for which all neighbours have a higher elevation. The user may also freely add additional resurgence points or sinks in order to adjust the scene.

Fig. 17. This example showcases a complex topography with sinkholes, tunnels and caves formed by our invasion percolation algorithm. The portion of terrain extends over 5.2×5.2km and the underground network of tunnels covers 2% of the surface. 165 773 sphere primitives were generated to create the caves.

While the queue Q has candidate points whose resistance is below a user-defined threshold, we perform the following steps (Figure 16): 1) Find the point $\mathbf{p}_k$ in Q with the least resistance $\rho(\mathbf{p}_k)$ and remove it from Q; 2) Locally carve the bedrock by blending the terrain with primitives with negative energy; 3) Propagate percolation by finding new points in the lower hemisphere at $\mathbf{p}_k$ and add them to Q. These steps are repeated until Q is empty.

In the original *invasion percolation* algorithm, step 1) is deterministic, always de-queuing the point with the least resistance. In our implementation, we slightly perturb the resistance by a random factor, whose range ε is controlled by the user. We set $\varepsilon \approx 0.1$ to allow for more randomness in the selection of candidates, and, consequently, in the shape of the generated networks. The second step carves the terrain only if the rock is sufficiently soft, specifically where $\rho(\mathbf{p}_k) < \rho_0$, with ρ_0 as a user-defined threshold. We modify the energy of the primitive according to the erosion effect, taking into account the stress and rock resistance $e(\rho, \sigma)$ (see Section 6.1). The third step generates new erosion directions. Since we approximate water infiltrating porous stone, we sample a set of random directions on an inverted hemisphere to account for the fact that water flows downwards. New samples are added to the queue Q only if their Poisson sphere does not intersect other candidates in the queue.

Our experiments demonstrate that tunnels extend organically and consistently with the geology of the terrain. Figure 14 illustrates this phenomenon: we used a set of parallel horizontal strata with some turbulence to produce layered and connected tunnel structures. Figure 17 shows an example where sinks were computed automatically on the plateau, leading to a complex set of tunnels emerging on the cliff.

6.3 Hoodoos and Goblins

Hoodoos are tall spires of rock that protrude from the base of arid basins and broken land. Their height varies from a few meters to more than 40 meters and their formation is the result of both frost wedging and rain.

A well-known location for hoodoos is the Bryce Canyon National Park, but they can be found elsewhere as well.

Creating such features using a physically-based approach would require an unreasonable number of erosion iterations. Therefore, we propose a procedural approach based on an open grammar method, inspired by the grammars introduced in plant modeling [Měch and Prusinkiewicz 1996]. The two-step algorithm is as follows: 1) We compute the probabilistic location of hoodoos according to drainage area, average slope and a prescribed user-mask; 2) We generate vertical hoodoo shapes with an open grammar, whose parameters are driven by the geology. The rules are based on the bedrock resistance function ρ: less durable bedrock will produce shapes differently to more durable bedrock.

Fig. 18. Examples of hoodoos generated with different symbols and production rules.

Figure 18 shows different types of hoodoos produced by our grammar rules.

Fig. 19. A more complex scene constructed with multiple hoodoo blocks.

Production rules		Probabilities	Terminal Symbols
$A(\mathbf{p}, s) \rightarrow B(\mathbf{p}, s)$		1	
$B(\mathbf{p}, s) \rightarrow \mathbf{b}(\mathbf{p}, s)$	$B(\mathbf{p} + s\,\mathbf{z}, \lambda\, s)$	$p_B(g(\mathbf{p}))$	
$B(\mathbf{p}, s) \rightarrow \mathbf{b}(\mathbf{p}, s)$	$C(\mathbf{p} + s\,\mathbf{z}, \lambda\, s)$	$1-p_B(g(\mathbf{p}))$	Box **d**
$C(\mathbf{p}, s) \rightarrow \mathbf{c}(\mathbf{p}, s)$	$B(\mathbf{p} + s\,\mathbf{z}, \lambda\, s)$	$\frac{1}{2}\, p_C(g(\mathbf{p}))$	Block **b**
$C(\mathbf{p}, s) \rightarrow \mathbf{c}(\mathbf{p}, s)$		$\frac{1}{2}\, p_C(g(\mathbf{p}))$	
$C(\mathbf{p}, s) \rightarrow \mathbf{d}(\mathbf{p}, s)$		$1-p_C(g(\mathbf{p}))$	Cone **c**

Fig. 20. Simplified production rules used in our open parameterized grammar to generate hoodoos and goblins. We represent non-terminal symbols with capital letters and their corresponding terminals in lowercase; A denotes the axiom of the grammar.

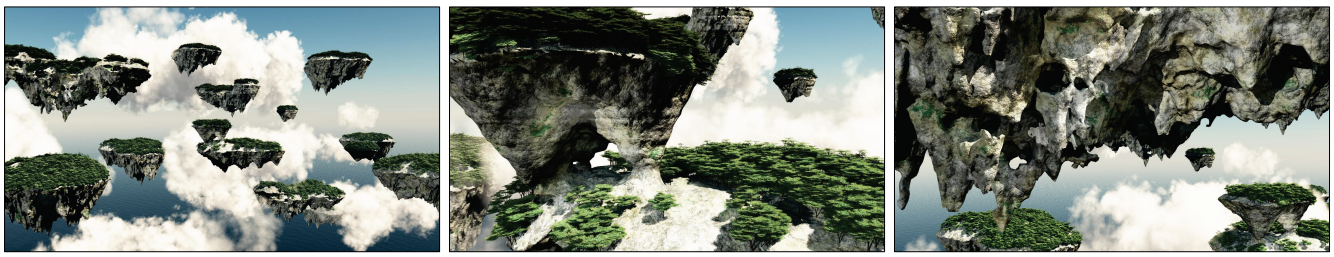


Fig. 21. The floating islands were created by combining implicitized heightfields; an erosion operator was then added to the construction tree to produce a precise stratification, and we finally carved some tunnels and caves into the bedrock of one island by applying the Invasion-Percolation algorithm. Individual islands are between 300m and 600m wide and were placed manually in the scene, for a total of 6mb in memory.

The probability of hoodoo growth is computed according to the drainage area A and the local slope s. Hoodoos are most likely to appear on talus or cliffs, which are characterized by a medium slope and low drainage area. We compute the probability of hoodoo growth by combining these criteria and then perform Poisson-disk sampling to generate starting positions, which will be fed as axioms to the grammar.

Hoodoos are created by assembling multiple terminal symbols using an open parameterized grammar method (see Figure 20). Our production rules are driven by the underlying geology, which impacts not only the probability but also the parameters of terminal symbols. We adapt the symbol size to the bedrock resistance $\rho(\mathbf{p})$. All the production rules start from an axiom A. We also add rotations to symbols to add variety to the generated shapes. Figure 19 shows a generated scene composed of multiple Hoodoo blocks.

7 POLYGONIZATION

At their core our terrains are implicit surfaces generated by evaluating an implicit 3D construction tree $\mathcal{T}$. Although implicit surface visualization can be achieved both directly using ray tracing, typically with interval arithmetic [Mitchell 1990] or Lipschitz [Hart 1996; Kalra and Barr 1989] techniques, and indirectly by first extracting a mesh [Lorensen and Cline 1987; Wyvill et al. 1986], doing so efficiently for highly-detailed terrains is challenging. Fortunately, in the case of an amplified terrain the volumetric carving and sculpting elements are bounded in extent and generally located on or below the input $2\frac{1}{2}$D terrain. This allows potential field queries $f(\mathbf{p})$ used for ray tracing and mesh extraction to be restricted to a spatial band, thereby reducing the number of field function evaluations.

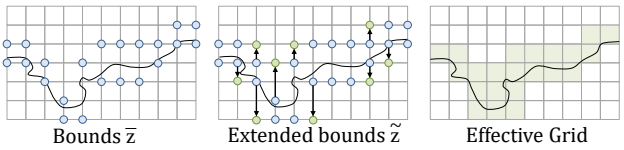


Fig. 22. A side-view summary of our polygonization algorithm: (1) We compute altitude bounds $\bar{z}$ for each grid vertex by querying the construction tree; (2) these bounds are dilated to ensure continuity in the output mesh, leading to extended bounds $\tilde{z}$, thereby (3) defining a minimal zone for cube traversal.

In our case, construction trees consist of thousands of complex skeletal primitives (Table 1), each requiring multiple cubic function evaluations. This makes ray tracing techniques less convenient than polygonization in the context of interactive editing, which is one advantage of our method. Therefore, we focus on polygonization techniques in this section. Note that while we frame subsequent presentation of our acceleration in terms of mesh extraction, the benefits also apply to ray casting where empty space skipping can be exploited [Kruger and Westermann 2003]. Our goal is to extract a C^0 surface from our implicit construction tree, and take advantage of the localized aspect of volumetric features.

It is useful to define a measure for the proportion of the domain occupied by 3D features. Let $n(\mathbf{p})$ denote the number of primitives whose vertical projection onto the ground plane encompasses the point $\mathbf{p} \in \mathbb{R}^2$. If the elevation has not been carved or sculpted and is determined solely by a heightfield primitive then $n(\mathbf{p}) = 1$, otherwise $n(\mathbf{p}) > 1$. The ratio of 3D coverage with respect to the domain is then defined as: $a = \widetilde{\mathcal{A}}/\mathcal{A}$, where $\widetilde{\mathcal{A}}$ denotes the area where $n(\mathbf{p}) > 1$ and $\mathcal{A}$ is the domain area. As Table 1 indicates for more extensive scenes (such as in Figures 11 and 17) this proportion tends to be small.

The original Marching Cubes algorithm extracts a mesh $\mathcal{M}$ from an implicit function f for values $f(\mathbf{p}) = T$. In our case, this would entail, given an input box $\mathcal{B}$ and a 3D virtual grid $\mathcal{G}$, querying the field function at every vertex $\mathbf{p}_{ijk}$ to extract the correct triangle configuration for cells in the grid. Fortunately, we can optimize surface extraction by leveraging the characteristics of the implicit 3D construction tree in two ways:

(1) *Surface Bounds.* We establish relatively tight bounds on the range of possible elevations and only process cubes, and hence query the field function $f(\mathbf{p})$, within this range.
(2) *Direct Elevation Extraction.* In regions unaffected by carving or sculpting (where $n(\mathbf{p}) = 1$) we derive vertex positions directly from the elevation function and avoid costly bisection search for $f(\mathbf{p}) = T$.

Surface Bounds. We prune the grid $\mathcal{G}$ to reduce the set of grid cells that require processing. Let $[a_{ij}, b_{ij}]$ denote an inclusive integer range representing the lower and upper elevation bounds for a given 2D vertex $\mathbf{p}_{ij}$ in the grid. To obtain these bounds we perform the following steps (see Figure 22):

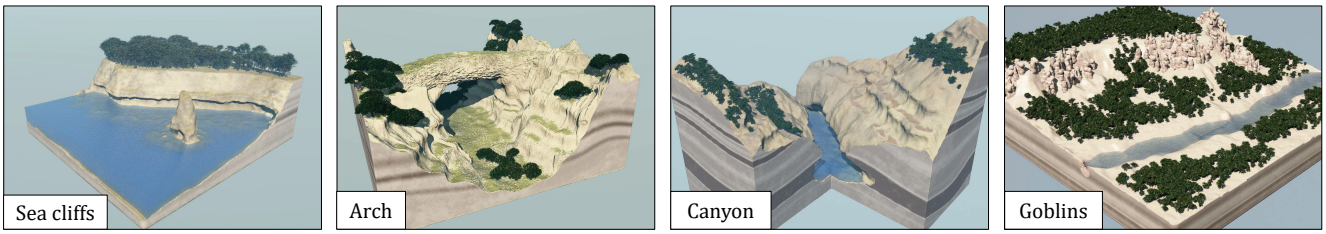


Fig. 23. Varied landforms generated using our implicit model on small terrains ($500 \times 500\,\mathrm{m}^2$): a coastal cliff with three sea erosion steps applied (left), a large arch forming a bridge (center-left), a canyon where river erosion has sculpted deep overhangs (center-right) and goblins placed along the banks of a river (right).

(1) The construction tree is queried to return elevation bounds $\bar{z} = [a_{ij}, b_{ij}]$ for each vertex $\mathbf{p}_{ij}$. This requires the definition of an $\mathbb{R}^2 \rightarrow \mathbb{Z}^2$ bounds function $\bar{z}_{ij} = \mathcal{B}(\mathbf{p}_{ij})$ that, for a given position, walks the tree to evaluate bounds on primitives and combine them using internal operators. For volumetric primitives, such as points and spheres, minimum and maximum altitude is based on the associated bounding box. For a heightfield primitive h a unique elevation $h(\mathbf{p}_{ij})$ is returned leading to equal upper and lower bounds after conversion in grid space: $a_{ij} = b_{ij}$. Next, binary operators such as carving or blending return the union of the bounds of their children z_1 and z_2: $\bar{z}_{ij} = \cup(\bar{z}_1, \bar{z}_2)$.
(2) To ensure C^0 continuity in the final mesh, we perform a dilatation of $\bar{z}$ in the 1-ring neighborhood of each $\mathbf{p}_{ij}$, leading to extended integer bounds $\tilde{z}$. Let V_{ij} denote the 1-ring neighborhood of $\mathbf{p}_{ij}$, then the dilated bound is: $\tilde{z}_{ij} = \cup_{(x,y)\in V_{ij}} \bar{z}_{xy}$. Intuitively, the dilated bound at a grid vertex represents the largest elevation range shared between itself and its neighbours.

The algorithm traverses (and selects triangle configurations for) the reduced subset of cells within the bounds specified by $\tilde{z}$. This leads to a speedup up to 12, depending on the proportion of volumetric features in the scene. Results and timings are reported in Table 1.

Direct Elevation Extraction. The intersection of a grid cell edge and the terrain surface is typically computed using bisection or Newton-Rhapson root finding, with repeated calls to the field function f. In $2\frac{1}{2}$D regions, there is no need for such iterative approaches since the elevation can be directly computed as $h(\mathbf{p}_{ij})$. Therefore, we approximate vertices using linear interpolation, which provides sufficient accuracy and results in a speedup ranging from 1.5 to 2.5.

Table 1 reports statistics for our visualization method. In particular, this shows the considerable reduction in computationally demanding field function calls ($\#C$ vs. $\#C_0$) and consequent acceleration by up to a factor of 12 (t vs. t_0).

Our optimized version profits from the localization of volumetric features. Thus the more widespread the volumetric features are the less comparatively efficient our approach becomes. This can be observed in the Benagil scene (Figure 25), where the ratio a is atypically high and the speedup is only 1.8. Thus, our improved version is most efficient in the context of realistic terrains with localized volumetric features.

8 RESULTS AND DISCUSSION

We implemented our system in C++. Experiments were performed on a desktop computer equipped with Intel® Core *i*7, clocked at 4 GHz with 16GB of RAM, and an NVidia GTX 970 graphics card. The output of our system was streamed into Vue Xstream® to produce photorealistic landscapes (Figures 1, 11, 15, 17, 21, 24, 25).

Our method is capable of generating a variety of landscapes amplified with local 3D features. Figures 11 and 8 show coastal cliffs procedurally eroded by sea, as well as complex features such as arches and caves created by interactive editing. Figures 17 and 14 depict procedural invasion-percolation simulation leading to the evolution of caves and tunnels deep below the surface. Figure 19 and 20 show hoodoos created with an open shape grammar based on the geology $\mathcal{G}$. Finally, Figure 21 demonstrates that capability of our framework in creating and authoring fantastical scenes.

8.1 Validation

Validation is a challenging issue for procedural methods. Real terrain data, with overhangs, cliffs, arches and karsts, are not readily available, making comparison difficult. Instead, we have included photographic images of real phenomena as a basis of comparison. It is difficult to quantify how closely our results match corresponding effects in nature and so we rely on visual inspection.

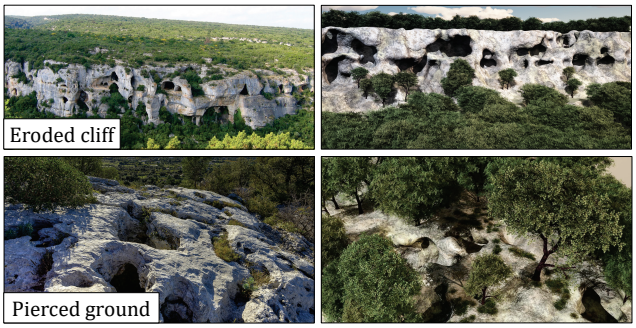


Fig. 24. A comparison between real (left) and synthetic karsts (right). From an initial $2\frac{1}{2}$D heightfield, we simulate water infiltrating soft strata and eroding the bedrock. In this example, the initial points for invasion percolation were distributed on the cliff faces and in depressions on the plateau.

Figure 24 shows a side-by-side comparison between a real karst and a volumetric model, synthesized with our method. The modified

invasion percolation algorithm generates a network of caves and tunnels that have a similar overall structure and appearance.

Figure 25 illustrates another example in which the user interactively sculpted a cave, inspired by a photograph of the Benagil Cave in Portugal. It required 10 minutes for an experienced user to author the scene from start to finish.

Fig. 25. Comparison between a real (left) and synthetic cave (right) in Benagil, Portugal. The scene was made by an experienced user in less than 10 minutes using skeletal brushes (spheres and curves) as sculpting tools.

8.2 Control

Our method provides several mechanisms for user control over landform generation. First, a user can define the regions to be amplified with erosion effects or landforms, by either directly painting a control region onto the 2D input terrain or by marking out a spatial volume. Effects can also be fine-tuned by changing their generative parameters. In contrast with most simulation-based methods, these parameters have a direct and intuitive physical interpretation (for example, the height and base radius of the hoodoos or the maximum depth of sea erosion).

Interactive editing is also supported. A user can directly and interactively sculpt the terrain with extruding or intruding skeletal primitives, or apply more complex brushes to form procedural arches and caves (see accompanying video). Table 1 reports the number of editing clicks $\#\mathcal{E}$ required to produce the different figures: Figures 11 and 17 were edited in less than five minutes and after multiple procedural erosion steps (Section 6). The user then placed sphere primitives to better sculpt the arches and the caves. Figures 15 and 19 were fully procedural. Figure 25 was entirely authored by an experienced user who interactively hollowed out the cave and sculpted the arches with point and sphere primitives to match the reference picture; the scene was completed in approximately 10 minutes. A key benefit of our framework is that our terrain models offer a single consistent global scene structure. This means that the user can seamlessly switch between manual authoring and procedural algorithms over as many cycles of iterative refinement as required. Interactive visualization during editing is made possible by delimiting the shaping tools and only repolygonizing over modified cells in the grid.

8.3 Performance

Table 1 reports the following statistics for the landscapes portrayed in our results: the extent of the input terrain (in $[km^2]$), the number of construction tree nodes produced by 3D augmentation, the amount of memory required, the time required to generate the construction tree (which excludes subsequent polygonization). We also report the amount of memory needed to model the same terrains using the Arches [Peytavie et al. 2009] model.

Speed. Our amplification methods generate the construction tree representing complex landforms at a precision of $\approx 10cm$ in less than 7 seconds for terrains that extend over $5km^2$. Performance could be improved by using the GPU, but it is beyond the scope of this paper and is left as future work. Note that such procedural methods are more time consuming when applied globally as opposed to locally during an editing session, where effects are restricted to a smaller domain to ensure interactivity.

Memory. Our hierarchical implicit construction tree is space efficient in modeling 3D landforms. One important aspect of our approach is that implicit primitives are only located where required. Thus, most of our scenes have a low occupancy ratio a compared to their extent. Exceptions are the Zion Canyon (Figure 15), which exhibits extensive overhangs resulting from hydraulic erosion, and the Benagil Cave (Figure 25), which is a small scene dominated by a sea cave. Our implicit model and the use of skeletal primitives enables us to represent local volumetric features with minimal information in memory. To achieve the results depicted in Table 1 we developed an instancing system that effectively halves the memory cost of replicating a skeletal primitive.

Control. Our system integrates user-control and authoring across different stages of the pipeline. Folds, faults and different geological strata can be specified easily. The procedural generation algorithms are parameterized with a limited set of intuitive parameters. The user can also directly sculpt landforms by merging the terrain with primitives or even subtrees. Moreover, our system efficiently combines procedural generation and authoring in a unified and coherent framework. This bridges the gap between editing and procedural generation, and supports iterative cycles of interactive refinement.

Extensibility. Our hierarchical implicit construction tree embeds heightfield representations at an extremely reduced cost, and allows us to augment $2\frac{1}{2}$D terrains with a wide range of 3D landforms. This extensibility is depicted in Figure 23, which shows the outcomes of a variety of processes. These scenes were created by extending the model with new operators, primitives and algorithms. We also believe that the model is suited to physical simulation, but testing this is left as future work.

8.4 Comparison with Other Techniques

Our primitive-based implicit model allows the generation of a wide variety of landforms with a low memory footprint. Such compactness is in contrast to other volumetric terrain models, such as Arches [Peytavie et al. 2009], which rely on voxels or material stacks. Table 1 compares the overall memory footprint for several terrains and demonstrates that our method uses two orders of magnitude less memory than material stacks, at the same precision. There are two main reasons for this: first, our hierarchical construction tree is a parsimonious vector-based representation that generates at appropriate locations the specific primitives required by a landform, and, second, we only rely on 3D primitives where needed, and resort to more memory efficient implicitized heightfield representations elsewhere.

Scene	Size	a	$\#\mathcal{N}$	Generation	$\#\mathcal{E}$	Memory		Meshing				
						Ours	Arches	Grid resolution	t	t_0	$\#C$	$\#C_0$
Sea (11)	6.0×6.0	0.01	50 693	5.5	156	3.0	300	$2000^2 \times 33$	13.3	91.5	9	210
Karst (17)	5.2×5.2	0.02	165 773	6.2	43	9.9	140	$1520^2 \times 447$	14.2	188.4	14	1000
Canyon (15)	1.1×1.1	0.20	137 043	6.8	0	9.3	110	$1000^2 \times 546$	32.6	159.0	30	569
Hoodoo (19)	0.35×0.35	0.05	55 619	2.1	0	5.5	10.2	$650^2 \times 245$	2.9	15.8	4	111
Benagil (25)	0.4×0.4	0.35	13 264	-	551	1.3	2.1	$550^2 \times 87$	2.7	3.9	8	20

Table 1. Statistics for different amplified terrains: size [km^2], percentage of $2\frac{1}{2}$D to 3D surface area a, number of nodes in the construction tree $\#\mathcal{N}$, generation time [s], editing click count done by the user $\#\mathcal{E}$, memory footprint of the construction tree excluding the base heightfield [Mb], memory consumption [Mb] using the model of Peytavie et al. [2009], meshing grid resolution, optimized and standard polygonization time t [s] and t_0 [s], number of calls to f with our optimized $\#C$ and with the standard algorithm $\#C_0$, in millions. Benagil was entirely authored by an experienced user using skeletal brushes in less than 10 minutes and therefore has no generation time.

9 CONCLUSION

We have introduced a novel method for augmenting $2\frac{1}{2}$D heightfields with 3D landforms. Such 3D landforms as sea cliffs, canyons with overhangs, network of caves and tunnels, hoodoos and goblins, and even floating islands, are essential scenic elements in synthetic environments, animated films and computer games.

Our compact hierarchical primitive-based implicit representation captures 3D features at resolutions as fine as 10 cm over large terrains up to 5 km^2 in extent with a memory footprint of at most a few megabytes. Structuring the terrain as an implicit hierarchy also enables significantly accelerated surface extraction. At the heart of our method is a system that analyzes an input $2\frac{1}{2}$D terrain to derive the location and characteristics of volumetric landforms, and which automatically generates their shape according to the relief of the terrain, environmental erosion effects, and the underlying structure of the geology.

Our system integrates user-control and authoring at different stages of the pipeline, from definition of the different strata, folds and faults of the geology, to direct sculpting of features. Crucially, the transition between editing and simulation is seamless, which supports iterative cycles of interactive refinement.

We have shown a wide variety of obtainable effects, including augmenting real $2\frac{1}{2}$D DEM data with features such as arches and caves, and synthesizing fantastical floating islands.

In future, we would like to investigate the implicit modeling of finely-detailed rock features at scales below 10cm, thereby bridging the gap between modeling and texturing for terrains. Currently, details such as cracks in granite or thin seams of limestone require creating complex geometric primitives or seeding thousands of tiny primitives, with a concomitant increase in both memory and computation.

ACKNOWLEDGMENTS

This work is part of the project *PAPAYA* funded by the *Fonds National pour la Société Numérique* and the project HDW ANR-16-CE33-0001, supported by Agence Nationale de la Recherche. This work also received a grant from Bourg en Bresse city and CCI de l'Ain. We would like to credit E-on software for providing Vue xStream for rendering our terrain models.

REFERENCES

Aurelien Barbier and Eric Galin. 2004. Fast Distance Computation Between a Point and Cylinders, Cones, Line-Swept Spheres and Cone-Spheres. *Journal of Graphics Tools* 9, 2 (2004), 11–19.

Richard Barnes, Clarence Lehman, and David Mulla. 2014. Priority-flood: An Optimal Depression-filling and Watershed-labeling Algorithm for Digital Elevation Models. *Computers & Geosciences* 62 (2014), 117–127.

M. Beardall, M. Farley, D. Ouderkirk, C. Reimschussel, J. Smith, M. Jones, and P. Egbert. 2007. Goblins by Spheroidal Weathering. In *Proceedings of Third Eurographics Conference on Natural Phenomena*. 7–14.

Michael Becher, Michael Krone, Guido Reina, and Thomas Ertl. 2017. Feature-based Volumetric Terrain Generation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '17)*. 10:1–10:9.

M. Becher, M. Krone, G. Reina, and T. Ertl. 2018. Feature-based Volumetric Terrain Generation and Decoration. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1.

Farès Belhadj and Pierre Audibert. 2005. Modeling Landscapes with Ridges and Rivers: Bottom Up Approach. In *Proc. International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. ACM, 447–450.

Norishige Chiba, Kazunobu Muraoka, and K. Fujita. 1998. 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.

Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Éric Galin, Adrien Peytavie, and Éric Guérin. 2016. Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion. *Computer Graphics Forum* 35, 2 (2016).

Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin. 2018. Sculpting Mountains: Interactive Terrain Modeling Based on Subsurface Geology. *IEEE Transactions on Visualization and Computer Graphics* 24, 5 (2018), 1756–1769.

Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. 2017. Authoring Landscapes by Combining Ecosystem and Terrain Erosion Simulation. *ACM Transactions on Graphics* 36, 4 (2017).

Benoît Crespin, Carole Blanc, and Christophe Schlick. 1996. Implicit Sweep Objects. *Computer Graphics Forum* 15, 3 (1996), 165–174.

Evgenij Derzapf, Björn Ganster, Michael Guthe, and Reinhard Klein. 2011. River Networks for Instant Procedural Planets. *Computer Graphics Forum* 30, 7 (2011), 2031–2040.

David S. Ebert, Forest Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. 1998. *Texturing and Modeling: A Procedural Approach* (3rd ed.). Elsevier.

T. Graham Freeman. 1991. Calculating catchment area with divergent flow based on a regular grid. *Computer and Geoscience* 17 (1991).

James Gain, Patrick Marais, and Wolfgang Strasser. 2009. Terrain sketching. In *Proc. Symposium on Interactive 3D Graphics and Games*. ACM, Boston, USA, 31–38.

James E. Gain, Bruce Merry, and Patrick Marais. 2015. Parallel, Realistic and Controllable Terrain Synthesis. *Computer Graphics Forum* 34, 2 (2015), 105–116.

Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. 2019. A Review of Digital Terrain Modeling. *Computer Graphics Forum (proceedings of Eurographics 2019 STAR)* 38, 2 (2019), 553–577.

Manuel Gamito and Steve Maddock. 2008. Topological Correction of Hypertextured Implicit Surfaces for Ray Casting. *The Visual Computer* 24, 6 (2008), 397–409.

Manuel N. Gamito and Kenton Forest Musgrave. 2001. Procedural landscapes with overhangs. In *Proc. of the Portuguese Computer Graphics Meeting*. Lisbon, Portugal.

Jean-David Génevaux, Éric Galin, Éric Guérin, Adrien Peytavie, and Bedřich Beneš. 2013. Terrain Generation Using Procedural Models Based on Hydrology. *ACM Transaction on Graphics* 32, 4 (2013), 143:1–143:13.

Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. 2015. Terrain Modelling from Feature Primitives. *Computer Graphics Forum* 34, 6 (2015), 198–210.

Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. 2016. Sparse representation of terrains for procedural modeling. *Computer Graphics Forum (Proceedings of Eurographics)* 35, 2 (2016), 177–187.

Eric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoit Martinez. 2017. Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks. *ACM Transactions on Graphics (proceedings of Siggraph Asia 2017)* 36, 6 (2017), 228:1–228:13.

John C. Hart. 1996. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

Houssam Hnaidi, Éric Guérin, Samir Akkouche, Adrien Peytavie, and Éric Galin. 2010. Feature based terrain generation using diffusion equation. *Computer Graphics Forum* 29, 7 (2010), 2179–2186.

Tomoya Ito, Tadahiro Fujimoto, Kazunobu Muraoka, and Norishige Chiba. 2003. Modeling rocky scenery taking into account joints. In *Proceedings of Computer Graphics International*. IEEE, Tokyo, Japan, 244–247.

M. Jones, M. Farlay, M. Butler, and M. Beardall. 2010. Directable Weathering of Concave Rock using Curvature Estimation. *IEEE Transactions on Visualization and Computer Graphic* 16, 1 (2010), 81–97.

D. Kalra and A. H. Barr. 1989. Guaranteed Ray Intersections with Implicit Surfaces. *SIGGRAPH Comput. Graph.* (1989).

Alex D. Kelley, Michael C. Malin, and Gregory M. Nielson. 1988. Terrain simulation using a model of stream erosion. *Computer Graphics* 22, 4 (1988), 263–268.

J. Kruger and R. Westermann. 2003. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03) (VIS '03)*. IEEE Computer Society, Washington, DC, USA, 38–. https://doi.org/10.1109/VIS.2003.10001

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH Comput. Graph.* 21, 4 (1987), 163–169.

Xing Mei, Philippe Decaudin, and Baogang Hu. 2007. Fast Hydraulic Erosion Simulation and Visualization on GPU. In *Pacific Graphics*. IEEE, 47–56.

Gavin Miller. 1994. Efficient Algorithms for Local and Global Accessibility Shading. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'94)*. ACM, 319–326.

Don P. Mitchell. 1990. Robust ray intersection with interval arithmetic. In *Proceedings on Graphics interface '90*. 68–74.

Forest Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. 1989. The synthesis and rendering of eroded fractal terrains. *Computer Graphics* 23, 3 (1989), 41–50.

Radomír Měch and Przemyslaw Prusinkiewicz. 1996. Visual Models of Plants Interacting with Their Environment *(SIGGRAPH '96)*. ACM, 397–410. https://doi.org/10.1145/237170.237279

Kenji Nagashima. 1998. Computer generation of eroded valley and mountain terrains. *The Visual Computer* 13, 9-10 (1998), 456–464.

Mattia Natali, Endre M. Lidal, Julius Parulek, Ivan Viola, and Daniel Patel. 2013. Modeling Terrains and Subsurface Geology. In *Eurographics State Of The Art*. 155–173.

Adrien Peytavie, Éric Galin, Stephane Mérillou, and Jérôme Grosjean. 2009. Arches: A Framework for modeling complex terrains. *Computer Graphics Forum* 28, 2 (2009), 457–467.

Przemyslaw Prusinkiewicz and Marc Hammel. 1993. A fractal model of mountains with rivers. In *Proceedings of Graphics Interface*. 174–180.

P. Roudier, B. Peroche, and M. Perrin. 1993. Landscapes Synthesis Achieved through Erosion and Deposition Process Simulation. *Computer Graphics Forum* 12, 3 (1993), 375–383.

Brennan Rusnell, David Mould, and Mark G. Eramian. 2009. Feature-rich distance-based terrain synthesis. *The Visual Computer* 25, 5-7 (2009), 573–579.

Flora Ponjou Tasse, Arnaud Emilien, Marie-Paule Cani, Stefanie Hahmann, and Adrien Bernhardt. 2014. First Person Sketch-based Terrain Editing. (2014), 217–224.

Flora Ponjou Tasse, James Gain, and Patrick Marais. 2012. Enhanced Texture-Based Terrain Synthesis on Graphics Hardware. *Computer Graphics Forum* (2012), 1959–1972.

Juraj Vanek, Bedřich Beneš, Adam Herout, and Ondřej Šťava. 2011. Large-Scale Physics-Based Terrain Editing Using Adaptive Tiles on the GPU. *Computer Graphics and Applications* 31, 6 (2011), 35–44.

David Wilkinson and Jorge F Willemsen. 1983. Invasion percolation: a new form of percolation theory. *Journal of Physics A: Math. Gen.* 16 (1983), 3365—-3376.

Brian Wyvill, Andrew Guy, and Éric Galin. 1999. Extending the CSG Tree - Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum* 18, 2 (1999), 149–158.

Geoff Wyvill, Craig McPheeters, and Brian Wyvill. 1986. Data Structure for Soft Objects. 2 (08 1986), 227–234.

Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. 2007. Terrain Synthesis from Digital Elevation Models. *Transactions on Visualization and Computer Graphics* 13, 4 (2007), 834–848.

Received February 2007; revised March 2009; final version June 2009; accepted July 2009