N° d'ordre NNT : 2022LYO1070

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

École Doctorale 512
InfoMaths

Spécialité de doctorat : Informatique

Présentée et soutenue publiquement le 8 mars 2023
par **Axel Paris**

Modeling and simulating virtual terrains

Devant le jury composé de :

Mme. CANI Marie-Paule	Professeure des Universités	CNRS, Ecole Polytechnique	Rapportrice
M. LÉVY Bruno	Professeur des Universités	Inria Nancy Grand-Est	Rapporteur
Mme. COLLON Pauline	Maîtresse de Conférences	Université de Lorraine	Examinatrice
Mme. HAHMANN Stefanie	Professeure des Universités	Université de Grenoble	Examinatrice
Mme. CHAINE Raphaëlle	Professeure des Universités	Université Lyon 1	Examinatrice
M. BARTHE Loic	Professeur des Universités	Université de Toulouse	Examinateur
M. GALIN Eric	Professeur des Universités	Université Lyon 1	Directeur de thèse
M. GUÉRIN Eric	Maître de Conférences	INSA Lyon	Co-Directeur de thèse

Laboratoire LIRIS
Adresse
Bâtiment Nautibus
Campus de la Doua
25 avenue Pierre de Coubertin
69622 Villeurbanne Cedex

École Doctorale InfoMaths
7, avenue Jean Capelle
69621 VILLEURBANNE Cedex

Contents

Remerciements

Résumé

Cette thèse, intitulée "Modeling and simulating virtual terrains" est dédiée à la création de contenus numériques et aux simulations scientifiques, dans le cadre des terrains virtuels dans les scènes naturelles. Les terrains sont composés de formes à différentes échelles (micro-échelle, meso-échelle, et macro-échelle), qui sont le résultat de plusieurs processus physiques entrelacés opérant à différentes échelles temporelles et spatiales. En informatique, ces formes sont habituellement représentées par des surfaces d'élévation, mais les formes telles que les arches ou les grottes requièrent une représentation volumique. Cependant, les besoins grandissants de réalisme et de taille des mondes virtuels amènent de nouveaux défis que les techniques et modèles actuels ne résolvent pas entièrement.

Cette thèse est séparée en deux parties. En premier lieu, nous observons que plusieurs formes de terrains à l'échelle macro, telles que les déserts et les méandres de rivières, ne peuvent pas être modélisées avec les techniques actuelles. Partant de cette observation, nous développons de nouvelles simulations inspirées de la géomorphologie pour modéliser ces phénomènes sur les terrains virtuels. Nous nous intéressons à la fois au réalisme de nos simulations et au contrôle utilisateur, qui est un aspect clé en informatique graphique.

Dans la seconde partie de cette thèse, nous nous intéressons à la modélisation et la génération de phénomènes volumiques de terrains. Les modèles existants utilisent sur les voxels et ont un coût mémoire important, ce qui empêche leur utilisation à grande échelle. À la place, nous développons un nouveau modèle basé sur les fonctions de distance signées pour représenter les formes de terrain volumiques, comme les arches, les surplombs et les grottes, avec un impact mémoire bien plus faible. Nous montrons comment cette représentation est adaptée pour générer des formes de terrain volumiques à plusieurs échelles (micro-échelle, meso-échelle, et macro-échelle).

Keywords: modèles de terrain, génération procédurale, simulations physiques, surfaces implicites

Abstract

This PhD entitled “Modeling and simulating virtual terrains” is related to digital content creation and geological simulations, in the context of virtual terrains. Real terrains exhibit landforms of different scales (namely microscale, mesoscale, and macroscale), formed by multiple interconnected physical processes operating at various temporal and spatial scales. On a computer, these landforms are usually represented by elevation models, but features such as arches and caves require a volumetric representation. However, the increasing needs for realism and larger worlds bring new challenges that existing techniques do not fulfil.

This thesis is separated in two parts. First, we observe that several macroscale landforms, such as desert landscapes made of sand dunes and meandering rivers, simply cannot be modeled by existing techniques. Thus, we develop new simulations, inspired by research in geomorphology, to generate these landforms. We particularly focus on the plausibility of our results and user control, which is a key requirement in Computer Graphics.

In the second part of this thesis, we focus on modeling and generating volumetric landforms in virtual terrains. Existing models are often based on voxels and have a high memory impact, which forbids their use at a large-scale. Instead, we develop a new model based on signed distance functions for representing volumetric landforms, such as arches, overhangs and caves with a low memory footprint. We show that this representation is adapted for generating volumetric landforms across a range of scales (microscale, mesoscale, and macroscale).

Keywords: terrain models, procedural modeling, physical simulations, implicit surfaces

Publications

Synthesizing Geologically Coherent Cave Networks
Axel Paris, E. Guérin, A. Peytavie, E. Galin
Computer Graphics Forum, 2021, vol. 40, num. 7, p. 277-287
Presented at *Pacific Graphics* 2021 and *JFIG* 2021

Modelling Rocky Scenery with Implicit Blocks
Axel Paris, E. Guérin, A. Peytavie, J-M. Dischler, E. Galin
The Visual Computer, 2020, vol. 36, num. 10, p. 2251-2261
Best Paper Award
Presented at *Computer Graphics International* 2020 and *JFIG* 2020

Terrain Amplification With Implicit 3D Features
Axel Paris, E. Galin, A. Peytavie, E. Guérin, J. Gain
Transactions on Graphics, 2019, vol. 28, num. 5, p. 147:1–147:15
Presented at *Siggraph Asia* 2019

Desertscape Simulation
Axel Paris, E. Galin, A. Peytavie, E. Guérin
Computer Graphics Forum, 2019, vol. 38, num. 7, p. 47–55
Presented at *Pacific Graphics* 2019

Amplification de Terrains avec des Caractéristiques Implicites 3D
Axel Paris, E. Galin, A. Peytavie, E. Guérin, J. Gain
Journées Françaises d'Informatique Graphique, JFIG 2018
2nd Best Paper Award

Large-scale terrain authoring through interactive erosion simulation
H. Schott, **Axel Paris**, L. Fournier, E. Guérin, E. Galin
Minor revision at *Transactions on Graphics*, 2023

Simulation, Modeling and Rendering of Glaciers
O. Argudo, E. Galin, A. Peytavie, **Axel Paris**, E. Guérin
Transactions on Graphics, 2020, vol. 39, num. 6, p. 177:1-177:14

Segment Tracing using Local Lipschitz Bounds
E. Galin, E. Guérin, **Axel Paris**, A. Peytavie
Computer Graphics Forum, 2020, vol. 39, num. 2, p. 545-554

Orometry-based Terrain Analysis and Synthesis
O. Argudo, E. Galin, A. Peytavie, **Axel Paris**, J. Gain, E. Guérin
Transactions on Graphics, 2019, vol. 38, num. 6, p. 199:1-199:12

Chapter 1

Introduction

In this thesis, we focus on natural scenes in the context of computer science, and more specifically computer graphics. During the past fifty years, computers have become increasingly powerful, providing us, computer scientists with many new possibilities. The increase in computational power, democratization of machine learning techniques and new virtual reality tools have drastically changed our world. This is particularly well illustrated by the entertainment industry: the video game industry and the cinema have been using computer generated worlds for a while now - so much that some movies are shot entirely in studio, with actors playing in front of a green screen. Natural sciences, such as physics, biology and geology also rely greatly on computer science. The ability to study a simplified version of our world on a computer at a given scale allows for a better understanding of the rules and processes at its origin.

Virtual worlds are a major component in the entertainment industry for setting the mood and context of a video game scene or a movie, as the surrounding landscape represents a large part of what people sees on the screen. Depending on the needs, the landscape may be realistic (a forest in a valley surrounded by mountains), or a fictional scene (an unknown planet with colorful alien vegetation). The deciding factor for the look and tone of a virtual scene is the artistic direction. Thus, computer models *must* be amenable to modification at any time. One key aspect of computer graphics is **control**.

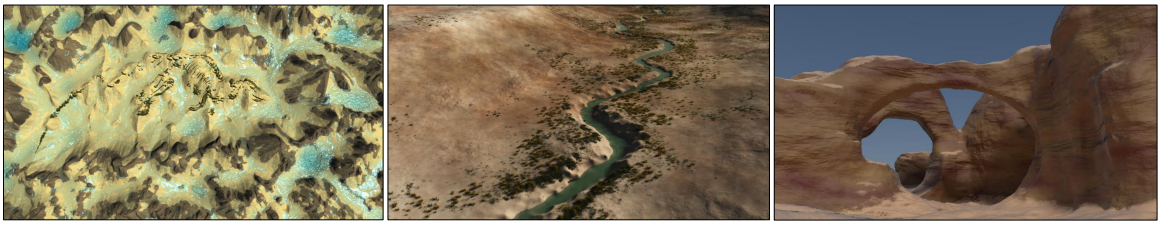
On the other hand, natural sciences are interested in approximating the rules and processes responsible for the creation of our environment: for instance, how does a mountain form? What characterize an underground cave? Extensive research have been made on classifications, from types of mountains to river archetypes. Classifications help us build an understanding of the world that surrounds us.

A natural scene is composed of many different complex elements: ecosystems (forests, bushes and grass), fluids (water bodies, clouds), urban elements (roads, villages) and terrains (mountains, coasts, caves). The challenges regarding natural scenes are many: from the shape of the terrain, to the animation of water bodies and clouds, vegetation placement and texturing.

Contributions and outline

Chapter 2

State of the art



Contents

2

Virtual terrains have been an active subject of research in the last decades, dating back to Musgrave *et al.* 1989 for the first terrain generation and erosion algorithms. Since then, an active research direction has been to generate terrains that exhibit realistic features, using a variety of algorithms and authoring techniques, with various applications in the entertainment industry, geology and scientific simulations.

In this chapter, we present an overview of *terrain models* and explain their advantages and limitations (Section 2.1). Particularly, we show how existing volumetric models are not adapted for representing large virtual worlds with landforms of multiple scales. Then, we address *terrain generation methods* (Section 2.2). These are usually classified into three categories: procedural techniques, physical simulations and synthesis from examples (Galin *et al.* 2019). We depart from this classification and analyze existing methods from a new perspective inspired by analysis made in geology, focusing on the spatial and temporal scales of terrain landforms. Despite decades of research on terrain modeling, we show that there is still no solution for generating certain terrain landforms, such as meandering rivers, cave networks covering dozens of kilometers, or desert landscapes with dunes and yardangs.

2.1 Terrain models

There are several ways to represent a terrain on a computer. This choice mainly depends on the target application (video games, scientific simulations), the type of landforms (mountain ranges, caves, hills), and the scale. We divide terrain models in two principal categories: *elevation* and *volumetric* representations. Elevation models are the most popular representation, as they provide a sufficiently accurate approximation of a terrain while being compact in memory. However, they cannot represent volumetric landforms, such as karstic networks, overhangs and arches, which are crucial scenic elements of virtual worlds. In this section, we focus on the most popular models but do not aim at describing all existing terrain representations. In particular, we set aside hexagon and triangular fields (Dixon *et al.* 1994) and combinatorial maps (Damiand *et al.* 2014; Crespin *et al.* 2014), and refer the reader to the associated papers for more details on these models.

2.1.1 Elevation models

Planar models (or heightfields) are the most common representation for virtual terrains. They are characterized by a function $h : \mathbb{R}^2 \to \mathbb{R}$ computing the elevation z at every point in a domain $\Omega \in \mathbb{R}^2$. The domain Ω is usually a rectangle $\mathcal{R}(\mathbf{a}, \mathbf{b})$ with $\mathbf{a}$ and $\mathbf{b}$ the opposite corners in the plane. The elevation function h may be defined by a combination of analytic primitives or discrete elevation data (Figure 2.1).

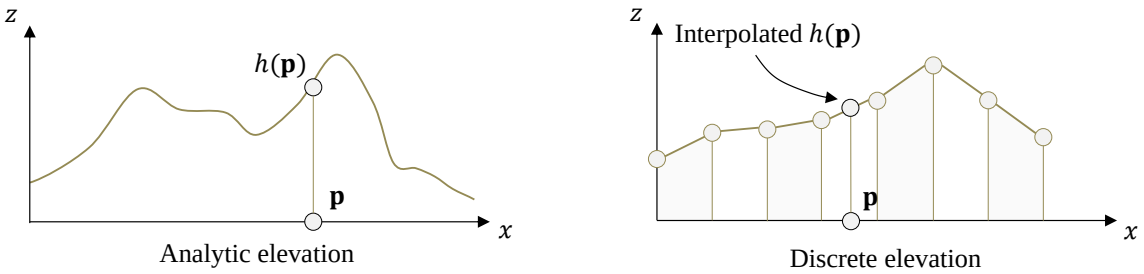


FIGURE 2.1: *Heightfields can be represented either by an analytic function (e.g. from an aggregation of primitives), or by discrete elevation data.*

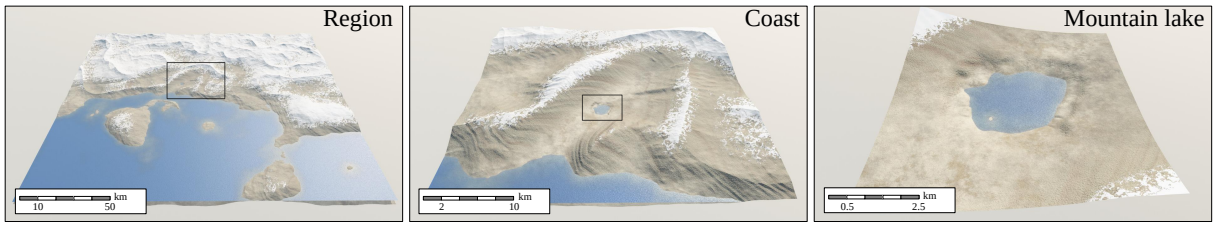


FIGURE 2.2: *The model from Génevaux* et al. *2015 allows to represent large-scale terrains with multiple levels of detail, here going from the macroscale to the mesoscale. However, because of the procedural representation of the terrain, erosion landforms such as gullies and ravines cannot be generated.*

Discrete representations define the elevation $h(\mathbf{p})$ as the interpolation of altitude at discrete points, usually distributed on a regular grid (Figure 2.1, right). Bilinear interpolation is the fastest method, with efficient implementation on graphics hardware for textures, but only provide C^0 continuity. On the other hand, biquadratic (C^1) and bicubic methods (C^2) require more neighboring values (16 for bicubic), are slower to compute, but provide a smoother reconstruction of the surface.

A key advantage of discrete models is the ability to define altitude from Digital Elevation Model (DEM) captured by remote sensing. In geology, this allows the analysis of real terrains, for extracting characteristics or predicting the evolution of the topography through time. In the entertainment industry, digital elevation models empower the creative process, where real and artificial terrains are combined together to create convincing landscapes.

Discrete models, particularly those based on regular grids, lends themselves for physical simulations (Musgrave *et al.* 1989; Cordonnier *et al.* 2017) as they provide a simple definition of the neighborhood for each cell, which is often needed to transport material between cells. Alternative discretizations, such as Triangular Irregular Networks (TIN), have also been used for simulations (Cordonnier *et al.* 2016) but require explicit computations for finding neighbors between cells. Discrete heightfields also lend themselves to machine learning techniques, (Guérin *et al.* 2017; Zhao *et al.* 2019; Zhang *et al.* 2022) and by-example synthesis (Zhou *et al.* 2007; Tasse *et al.* 2012; Gain *et al.* 2015; Argudo *et al.* 2017; Scott *et al.* 2021).

Function-based representations also referred to as procedural models, define the elevation by a closed-form mathematical expression. They were first introduced by Génevaux *et al.* 2015 for modeling large-scale terrains (Figure 2.2). The elevation function is defined by a construction tree of implicit primitives and operators, taking inspiration from the Blob Tree (Wyvill *et al.* 1999). Let $f : \mathbb{R}^2 \to R$ denote the evaluation function of the construction tree (Figure 2.3), the surface $\mathcal{H}$ of the terrain is defined as the set of points $(\mathbf{p}, f(\mathbf{p}))$ in space within a domain $\Omega \in \mathbb{R}^2$:

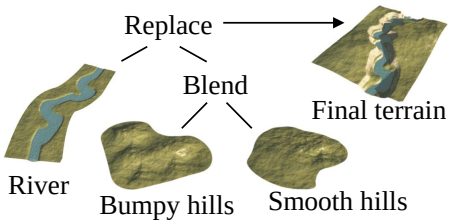


FIGURE 2.3: *Hierarchical terrain model from Génevaux* et al. *2015.*

$$\mathcal{H} = \{(\mathbf{p}, f(\mathbf{p})) \in \mathbb{R}^3, \mathbf{p} \in \Omega\}$$

Primitives are constructed by combining an elevation function $h : \mathbb{R}^2 \to \mathbb{R}$ and a weight function $\alpha : \mathbb{R}^2 \to [0, 1]$. The elevation is usually defined as a combination of carefully-designed noise functions, and the weight is a classical falloff function (Wyvill *et al.* 1999) that describes how primitives are merged together within operators. The construction tree can be exploited to compute Lipschitz bounds hierarchically, which in turn allows to render the terrain directly with sphere tracing (Hart 1996).

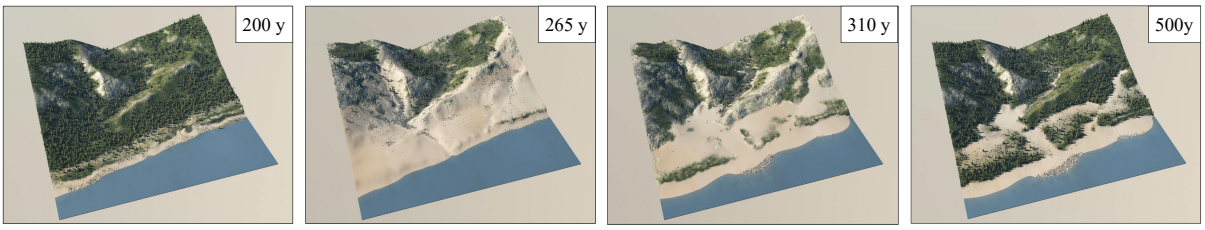


FIGURE 2.4: *Cordonnier* et al. *2017 use multiple material layers, such as sand, rocks, and vegetation to generate complex virtual terrain models and simulate their evolution through time.*

This representation is compact in memory and theoretically provides infinite precision. It also lends itself for compression and amplification, for instance by using a sparse representation based on a dictionary of patches (Guérin *et al.* 2016). However, the evaluation of $h(\mathbf{p})$ may be computationally demanding, and modeling realistic landforms requires a fine-tuning of noise-based primitives, which can be tedious. Another crucial limitation of procedural models is that they cannot be used with simulations, as it is not possible to transport material between different locations because of the implicit nature of the representation.

Multi-layer extensions for handling multiple material layers can be defined easily by using one height-field (discrete or procedural) per material. This *layer-field* data structure can be constructed as a function $l : \mathbb{R}^2 \to \mathbb{R}$ defined from multiple elevation functions h_i as:

$$l(\mathbf{p}) = h_0 + \sum_{i=1} h_i(\mathbf{p})$$

The term h_0 denotes the elevation of the bare bedrock, and h_i the thickness of the different materials such as sand, water or rocks. Layer-fields were first used by Musgrave *et al.* 1989 which modelled a layer of sediments on top of the bedrock layer for simulating hydraulic erosion. Additional layers, such as sand, rocks, and vegetation are also possible, at the expense of additional memory (Beneš *et al.* 2001; Génevaux *et al.* 2015; Cordonnier *et al.* 2017). The ability to define multiple layers allows to simulate a wider range, such as the combination of ecosystem and terrain simulation as demonstrated by Cordonnier *et al.* 2017 (see Figure 2.4).

Conclusion. Elevation models are used extensively in video games, Geographic Information Systems (GIS), and physical simulations, as the representation is compact in memory and can be used efficiently with LOD techniques, therefore allowing the modeling of large-scale terrains. However, this representation forbids the modeling of volumetric features such as arches and overhangs. This limitation is partially alleviated by placing assets often defined as triangle meshes (for instance in video games), or by using displacement techniques (Gamito *et al.* 2001). Still, modeling truly volumetric landforms such as caves and arches simply cannot be done with elevation models.

2.1.2 Volumetric models

Volumetric landforms, such as cave networks, overhangs, and arches, are crucial visual elements of virtual terrains with cannot be captured by elevation models (Section 2.1.1). They are defined by a function $v : \mathbb{R}^3 \to \mathbb{N}$ which computes the material index for every point in space. The simplest representation uses 0 for air and 1 for bedrock, but other materials such as sand or different types of bedrock can be

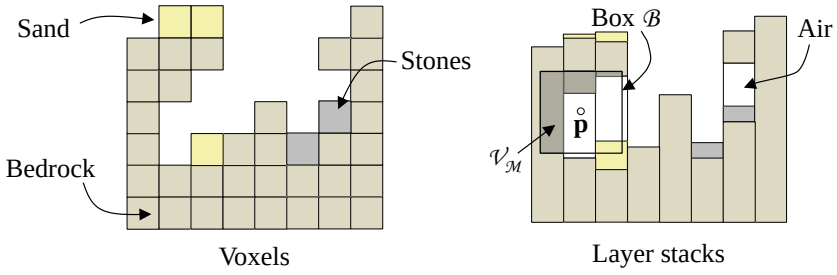


FIGURE 2.5: *Existing volumetric models include voxels (left) and layer stacks (right). In both cases, the memory footprint is high, which limits the extent of the domain or the precision of the terrain.*

defined. The function v can be constructed in different ways, for instance using voxels or layer stacks (Figure 2.5).

2.1.2.1 Voxels

Voxels are a fully explicit representation where the terrain is stored in a three-dimensional grid (Figure 2.5, left). Each cell is assigned an integer value representing the associated material, such as bedrock, sand, water, or air. They provide a convenient authoring framework to the user, with the ability to sculpt the landscape by using dedicated brushes for adding or removing different materials, or by using control curves (see Figure 2.6 and Becher *et al.* 2019). As for discrete elevation models, voxels are well suited for physical simulations, with applications to erosion (Beardall *et al.* 2007; Jones *et al.* 2010). The smooth surface of the terrain can be reconstructed by polygonizing the voxel field using Marching Cubes (Wyvill *et al.* 1986; Lorensen *et al.* 1987).

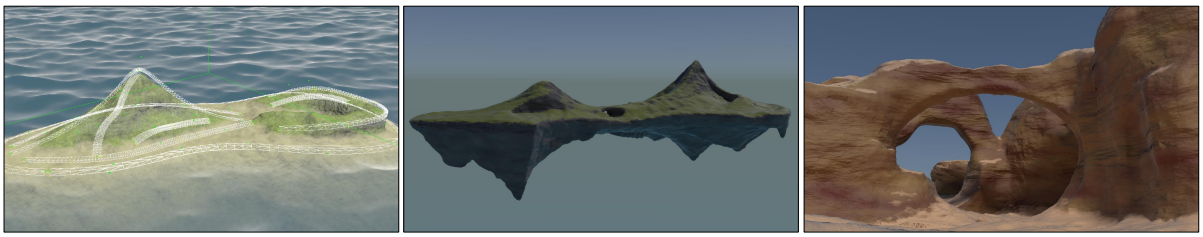


FIGURE 2.6: *Becher* et al. *2019 use voxels and 3D features curves to generate volumetric landforms such as arches and fictional floating islands.*

The main limitation of voxels is the $O(n^3)$ memory cost, which makes the representation unsuited for large and detailed landscapes. Compression techniques such as Sparse Voxel Octrees (SVO) (Laine *et al.* 2010) only partially alleviates the problem, even when exploiting symmetry or treating the octree as a directed acyclic graph (Kämpe *et al.* 2013; Villanueva *et al.* 2017). Still, voxels remain a popular model in the industry for authoring 3D models (in softwares such as MagicaVoxel) or generating complex landscapes (in video games such as Minecraft).

2.1.2.2 Hybrid models

FIGURE 2.7: *The Arches model from Peytavie* et al. *2009b.*

Layer stacks, inspired by multi-layer heightfields, include air as a material for representing volumetric terrains. They represent the terrain as intervals of constant material stacked on top of each other (see Figure 2.5, right). A smooth representation $f : \mathbb{R}^3 \rightarrow [-1, 1]$ of the terrain can be reconstructed by applying a convolution operator, denoted as $\circledast$, between the discrete layer stacks v and a box filter k:

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

With $\mathcal{V}_{\mathcal{M}}$ the volume of the material intersecting the box, and $\mathcal{V}_{\mathcal{B}}$ the volume of the convolution box $\mathcal{B}$ (see Figure 2.5, right). Layer stacks belong to hybrid models, in between procedurally-defined functions and discrete representations based on (adaptive) grids or triangular irregular networks. They have been used in (Peytavie *et al.* 2009b; Peytavie *et al.* 2009a) for modeling landscapes featuring arches, overhangs, and piles of rocks (Figure 2.7). However, the smooth reconstruction of the surface is computationally intensive because of the convolution operator, and the memory impact still forbids the use of layer-stacks for large terrains.

Conclusion

The main restriction of existing volumetric models lies in the discrete nature of the representation. They have a high memory requirement, which limits the precision and therefore the extents of the terrain. This makes existing volumetric models unsuited for representing large-scale or detailed terrains, which is a key requirement for virtual worlds.

While the percentage of volumetric landforms in real terrains is rather small, they have a major impact on the overall visual perception of the scene. There is thus a need for a *sparse* representation of volumetric features compatible with elevation models used for representing large-scale terrains. Interestingly, procedural models created using a construction tree of closed-form expression primitives have been applied successfully with elevation terrains (Génevaux *et al.* 2015), but they have not yet been used in the context of volumetric terrains. In Part II of this thesis, we present a procedural model for volumetric terrains based on signed distance functions.

2.2 Classification of terrain generation methods

Terrain generation methods are usually classified in three categories: example-based synthesis, simulation and procedural techniques (Galin *et al.* 2019). This classification builds on the underlying technical aspects of the methods. So-called procedural methods encompass phenomenological approaches, *i.e.* techniques that directly reproduce the appearance of a terrain landform from observations. In contrast, simulation techniques approximate natural phenomena such as hydraulic, thermal or aeolian erosion to simulate the evolution of the terrain throughout time. Finally, example-based methods aim at generating terrains by combining different real world exemplars from a large dictionary. However, as real terrains are the result of complex geomorphological processes, it is interesting to analyze terrain generation techniques from a more geological perspective. Particularly, studying the spatial and temporal scale of terrain landforms has been done for centuries by geologists and provide great insight regarding how landforms emerge and evolve.

Scale	Size range	Example of landforms
Microscale	10cm-10m	Sand ripples, rock arrangements, ventifacts
Mesoscale	10m-1km	Cliffs, arches, ravines and gullies
Macroscale	1km-100km	Sand dunes, river networks, cave networks
Megascale	100km-10000km	Mountain ranges, continents

TABLE 2.1: *Scale classification used in this thesis.*

In this chapter, we take inspiration from geology and geomorphology and categorize terrain generation methods depending on the spatial scale at which they operate, and on the landforms they are trying to reproduce. We distinguish four spatial scales: *microscale* (10cm-10m), *mesoscale* (10m-1km), *macroscale* (1-100km) and *megascale* (100-10000km), as denoted in Table 2.1. Whenever possible, we classify terrain landforms in groups. For instance, we do not consider a single mountain, but the entire mountain ranges it belongs to, as a single landform. To determine the size, we consider the horizontal extent of the terrain feature.

Figure 2.8 depicts such a classification for important landforms, each one annotated with the number of research papers dealing with these terrain features. Recall that this figure is subjective and should not be interpreted as strict classification. However, it is useful to emphasize global areas of interest in terrain modeling from the Computer Graphics community. For instance, many papers are interested in the generation of mountainous landscapes, possibly with erosion features for creating ravines and gullies, while volumetric landforms such as arches and overhangs received less attention.

Generating microscale features such as sand ripples and details over the bedrock, is often done using noise, which has been thoroughly investigated (Perlin *et al.* 1989; Worley 1996; Lagae *et al.* 2009; Tricard *et al.* 2019). Larger details of a few meters wide, such as piles of rocks or arrangements of 3D objects, can be generated using specific techniques (Peytavie *et al.* 2009a; Grosbellet *et al.* 2016; Guérin *et al.* 2016). For the mega-scale, dedicated methods operate at continental or even planet-scale (Derzapf *et al.* 2011; Cortial *et al.* 2019; Cortial *et al.* 2020). We refer the reader to (Galin *et al.* 2019; Wei *et al.* 2009; Dong *et al.* 2020) for more details on mega-scale terrain generation methods and texture synthesis techniques.

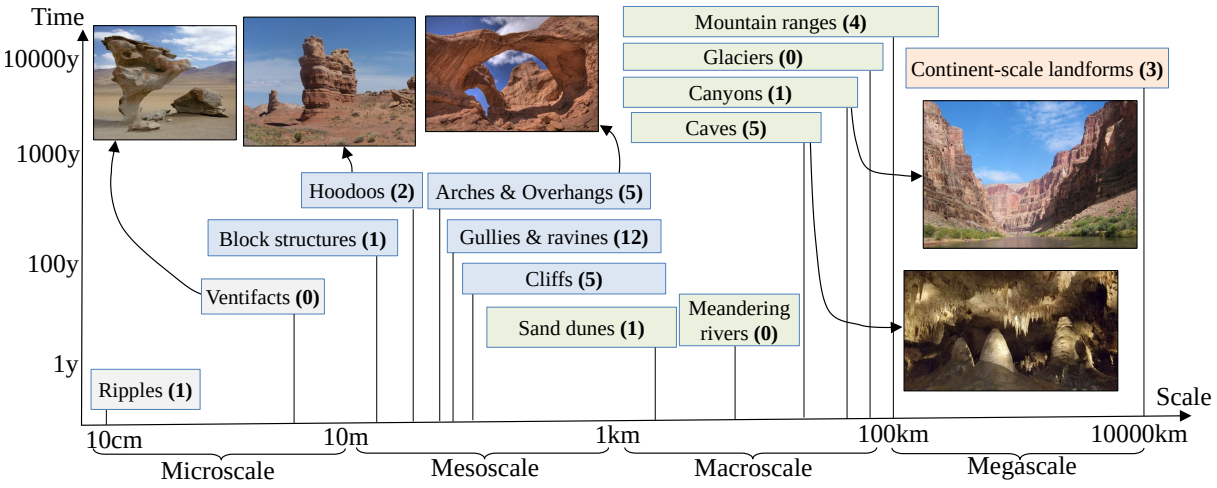


FIGURE 2.8: *Landform classification based on their temporal and spatial scale, with numbers in parenthesis representing the amount of papers focused on generating these landforms in Computer Graphics.*

In this thesis, we particularly focus on the *mesoscale* and the *macroscale*. The following sections classify existing terrain generation methods depending the generated landforms and their associated spatial scale. We often refer to terrain features by using their geological name, and refer the reader to Appendix B for a visual depiction of terrain landforms.

2.3 Scale-agnostic methods

Many methods cannot be classified depending on their spatial scale, as they provide a framework or a system that can generate landforms at different levels of detail. This category includes noise-based methods, example-based techniques (including machine learning), and interactive systems that rely on procedural brushes as well as sketching frameworks. Here, we review such methods and discuss the limitations below.

2.3.1 Noise-based techniques

The first terrain generation method was introduced by Musgrave *et al.* 1989 and defined the elevation as a fractal sum of noises (Perlin *et al.* 1989; Ebert *et al.* 1998), also referred to as *turbulence*. Let $n : \mathbb{R}^2 \rightarrow [-1, 1]$ denote a noise function with a frequency of 1, *i.e.* where n interpolates values or gradients defined at every integer position, we define the turbulence t as:

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

Where o denotes the number of octave in the fractal, a_i refers to the amplitudes, and φ_i to the frequencies. These two terms are defined as geometric series: $a_i = a_0 p^i$ and $\varphi_i = \phi_0 l^i$, where a_0 and φ_0 are the base amplitude and frequency, $l \in [0, 1]$ is the lacunarity, and $p \in [0, 1]$ denotes the persistence. The persistence and lacunarity defines how the amplitude and frequency decrease in the successive octaves, respectively. This definition of the turbulence, based on Perlin or Simplex noise (Perlin *et al.* 1989), allows to create smooth uniform landscapes, and subsequent works have been interested in generating more variations. Parberry 2015 tunes Perlin noise to ensure an exponential distribution of the gradient, with a view to fitting slope distributions to histograms observed on real terrains, but is still limited regarding the generated features. *Ridge noise* was introduced to better represent sharp crests and ridges,

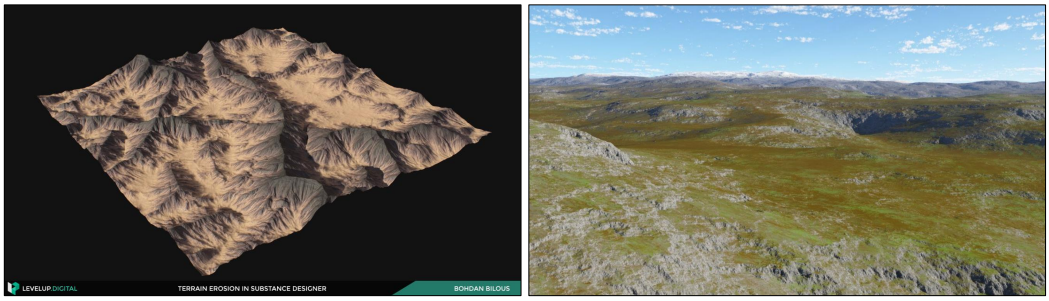


FIGURE 2.9: *Noise-based methods exhibit self-similar characteristics across their whole domain. There is thus a need for improving the terrain by adding hydraulic erosion landmarks (Musgrave* et al. *1989, left. Image from Bohdan Bilous), or modulating the noise function to better reproduce slope distributions of real terrains (Parberry 2015, right).*

and is simply defined by using an absolute value $r(\mathbf{p}) = 1 - |n(\mathbf{p})|$ (Ebert *et al.* 1998). Another possibility for introducing more variation is through the use of *multi-fractals*. These are defined by modifying the fractal sum with a function α, so that the amplitude at step $k+1$, denoted as a_{k+1}, should be weighted according to the value obtained at the previous octave:

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

Using this formulation means that lower elevations at step k, denoted as $t_k(\mathbf{p})$, scale down higher frequencies and leads to smooth valleys, whereas high values of $t_k(\mathbf{p})$ boost high frequencies to enhance mountain peaks with smaller details. Finally, another possibility to generate noise with variations is by applying deformations through the use of warping functions, usually made of rotations and translations, which transforms the point $\mathbf{p}$ using a 2×2 rotation matrix or a two-dimensional translation vector. Procedural noises are compact in memory (only the definition of the function must be stored), can be evaluated on the fly, and can indeed be used to generate mountain-like landscapes at a certain scale. A step forward in controlling the extent of terrain landforms was proposed by Génevaux *et al.* 2015, which uses compact noise primitives for defining the elevation, and operators to combine them and construct the final terrain.

The main limitation of noise-based methods is the self-similarity of the generated pattern leading to unrealistic terrains, which limits its use to a restricted domain. Noise is also completely scale-agnostic, which is both an advantage as it can be used for generating textures, landscapes or other volumetric effects, and a limitation as the generated features are not realistic from a geological point of view.

2.3.2 Faulting approaches

The faulting algorithm was introduced to generate fractal terrains (Mandelbrot 1983; Ebert *et al.* 1998). The method proceeds by repeatedly generating random vertical faults ϕ_i, and displacing the points upwards or downwards on either side depending on their distance to the faults $d(\mathbf{p}, \phi_i)$. More formally, let g denote a smooth step function parameterized by a radius of influence R, the elevation of the terrain is defined by summing the influence of the faults as:

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

Where a_i denotes the vertical displacement of the fault ϕ_i at step i. The faulting algorithm can be applied to a sphere for generating planets (Fournier *et al.* 1982a). As with noised-based approaches, several papers took interest in adding control over the process, for instance by controlling the elevation of certain region in the domain (Kamal *et al.* 2007). Although the methods differ, the result is similar to sum of fractal noise and suffer from the same limitations. Faulting approaches are also more computationally intensive as they cannot be evaluated on the fly, as opposed to noise-based methods.

2.3.3 Subdivision schemes

Subdivision schemes generate a terrain iteratively by subdividing the spatial data structure (usually a regular grid) to add more details. The midpoint subdivision algorithm was introduced by Fournier *et al.* 1982a; Fournier *et al.* 1982b, and recursively adds details by refining a grid and adding random displacements to the new points. Additional control is possible by constraining the subdivision for generating river networks (Kelley *et al.* 1988; Derzapf *et al.* 2011), or crest features

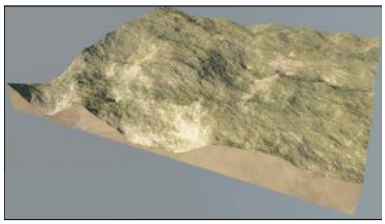


FIGURE 2.10: *8 steps of the diamond-square subdivision algorithm (image from Galin* et al. *2019).*

(Belhadj *et al.* 2005). Recently, subdivision schemes were adapted for the real time amplification of planetary terrains in Cortial *et al.* 2020. The method starts from an initial low resolution map of the main zones of the planet (mountain ranges, deserts, seas), refines a spherical mesh in real time depending on the viewpoint of the camera, and finally assigns elevations according to the low resolution map to create terrain landforms. The technique allows for a real time exploration of planetary-scale terrains up to a resolution of a few meters and generates varied terrain landforms. At the exception of Cortial *et al.* 2020, subdivision schemes lack user control over the generated features. Furthermore, none of these methods can represent volumetric landforms, as the presented subdivision schemes are based on elevation models.

2.3.4 Example-based techniques

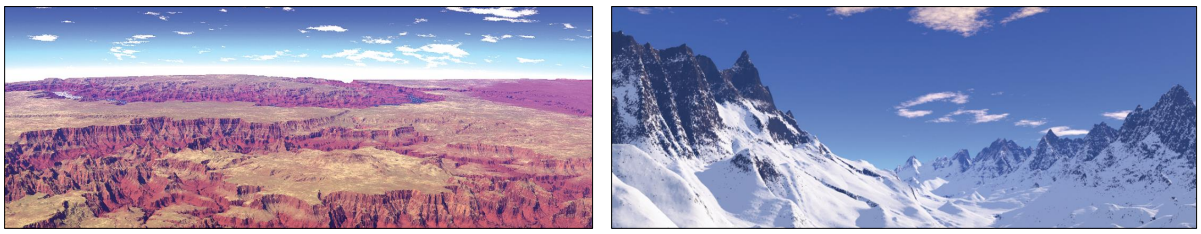


FIGURE 2.11: *Terrain generated by stitching patches from real exemplars from Zhou* et al. *2007. Given a carefully crafted dictionary, the method can reproduce varied landforms such as canyons and mountain ranges.*

Apart from noise-based techniques, there are other methods that are *scale-agnostic*. Example-based techniques are a powerful tool for generating realistic terrains by stitching real terrain patches (Brosz *et al.* 2007; Zhou *et al.* 2007; Tasse *et al.* 2012; Gain *et al.* 2015) or patches (Guérin *et al.* 2016) together (Figure 2.11). Machine learning methods (Guérin *et al.* 2017; Zhao *et al.* 2019; Zhang *et al.* 2022) learn a correspondence between a user sketch map and scans of real terrains, and thus belong to this category as well.

One limitation of example-based methods is that even though the generated terrain is built as a combination of real exemplars, there is no guarantee to get a realistic terrain in output. In particular, the drainage network of the final terrain might contain pits (cells with no outgoing flow), which are not realistic from a geological point of view. These methods are also heavily reliant on sourcing high quality digital elevation models (DEMs) for constructing the exemplar or training database. Even though it is theoretically possible to reproduce different archetypes of landforms using well-made dictionaries, researchers mostly concentrated their efforts on mountains and canyons, whereas other features such as deserts, rivers, cliffs and hills have been neglected. These mesoscale landforms require precise elevation data which may not be readily available, and using coarser maps as exemplars does not allow to reproduce features such as the sharp crests of sand dunes or escarpments in cliffs.

2.3.5 Editing and sketching frameworks

There are other papers that describe editing framework or sketching interfaces, with the vast majority also focusing on alpine mountain generation (Carpentier *et al.* 2009; Gain *et al.* 2009; Passos *et al.* 2013; Tasse *et al.* 2014). A notable exception is the feature-curve framework from (Hnaidi *et al.* 2010), which is capable of generating barchan dunes, canyons, and terrains with rivers enforced by the user. A diffusion algorithm operating on a multigrid is used to reconstruct the surface of the terrain. The gradient-based

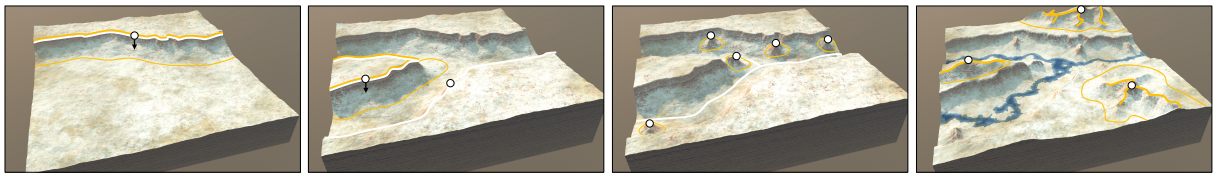


FIGURE 2.12: *Gradient-domain authoring (Guérin* et al. *2022) offers powerful tools to the user, where elevation (white) and gradient (yellow) constraints on points and along curves can be placed to create different landforms such as peaks, table mountains, crevasses and canyons.*

authoring framework presented by Guérin *et al.* 2022 exhibits similar capabilities and provides more intuitive control to the user, such as cut-and-paste operations and sketching tools that supports both point and curve constraints (Figure 2.12).

Conclusion

Scale-agnostic methods are useful for reproducing landforms at different scales (mainly at mesoscale and macroscale), but cannot reproduce the entire variety of terrain landforms. Noise-based techniques are limited to features of a certain scale, and do not contain any geological features such as erosion landmarks. Editing and sketching frameworks provide an efficient authoring environment for the user, but they cannot be used solely to create landscapes, as sketching an entire terrain from scratch is a tedious and time-consuming task. Overall, scale-agnostic methods are a complementary tool for generating and authoring virtual terrains but must be complemented with dedicated algorithms that focus on the generation of specific landforms, such as erosion patterns, canyons or desert features.

2.4 Mesoscale landforms

We classify mesoscale terrain features in the spatial range of $10\mathrm{m}$-$1\mathrm{km}$. This includes landforms that are usually modeled using elevation models such as gullies, ravines, and cliffs, but also the vast majority of volumetric landforms, including arches, overhangs, and hoodoos. We list existing methods for reproducing these terrain features below and also discuss the limitations of elevation models regarding the representation of mesoscale cliffs.

2.4.1 Gullies, ravines and floodplains

Gullies and ravines are usually present on steep slopes of mountains and hills. They are characterized by narrow passages that can span hundreds of meters in width and dozens of meters in depth. These landforms are the result of *hydraulic erosion*, where rainfall water progressively carved the erodible material when running down the terrain slope. Gullies and ravines are complementary to sediment floodplains, as the eroded material is transported downhill by water and deposited on flat areas of the terrain.

These terrain features have been extensively studied in Computer Graphics since the seminal work of Musgrave *et al.* 1989, which proposed the first model of hydraulic erosion. This method mimics the behavior of erosion on a discrete elevation model using simple transport rules, where sediment are carried by water running down the slope which progressively creates erosion landforms. Roudier *et al.* 1993 extended this approach to account for multiple materials with different geological properties to generate terrains with more variations. The main issue with these methods is that they do not work with

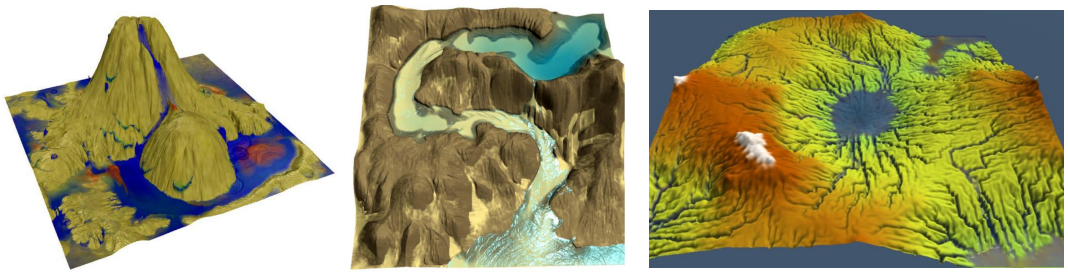


FIGURE 2.13: *Gullies and ravines generation has been an active subject of research in terrain modeling, with methods based on smooth particles hydrodynamics (Krištof* et al. *2009, left), shallow-water simulations (Šťava* et al. *2008, center) or grid-based hydraulic erosion (Jákó* et al. *2011, right).*

any physical units, and the transport rules are relatively simple when compared to a real fluid simulation. Thus, they belong to phenomenological approaches rather than accurate simulations of hydraulic erosion, even when computing the water flow from a velocity field as done in Chiba *et al.* 1998.

More physically-based methods were then developed to simulate the behavior and erosion action of water on a terrain, for instance using Navier-stokes equations (Neidhold *et al.* 2005; Beneš *et al.* 2006). Under the right initial conditions, these methods can theoretically reproduce varied terrain landforms, including gullies and ravines but also waterfalls and small meanders. However, they are computationally intensive and require a voxel representation, which limit their use to small domains. The shallow-water model used by Benes 2007 is a simplification of the Navier-Stokes equations and is a fast and simple method to simulate hydraulic erosion that can be efficiently implemented on graphics hardware (Mei *et al.* 2007; Šťava *et al.* 2008; Jákó *et al.* 2011; Vanek *et al.* 2011). In practice, a shallow-water erosion model is usually preferred to a full Navier-Stokes simulation, mainly for computational efficiency reasons which allows its use on larger domains.

Alternative approaches represent the fluid using smooth particle hydrodynamics (SPH) (Krištof *et al.* 2009; Skorkovská *et al.* 2015). These methods can reproduce visually convincing erosion landforms, but require hundreds of thousands of particles even for small terrains, thus they cannot be used for generating gullies and ravines on large mountain ranges.

Screes and debris accumulation

Independently of the underlying simulation method, hydraulic erosion is often coupled with material stabilization (also referred to as thermal weathering), which was first introduced by Musgrave *et al.* 1989 and later formalized with the layer field representation by Beneš *et al.* 2001. Thermal weathering describes the stabilization process of a granular material based on its repose angle, which is the steepest angle to which the material can be piled without slumping. This process is particularly important in the case of sediments carried by water and deposited on flat areas of the terrain, or in the case of sand dunes in desert landscapes. Using thermal erosion, sediments are treated as a granular material that stacks progressively in a physically plausible way.

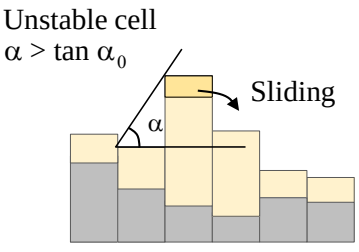


FIGURE 2.14: *Granular material stabilization process.*

2.4.2 Cliffs

A cliff is a vertical bedrock landform, generally defined by a steep angle. They are formed by erosion processes and can be found in numerous places such as coasts, mountains or along rivers. Cliffs are very important landforms in terrain modeling, with applications in urban planing (where it is important to understand the evolution of the cliff through time) and also in the entertainment industry (where cliffs are used as a central part of the gameplay). Cliffs can have different visual appearances, exhibiting clear and structured stratification, or a more uniform look depending on the properties of the underlying bedrock material. Thus, they are often composed of multiple volumetric features, such as small overhangs or visible strata of different sizes.

FIGURE 2.15: *A cliff with overhangs modeled in the Arches system (Peytavie* et al. *2009b).*

In practice, cliffs are usually represented with elevation models (Section 2.1.1), even though the steep nature of these landforms makes it challenging to represent with such structure due to the insufficient precision on the vertical parts of the terrain. The method described in Benes *et al.* 2005, which took interest in modeling table mountains with mesas and slopes of sediments, is a clear example of such limitation. Elevation models are well suited for representing planar areas, gentle slopes, and even mountains that are not too steep, but are limited when it comes to vertical parts (steep parts of mesas). Furthermore, overhangs and structured cliff faces are not easy to represent with elevation models.

Warping the terrain horizontally as described in Gamito *et al.* 2001 can potentially solve this issue, and has been used to represent overhangs near waterfalls in Emilien *et al.* 2015. However, the warping strength must remain small enough to avoid non-manifold configurations in the terrain, thus it can only represent small overhangs. To fully capture the volumetric landforms of a cliff, one must resort to voxels (Ito *et al.* 2003) or layer stacks (Peytavie *et al.* 2009b). These methods suffer from current volumetric models limitations, and are thus constrained to range-limited domains. Overall, capturing the complex microscale and mesoscale volumetric landforms of cliffs is a difficult task which is still an open research question.

2.4.3 Arches and overhangs

Arches are one of the most scenic volumetric landforms in virtual terrains. An arch (also called a bridge) is a mesoscale landform possibly spanning dozens of meters, commonly found near inland cliffs and coasts. They are usually formed from narrow bedrock formations with different materials, which are progressively eroded. An alcove is slowly created due to softer bedrock, which finally leads to an arch when it collapses on the ground. In the same category, we include large overhangs of dozens of meters (not the small scale overhangs found on cliff faces, discussed in Section 2.4.2) which are a common feature in coastal cliffs, created by the physical action of repeating waves hitting the cliff.

FIGURE 2.16: *Arches modeled by Peytavie* et al. *2009b.*

Modeling such visually arresting landforms has attracted the attention of Computer Graphics researchers for a while. As these features are volumetric at their core, they cannot be properly represented by elevation models, even when using

warping (Gamito *et al.* 2001). Existing volumetric solutions are most often based on authoring frameworks, and use feature curves (Becher *et al.* 2019), open-shape grammars (Dey *et al.* 2018), or sculpting brushes (Peytavie *et al.* 2009b) to create such volumetric landforms.

Interestingly, the automatic generation (by using procedural methods or simulations) of arches and overhangs have not been investigated, at the exception of Crespin *et al.* 2014 which modeled 3D erosion with generalized maps. However, the method is limited to small domains, is presented under relatively simple initial conditions, and is computationally intensive. In the second part of this thesis, we tackle the efficient modeling and procedural generation of arches and overhangs (Chapter 6).

2.4.4 Hoodoos

FIGURE 2.17: *Hoodoos generated by Jones* et al. *2010.*

A hoodoo (also called a goblin or a fairy chimney) is a tall vertical spire of rock formed by erosion. The alternating pattern of soft and hard bedrock layers leads to columns with lots of small and large overhangs. Hoodoos are particular terrain landforms that usually exhibit unique shapes carved by wind and hydraulic erosion (many have the shape of a mushroom) and sizes (from a few meters to dozens of meters). From a terrain modeling perspective, their very vertical aspect makes them a challenging landform that requires a volumetric model to be properly reproduced. Existing methods focus on the generation of a single hoodoo by using spheroidal erosion (Beardall *et al.* 2007; Jones *et al.* 2010) on a voxel model with different material properties to generate bedrock formation with diverse rounded shapes (Figure 2.17). While the generated hoodoos are visually convincing, the method requires a fine voxel grid to capture the mesoscale and microscale details of the shape.

Another interesting aspect of hoodoos is that they are often found in large fields, for instance in the Bryce Canyon National Park, Utah. Modeling such a vertical feature over large spatial zones is challenging for current volumetric models, and to the best of our knowledge, there are no methods for generating large fields of hoodoos in a virtual terrain, which is a problem we tackle in Chapter 6.

Conclusion

While there are numerous methods for generating mesoscale landforms on virtual terrains, there are still open challenges. In particular, steep landforms such as cliffs (Section 2.4.2) and volumetric features such as arches and overhangs (Section 2.4.3) can only be properly reproduced using fully volumetric or hybrid models, both of which have a high memory impact (Section 2.1.2). This limitation also forbids the creation of large-scale volumetric landforms, such as overhangs along coastal cliffs, or fields of hundreds of hoodoos. In the second part of this thesis, we introduce a new model based on implicit surfaces that provides an efficient representation of vertical areas of the terrain, which in turn allows to generate detailed cliff faces, overhangs, and arches over large spatial scales.

2.5 Macroscale landforms

Macroscale terrain features are classified in the spatial range of $1 - 100\text{km}$. They include popular landforms such as mountain ranges, rivers, and canyons usually represented by elevation models, but also underground karstic networks made of tunnels and chambers, which require a volumetric representation.

We demonstrate that the vast majority of techniques focus on the formation of mountain ranges and that several important macroscale landforms have been neglected.

2.5.1 Mountain and hill ranges

A mountain is defined as a steep portion of terrain with a peak that is the higher elevation point. While the difference between a hill and a mountain is largely subjective, hills are usually viewed as less steep and not as tall as mountains, and exhibit rounder shapes overall. A mountain or hill range is simply defined as a series of mountains or hills arranged together, usually following a line pattern. They are the result of complex geological processes operating over large (millions or billions of years) time scales, including tectonic forces, weathering, and glacial erosion.

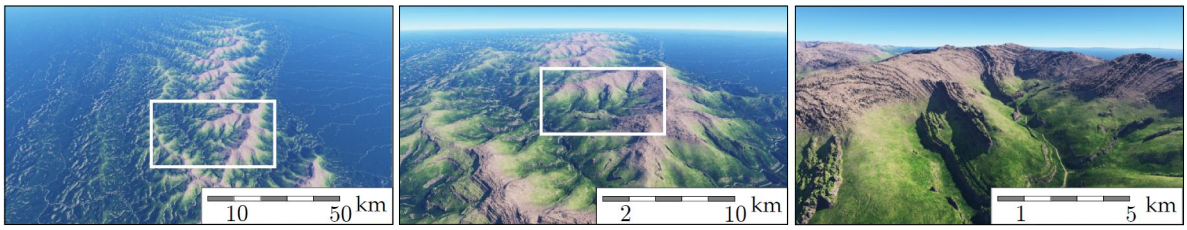


FIGURE 2.18: *Large-scale terrains with realistic mountain ranges generated with the stream power simulation from Cordonnier* et al. *2018a.*

Mountain ranges have been a popular research subject in the past decades, as they are one the most popular terrain landforms. Until recently, the majority of methods for generating mountain ranges were either based on noise or a combination of noise-based primitives (Musgrave *et al.* 1989; Génevaux *et al.* 2015), or example-based synthesis techniques (Zhou *et al.* 2007; Tasse *et al.* 2012; Gain *et al.* 2015). We discussed these scale-agnostic methods in Section 2.3, and focus here on geologically-based approaches such as the work of Cordonnier *et al.* 2016. They introduced the Stream Power equation to the Computer Graphics community, which models the evolution of mountains as an equilibrium between tectonic uplift (which grows mountains) and fluvial erosion (which carves erosion landmarks) using the following equation:

$$\frac{\partial h}{\partial t} = u - s^n \, a^m$$

Where u denotes the uplift, s is the local slope, a is the drainage area, and n and m are exponents that depends on the type of the terrain. This equation states that the rate of change of surface topography is controlled by the balance between the uplift u and the fluvial erosion term $s^n \, a^m$. Realistic mountain ranges with dendritic patterns spanning hundreds of kilometers can be reproduced by this method. Subsequent work (Cordonnier *et al.* 2018a) incorporates the effects of different geological strata and presents a convenient authoring framework where users can move around tectonic faults in three dimensional space and see the result interactively (see Figure 2.18).

However, even though the represented domain is larger than any other terrain generation methods, these simulations are computationally intensive and thus limited in precision. Furthermore, as with other simulation techniques, user control is indirect through the sketching of the uplift map, and therefore limited. Providing direct control over the location of ridges and valleys is still an open research question. Another interesting topic would be to take into account other important geological processes such as glacial erosion as done in Mieloszyk 2017.

2

2.5.2 Canyons

A canyon is a path between two steep cliffs, possibly spanning hundreds of kilometers in length and dozens of meters in depth. They are often also composed of multiple terraces (or plateau) which creates an interesting and complicated landscape. Canyons are usually formed by the erosive action of a river which follows the canyon trajectory. The different bedrock materials combined with erosion result in microscale and mesoscale overhangs in the sides of the canyon. Canyons are also a popular terrain landforms due to visual arresting occurrences such as the Grand Canyon in Arizona, or the Gorges de l'Ardèche in France.

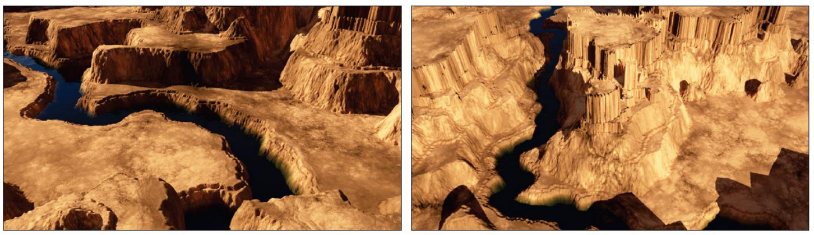


FIGURE 2.19: *Canyons generated with the method from Carli* et al. *2014. Representing the steep nature of table mountains remains challenging using with elevation models.*

As with cliffs, canyons are usually represented and generated with elevation models. The method presented in Carli *et al.* 2014 focuses specifically on the generation of canyons from an initial fractal terrain (Figure 2.19). The elevation is first clamped to create terraces at multiple altitudes, and mesas as well as the river trajectory are then generated using a shortest path algorithm. The method greatly suffers from the lack of precision of elevation models on steep areas of the terrain.

2.5.3 River networks

Rivers are among the most complex phenomena on earth and have been thoroughly studied by geologists and hydrologists. A river can take numerous shapes, from a small and narrow stream near its source, to a delta possibly spanning dozens of kilometers when connecting to the sea, while going through complex meandering or braided systems in the floodplain. These different stages exist because of the differences in flow, terrain topography and climate conditions along the path of the river.

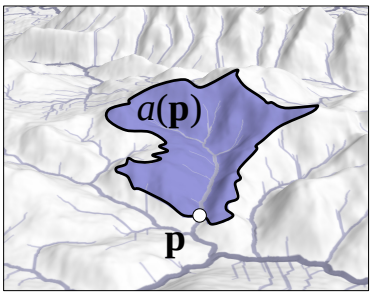


FIGURE 2.20: *Drainage area of a point and its highlighted watershed.*

Close to the river network is the notion of drainage area of a terrain. The drainage area is commonly used in geomorphology to characterize real terrains, and is defined as the upstream area (also called watershed) draining through a point $\mathbf{p}$ when following the gradient of the terrain. In other words, it is a measure of how much water is flowing towards a given point of the terrain. At the exception of lakes, pits (cells with no outgoing flow) are considered very rare in real terrains. Thus, one way to improve the realism of a virtual terrain is to ensure that it does not contain any pit cells so that the flow is correctly directed towards the borders of the domain. Having a hydrologically-correct drainage area is important, for instance in the case of river detection where trajectories with the highest flows must be detected to compute the main river channels, as done in Peytavie *et al.* 2019.

The notion of drainage area was first used in Computer Graphics by Kelley *et al.* 1988, which enforces the precise trajectories of rivers in the generation process by providing an initial drainage system. Later

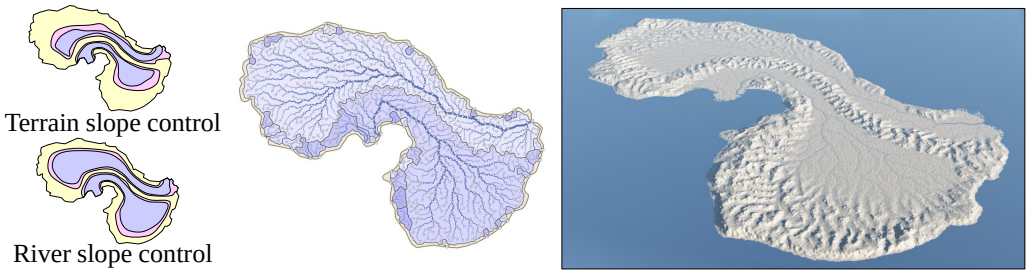


FIGURE 2.21: *Génevaux* et al. *2013 procedurally computes the entire river network, which is then used to create the terrain that conforms to the specified hydrology. The smooth surface if the terrain is reconstructed using a construction tree of procedural primitives merged together to define the elevation.*

works from Génevaux *et al.* 2013 and Teoh 2009 generate the entire hierarchical drainage network, represented as a geometric graph, over an input domain. After covering the entire domain with rivers, the method of Génevaux *et al.* 2013 classifies the edges of the graph according to Rosgen classification of rivers (Rosgen 1994). The surface of the terrain is finally reconstructed by using a construction tree of primitives (Section 2.1.1) representing the riverbed and the surrounding mountains (Figure 2.21). These methods are useful for generating a terrain that complies with hydrological constraints, but do model the evolution of the network throughout time. Also, some important river landforms such as deltas, meandering and braided rivers are not reproduced. Meanders could be generated by placing carefully-designed curvilinear primitives (as done in Peytavie *et al.* 2019), but the result would only be static and would not reflect the complex and chaotic nature of the phenomena.

A dedicated method is required to model such complex river landforms. However, existing simulations (Krištof *et al.* 2009; Šťava *et al.* 2008; Skorkovská *et al.* 2019) are computationally intensive and cannot be used to simulate the evolution of an entire river network covering hundreds of kilometers. Thus, simulating the dynamic evolution of the different stages of rivers remains an open research question. In Chapter 3, we address the problem of simulating meandering rivers over an entire network.

2.5.4 Karstic networks

A karstic system is the geological name for a cave network. Karsts are characterized by underground networks composed of conduits and caves that have grown by the dissolution of the bedrock. Karstic systems exhibit a large variety of visually appealing and complicated mesoscale and microscale landforms called speleothems, such as stalactites and stalagmites, draperies, columns and more. At a larger scale, conduits vary in size from a few centimeters to several meters wide with diverse shapes (such as *keyhole* or *canyon* tunnels), and can spread across dozens of kilometers under the surface.

FIGURE 2.22: *Cave made of tunnels, bridges, and rocks created by authoring with a layer-stack model (Peytavie* et al. *2009b).*

Karsts cannot be modeled using elevation models and require a volumetric structure. There are a few methods that focus on the microscale landforms found in caves, for instance using procedural techniques for placing stalagmites and stalactites (Cui *et al.* 2011). Layer stacks have been used to create caves with columns and tunnels by Peytavie *et al.* 2009b, but the global structure of the tunnel is reproduced solely by authoring, which can be tedious (Figure 2.22). On the other hand, the generation of the large-scale karstic network has been done using procedural L-system, as described in Mark *et al.* 2015. The resulting trajectories are then carved in a voxel model, and the smooth surface of the cave is reconstructed using polygonization techniques (Wyvill *et al.* 1986; Lorensen *et al.* 1987). This approach generates caves with insufficient precision due to the underlying voxel grid and does account for the geological characteristics of the terrain.

Pytel *et al.* 2015 proposed a two-stage simulation pipeline for modeling large-scale karstic networks. While they account for geological parameters such as rock porosity, the resulting tunnels are defined as a set of connected cubes in the underlying voxel model, and no solution is provided regarding the synthesis of the detailed mesoscale geometry of the tunnels. Another geologically-based approach was proposed by Franke *et al.* 2022, which simulates water flowing through a 3D grid to generate the trajectories of the karstic conduits. This method is able to reproduce different archetypes of karstic networks, such as rectilinear or anastomotic mazes (see Figure 2.23). However, important geological factors such as inception horizons are not taken into account and no control is provided over the generation process. Finally, the mesoscale geometry of the tunnels does not exhibit plausible shapes, as it is also simply generated by polygonizing the voxel grid (Wyvill *et al.* 1986; Lorensen *et al.* 1987).

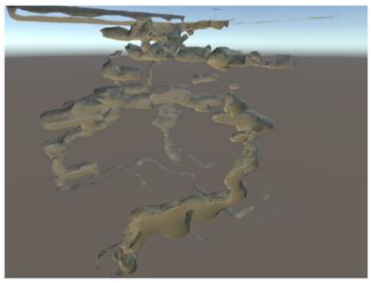


FIGURE 2.23: *Rectilinear karstic network generated by Franke* et al. *2022.*

2.5.5 Deserts

Deserts are regions with low water supply. Depending on the amount of precipitation, they are classified as hyper-arid, arid, or semi-arid. Deserts can span hundreds of kilometers and exhibit different landforms such as sand dunes, yardangs and steep cliffs or canyons. As the water supply is usually low, aeolian processes are central in the formation of desert landforms.

The vast majority of desert features have been neglected from the Computer Graphics community, with only a few papers interested in simulating sand ripples (Onoue *et al.* 2000; Beneš *et al.* 2004). While both ripples and dunes are formed by the action of the wind, they are two different landforms with vastly different scales which cannot be simulated with the same technique. Furthermore, landforms such as yardangs and ventifacts cannot be modeled by these methods. Thus, generating convincing macroscale desert scenery remains an active area of research.

Conclusion

Macroscale landforms in virtual terrains are becoming increasingly important because of the need to model, simulate and author larger worlds. While there has been numerous works interested in generating mountain ranges (Section 2.5.1), other macroscale features such as river networks (Section 2.5.3), karstic systems (Section 2.5.4), and desert landscapes (Section 2.5.5), received less attention. These features cannot be fully captured by existing methods and may require dedicated procedural or simulation techniques. In the first part of thesis, we introduce new simulation methods inspired by geomorphology for simulating desert scenery (Chapter 4), and meandering rivers (Chapter 3).

2.6 Conclusion

Terrain generation methods progressively evolved over time from procedural techniques to simulations, mainly because of the increase in computational power which allowed the use of more complex models elaborated by scientists. Even though procedural and phenomenological approaches provide great control to the user, they cannot always fully capture the complex dynamics of terrain landforms. Therefore, they are often complemented with simulations. Over the last decade of research in terrain modeling, the tendency has been to move towards more geologically-based techniques, mainly because of the ever increasing need for realism which cannot be fulfilled without understanding how terrain landforms emerge. Geologists and geomorphologists have been studying the underlying processes responsible for the formation of terrain features, and such knowledge is a great source of inspiration for the field of Computer Graphics.

We proposed a new classification of terrain generation methods based on the spatial scale at which they operate and the landforms they are trying to reproduce. Considering the macroscale, the majority of techniques has focused on the generation of mountain ranges. Features such as sand dunes (and more generally desert landforms) as well as meandering rivers, canyons and glaciers have been neglected, even though they represent a non negligible percentage of earth landform. In Chapter 3 and Chapter 4, of this thesis, we focus on the simulation of meandering rivers and deserts, which are highly dynamic phenomena that shape the landscapes on a yearly basis, while spanning dozens of kilometers at the same time. We study existing numerical models from Geomorphology and propose new techniques adapted to the needs of Computer Graphics with direct control tools for artists.

Another crucial realization of our classification lies in volumetric terrain landforms, such as arches and overhangs. These have been studied extensively, but as demonstrated in Section 2.1.2, existing volumetric models (voxels and layer stacks) are limited by their memory requirement, which forbids their use for large terrains. There is thus a need for a compact representation of volumetric features, compatible with elevation models used for representing large-scale terrains. In Chapter 5, 6, 7 and 8, we present a new procedural model for volumetric terrains based on signed distance functions. We study volumetric landforms across the whole range of scales (macroscale, mesoscale and microscale) and develop new primitives and operators suited for representing detailed terrain features.

Part I

Simulation of dynamic phenomena on terrains

Abstract

TODO: simulation of highly dynamic phenoma, focus on short time scale and large spatial scale, etc...

Chapter 3

Simulating meandering rivers

Contents

FIGURE 3.1: *Given an input terrain, we synthesize an initial river network by breaching the elevation to avoid unwanted pits and then computing the drainage area. We then simulate meandering rivers and take into account the slope of the terrain to preserve steep slope from meandering. The computation of the river network allows us to approximate the sediment deposition processes, which in turn, combined with the wetness maps derived from the river channels,*

3.1 Introduction

Researchers have made considerable progress toward developing efficient methods for synthetic terrain generation. Procedural models, erosion simulations and image-based synthesis have proved to be powerful tools to create virtual terrains. Existing methods most often focus on the creation of mountainous landforms with steep slopes, such as dendritic mountain ranges and erosion landmarks such as gullies and ravines. In contrast, the generation of smooth sedimentary valleys with complex river networks received less attention.

Numerous methods exists for realistic rendering of water bodies (**???**; **???**; Peytavie *et al.* 2019), however the generation of river trajectories is still an open research question. The challenge stems from the fact that rivers are highly dynamic objects, especially in sedimentary valleys where the overall topography is flat and the river exhibits a *meandering* pattern. The lateral migration of meanders can be up to $25\mathrm{m}$ a year, thus studying and understanding their behavior is an important subject in urban planning.

Modeling river networks on virtual terrains is a complex task, usually accomplished with dedicated techniques that first generate the river system and the terrain that conforms to the specified hydrology in a second time (Kelley *et al.* 1988; Génevaux *et al.* 2013). Another method from Peytavie *et al.* 2019 computes the river network from an existing real or synthetic terrain, carves the trajectories in the bedrock, and finally generates a plausible water surface. However, the resulting rivers are static and their evolution through time is not modeled. To the best of our knowledge, there are no methods for simulating the complicated evolution of river trajectories, and especially meandering patterns which are among the most dynamic ones. In this chapter, we introduce a new simulation method to reproduce the large-scale meandering behavior of rivers. We use a curvature-based approach to deform the trajectories, taking into account environment parameters such as the local slope, upstream water discharge, and control fields. We reproduce well-known phenomena such as complex bend development, oxbow lake formation due to cutoffs, and avulsion events, leading to a terrain with realistic river trajectories. Finally, we show how such simulation can be used to compute abiotic parameters, useful for later determining the surrounding ecosystem on the terrain. We provide several direct and indirect control tools: pn top of modifying the topography of the terrain as well as different control fields in real time, the user can also prescribe the precise trajectories or place specific river junction patterns to improve realism.

The main contributions outlined in this chapter include:

1. A curvature-based approach to simulate the behavior of meandering rivers over a large-scale network while taking into account environment parameters.

2. A procedural approach for computing abiotic parameters from the entire simulation history, which are used to compute vegetation cover in the terrain.

3. Indirect controls such as attractive and repulsive points for influencing the simulation, and intuitive direct tools for modifying the terrain, and prescribing the precise trajectories of the river.

4. An interactive application showing the capabilities of our system. We demonstrate that our simulation can be smoothly integrated in a production pipeline, as it is compatible with other state of the art terrain modeling techniques.

The work presented in this chapter has not yet been published and is thus a work in progress.

3.2 Geomorphology background

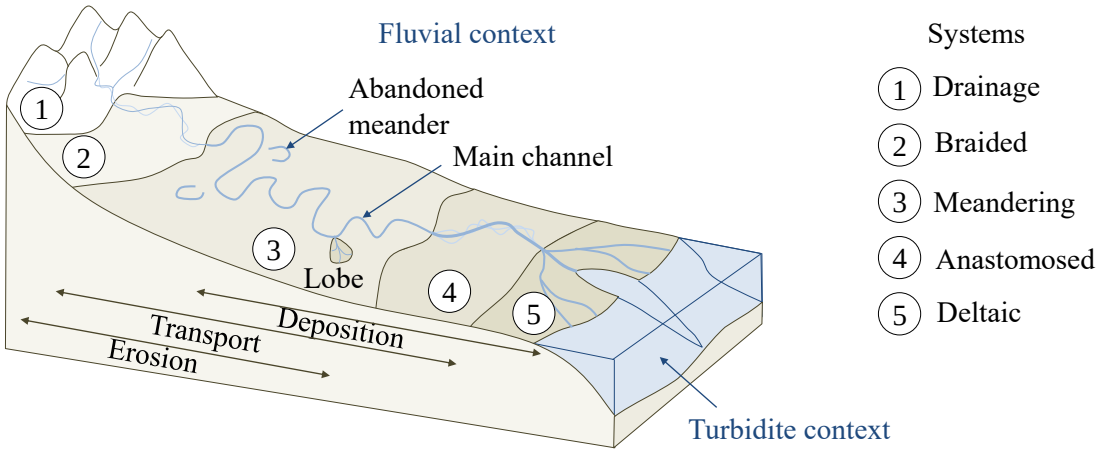


FIGURE 3.2: *Background on the geomorphology of rivers: our work focuses on the meandering systems, which can be found in plains.*

Rivers are among the most dynamic phenomena on earth. They can take a wide variety of shapes such as braided, meandering or anastomosed, depending on their location (Figure 3.2). Each shape is the result of different physical processes, and their understanding is an important research area in geology. In particular, lots of work in geomorphology has been dedicated to the meandering stage of the river, as meanders can migrate several meters per year (Figure 3.3). Interestingly, meandering patterns have been observed both in turbiditic (underwater) and fluvial context, as the governing processes share some similarities. In this chapter, we focus specifically on the fluvial part of the river, but the presented method could also be applied in a turbiditic context with some adjustments. The basic mechanism of meanders consists of erosion on the outer bank of the river, and deposition on the inner bank, which leads to a progressive lateral migration of the riverbed (also referred to as channel). The migration process has been found to be highly dependent on the river curvature (Howard *et al.* 1984).

Prior work on meandering rivers. The simulation of meandering rivers is an ongoing subject of research in geomorphology. We briefly review important papers from this area below.

Seminal work on meandering rivers (Ikeda *et al.* 1981; Howard *et al.* 1984) identified a non-linear relation between channel lateral migration and the local and upstream curvature. They also propose the first algorithm to model neck cutoff events, which are a key stabilization process in meandering

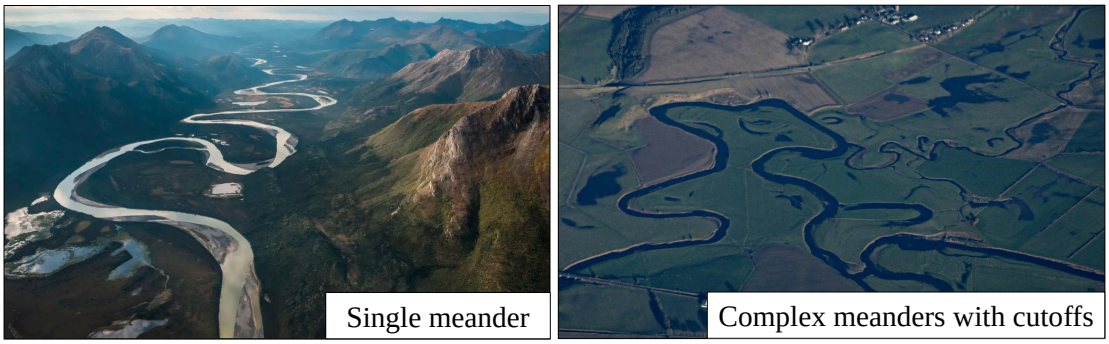


FIGURE 3.3: *Examples of real meandering rivers. Left shows a meander constrained within mountain ranges, and right shows a more complex meandering pattern with oxbow lakes.*

systems. Latter work focused on specific phenomena such as lateral or downstream bend migration (Posamentier *et al.* 2003), channel bend retro-migration (Nakajima *et al.* 2009), or avulsion (Pyrcz *et al.* 2009; Rongier *et al.* 2017). Recent work from (Sylvester *et al.* 2019) analysed the time-evolving behavior of real meanders using precise river elevation data. Their results suggest that lateral migration may follow a more simple linear relation with local and upstream curvature.

While the majority of these methods focus on the planar evolution of the river trajectory, meanders also modify the elevation of the surrounding terrain, also known as aggradation (or vertical migration), which moves the channel upward and thus require a more complex 3D simulation (Peakall *et al.* 2000; Rongier *et al.* 2017). However, this phenomena operates at the microscale (in the order of millimetres per year), and can thus be neglected in our context where we aim at modeling large-scale river networks spanning dozens of kilometers.

Existing work in geomorphology usually study the evolution of a single idealized channel through time, which is a key limitation in our context where artists generally aim at producing complex networks. In this chapter, we present a method for simulating meandering rivers on the entire terrain. We simulate the evolution of the trajectory on each channel, taking into account local slope and user defined constraints. Particularly, we focus on aspects such as interactivity and user control, while retaining the realism of the simulation.

3.3 Overview

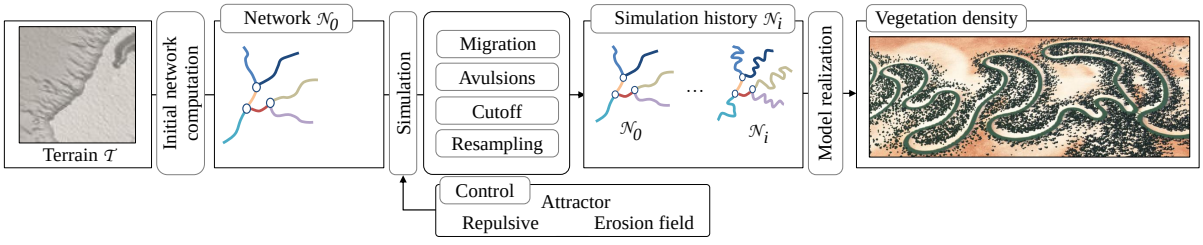


FIGURE 3.4: *Overview of our method. We first compute an initial river network $\mathcal{N}_0$ from the input terrain, and then simulate its evolution through time with processes such as migration, avulsions and cutoffs. We finally resolve collision between river sections. The network history allows us to compute abiotic parameters, which are finally used to create a plausible vegetation cover on the terrain.*

Starting from a terrain $\mathcal{H}_0$ defined by an elevation function $h : \mathbb{R}^2 \to \mathbb{R}$, an initial river network graph $\mathcal{N}_0$ must be provided (Figure 3.4). It can be computed automatically as described in Peytavie *et al.* 2019, or provided by the user. At the heart of our model is the representation of the river network as a set of channels $\{r_i\}$. A channel is defined by a discrete set of equally spaced control points $\mathbf{p}_k \in r_i$, and an associated width w and depth d (Figure 3.5). These can be automatically computed from well known power laws in geomorphology (Dunne 1978), which relates the drainage area to the flow and width of the river channel. Points within a channel have exactly one upstream and downstream neighbor, except for the first and last points, which may be linked to other channels in the network.

Starting from the initial $\mathcal{N}_0$, the simulation computes new networks $\mathcal{N}_i$ at each step. It is performed independently on each channel of the network (Section 3.4), and is separated in four processes (see Figure 3.4). First, we compute *channel migration* which moves points of the curve laterally by taking into account local and upstream curvature, terrain slope and environment conditions. Then, we simulate punctual events such as *cutoff* and *avulsion*, which are crucial processes in meandering systems that can drastically modify the trajectory of the channel. We then deal with collisions between channels, which we express as simple topological operations on the river graph (Section 3.5), and perform a resampling step to ensure that points are equally spaced with a given section. The exact trajectory of the river is finally constructed as the piecewise cubic curve passing through the control points of the channels.

Ecosystems and rivers are known to be interconnected processes, however simulating their interactions would be too complex. Instead, we decouple the ecosystem generation and use the recording the networks $\mathcal{N}_i$ of the simulation to compute abiotic conditions such as the wetness index. These abiotic parameters can then be used to determine the placement of vegetation in the final terrain (see Section 3.6).

We incorporate several level of controls in the simulation (Section 3.7). On top of providing the initial network $\mathcal{N}_0$, the user can place sparse constraints such as attractive or repulsive points, which locally modify the river trajectories. The elevation can also be modified interactively to create gorges or canyons, constraining the river path. We reproduce several known phenomena such as complex bend formations, downstream migration of meanders, oxbow lakes with cutoffs and new channel carving with avulsion.

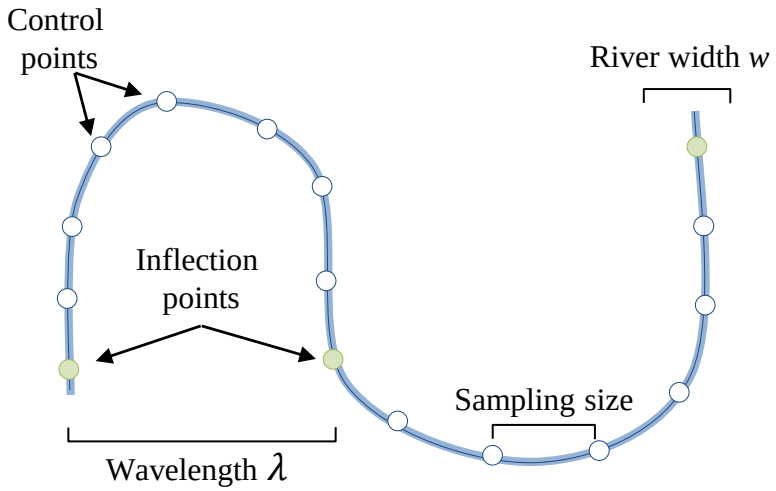


FIGURE 3.5: *A channel is characterized by width w_i, its depth d_i and the centerline, which is defined as a set of equally space control points along the trajectory.*

3.4 Simulation

For a given channel, the simulation proceeds to compute new positions for the control points of the river Γ. Each point $\mathbf{p}$ migrate towards the direction of the normal to the channel centerline $\mathbf{n}$:

$$\mathbf{p}(t + \delta t) = \mathbf{p}(t) + \mathbf{n}\, \mu(\mathbf{p}(t))\, \delta t \tag{3.1}$$

With δt the time step, and μ is the migration rate (Section 3.4.1). The challenge stems from the computation of the migration rate μ, which we detail in Section 3.4.1. After displacing the points of the channel, we look for cutoff and avulsion events, which can drastically change the river path (Section 3.4.2). Finally, we perform a resampling step to ensure that points along the channel are equally spaced, which is crucial for the stability of the simulation (Section 3.4.3).

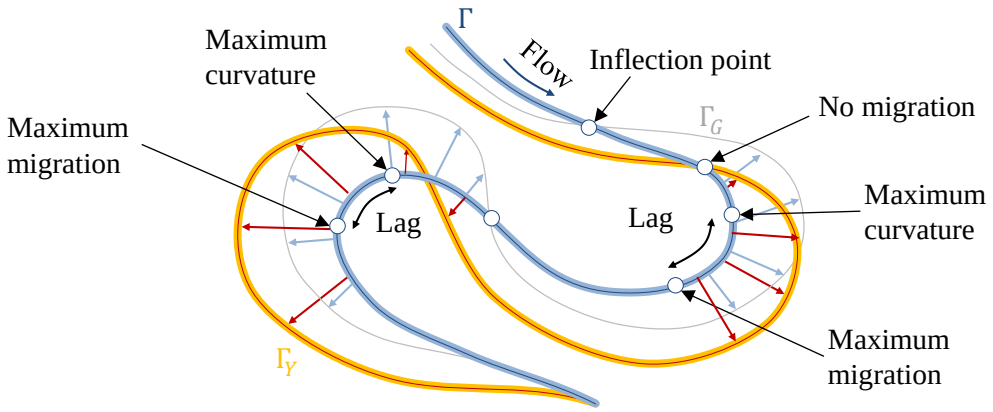


FIGURE 3.6: *Comparison between the local migration rate μ_0 and the global migration rate μ, taking into account the influence of the upstream curvature. The global migration rate introduces a lag between the maximum curvature point and the maximum migration point.*

3.4.1 Migration rate

The migration rate of a point along the channel denotes its movement speed through time. It was found to be directly correlated to the curvature of the river (Howard *et al.* 1984). Put simply, high curvature points are linked to the fastest moving part of the river. Let $\Gamma : \mathbb{R} \to \mathbb{R}^2$ denote a parameterized curve in the plane, the local curvature $\phi_\Gamma(u) = (x(u), y(u))$ is defined as:

$$\phi_\Gamma = \frac{x'\, y'' - y'\, x''}{(x'^2 + y'^2)^{3/2}} \tag{3.2}$$

Where x' and x'' (respectively y' and y'') denote the first and second derivatives. In our case, the channel curve is defined by a set of discrete points, thus derivatives are approximated using central differences. The local migration rate μ_0 is then defined by scaling the curvature according to the river width w and the migration rate constant k_1:

$$\mu_0(u) = w\, k_1\, \phi_\Gamma(u) \tag{3.3}$$

The linear scaling with w implies that the frequency of meander bends is proportional to the dimension of the river (width and depth), which has been identified and reported in multiple studies in geomorphology (Leopold *et al.* 1960; Williams 1986). Our simulation conforms to this observation, as discussed in Section 3.8.2.

Using the local migration model transforms the original curve Γ into a new curve Γ_G with increased bends (see Figure 3.6, blue and grey curves). In this case, the maximum curvature point corresponds to the maximum migration point. Inflection points, *i.e.* such that $\phi(\mathbf{p}) = 0$, correspond to the intersections between Γ and Γ_G. However, meanders are also influenced by upstream trajectories, as high curvature

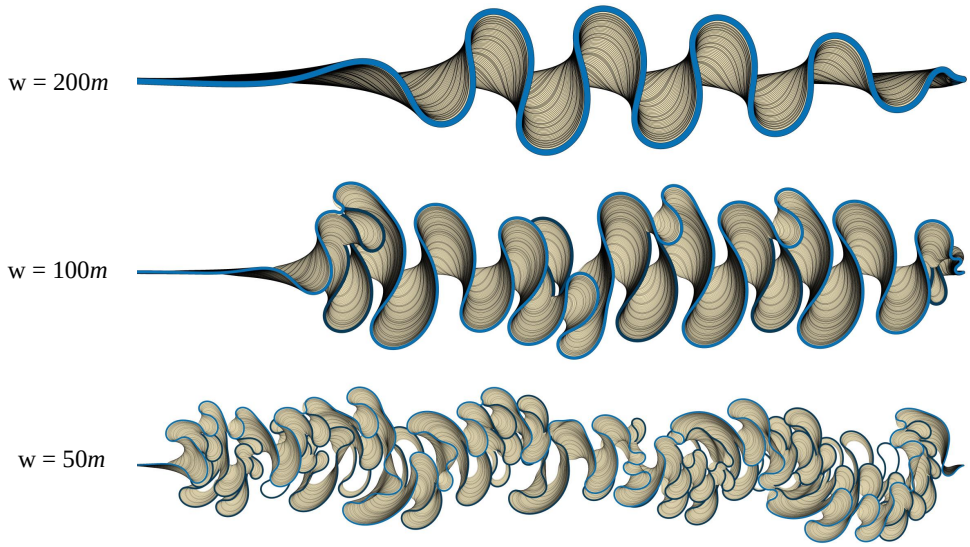


FIGURE 3.7: *Comparison between meandering rivers of different widths (*100 *years simulation): the meander size is directly related to the river width w, which in turn follows a power law with the drainage area a.*

upstream bends tend to increase the migration rate of a given point (Howard *et al.* 1984). Our model takes inspiration from (Sylvester *et al.* 2019) by taking into account the upstream curvature to compute a more accurate global migration rate μ. Let σ denote the sinuosity of the channel, which is computed as the ratio between the length of the curve l and the distance between the first and last points of the section:

$$\sigma = l/\|\mathbf{p}(0) - \mathbf{p}(1)\| \qquad l = (\int_{\zeta=0}^{\zeta=1} \|\nabla c(\zeta)\| \mathrm{d}\zeta) \tag{3.4}$$

The global migration rate for a point $\mathbf{p}$ is defined as:

$$\mu(\mathbf{p}) = \omega\, \mu_0(\mathbf{p}) + \sigma^{-2/3} \left[\gamma \int_0^{\infty} \mu_0(\mathbf{p} - \zeta) k(\zeta) d\zeta \right] \left[\int_0^{\infty} g(\zeta) d\zeta \right]^{-1} \tag{3.5}$$

The variable ζ denotes the upstream distance from the point $\mathbf{p}$, and ω, γ are weighting parameters. Multiplying by σ implies that a meander with lots of bends will migrate faster. The kernel function k is an exponential weighting function:

$$k(\zeta) = e^{-\alpha\zeta} \tag{3.6}$$

The term α is defined from a friction factor $\mathcal{C}_f$ and the river depth D, assumed to be uniform along the channel:

$$\alpha = 2\mathcal{C}_f / D \tag{3.7}$$

It defines a relation between the river depth D and the influence of upstream points: a point in a deeper river is influenced by points further away upstream.

The definition of the global migration rate μ in Equation 3.5 introduces a lag between the maximum

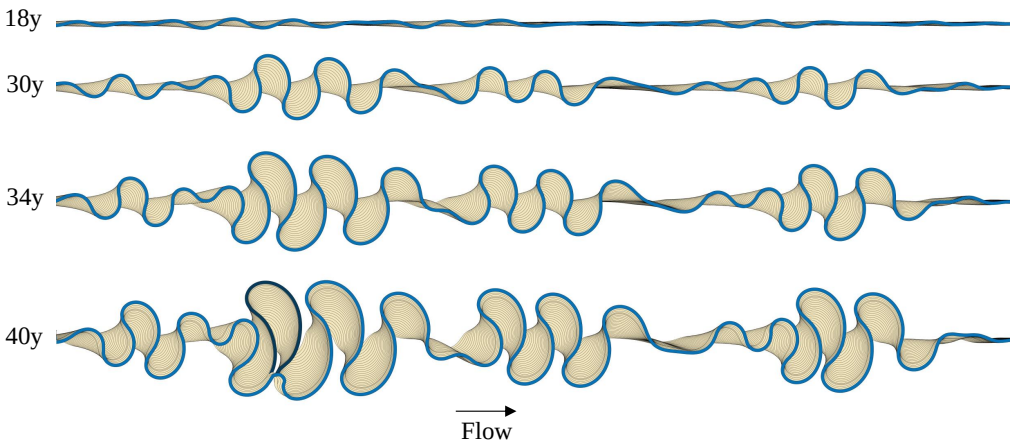


FIGURE 3.8: *Snapshots of our meandering river simulation on a single channel. The latest trajectory is shaded in light blue, while older paths are shaded in brown. Oxbow lakes due to cutoff events are in dark blue.*

curvature point and the maximum migration point, transforming the original curve Γ into a new curve Γ_Y (see Figure 3.6, blue and yellow curves). Using this formulation, we are able to reproduce well-known phenomena such as downstream migration, where meander loops progressively move downstream as time passes (Figure 3.8).

3.4.2 Catastrophe events

The dynamic of meandering rivers does not only depend on the migration of the channel, but also on catastrophe events. In particular, cutoffs (Howard *et al.* 1984), the presence of crevasses, and avulsions (Cojan *et al.* 2005) are known to play a key role in the evolution of the river.

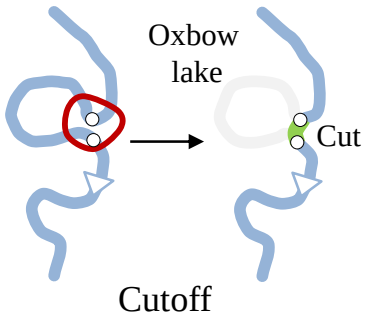


FIGURE 3.9: *Cutoff event leading to an oxbow lake formation.*

Cutoffs (also called neck-cutoffs) occur when the channel starts to intersect itself, due to the high local curvature of a river. When a cutoff occurs, the channel abandons its current trajectory, which becomes an oxbow lake (see Figure 3.9) and continues onto a new path. The oxbow lake can remain partially filled with water, or dry out depending on environment conditions. Vegetation may grow in these locations due to the water stored in the ground. Cutoffs events are fundamental process of meandering rivers as they regulate the bend formation.

In our simulation, we trigger a cutoff event when the distance between two points within a channel is inferior to the river width w. An edge is created between these two points, and the abandoned part of the meander is removed from the channel and saved in the simulation history. The cutoff process during the simulation can be seen in Figure 3.10. An interesting aspect of oxbow lakes is their influence on the surrounding landscape. Figure 3.11 shows the history of oxbow lake formation through the simulation. The channel belt, which is the embedding of all previous channel trajectories, mainly depends on the river width and local terrain topography. In Section 3.6, we exploit the simulation history of oxbow lake to compute abiotic parameters which can be used for placing vegetation in the terrain.

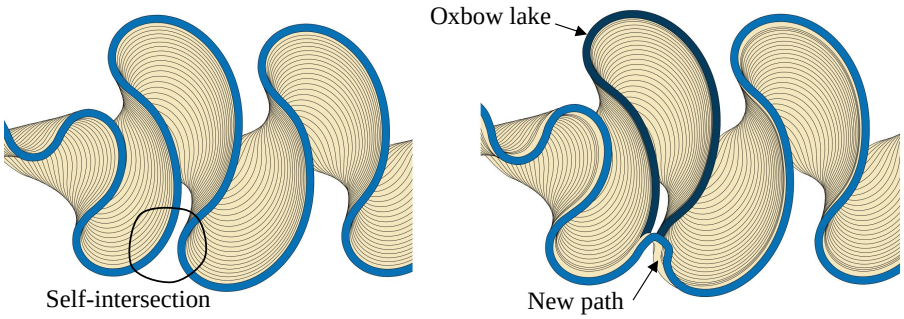


FIGURE 3.10: *Showcase of the formation of an oxbow lake where a meander start to intersect itself, leading to an abandoned channel.*

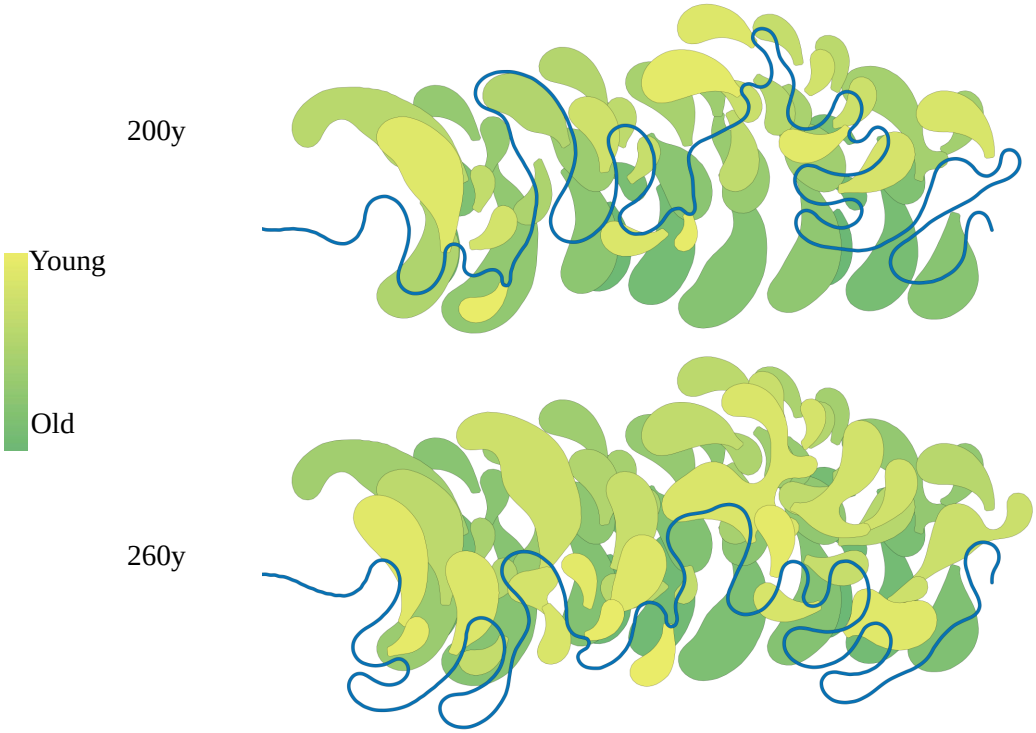


FIGURE 3.11: *History of oxbow lakes through the simulation, shaded from oldest (green) to youngest (yellow). The union of all ancient trajectories form the meander belt.*

Crevasses occur when water starts to slowly exceeds the channel capacity. They are characterized by large sediment lobs that deposit progressively at the location of overflow. They usually occur in locations of high curvature (Figure 3.13, left), but the exact formation process remain an active area of research in geomorphology. We model crevasses stochastically in our simulation. At each step, we look for points with a curvature above a given threshold t_c, and trigger the creation of a crevasse based on a probability ρ_c. Knowing the location of crevasses is interesting for adding procedural details on the terrain. We procedurally distribute sediments at crevasses in a circular domain, and perform material stabilization (Musgrave *et al.* 1989) on the deposited material (Figure 3.12). When a crevasse is created, it becomes a candidate for a future avulsion event, which drastically changes the river trajectory (Slingerland *et al.* 2004).

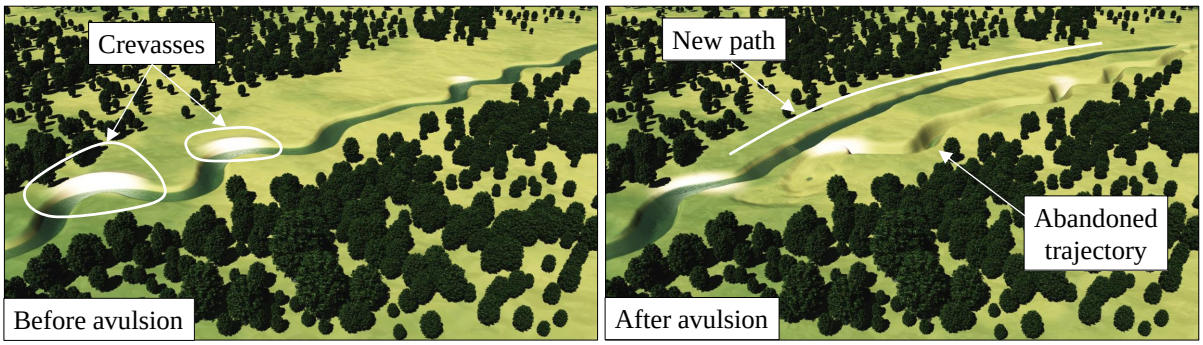


FIGURE 3.12: *We procedurally place sediment lobs at the location of crevasses (left), and stochastically trigger avulsion events which modify the trajectory of the channel.*

Avulsions trigger at the location of crevasses. The exact triggering conditions of avulsions are still not well understood, but they may be due to an intense rain causing an overflow in the channel. The upstream part of the channel remains unchanged, and the downstream part is abandoned and a completely new path is formed (see Figure 3.13, right). Simulating avulsion is difficult, as carving a new trajectory while ensuring a correct flow may not always be possible depending on the terrain topography. One solution would be to use an anisotropic shortest path algorithm, but this would be computationally intensive. Thus, we adopt a procedural approach.

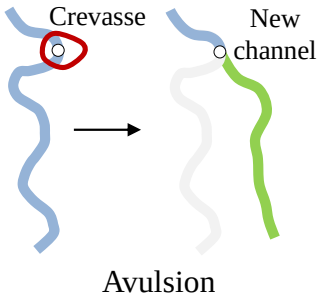


FIGURE 3.13: *Avulsion process in a channel.*

In our model, we only model local avulsions which occur within a single channel. A more complex simulation could incorporate global avulsions, but we left this as future work. An avulsion may be triggered with a probability ρ_a at the location of a crevasse, denoted at $\mathbf{a}$. The end point of the avulsion $\mathbf{b}$ is computed randomly within the channel, thus the goal is to compute a new path from $\mathbf{a}$ to $\mathbf{b}$. We first define the global avulsion direction $\vec{\mathbf{d}}$, computed as a random unit direction within the cone at $\mathbf{a}$ and oriented toward $\mathbf{b}$, with an angle of ϕ, set to 45 degrees. Let $\mathbf{p} = \mathbf{a}$ denote the starting location of the new path, we compute the new position as $\tilde{\mathbf{p}} = \mathbf{p} + \vec{\mathbf{d}}$. At each step, the direction $\vec{\mathbf{d}}$ is linearly interpolated towards the unit vector $\widehat{\mathbf{b} - \mathbf{p}}$, depending the distance between the points. This process is repeated until the distance between $\mathbf{p}$ and $\mathbf{b}$ is inferior to the sampling size of the channel.

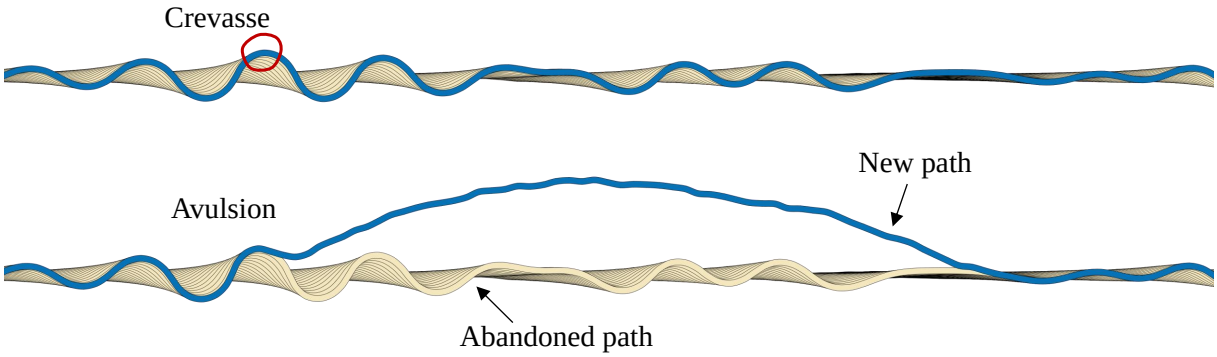


FIGURE 3.14: *Showcase of an avulsion event, where the crevasse became the starting point of a new path carved by the river.*

While this technique is fast and allows control through the starting angle ϕ, the created path may not

respect hydrological constraints (*i.e.* it may not go up a terrain slope). To solve this issue, we check at each step that the slope between $\mathbf{p}$ and $\tilde{\mathbf{p}}$ is negative. If that is not the case, we may carve the terrain to correct the flow (if the slope is small), or abort the avulsion completely. However, because meandering rivers occur in plains, the overall gradient of the terrain remains small, so it is usually possible to carve the terrain to ensure hydrological constraints.

3.4.3 Resampling

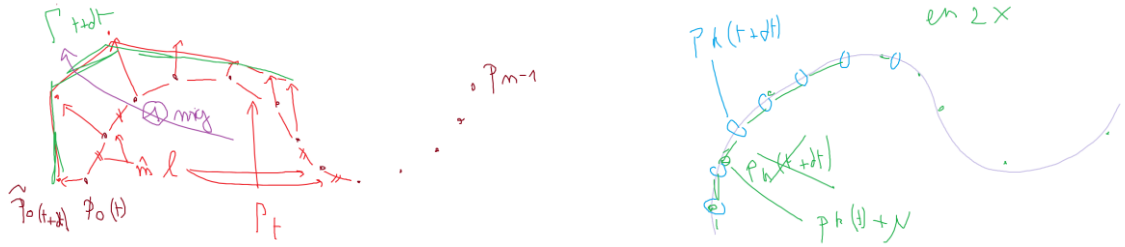


FIGURE 3.15: *Resampling ensures that points are evenly spaced on the trajectory.*

The migration step, as it moves the points in the direction of the normal to the channel, leads to unevenly spaced points, which in turn leads to an unstable and incorrect simulation. To solve this issue, we perform a resampling at the end of each simulation step, after point migration (Figure 3.15).

TODO: notation + formulas

We first compute the cubic spline approximation of the channel passing through the control points. Then, we sample the curve at regular intervals and use these samples as the new control points of the channel. Using a cubic approximation allows a precise and smooth reconstruction of the trajectories.

3.5 River network

3.5.1 Dealing with junctions

Channels within the river graph are connected through junction points. These specific points are defined when two upstream channels meet and merge into a single one downstream. Simulating the evolution of junctions would be complex, and could not be done with our simulation as they are the result of different physical processes (Guillén-Ludeña *et al.* 2016). However, geomorphologists identified simple rules regarding the connection angle between different channels based on their respective flows. When the junction involves two rivers with significantly different water flows, the connection angle is near perpendicular, whereas the junction of two rivers of similar flow will cause a small connection angle (Hooshyar *et al.* 2017).

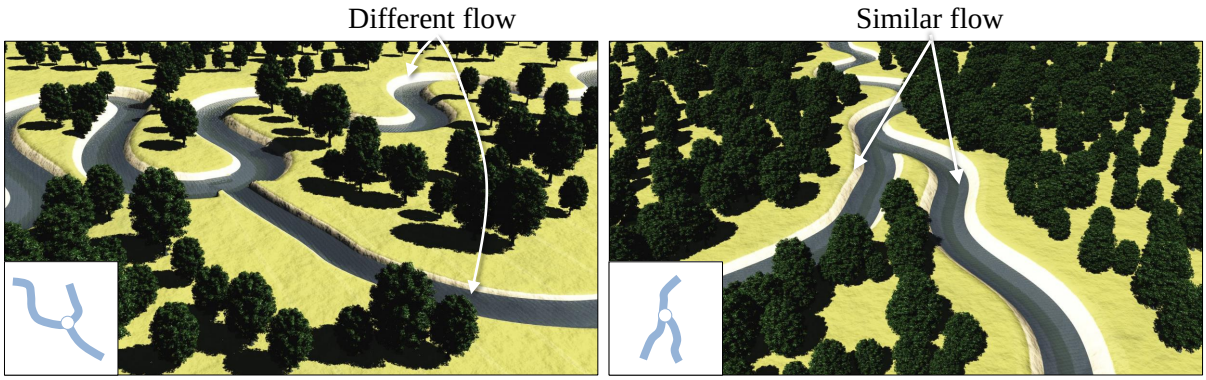


FIGURE 3.16: *We define procedural junction templates for modeling channels intersections (shown as insets), with a near perpendicular angle for rivers with different flow values (left), and a small angle for channels with similar flow values (right).*

Following these observations, we adopt a procedural approach focusing on control and stability for modeling junctions. We apply a linear falloff at the beginning and end parts of the channel to avoid moving the control points during the simulation. This ensures that channels will not collide at these locations. Then, we automatically place procedural river junction templates based on the flows of the merging channels (see Figure 3.16).

3.5.2 Collision between sections

TODO(Axel): Describe how to modify the graph when two sections collide. Two cases can arise: downstream collision, or upstream collision.

FIGURE 3.17: *Example of two cases that arise for collision detection. The junction point is modified and the ancient river trajectory is abandoned.*

3.6 Model realization

Parler de pourquoi notre simulation est utile dans le cadre d'autres choses: terrain carving, or ecosystem simulations.

3.6.1 Abiotic parameters computation

Abiotic parameters, such as soil moisture, temperature and sunlight exposure are key development factors for simulating ecosystems (Gain *et al.* 2017). As the trajectory of meandering rivers evolve rapidly (sometimes in the order of dozens of meters per year), they have a drastic impact on the surrounding ecosystem. In the outer bank (where erosion occurs), vegetation is progressively destroyed, while the inner bank, filled with deposited sediment and having a high moisture, is amenable to vegetation growth. Reproducing the complex dynamic between ecosystem and meanders would involve a joint simulation of the two phenomena, which is computationally intensive, and is still an open question in geomorphology (**???**).

Instead, we adopt a more procedural approach. By taking into account the history of the simulation (ancient trajectories, oxbow lakes, and avulsion paths), we are able to compute abiotic parameters such as soil moisture and sediment covers at a given time step, which can then used as inputs for an ecosystem simulation.

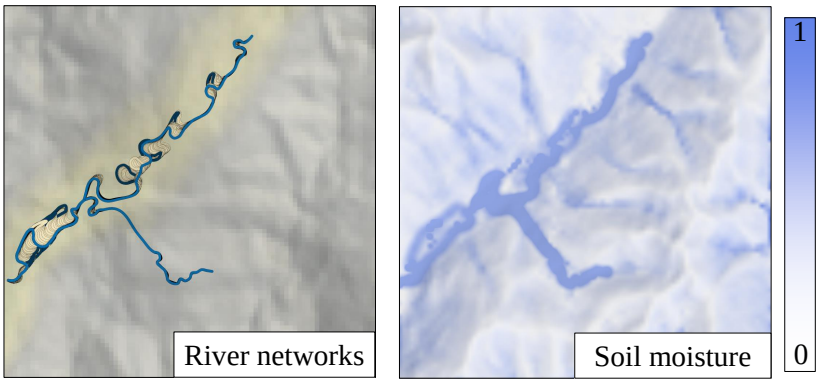


FIGURE 3.18: *Soil moisture is an important development factor for ecosystems. We compute it as a function of past river trajectories.*

Soil moisture is one of the key development factor of ecosystems, and is governed by precipitation and soil type, among other factors. It is usually approximated as a function of slope and drainage area (Gain *et al.* 2017). Steeper slopes are usually made of rocks, which are less amenable to vegetation growth, while plains receive runoff water from precipitation and have a high reservoir soil capacity. In our case, as the river network is rapidly evolving, it it necessary to take into account past trajectories to determine a global measure of soil moisture. We compute soil moisture as a scalar field $m : \mathbb{R}^2 \to \mathbb{R}$, defined from the combination of the stream power field $p : \mathbb{R}^2 \to \mathbb{R}$ (Cordonnier *et al.* 2016) and the sum of past moisture fields $m_i : \mathbb{R}^2 \to [0, 1]$:

$$m(\mathbf{p}) = p(\mathbf{p}) + \sum_{i=0}^{n} m_i(\mathbf{p}) \qquad m_i(\mathbf{p}) = \varepsilon_m \, (g \circ d(\mathbf{p}, \mathcal{N}_i))$$

The stream power p allows to take into account both the drainage area and the terrain slope. The parameter $\varepsilon_m \in [0, 1]$ represents the moisture accumulated at each step for a given channel, and is set by the user. High values of n take into account more ancient trajectories in the computation of moisture, while $n = 1$ only takes into account the current river network. The function g is a falloff function, taking the

distance to the nearest river channel $d(\mathbf{p}, \mathcal{N}_i)$ and its radius of influence R_i:

$$g(\mathbf{x}, R_i) = \begin{cases} (1 - (\mathbf{x}^2/R_i))^2 & \text{if } \mathbf{x} < R_i \\ 0 & \text{otherwise.} \end{cases}$$

As rivers infiltrate deep into the underground water tables, they can have an impact over large spatial zones, thus the radius of influence R_i is set to twice the size of the channel width. The resulting soil moisture has a high value near the current and past river channels. Figure 3.18 shows the soil moisture computed with this method from the latest 100 simulation steps.

Sediments are deposited along the trajectory of the river during the simulation, and progressively transform into clay that favourizes vegetation growth. The deposited sediment fields is computed from the past trajectories and modulated with a turbulence function to account for the stochastic nature of the phenomena. On top of using as input to an ecosystem simulation, the sediment field can be used to add more variety in the final terrain shading, as demonstrated in Figure 3.1.

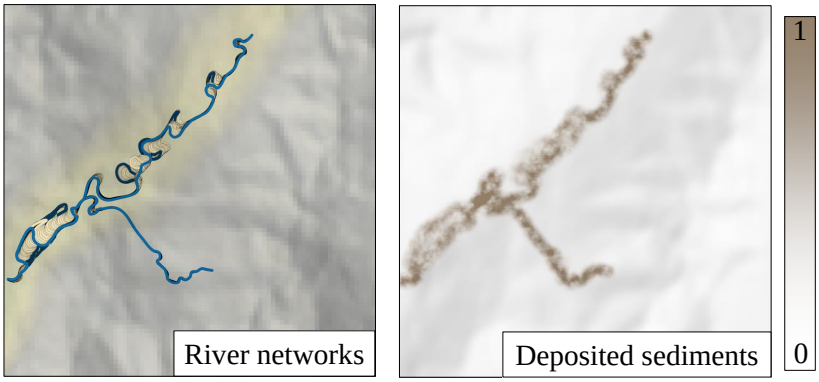


FIGURE 3.19: *TODO(Axel)*

3.6.2 Vegetation placement

We use a simplified model, in the spirit of Gain *et al.* 2017 or Cordonnier *et al.* 2017, or another ? (find closest model that we use). We use the current river trajectory as a mask to avoid ecosystem development inside the channels.

3.7 Simulation controls

3.7.1 Terrain influence

Meandering rivers tend to develop in low-gradient areas, as strong slopes forbid the lateral migration of the channel. We model this by multiplying Equation3.5 with a slope term $\mathcal{S}$. It is defined as the combination of a falloff function with the topographic gradient of the elevation function:

$$\mathcal{S}(\mathbf{p}) = g \circ ||\nabla h(\mathbf{p})||$$

Where $g : \mathbb{R} \to [0, 1]$ is the smoothstep function with values of 0.0 above a threshold s_{max} specified by the user. Figure 3.20 shows a river carved in a mountainous terrain followed by a large floodplain, where meanders remain contained between the mountains and develop only in the flat area.

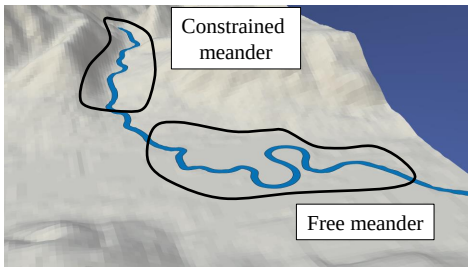


FIGURE 3.20: *Showcase of the influence of terrain slope: meanders develop only in low gradient areas.*

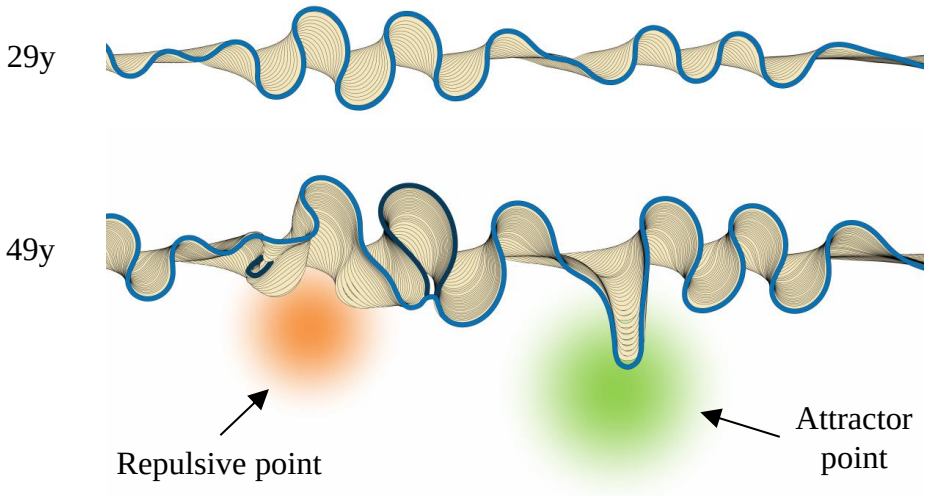


FIGURE 3.21: *Showcase of the influence of an attractive point on the simulation. Show domain of influence and time step in which influence starts.*

3.7.2 Erodability maps

Erodability maps correspond to the global permeability of the ground. More permeable ground favours the development of meanders.

3.7.3 Control points

Attractive and repulsive constraints. allow the user to specify precise location for meanders. We introduce a scalar control field $c : \mathbb{R}^2 \to \mathbb{R}$, which is constructed and edited using primitives (points, curves) and operators (min, max, blend). Negative (respectively positive) values indicate a repulsive (respectively attractive) constraints. The migration direction is then computed as:

$$d(\mathbf{p}) = \frac{n(\mathbf{p}) + \nabla c(\mathbf{p})}{2}$$

With $n(\mathbf{p})$ the normal to the river. Figure 3.21 shows an example of an attractive and repulsive point on the simulation. While repulsive points indirectly influence the development of meanders and are intuitive to control, we found that attractive constraints can be difficult to control as they tend to over constrain the simulation. For controlling the precise trajectory of the river, we let the user specify fixed points in the river sections which'll not migrate through the simulation.

Direct control points. do we need to talk about those ? Maybe a little at the beginning of this section ?

3.8 Results and discussion

We implemented our method in C++. Experiments were performed on a desktop computer equipped with Intel® Core $i7$, clocked at 4 GHz with 16GB of RAM, and an NVidia GTX 1080ti graphics card. The output of our system was streamed into Vue XStream® to produce photo realistic landscapes. The code for producing the results shown in the paper is available at [link will be added upon acceptance]. Table 3.1 reports the values for the different parameters and constants shown in the paper, and Table 3.2 reports the statistics for different scenes shown throughout the paper.

Parameter	Value	Unit
k_1	0.164	m/d
δt	106	Days
$\mathcal{C}_f$	0.011	Dimensionless
γ	2.5	Dimensionless
ω	-1	Dimensionless
s_{max}	0.15	m
t_c	0.1	Dimensionless
$\mathcal{P}_c$	0.05	Dimensionless
$\mathcal{P}_a$	0.05	Dimensionless

TABLE 3.1: *Values for the different constants and parameters of our simulation.*

3.8.1 Performance

The simulation runs in real time even for a complex network featuring hundred to thousands of river sections (see Table 4.1). The bottleneck is the visualization based on the rasterization of cubic Bézier curves, which becomes computationally intensive beyond a few hundred river sections. Our prototype relies on Qt.6 *Q*GraphicsScene class to render high resolution images of the river network, and could be replaced with another system taking advantage of graphics hardware; however this development was beyond the scope of this paper.

3.8.2 Validation

As rivers are an active subject of research, different metrics exist to describe, compare and quantify meandering rivers. While we do not aim at an exhaustive comparison, we implemented several metrics

Figures	$\#\Gamma$	Size	Years	Steps	t_s (ms)	t_c (s)
3.1		X	X	X	X	X
3.7	25 21			350	7.6	2.7
3.8	403			130	0.6	0.8
3.21	323			160	0.6	1.0
3.20	X	X	X	X	X	X

TABLE 3.2: *Performance of the simulation for different scene in the paper: point count $\#\Gamma$, simulation steps, time for single step t_s (ms) and total simulation time t_c (s).*

and compare the obtained values against real data. In particular, we focus on meander wavelength and sinuosity of the channels.

Meander wavelength is defined as the spacing between two points of similar curvature along the channel. In our implementation, we measure the wavelength λ as the distance between inflection points, as advocated by Reinfelds *et al.* 1998. Studies found that the ratio of meander wavelength and river width is within $[6.2, 12]$ for real rivers (Leopold *et al.* 1960; Williams 1986). Our generated network shows similar values, except for the rivers with $w = 100\text{m}$ and $w = 50\text{m}$ of Figure 3.7. This can be easily explained by the fact that wavelength is mostly relevant for regular meanders (Reinfelds *et al.* 1998), and these channels exhibits very irregular trajectories.

Recall that sinuosity σ is defined as the ratio between the length of the curve and the distance between the first last points of a section (Equation 3.4). Sinuosity is a useful metrics for rivers, and a commonly accepted classification states that the meandering stage occur for $\sigma > 1.5$, which is obtained consistently in our simulation.

Figures	λ/w	σ
Figure 3.7 [200m]	8.4	1.9
Figure 3.7 [100m]	14.4	3.6
Figure 3.7 [50m]	14.2	3.8
Figure 3.8	10.8	2.4
Figure 3.21	9.2	1.9
Observed range	$[6.2, 12]$	> 1.5

TABLE 3.3: *Geomorphological properties of different scenes: ratio λ/w, sinuosity σ.*

We provide a qualitative validation in Figure 3.22, where we compare the result of our simulation with photographs of real meandering rivers. In this example, the user modified the influence of past meandering trajectories to prevent the growth of vegetation (Figure 3.22). While we do not aim at reproducing the exact meandering patterns and vegetation placement, we show that our method can be used to create scenes with similar visual aspects as real rivers.

3.8.3 Limitations

Our approach currently makes some approximations. We no not take vegetation into account in the simulation. This is a typical tradeoff between interactivity and an accurate simulation. Furthermore, the link between ecosystems and meanders it not clear and depends on more global climate parameters.

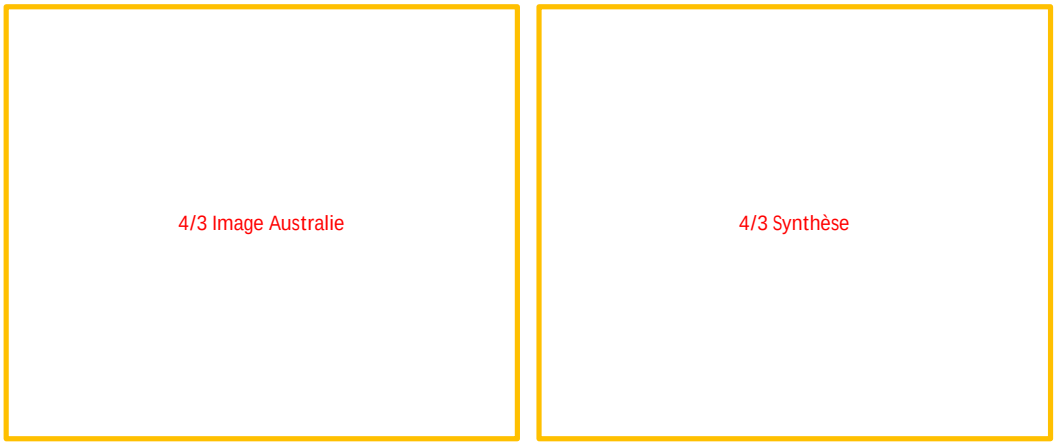


FIGURE 3.22: .

We only simulate local avulsions within a channel. A more complex algorithm is needed to ensure that global avulsions can work, by redirecting flow properly between channels.

Attractive points can over constrain the simulation and lead to unrealistic meander shapes.

3.9 Conclusion

This work open several avenues for future research. A direct extention of this work consists in improving the simulation loop by taking into the effect of vegetation, and couple the meandering similation with an ecosystem simulation. Another direction worth investigating is the simulation of braided rivers, and deltas.

Chapter 4

Desertscape simulation

Contents

4.1 Introduction

Hydraulic erosion attracted the most attention in the computer graphics community, focusing on Alpine mountain ranges with specific features resulting from the action of water, such as dendritic river and ridge networks, eroded mountain ranges with sedimentary valleys. In contrast, the impact of wind erosion over terrains in hot and arid regions has seldom been addressed despite their significant earth coverage (about $1/3$ of earth land surface) and scenic visual aspect. One notable exception is the influence of wind over the growth of tree (Pirk *et al.* 2014) and snow simulation (Cordonnier *et al.* 2018b). Yet, desert landscapes have not been studied.

Wind is an crucial erosion agent in hot deserts, where annual rainfall is low. In particular, hot desert landscapes are characterized by distinctive landforms modeled by the action of wind, such as dunes of different shapes and sizes, eroded table mountains, and bedrock sculpted by the abrasion of sand blown by the wind. These phenomena are highly dynamic and can quickly change the landscape, moving entire dunes sometimes up to $25\mathrm{m}$ per year.

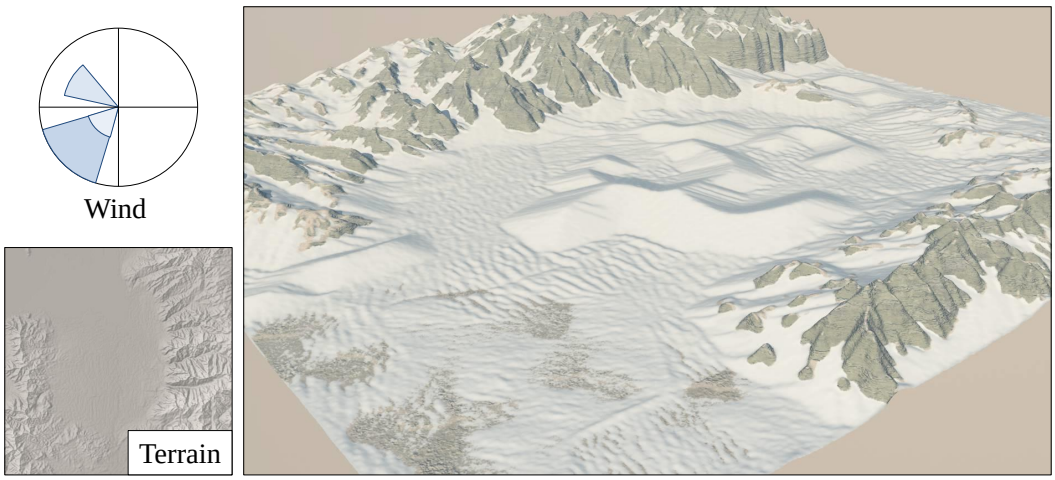


FIGURE 4.1: *Example of a desert landscape. The user defined the high-altitude wind regime, added local turbulence, and finally placed sand at the center of the terrain. Our model automatically created a mega barchan and a star-shaped dune. The turbulence also created asymmetric transverse dunes as well as a linear dune.*

This chapter presents an original framework simulating aeolian processes described in geomorphology: namely *saltation*, *reptation*, and *avalanching*, which take place in the generation of many desert features such as dunes created by the accumulation of sand, nabkha produced by the surrounding vegetation, and yardangs created by abrasion of the bedrock (Figure 4.1).

We model the terrain using a layered data structure combining bedrock, sand, and vegetation density. Stochastic rules simulate how sand is transported by the wind and its interaction with bedrock and vegetation. Given a heightfield and time-varying high-altitude wind field, our method automatically computes the surface wind taking into account the relief, simulates sand transport across the terrain, and performs abrasion of the bedrock. The simulation runs at interactive rates and provides multiple levels of control: at any time during the simulation the user may add or remove sand, change the wind regime or modify the vegetation density and directly see the evolution of the system.

The main contributions outlined in this chapter are:

- A procedural model for approximating the wind flow over the relief of the terrain with a multi-scale warping.
- An interactive aeolian erosion simulation derived from algorithms in geomorphology, combining a set of stochastic sand transportation rules operating on a layered terrain model.
- A controllable wind-based approach for authoring desert landscapes, providing a variety of direct and indirect interactive control tools to the user.

After introducing the necessary geological knowledge to the reader (Section 4.2), we provide a high level overview of the simulation pipeline (Section 4.3), and detail how to compute the surface wind from the high-altitude wind regime (Section 4.4). We then explain how sand is transported across the terrain by phenomena such as saltation, reptation, and abrasion (Section 4.5). We also show the potential of such simulation as an input for terrain amplification (Section 4.6) and, finally, discuss implementation details (Section 4.7), and show the results and limitations of our method (Section 4.8).

The work presented in this chapter was published in Paris *et al.* 2019b and received the Replicability Stamp.

4.2 Geomorphology background

Deserts are regions with low water supply. Depending on the amount of precipitation, they are classified as hyper-arid, arid, or semi-arid. Hyper-arid and arid deserts, which receive less than 250mm of rainfall per year, represent almost 25% of the earth surface and have been extensively studied in geomorphology (Huggett 2003). In this work, we focus exclusively on these two types, as they encompass interesting and varied landforms.

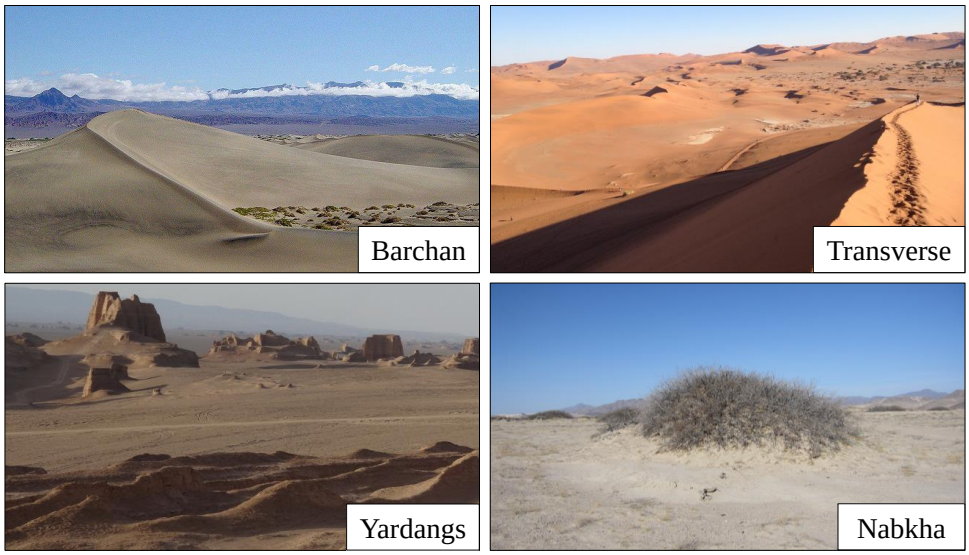


FIGURE 4.2: *Natural desert landscapes exhibit a wide range of landforms. Barchan and transverse dunes emerge from unimodal wind regime, while nabkha appear under the presence of vegetation. Yardangs are elongated shapes in the bedrock that form due to the abrasive action of the wind.*

Geomorphological phenomena. Deserts are composed of different kinds of landforms created by aeolian processes: wind eroding, transporting, and depositing materials. Wind erosion effects include deflation, which is the removal of loose, fine-grained particles due to turbulent wind; and abrasion, which is the wearing down of the bedrock by the grinding action of sand. The transport and accumulation of the sand lead to the formation of dunes. Figure 4.2 illustrates the variety of landforms in hot deserts.

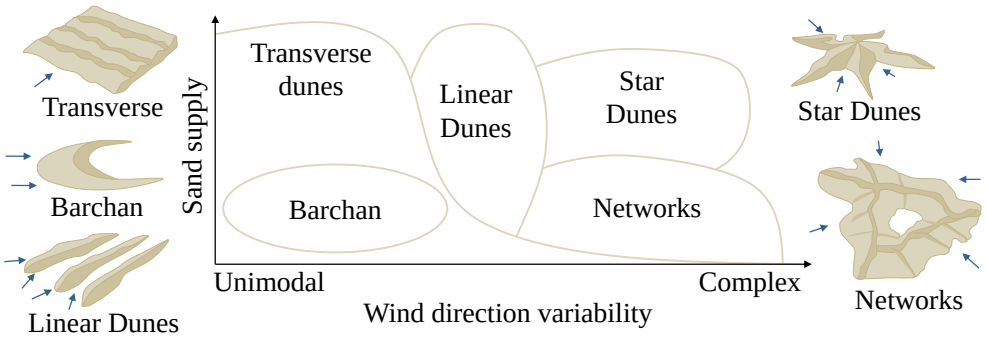


FIGURE 4.3: *Different types of dunes in relation to the variability of wind direction and available sand supply.*

The complexity and variability of wind direction play a key role in the formation of different dunes (Figure 4.3): transverse and barchans dunes are formed from unimodal winds (*i.e.* wind blowing from one major direction), whereas star dunes and network dunes featuring complicated patterns are known to be shaped by more complex wind regimes with local turbulences and vortices, even though the exact processes remain an active field of research. The relief of the terrain and the presence of vegetation also influence sand transport, and thus the different types of dunes formed (Lancaster *et al.* 2013). Anchored dunes such as nabkha are those created under the influence of vegetation (Baas 2002), as opposed to free dunes such as transverse, barchan, and star-shaped. Wind erosion can also carve the bedrock, creating elongated ridge-like formations called yardangs.

Prior work on dune simulation. Few computer graphics works have addressed the formation of sand dunes. Existing methods either address the modeling of sand as a continuous fluid (Yan *et al.* 2016; Daviet *et al.* 2016) or the modeling of sand ripples (Onoue *et al.* 2000; Beneš *et al.* 2004). In contrast, desertic landscapes and in particular sand dunes have received a lot of attention in physics and geomorphology.

Stochastic models (Werner 1995) have been proposed for simulating the transport of sand by the wind and the formation of different types of dunes. Barchan and transverse dunes can be simulated efficiently with a uniform and unidirectional wind flow under different sand supply conditions. This model was improved by approximating the effect of wind acceleration on the windward side of dunes (Momiji *et al.* 2000), which produces more asymmetric dunes and allows to better simulate their movement throughout time. The impact of vegetation on the shaping of sand dunes was studied in Baas 2002 by introducing an additional layer representing vegetation density.

The simulation of more complex dunes such as star-shaped or network dunes remains challenging. Sand accumulation results from the interaction between irregular wind fields and the terrain, and capturing the dynamics of winds over a constantly evolving terrain remains a computationally intensive problem. Therefore, although changes in the wind direction and speed strongly influence the type and shape of dune formations, most existing techniques rely on 2D wind flow approximations and avoid computationally intensive wind simulation. Narteau *et al.* 2009 developed a complex wind model to represent complex effects such as transport, gravity and diffusion. This model was extended in Zhang

et al. 2012 to reproduce star dunes, which are complex to create by simulations. Aeolian erosion is also known to be responsible for the formation of yardangs created by abrasion. Despite several field studies and wind tunnel experiments (Ward *et al.* 1984), we are not aware of any numerical model capable of creating yardangs.

In this work, we depart from complex physical simulations (Narteau *et al.* 2009) and prefer simpler approaches that allow us to include user control in the simulation process. In particular, we take inspiration from Werner 1995 and rely on stochastic rules to transport the sand across the terrain. We simulate phenomena such as *saltation*, *reptation* and *avalanching* while taking into account vegetation presence and bedrock resistance and providing direct control to the user.

4.3 Simulation pipeline

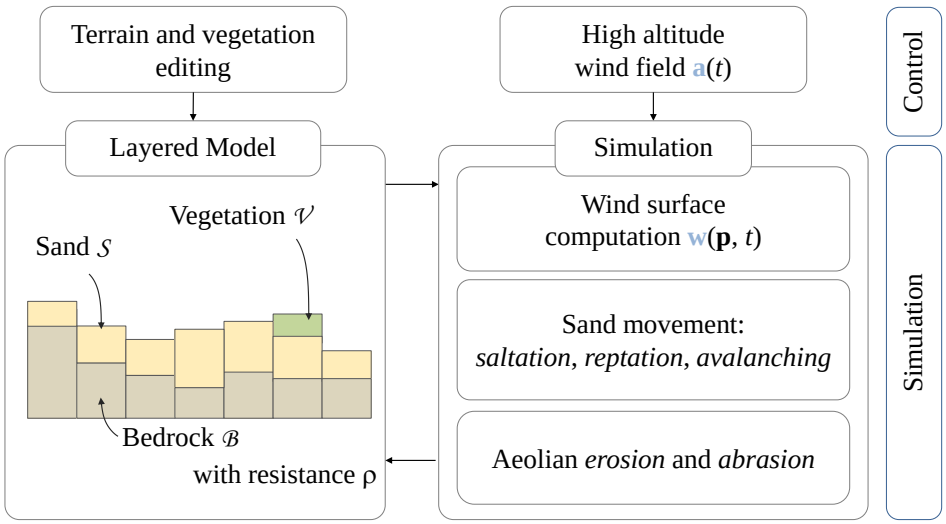


FIGURE 4.4: *Synthetic overview of the simulation.*

Terrain Representation. The layered terrain model, denoted as $\mathcal{H}$, is a discrete regular grid of size $n \times n$ cells. It represents a multi-layer ordered data-structure to represent different terrain materials and plant density in every cell (Figure 4.4). The sand $\mathcal{S}$ layer represents the material thicknesses on top of a bedrock layer $\mathcal{B}$, which defines the base elevation. Plants are represented using a generic vegetation density layer $\mathcal{V}$ which takes values in $[0, 1]$ and represents vegetation cover. The resistance of the bedrock $\rho : \mathbb{R}^2 \rightarrow [0, 1]$ defines the resistance to erosion; regions with a high resistance to erosion ($\rho = 1.0$) are eroded slower than low resistance ones ($\rho = 0.0$).

Wind and sand transport simulation. At the heart of our method is a sand transport algorithm based on the computation of the wind at the surface of the terrain which takes into account the relief. At every time step, we compute the evolution of terrain model $\mathcal{H}(t + \Delta t)$ according to the wind conditions $\mathbf{w}(\mathbf{p}, t)$. Figure 4.4 presents an overview of our simulation. Given an initial input layered representation of the terrain and a high altitude wind field $\mathbf{a}$, we compute a time varying wind field $\mathbf{w}$ over the surface of the terrain (Section 4.4), which is used to compute the movement of sand.

The transport and collision of sand with the relief and vegetation form different types of dunes (Section 4.5.1), and at the same time erode the terrain through abrasion (Section 4.5.2). The vegetation layer

plays a role in the formation of anchored dunes such as nabkha, which only form around plants, and greatly influences the shape of the dunes by preventing the sand from being blown by the wind. Since the simulation is performed at a $1 - 10\,\mathrm{m}$ range per cell, we finally add procedural details to the sand layer to account for small bumps around plants and sand ripples (Section 4.6). Similarly to Werner 1995, the simulation is performed on a toric domain: the sand moved beyond one bound is transported to the opposite bound, which preserves the overall volume of sand during the simulation. Note that the bedrock layer has to be tileable so as to avoid artifacts caused by different elevations on opposite sides.

Control. The high altitude wind regime is defined by a wind rose that prescribes the wind direction and speed distribution throughout time. At any time in the simulation, the user can change the high altitude wind field $\mathbf{a}$ and edit the wind field at the surface of the terrain by adding local procedural wind primitives (Section 4.4.4). Moreover, throughout the simulation, it is possible to add or remove sand, sculpt the underlying bedrock, and modify the density of vegetation.

4.4 Wind surface computation

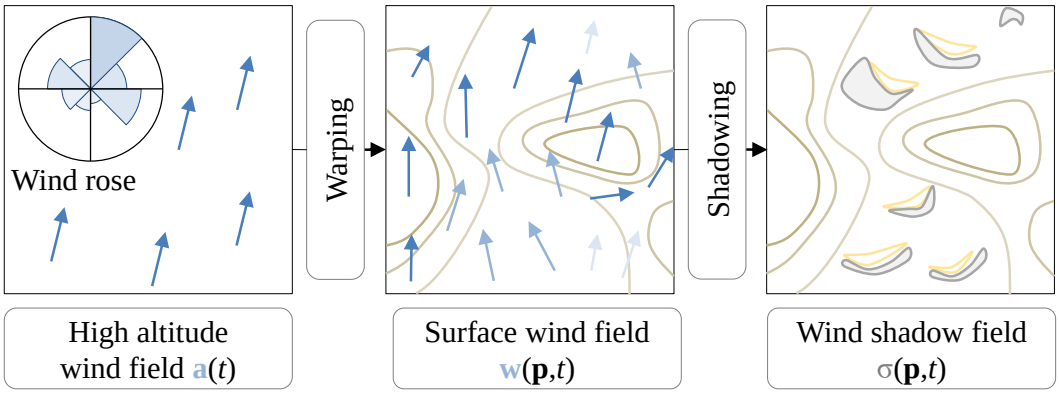


FIGURE 4.5: *Overview of the wind field computation.*

We define the wind field over the surface of the terrain $\mathbf{w} : \mathbb{R}^3 \to \mathbb{R}^2$ by constructing a high-altitude wind field, denoted as $\mathbf{a}$ and computed from a wind distribution (wind rose), and then warping it at different scales according to the relief $\mathcal{H}$. We define the wind field $\mathbf{w}$ as:

$$\mathbf{w}(\mathbf{p}, t) = \sigma(\mathbf{p}, t)\, \omega \circ \mathbf{a}(t) + \mathbf{u}(\mathbf{p}, t)$$

The function $\omega : \mathbb{R}^2 \to \mathbb{R}^2$ denotes a multi-scale warping taking into account the relief of the terrain at different scales which deforms the wind field (Figure 4.5). The function $\sigma : \mathbb{R}^3 \to [0, 1]$ scales the speed by computing the shadowing effects that are generated by small scale elevation features such as sand dunes and small steep cliffs. Eventually, $\mathbf{u} : \mathbb{R}^2 \to \mathbb{R}^2$ is a user control local wind field perturbation which allows the user to edit the wind field $\mathbf{w}$ locally, for instance by adding swirls or turbulence. The final resolution of the wind field $\mathbf{w}$ is $n \times n$, which is the same as the other sand and bedrock layers of our simulation.

4.4.1 High-altitude wind field

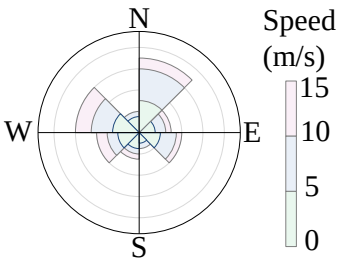


FIGURE 4.6: *Wind rose.*

The high altitude wind field $\mathbf{a} : \mathbb{R} \rightarrow \mathbb{R}^2$ is approximated as a time varying wind of uniform direction and speed over the terrain, since we focused on relatively small domains (up to $10 \times 10\,\mathrm{km}^2$).

In our framework, users can either specify the variation of direction and speed throughout time, or rely on generic wind regime models. Wind regimes are specified by a wind rose that represents the distributions of wind speeds and directions (Figure 4.6). At a given time step, the distribution is sampled and a direction is chosen for the whole terrain.

4.4.2 Warping

The high altitude wind field $\mathbf{a}$ is then warped according to the relief of the terrain at different scales. We compute the smoothed elevation function of the terrain at different resolutions: let R a smoothing radius, we define $\widetilde{\mathcal{H}}_i = \mathcal{H} \circledast g_{R_i}$ as a convolution between the terrain $\mathcal{H}$ and the Gaussian kernel of radius R_i.

We first account for Venturi effects, which accelerate wind at higher altitudes, according to the base elevation:

$$\mathbf{v}(\mathbf{p}, t) = \mathbf{a}(t)\,(1 + k_{\mathcal{W}}\mathcal{H}(\mathbf{p}))$$

The term $k_{\mathcal{W}}$ is a scaling parameter, set to 5×10^{-3} in our model. While wind is usually approximated using a power law, this linear approximation lends itself for interactive purposes as it is efficient and easily controllable.

We then change the direction of the wind according to the gradient of the surface of the terrain at multiple scales. We define the wind at surface as a weighted sum:

$$\sum_{i=0}^{i=n} c_i\,\omega_i \circ \mathbf{v}(\mathbf{p}, t)$$

The term $\omega_i \circ \mathbf{v}$ denotes the warping of $\mathbf{v}$ at scale i weighted by coefficient c_i. The warping operator is in turn defined as:

$$\omega_i \circ \mathbf{v}(\mathbf{p}, t) = (1 - \alpha)\,\mathbf{v}(\mathbf{p}, t) + \alpha\, k_{\mathcal{T}i}\,\nabla\widetilde{\mathcal{H}}_i^{\perp}(\mathbf{p}) \qquad \alpha = \|\nabla\widetilde{\mathcal{H}}_i(\mathbf{p})\|$$

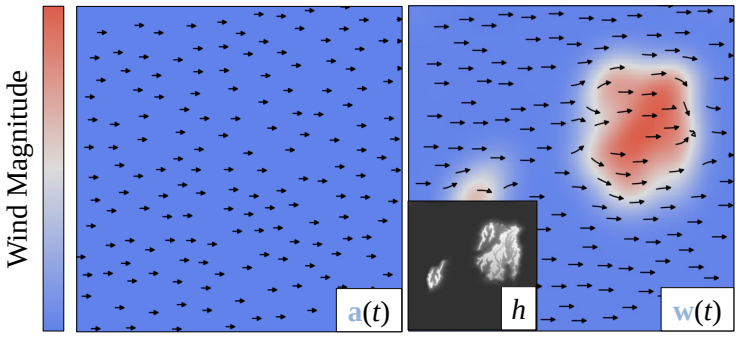


FIGURE 4.7: *Deformation of the constant high altitude wind field* $\mathbf{a}$ *to account for terrain obstacles in* $\mathcal{H}$*, leading to a surface wind field* $\mathbf{w}$.

The term $k_{\mathcal{T}i}$ is a deviation coefficient set by the user, and α the normalized slope of the smoothed terrain. $\nabla\widetilde{\mathcal{H}}_i^{\perp}(\mathbf{p})$ denotes the orthogonal vector to the terrain gradient in the direction of $\mathbf{v}$, thus $\mathbf{v}\cdot\nabla\widetilde{\mathcal{H}}_i^{\perp}(\mathbf{p}) > 0$. In our experiments, we used $n = 2$: convolution radii were set to $200\,\mathrm{m}$ and $50\,\mathrm{m}$, with corresponding weights 0.8 and 0.2, and deviation coefficients of 30.0 and 5.0 respectively. This allows us to redirect the wind with respect to mountains and smaller cliffs or mesas (Figure 4.7).

4.4.3 Wind shadowing

Wind shadowing occurs in the lee side of terrain relief or vegetation, *i.e.* those areas where wind flow has been slowed down sufficiently to suppress any further transport of sand. This complex phenomenon is fundamental in the formation of all sand dunes (Baas 2002).

Our simulation conforms to experiments in geomorphology demonstrating that wind shadowing takes place under a 15 degree accessibility angle. Therefore, wind shadowing at a point $\mathbf{p}$ is approximated using a dampening function $\sigma(\mathbf{p})$, computed as follows. Starting from $\mathbf{p}$, we march in the opposite direction of $\mathbf{w}(\mathbf{p}, t)$ and check intersection with the terrain (Figure 4.8). The maximum marching distance is a control radius R_s set to $10\,\mathrm{m}$, and the marching step is set to $0.5\,\mathrm{m}$. We keep the point $\mathbf{q}$ with the maximum elevation difference regarding $\mathbf{p}$, and compute the shadowing angle α as:

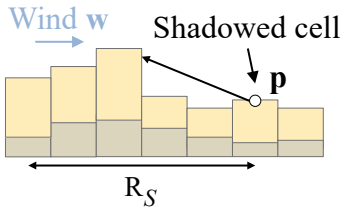


FIGURE 4.8: *Shadowing.*

$$\tan\alpha = (\mathcal{H}(\mathbf{p}) - \mathcal{H}(\mathbf{q}))/(\|\mathbf{p} - \mathbf{q}\|)$$

We finally compute shadowing as the linear interpolation between 10 and 15 degrees from the angle α, which will progressively suppress sand transport (see Figure 4.9).

Note that the computation of $\mathbf{w}$ is performed at every step of the simulation to account for the time-varying high altitude wind field $\mathbf{a}$, as well as the constant movement of dunes, which leads to different warping and shadowing effects as the landscape evolves. While this update is computationally intensive, the multi-scale approach generates realistic sand transport effects by weighting obstacles accordingly to their size: a small hill does not have the same impact on the wind direction as a mountain peak.

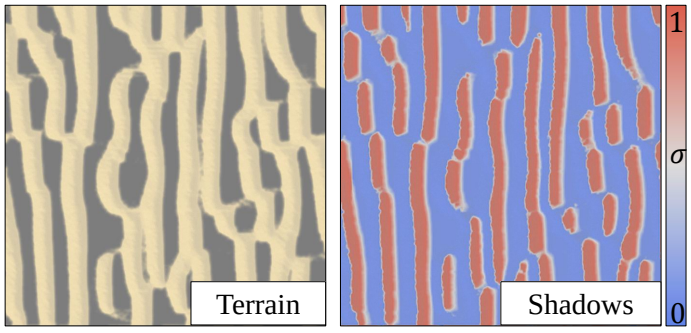


FIGURE 4.9: *Shadowing map for a set of transverse and barchan dunes. Red indicates maximum shadowing with $\sigma = 1$ ($\tan\alpha > 15$ degrees) while blue means complete exposition to wind with $\sigma = 0$ ($\tan\alpha < 10$ degrees).*

4.4.4 Control

The user-controlled perturbation field $\mathbf{u} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is constructed by combining time varying procedural wind primitives as presented in Bridson *et al.* 2007. This approach provides fine user-control over the simulation process and guarantees that $\mathbf{u}(\mathbf{p}, t)$ should be divergence-free (*i.e.*that it represents an incompressible fluid with no sources or sinks). The perturbation field provides the user with local control and enables her to add eddies or turbulences at prescribed locations (see Figure 4.10), which are important for some specific phenomena such as asymmetric transverse or star dunes (Section 4.8).

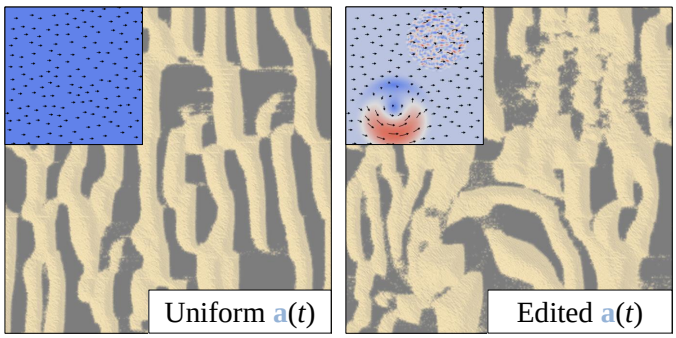


FIGURE 4.10: *Comparison between a uniform (left) and a user-edited (right) wind field using a vortex and a turbulence primitives. The latter produces more asymmetric dunes.*

4.5 Sand simulation

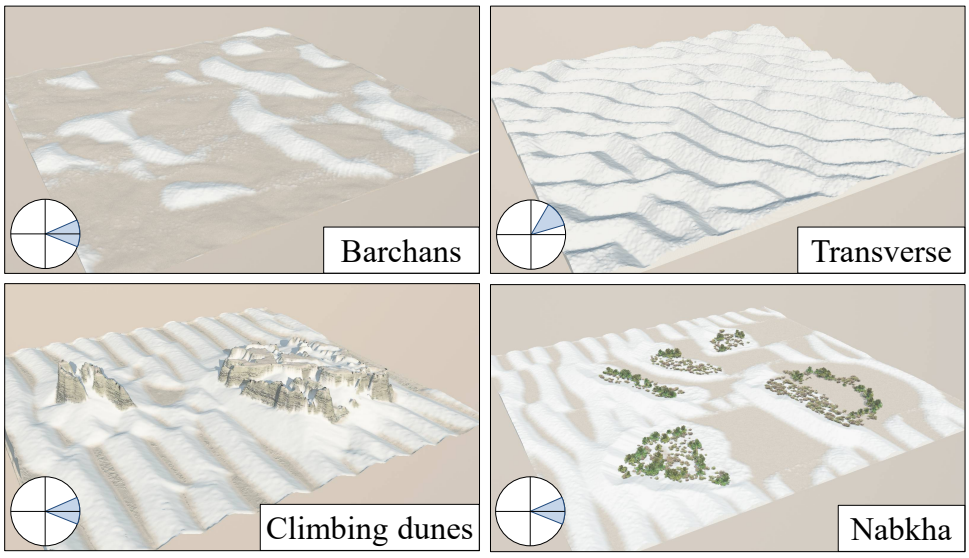


FIGURE 4.11: *Our method generates different types of dunes: free dunes such as barchan and transverse dunes (top row) form with uniform wind conditions; anchored dunes are influenced by their environment: climbing dunes appear at the bottom of small cliffs, and nabkha are created near vegetation.*

The sand transport model involves three different sand movements: *saltation*, *reptation* and *avalanching* which are modelled as stochastic processes (Figure 4.12). The fundamental sand transport process leading to the formation of dunes is *saltation* (Section 4.5.1).

Depending on its strength, wind can lift and carry sand along its direction, bouncing possibly multiple times before being deposited at some other location. In turn, these bounces and depositions produced by *saltation* can trigger *reptation*, also referred to as *creep*, which is the movement of sand grains to adjacent positions upon impact by other sand grains. Finally, when the deposited sand creates a local slope greater than the angle of repose threshold, *avalanching* events are triggered. Note that *saltation* moves sand in wind direction, whereas *reptation* and *avalanching* may transport it in different directions.

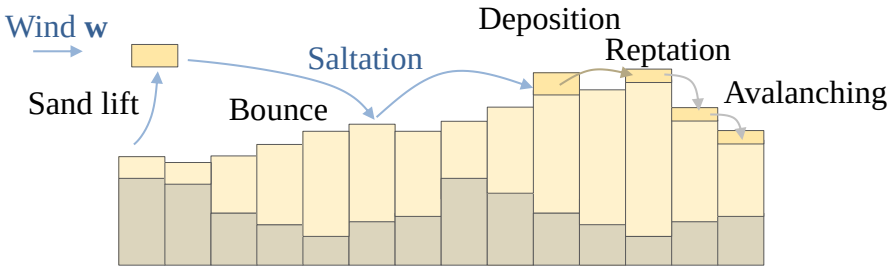


FIGURE 4.12: *Processes involved in sand transport: saltation lifts sand in the air and transports it over a few meters, possibly with multiple bounces. Deposition eventually occurs based on stochastic rules involving the presence of sand, vegetation and wind shadowing. Reptation is triggered by the deposition of sand at each bounce during saltation.*

Wind also erodes the surface of the terrain by an aeolian *abrasion* process, which occurs whenever a small amount of sand blown by the wind is transported over bare bedrock (Section 4.5.2).

4.5.1 Sand transport

For every cell $\mathcal{C}$ of the grid, we successively trigger a series of events according to the surface wind $\mathbf{w}(\mathbf{p})$ at the location of the cell: starting with saltation, a small fixed amount of sand, often referred to as *slab* in geomorphology (Werner 1995), is lifted and moved over the grid by successive saltation steps or bounces, triggering in turn reptation events until sand is eventually deposited back to the ground, which might trigger avalanching events.

Saltation. We approximate *saltation* as a stochastic event on a given cell $\mathcal{C}$ in three steps. First, *lifting* removes a small amount of sand ϵ_S (set to 0.1m). Then, wind transports ϵ_S to a target cell denoted as $\mathcal{N}$ located at $\mathbf{q} = \mathbf{p} + d \circ \mathbf{w}(\mathbf{p}, t)$ where d denotes the saltation distance function of the wind. The saltation distance is linear to the intensity of the wind $\|\mathbf{w}(\mathbf{p}, t)\|$ and transports the sand along its direction. When this sand slab hits the ground, it can either bounce or be deposited on the ground according to a probability β. This probability depends on wind shadowing $\sigma(\mathbf{q})$, the presence of sand $\mathcal{S}(\mathbf{q}, t)$ and the vegetation density $\mathcal{V}(\mathbf{q}, t)$ of the target cell $\mathcal{N}$:

$$\beta = \sigma(\mathbf{q}) + f_{\mathcal{S}}(\mathcal{S}(\mathbf{q}, t)) + f_{\mathcal{V}}(\mathcal{V}(\mathbf{q}, t))$$

Note that we clamp the value of β in $[0, 1]$. The transfer functions $f_{\mathcal{S}}$ and $f_{\mathcal{V}}$ are defined as follows. Following Werner 1995, we model $f_{\mathcal{S}}$ as the step function $f_{\mathcal{S}}(0) = 0.4$ and $f_{\mathcal{S}}(x) = 0.6$ for $x > 0$. The action of saltation is more intense in those places where vegetation is sparser since sand grains are unconsolidated. Therefore, we used a linear decay: $f_{\mathcal{V}}(x) = 1 - x$ for the vegetation density term.

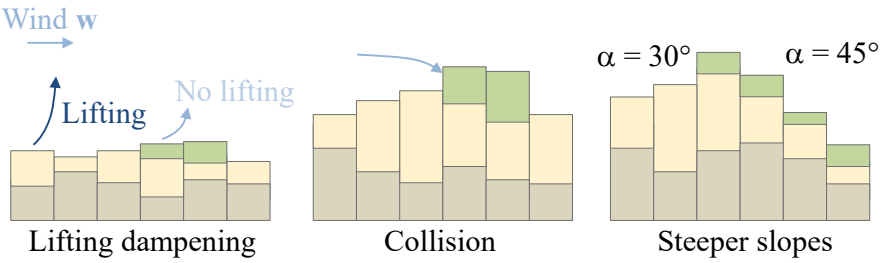


FIGURE 4.13: *Vegetation limits sand lifting during saltation, decreases the probability of bouncing, and prevents avalanching and reptation.*

Reptation. is a process whereby sand grains collide with others during bounces in saltation, prompting them to move in the slope direction. It is also, with *avalanching*, a process where sand grains can move laterally to the wind direction.

We model reptation as a stochastic event that is triggered by the sand slabs bounces and depositions during saltation transport. In reptation movements, a small amount of sand ϵ_R is displaced to neighboring positions depending on the slope. We transport sand to the n steepest neighbors of the current cell, and distribute the quantity ϵ_R (set to 0.1m in our simulation) to each neighbour proportionally to their steepness. We empirically found that $n = 2$ was enough to account for the chaotic nature of this phenomenon; adding more neighbors can lead to oscillations and visual artifacts. Vegetation also influences *reptation* as it shields and retains sand from being moved by collision during bounces. In our implementation, the probability β_R of a *reptation* event is defined as:

$$\beta_R = 1 - \mathcal{V}(\mathbf{q}, t)$$

While *saltation* is the fundamental transport process in sand dunes, the importance of *reptation* is still an open question in geomorphology. Cooke *et al.* 2006 states that the importance ratio between *reptation* and *saltation* has been found to be $1:3$, while Nickling 1978 found that *reptation* might only account for less than 4% of sand grain movements. We found the importance of *reptation* to be negligible as it did not change the global dune shape and distribution, even when increasing the amount of neighbors n.

Avalanching occurs when the local slope $s(\mathbf{p})$ of the sand is greater than a given threshold defined by the repose angle: $s(\mathbf{p}) > \tan\alpha$ (Figure 4.14). Sand slides in the direction of the steepest slope only, making avalanching a deterministic process as opposed to saltation and reptation. The avalanching process participates to the formation of climbing dunes and talus on the leeward side of steep cliffs (see Figure 4.11).

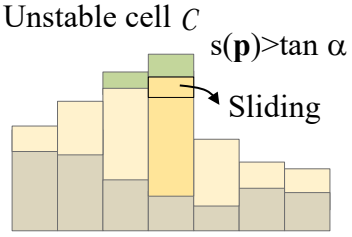


FIGURE 4.14: *Sliding.*

It is a fundamental stabilization process in sand simulations and neglecting it would lead to large unrealistic piles of sand. When *saltation* has transported sand from one cell to another, we check stability on both cells and trigger sand slides if necessary. We model this process in the same way as the granular material stabilization process described in Cordonnier *et al.* 2017 by checking stabilization on a per-cell basis, propagating material to neighboring cells and triggering avalanching events to those cells.

Vegetation prevents avalanching by retaining sand (Figure 4.13). In our model, vegetation density changes the angle of repose threshold

value: we linearly interpolate between $\alpha = 30°$ for bare sand and $\alpha = 45°$ for sand covered by vegetation ($\mathcal{V} = 1$).

Parameters Existing models in geomorphology are often scale-independent, which allows them to reproduce different phenomena occurring at different scales, such as sand dunes and sand ripples. We investigated several studies in geomorphology to determine the parameters of the simulation. In deserts, sand grains are lifted by the wind and then transported over short distances. By setting a time step $\Delta t = 10$ days and a maximum saltation distance of $d_0 = 8\,\mathrm{m}$ (per iteration), an entire barchan dune moves by ≈ 25 meters in a year, which is consistent with observations made on barchan dunes (Groh *et al.* 2008). Note that this parameter highly depends on the wind regime and the average sand supply and is only valid in the context of arid deserts.

4.5.2 Bedrock abrasion

Abrasion is the erosion of bedrock by the wind, more precisely by sand hitting the surface and bouncing off during saltation. Typical desert landforms such as yardangs and ventifacts present in the Gobi desert are produced by the action of the wind, which carves the bedrock. In areas with low sand supply and high wind speed, sand carried by the wind hits the bedrock, thus eroding the surface. Softer bedrock erodes faster, which leads to the creation of characteristic landform as depicted in Figure 4.16.

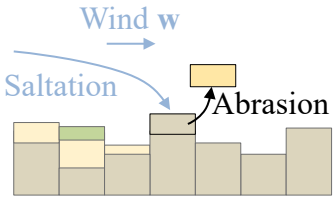


FIGURE 4.15: *Abrasion.*

We simulate bedrock abrasion during the *saltation* step, where sand moves from one cell to another with possibly multiple bounces (Figure 4.15). If a bounce occurs on a cell $\mathcal{C}$ with a low sand thickness value ($s(\mathbf{p}) < 25\,\mathrm{cm}$), we trigger an abrasion event for this cell. The abrasion process transforms a small amount ϵ of bedrock into sand, which may be transported by the wind in future saltation steps and may stabilize according to the *avalanching* process. The eroded amount of material ϵ is computed as a function of the wind speed, bedrock resistance and vegetation density:

$$\epsilon = k_a \left(1 - \rho(\mathbf{p})\right) \|\mathbf{w}(\mathbf{p})\| \left(1 - \mathcal{V}(\mathbf{p})\right)$$

Abrasion is more important as the speed of the surface wind $\mathbf{w}$ is high. The term $(1 - \rho(\mathbf{p}))$ denotes that abrasion is less intense as the bedrock is more resistant. Vegetation dampens abrasion and acts as a shield, protecting the bedrock surface.

Abrasion may erode up to 4 millimeters of rock per year (Cooke *et al.* 2006) and therefore acts on a larger time scale than saltation and reptation. The constant k_a, experimentally set to 12.5, is a user-defined factor used to accelerate the effects of abrasion in the simulation ($\Delta t = 125$ days in the case of abrasion). In our model, abrasion does not take into account the angle between the wind and the surface and the proposed framework would allow for more complex models taking into account the curvature or the slope of the terrain.

One notable limitation of our simulation regarding yardangs is that they often exhibit more complex volumetric landforms, which cannot be modeled on a 2D layered representation of the terrain. In Chapter 6, we show how to overcome this limitation using fully volumetric method based on implicit surfaces. Yardangs and ventifacts are modeled using arrangements of skeletal primitives which are later eroded using a ballistic approach (see Figure 4.17).

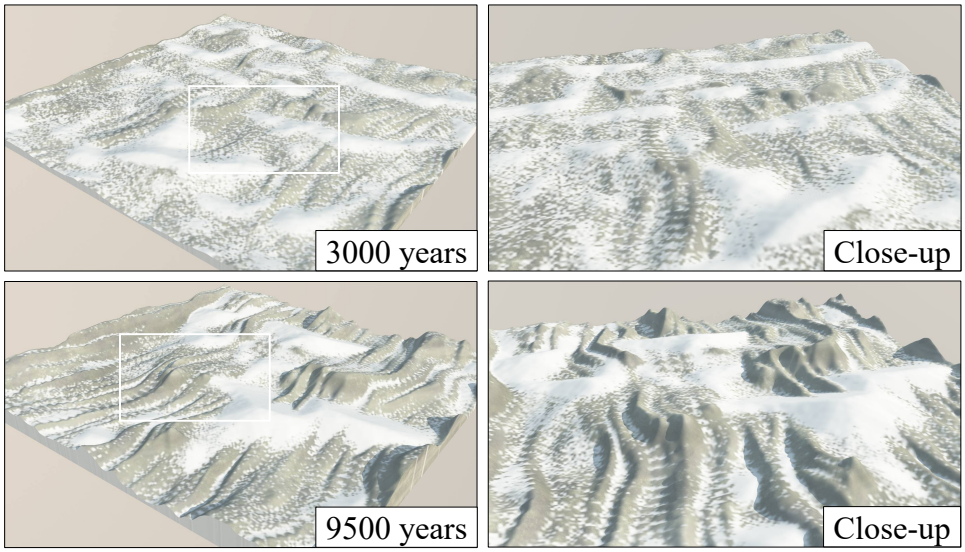


FIGURE 4.16: *Yardangs modeled in our simulation with large time steps. Abrasion shapes the bedrock layer into lines parallel to the major wind direction during saltation, depending on the bedrock resistance (defined as a warped noise), showing the footprint of the wind.*

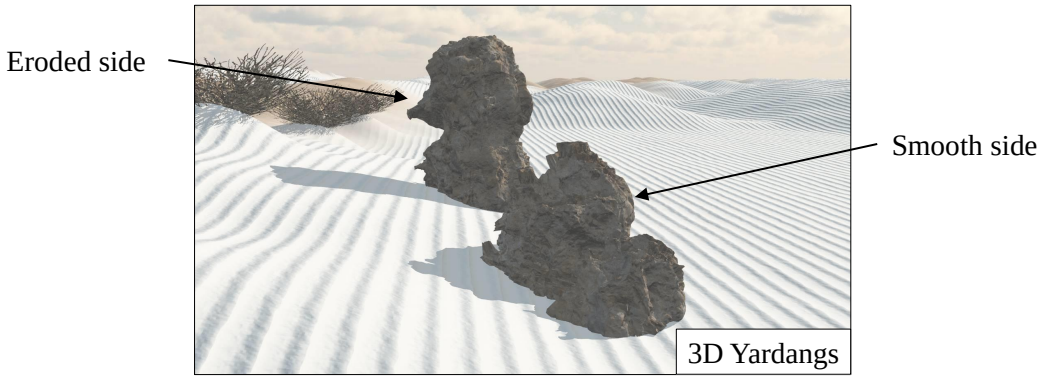


FIGURE 4.17: *Volumetric yardangs generated using a ballistic approach, where volumetric spheres progressively carved an implicit surface base model.*

4.6 Amplification

Recall that our simulation generates a multi-layered representation of desert landscapes at a resolution of $1 - 10\,\mathrm{m}$ per grid cell. Smaller details such as sand ripples or sand accumulation at the base of smaller obstacles, such as plants, cannot be simulated. Thus, there is a need for an *amplification* step to increase the final resolution of the terrain and add microscale details. We procedurally generate details as a post processing step: the final sand elevation is defined as $\widetilde{\mathcal{S}} = \mathcal{S} + \mathcal{R} + \mathcal{B}$ where $\mathcal{R}$ and $\mathcal{B}$ denote the wind ripples and sand bumps caused by small obstacles.

Sand ripples are smaller than dunes, with a width range of $1 - 20\,\mathrm{cm}$. In our implementation, we define the presence and shape of ripples as a function of the wind direction. We relate the ripple size r linearly to the wind speed $\|\mathbf{w}\|$. Asymmetrical ripples profiles are observed when the wind blows in a single direction, whereas symmetrical ripples form when the wind blows in several directions (see Figure 4.21 right). Parallel asymmetric ripples are generated and oriented orthogonally to the wind direction $\mathbf{u} = \mathbf{w}^{\perp}$. We also weight the presence of ripples according to the wind shadowing effect of the

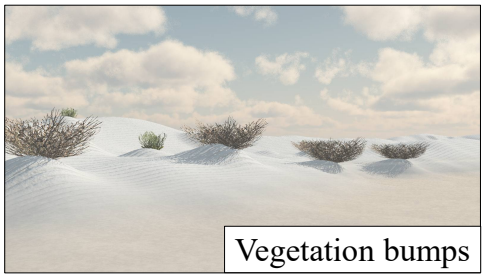


FIGURE 4.18: *Procedural sand bumps located around plants, defined as two blended point primitives.*

relief of the sand dunes.

Small sand bumps that form near rocks and plants result from the collision of the wind-transported sand and obstacles. Sand is accumulated on the windward side of obstacles, the sand relocation and wind-ripples are diminished on the leeward side. We approximate those effects and procedurally generate sand displacement according to the wind and sand fields $\mathbf{w}$ and s respectively (Figure 4.18).

4.7 Optimized implementation

The simulation has been implemented in C++ and is available at

https://github.com/aparis69/Desertscapes-Simulation

The provided code uses OpenMP for parallelization but relies on atomic operations, which are known to be slow. While this remains interactive for scenes up to 1024×1024 resolution, larger domains cannot be processed efficiently. In this section, we reformulate the sand transport algorithm to make it massively parallel without requiring atomic operations. While we do not provide a complete GPU implementation, the optimized version of the algorithm implemented on the CPU performs up to $4\times$ faster than the original implementation.

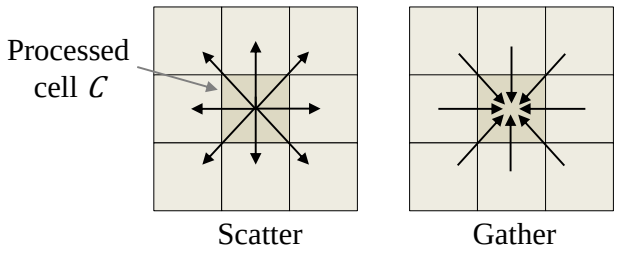


FIGURE 4.19: *Comparison between the scattering and gathering models.*

Grid-based simulations usually involves writing data to neighboring cells. A straightforward implementation is usually based on the *scatter* principle (Figure 4.19, left): when processing a given cell C, the algorithm distributes a certain amount of material to its neighbors depending on the simulation conditions. While easy to understand and simple to implement, this approach prevents an efficient parallel implementation as multiple threads can be writing to the same cell, leading to a race condition.

Instead, the *gather* principle is usually preferred in a parallel context: instead of scattering to neighboring cells, the algorithm checks the amount of material arriving on the current cell from its neighbors

(Figure 4.19 right). Such parallel implementation provides considerable speed-up. The following sections explain how to apply the *gather* principle to saltation and avalanching.

4.7.1 Saltation

Recall that saltation is the main transport process leading to the formation of sand dunes. Sand is lifted from a cell, transported on a maximum distance d_s with possibly multiple bounces in between before deposition eventually occurs on another downwind cell.

To apply the *gather* principle, we must compute the amount of sand arriving on a given cell $\mathcal{C}$ from its upwind neighbors. The complete neighborhood $\mathcal{N}_\mathcal{C}$ thus depends on the maximum saltation distance d_s. The left side of Figure 4.20 shows $\mathcal{N}_\mathcal{C}$ without taking into account wind regime. In practice, exploring the complete neighborhood for every cell increases the overall complexity of the algorithm by an order of magnitude.

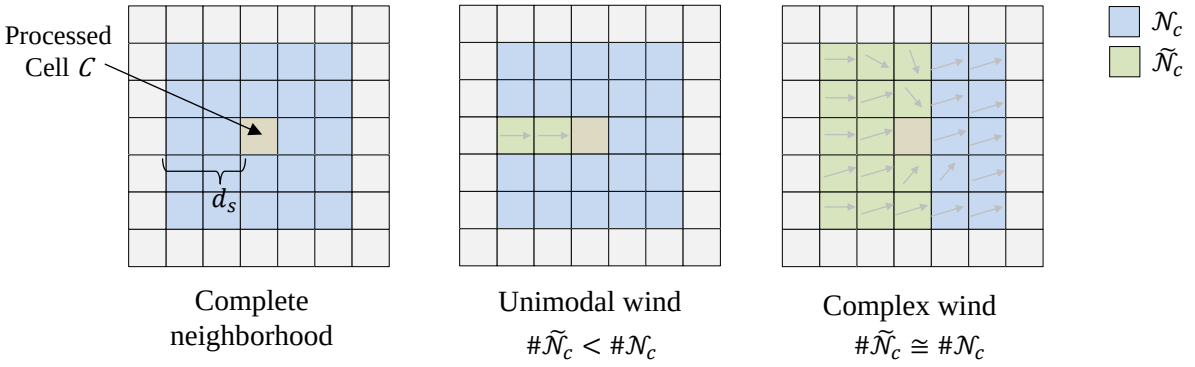


FIGURE 4.20: *Complete neighborhood of a cell $\mathcal{N}_\mathcal{C}$ (left), pruned neighborhood $\tilde{\mathcal{N}}_\mathcal{C}$ under unimodal wind (center), and under complex wind regime (right).*

The wind regime also has a crucial influence on a cell neighborhood: simplifications can be made by pruning a large number of cells from which sand cannot be arriving, leading to a modified neighborhood $\tilde{\mathcal{N}}_\mathcal{C}$ (Figure 4.20, center). This pruning is completely dependent on the wind regime. The majority of cells can be pruned under unimodal wind, leading to $\#\tilde{\mathcal{N}}_\mathcal{C} < \#\mathcal{N}_\mathcal{C}$ (Figure 4.20, center). Under complex wind regime with eddies and turbulences, sand can arrive from more directions, leading to $\#\tilde{\mathcal{N}}_\mathcal{C} = \#\mathcal{N}_\mathcal{C}$ in the worst case scenario (see Figure 4.20 for a complex case). In practice, users tend to favour unimodal or almost unimodal wind regime for modelling the main landforms, and only use more complex wind regime at the end of the simulation scenario to add more diversity.

4.7.2 Avalanching and reptation

In essence, *avalanching* is very similar to the material stabilization described by Musgrave *et al.* 1989 and later by Cordonnier *et al.* 2017. The algorithm can be easily implemented in parallel: for a every cell $\mathcal{C}$, we need to check the amount of material arriving from its 8 direct neighbors. This stabilization step can be performed independently after the saltation and reptation processes of the simulation.

While *reptation* is not the key process accounting for the formation of sand dunes in our simulation, it is still interesting to investigate its efficient parallel implementation. Reptation is heavily linked to saltation, as it is triggered stochastically by a bounce during sand movements, and transports a certain amount of material to the n lower steepest neighbors. By recording the number of bounce for every cell

during a simulation step, reptation can be simulated independently after saltation, by checking the amount of material arriving from the 8 direct neighbors of a cell. In our experiments, this refined algorithm leads to a speedup up to a factor between 2 and 4 compared to the original implementation, depending on the wind regime. Moreover, this scheme should allow an efficient implementation on graphics-hardware, but it is beyond the scope of this research and is left as future work.

4.8 Results and discussion

Experiments were performed on a desktop computer equipped with Intel® Core $i7$, clocked at 4 GHz with 16GB of RAM. The output was streamed into Vue XStream® to produce photorealistic landscapes. Table 4.1 reports the statistics for different landscapes shown throughout this chapter. The simulations were performed with cells ranging from 1 m to 10 m in size. The yardang terrain (Figure 4.16) featuring bedrock abrasion involved the computation of the wind $\mathbf{w}$ according to the varying bedrock at every iteration, hence a higher iteration time. Recall that the code for the simulation is available at https://github.com/aparis69/Desertscapes-Simulation.

Scene	Figure	Size	Grid	Step	Time
Dunes	4.11	0.5×0.5	512	0.12	36
Yardangs	4.16	1×1	512	0.53	371
Mountain	4.1, 4.21	4×4	1024	0.60	300

TABLE 4.1: *Statistics for the scenes shown in this chapter. Terrain size (in* km*), grid discretization, average time of a single simulation step (in seconds) and total time of the simulation (in seconds).*

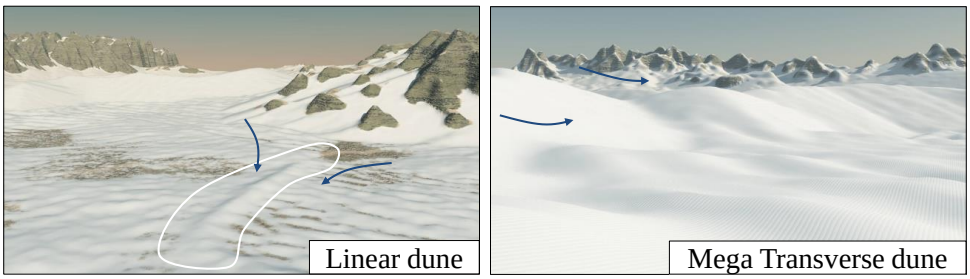


FIGURE 4.21: *Linear dune formed by opposite wind directions, and closeup view of a mega transverse.*

Our method is the first capable of generating a variety of desert landscapes. Figures 4.11 and 4.21 show the variety of sand dunes that can be achieved. Figures 4.1, 4.16 and 4.22 show complex interactions between bedrock and sand with different wind regime conditions prescribed by the user. Figure 4.24 shows several time steps of an editing session and demonstrates the capabilities of our model regarding interactive editing and fine tuning by an experienced user.

4.8.1 Control

The user can interactively change the sand supply at any cell, as well as vegetation density during the simulation. Wind direction, which is the key element of aeolian landscapes, can also be changed at any time. An interactive simulation is a necessary component for authoring: dunes emerge and disappear quickly depending on the wind regime, making interactive visual feedback necessary to allow the user to achieve her particular intent.

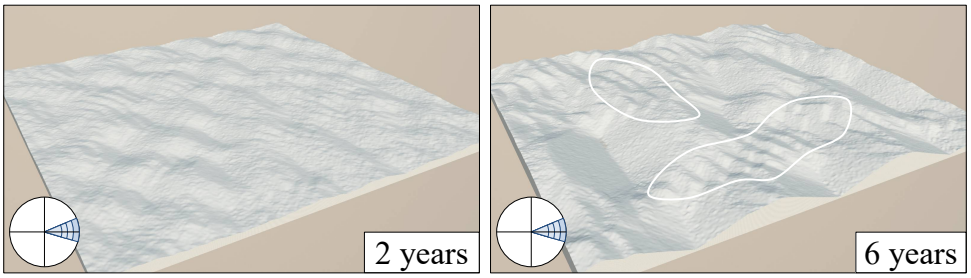


FIGURE 4.22: *Using high speed wind regime increasing linearly over time, we are able to produce multi-scale dunes as found in nature.*

A variety of control mechanisms and the resulting landscapes are showcased in Figure 4.24. Here, the designer modeled an arid terrain covered with a shallow layer of sand, creating barchans dunes. She then triggered abrasion which produced yardangs, which were then covered by adding more sand to get transverse dunes. Nabkha were created by increasing vegetation density and complex star dunes by playing with different wind regimes.

The user may also change the elevation of the bedrock layer to create features such as falling and climbing dunes (see Figure 4.11).

FIGURE 4.23: *Comparison of real (left) and synthesized (right) barchan dunes; no user interaction was needed to create this terrain.*

4.8.2 Validation

Figure 4.23 shows a comparison between real and synthesized barchan dunes. While the generated dunes lacks sharp ridges at their top, we succeed in capturing the overall shape and placement of the dunes. Moreover, we are able to recreate a large number of desert features: transverse, barchan, linear, climbing and star dunes, as well as yardangs produced by abrasion (Figures 4.11, 4.16, 4.21 and 4.24). This qualitatively validates the overall coherency of our simulation. We managed to reproduce complex phenomena and results are consistent with observations and numerical simulations done in geomorphology (Werner 1995; Baas 2002; Momiji *et al.* 2000).

A more complete quantitative validation of our approach is difficult: current available elevation data is not accurate enough to capture detailed desert landforms. A comparison with dense and accurate real elevations would be an interesting research direction worth investigating.

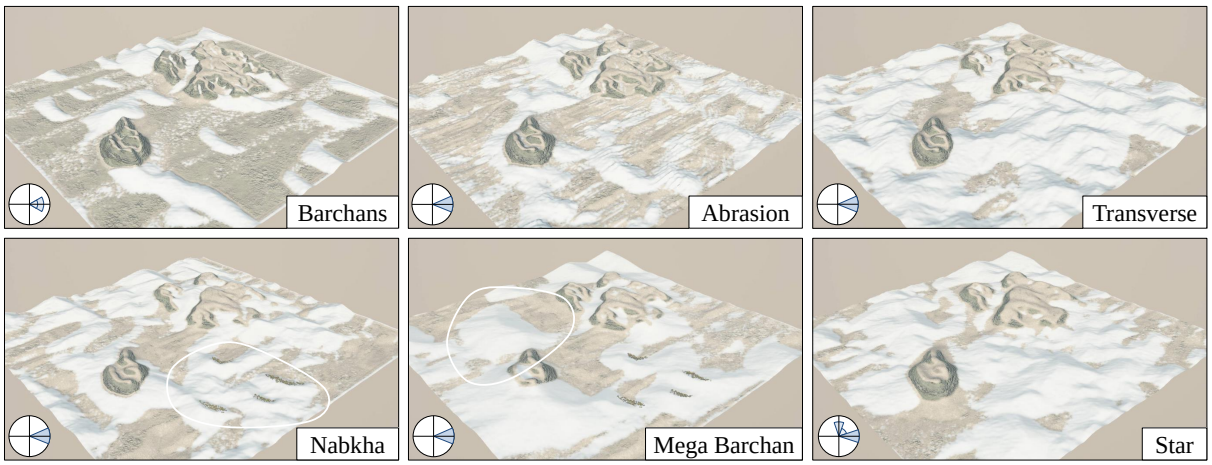


FIGURE 4.24: *Snapshots of an interactive edition session. Starting with a bare bedrock covered by a small amount of sand, barchans emerged because of the low sand supply and the uniform wind conditions. The user then triggered abrasion which shaped yardangs parallel to the wind direction. Then, sand was uniformly added over the entire terrain, which transformed barchans into transverse dunes. Vegetation was later added on the right side, which yielded nabkha. Mega-barchans started to appear after many iterations, as observed in nature. Finally, the user removed some sand to get a more uniform sand layer, destroyed vegetation, and created large dome dunes by changing wind rose parameters.*

4.8.3 Comparison with other techniques

To the best of our knowledge, our model is the first to capture such a wide variety of desert landforms. Previously published methods either focus on generic hydraulic or thermal erosion which primarily generate Alpine mountain ranges.

Previous works directly dealing with the saltation process (Onoue *et al.* 2000; Beneš *et al.* 2004) target the specific application of sand ripples at a much lower scale. Applying a simple scaling factor to the results would not work as the phenomena are not linear. In contrast, our approach is more general and thus achieves a wider range of desert features. Other specific works only apply to a limited range of ventifacts such as Goblins (Beardall *et al.* 2007; Jones *et al.* 2010) which also belong to the small scale class of phenomena. In a sense, these works can be seen as complementary to ours, as they can be used to amplify the generated landscapes with details.

Closer in spirit to our approach is the work of Cordonnier *et al.* 2017 that combines (hydraulic, thermal and lightning) terrain erosion and ecosystem simulation in a unified framework. Our method extends this work and could be seamlessly integrated to it as it also relies on the definition of stochastic events.

4.8.4 Limitations

Our model produces dune topography and landforms similar to the ones observed in geomorphology but does not come without limitations. First, the simulation grid is currently limited to 1024×1024 resolution in order to maintain acceptable computation time and to allow for simultaneous interactive editing and control. The simulation could be accelerated by carefully implementing the algorithm on graphics hardware using the algorithm described in Section 4.7.

Another limitation commonly accepted for all grid-based terrain erosion simulations is the lack of precision. Sand dunes may have sharp features, such as crests or ridge lines, which are not captured by

the simulation even with high resolution grids. These sharp features are the result of more complex wind processes, which we do not model. An efficient, artist-oriented solution consists in using the amplification combined with procedural primitives in the spirit of Génevaux *et al.* 2015 to restore the sharpness of the terrain. This was the essence of our amplification approach for modeling sand accumulation and ripples (Section 4.6).

A wider range of effects observed in geomorphology could be incorporated in our model. For instance, echo dunes (Tsoar 1983) which form on the windward side of cliffs or escarpments, are separated from the scarp by a sand-free region and are the product of the complex movement of wind, forming a fixed eddy between the escarpment and the dune. Such dunes could be obtained by improving the procedural wind warping and shadowing model with procedurally generated edits. Small scale user-controlled wind field currently partially leverages this limitation.

4.9 Conclusion

We introduced a complete aeolian erosion and transport simulation to the field of computer graphics. Derived from the high altitude wind is the fast approximation and computation of the surface wind, taking into account relief shadowing at different scales, which is central to the simulation of sand transport. In turn, *saltation*, *reptation* and *avalanching* processes are simulated in a consistent framework and combined with bedrock erosion to simulate abrasion as well as vegetation shielding to create nabkha.

Our model is versatile and capable of generating all the different dune types as well as abrasion effects on the bedrock. However, obtaining a specific distribution of sand dunes from a simulation is a challenging task, requiring many trial and errors, especially for complex dune shapes. A direct extension to this work would be to incorporate a variety of complex wind scenarios to guide the simulation. An even more noteworthy albeit challenging avenue of future research would be to learn the correlations between the wind and the generated features in an inverse procedural way. Learning which parameters of the simulation lead to some specific dunes distributions and shapes could lead to new insights on how certain types of dunes form and evolve.

Part II

Volumetric terrains: from microscale to macroscale

Abstract

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). Existing solutions, in the literature do not permit the modeling of these varied volumetric features efficiently. In the second part of this thesis, we propose a complete framework for modeling, generating and authoring volumetric terrains across a range of scale (macroscale, mesoscale, and microscale). At the heart of our method is a novel representation for representing volumetric terrains based on implicit surfaces and a construction tree paradigm that arranges primitives to create 3D landforms. This representation is compact in memory, amenable to modifications by the user, and can be used as a basis for complex simulations and procedural algorithms.

Chapter 5 provides the necessary background on implicit modeling for understanding our latter contributions. In particular, we focus on analytic signed distance fields and explain how to properly extract signed distance bound primitives and operators.

In Chapter 6, we show how to generate smooth large-scale 3D landforms as arrangements of skeletal primitives positioned using various techniques, including Poisson sampling, open shape grammars and invasion percolation processes. We show that our method allows generating volumetric features such as canyons, arches, hoodoos from a base 2D heightfield provided as input by the user.

In Chapter 7, we investigate the generation of geologically coherent karstic networks - a set of connected tunnels deep under the terrain surface - through an anisotropic shortest path algorithm and geometric graph generation. Geological conditions such as inception horizons and fracture distributions influence the trajectory of the tunnels and allows reproducing patterns observed in real karstic networks. The mesoscale geometry of the network is modeled using specific implicit primitives and operators that exhibit correct mathematical properties and reproduce identified archetypes from geomorphology.

Finally, Chapter 8 explains how to model detailed volumetric block structures for amplifying a smooth input terrain. Tiles of blocks are first generated using a greedy fracturing algorithm based on a geomorphological classification, and finally replicated on vertical parts of the terrain. On top of the generation of tiles with different geological characteristics, we introduce new primitives and operators allowing for the modeling of mesoscale and microscale features in volumetric terrains.

Chapter 5

Background on implicit modeling

Contents

5.1 Introduction

Implicit surfaces are a powerful tool for modeling and animating shapes of arbitrary topology. They provide a simple and consistent framework for geometric operations such as Boolean operations, blending, and warping. They are compact in memory and theoretically provide infinite precision. They have been used for decades in numerous applications, including fluid animation (Stam *et al.* 2011; Desbrun *et al.* 1995), modeling of molecular structures (Parulek *et al.* 2012), volumetric sculpting (Schmidt *et al.* 2006), and more. They recently regained popularity thanks to platforms such as ShaderToy, clay-based modeling software (MagikaCSG, Clayxels, Adobe Substance Modeler), and for their use in machine learning methods (Park *et al.* 2019; Sitzmann *et al.* 2020).

We distinguish between two categories of implicit surfaces: discrete representations that store the value of the function in a regular or an adaptive grid (Frisken *et al.* 2000), and procedural (or analytic)

models that directly encode the mathematical expression of the function from a list of primitives organized in a construction tree (Wyvill *et al.* 1999). In this thesis, we focus on analytic representations and investigate their use for volumetric terrain modeling. As opposed to (Peytavie *et al.* 2009b; Becher *et al.* 2019), we generate a fully-implicit model of the terrain using a hierarchical structure inspired by the Blob Tree (Wyvill *et al.* 1999). The compact memory aspect and expressiveness of the model are key advantages that we leverage in the following chapters.

Note that visualizing implicit surfaces can be performed either directly using ray tracing approaches (Kalra *et al.* 1989; Hart 1996) or indirectly using polygonization (Araújo *et al.* 2015). A complete description and analysis of those methods is beyond the scope of this thesis and will not be detailed here. However, we focus on Lipschitz techniques, which provide a consistent background for computing ray-surface intersection and other queries that will be used throughout this thesis.

The first part of this chapter presents a general introduction to implicit surfaces and signed distance functions (Section 5.2). The second part introduces the hierarchical tree structure used to define the field function (Section 5.3), along with classical primitives (Section 5.4) and operators (Section 5.5) that we use in the context of volumetric terrain modeling.

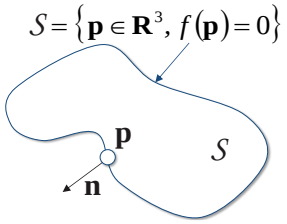
5.2 Fundamentals and notations

In this section we recall the definition of an implicit surface and review some of their properties under simple hypotheses that provide the implicit model with efficient tools for queries, *i.e.* detecting whether a region is inside, outside or straddling the surface, which provides us with efficient adaptive algorithms for computing the intersection between the object and a ray.

5.2.1 Implicit surface

An implicit surface $\mathcal{S}$ is defined as the set of points in space $\mathbf{p} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$ for which a function $f : \mathbb{R}^3 \to \mathbb{R}$ satisfies $f(\mathbf{p}) = 0$:

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



We consider the set of points inside of $\mathcal{S}$ as $\mathbf{p} \in \mathbb{R}^3, f(\mathbf{p}) < 0$ and outside as $\mathbf{p} \in \mathbb{R}^3, f(\mathbf{p}) > 0$ respectively. Additional queries such as the gradient ∇f is defined from the field function:

$$\nabla f(\mathbf{p}) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

It may be computed in closed-form or approximated numerically. Let $\varepsilon > 0$, we have:

$$\nabla f(\mathbf{p}) \approx \frac{1}{2\varepsilon} \begin{pmatrix} f(\mathbf{p} + \varepsilon\mathbf{x}) - f(\mathbf{p} - \varepsilon\mathbf{x}) \\ f(\mathbf{p} + \varepsilon\mathbf{y}) - f(\mathbf{p} - \varepsilon\mathbf{y}) \\ f(\mathbf{p} + \varepsilon\mathbf{z}) - f(\mathbf{p} - \varepsilon\mathbf{z}) \end{pmatrix}$$

Computing the approximation of the gradient can be computationally intensive, as it requires six field function evaluations. Therefore, gradient-based operators and algorithms need to limit the number of gradient queries to perform efficiently. A less computationally intensive approximation requires only

four field function evaluation at the cost of a less accurate approximation:

$$\nabla f(\mathbf{p}) \approx \frac{1}{\varepsilon}\begin{pmatrix} f(\mathbf{p}+\varepsilon\mathbf{x}) - f(\mathbf{p}) \\ f(\mathbf{p}+\varepsilon\mathbf{y}) - f(\mathbf{p}) \\ f(\mathbf{p}+\varepsilon\mathbf{z}) - f(\mathbf{p}) \end{pmatrix}$$

Using the convention $f(\mathbf{p}) < 0$ inside the object, the normal $\mathbf{n}$ of the surface at $\mathbf{p}$ may be derived from the gradient as:

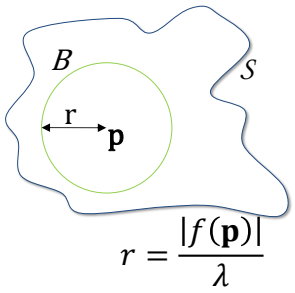
$$\forall \mathbf{p} \in \mathcal{S}\ \mathbf{n}(\mathbf{p}) = \hat{\nabla} f(\mathbf{p}) = \frac{\nabla f(\mathbf{p})}{\|\nabla f(\mathbf{p})\|}$$

5.2.2 Lipschitz property

A function f is said to be Lipschitz over Ω if and only if there exists a positive constant $\lambda > 0$ such that:

$$\forall (\mathbf{p}, \mathbf{q}) \in \Omega \times \Omega, |f(\mathbf{p}) - f(\mathbf{q})| \leq \lambda \|\mathbf{p} - \mathbf{q}\|$$

The Lipschitz constant of f is the minimum value satisfying this equation. Any value overestimating it is called a Lipschitz bound. A function with a Lipschitz constant λ is commonly called a λ-Lipschitz function.

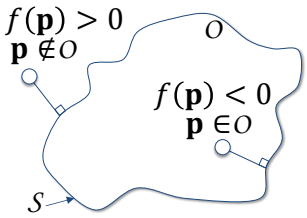


Let $B(\mathbf{c}, r)$ denote the sphere of center $\mathbf{c}$ and radius r, the Lipschitz exclusion criteria (Hart 1996) states that the intersection between the ball B and the surface is empty, thus:

$$\forall \mathbf{p} \in \mathbb{R}^3 \quad B(\mathbf{p}, |f(\mathbf{p})|/\lambda) \cap S = \varnothing$$

Lipschitz conditions are fundamental to implicit surface processing as they provide surface exclusion criteria (Kalra *et al.* 1989; Hart 1996) and monotonicity (Kalra *et al.* 1989), which are essential for deriving guaranteed ray-surface intersection algorithms.

5.2.3 Signed distance function



Signed distance functions, commonly referred to as SDF, are a subset of implicit surfaces, where the function computes a geometric distance to the object's surface. Let $d : \mathbb{R}^3 \to \mathbb{R}$ denote the positive *Euclidean* distance to the surface $\mathcal{S}$:

$$d(\mathbf{p}) = \min_{\mathbf{q} \in \mathcal{S}} ||\mathbf{p} - \mathbf{q}||$$

The *signed* Euclidean distance function $f : \mathbb{R}^3 \to \mathbb{R}$ to the surface $\mathcal{S}$ of an object $\mathcal{O}$ is defined as the positive Euclidean distance outside, negative inside, and 0 on the surface:

$$f(\mathbf{p}) = \begin{cases} d(\mathbf{p}) & \text{if } \mathbf{p} \notin \mathcal{O} \\ 0 & \text{if } \mathbf{p} \in \mathcal{S} \\ -d(\mathbf{p}) & \text{otherwise.} \end{cases}$$

A signed distance *bound* is formally defined as a lower bound function $b : \mathbb{R}^3 \to \mathbb{R}$ such that:

$$\forall \mathbf{p} \in \mathbb{R}^3, |b(\mathbf{p})| \leq |f(\mathbf{p})|$$

The function b always provides a lower bound to the distance to the surface, thus it can be used as a safe marching distance for intersection algorithms, such as sphere tracing (Hart 1996). It is possible to derive a signed distance bound from any pseudo-distance function f through the use of one of its Lipschitz bounds λ, by using $f(\mathbf{p})/\lambda$ as the field function. The Lipschitz constant of a C^1 function f may be defined as the upper norm of the gradient of the underlying function $\lambda = \max ||\nabla f||$. In some cases where f is C^2, this can be achieved by finding the roots of the second derivative of f and plugging those roots into f'.

In this thesis, we aim at defining continuous 1-Lipschitz functions that are lower signed distance bounds to the surface. We derive all Lipschitz constants in Appendix A.

5.3 Hierarchical model

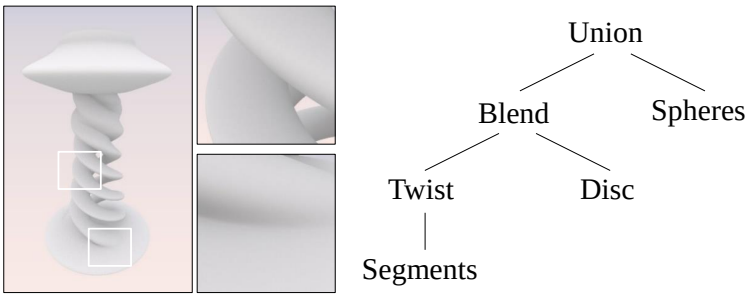


FIGURE 5.1: *A construction tree with primitives and operators defining a candlestick (after (Wyvill* et al. *1999), see Figure 1.3 for inspiration).*

We propose a hierarchical model for constructing the field function f as a signed distance field. Our approach takes inspiration from the Blob Tree (Wyvill *et al.* 1999). Instead of combining compactly supported primitives built with a falloff function combined with skeletal primitives, we aim at constructing signed distance functions directly, with leaves of the tree defining a signed distance to the surface. A depth-first walk of such a tree is equivalent to a function evaluation for a point $\mathbf{p}$. This data structure allows taking full advantage of the compact memory aspect of implicit surfaces, as it is possible to define complex shapes with very few nodes (see Figure 5.1). As opposed to the Blob Tree, we encode shapes as 1-Lipschitz continuous signed distance functions and adapt the operators accordingly. A similar construction tree operating with signed distance field primitive was introduced in Reiner *et al.* 2011, however, few details are provided regarding the Lipschitz properties of the proposed distance functions.

5.4 Skeletal primitives

A vast variety of skeletal primitives exist in the scope of the Blob Tree model, from simple points or line segments (Wyvill *et al.* 1986), anisotropic distance primitives (Crespin *et al.* 1996), and curve or volume based primitives such as cylinders, discs, cones or curves introduced in Barbier *et al.* 2004, which are useful for modeling and morphing more complex skeletal models. These primitives are defined by the signed Euclidean distance to their skeleton, thus they are 1-Lipschitz by construction. Here we briefly review some primitives that are commonly used in our system: simple sphere and box primitives are used to carve volumetric terrain landforms such as arches, overhangs and hoodoos (Chapter 6), while complex curves and segments are used for modeling karstic tunnels (Chapter 7).

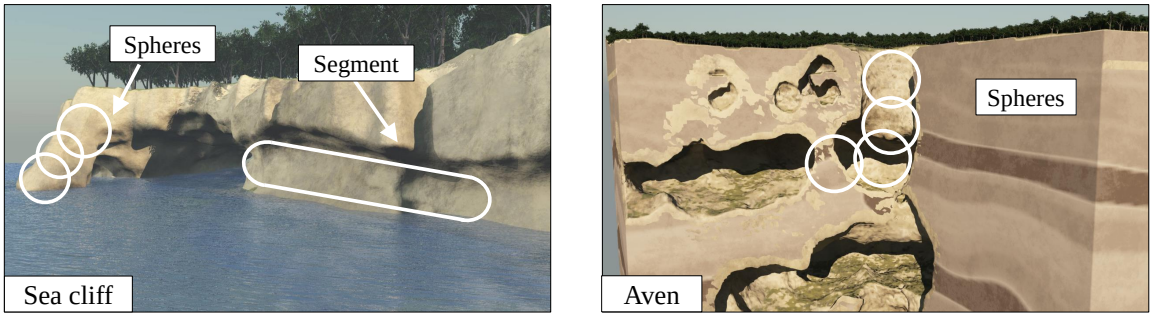


FIGURE 5.2: *Skeletal primitives are useful for modeling volumetric features in virtual terrains. Segments are useful for carving overhangs over large zones, while spheres can be used to model arches and adding irregularities (Chapter 6).*

5.4.1 Sphere

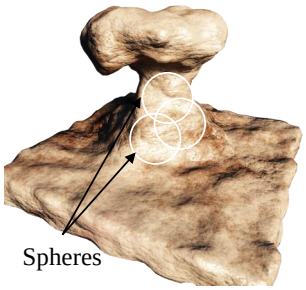


FIGURE 5.3: *Ventifact made with smooth union of spheres.*

The simplest signed distance primitive is the sphere. Let $\mathbf{c}$ denote the center of the sphere and r its radius, the distance function is:

$$f(\mathbf{p}) = ||\mathbf{p} - \mathbf{c}|| - r$$

Spheres are often combined to create more complex shapes such as ventifacts (Figure 5.3). In Chapter 6, we show how to carve overhangs and build arches using arrangements of spheres, placed by an invasion-percolation process (Figure 5.2).

Instead of using the Euclidean distance $\|\mathbf{p} - \mathbf{c}\|$, it is possible to use the L^p norm and generate a variety of shapes using $f(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}\|_p$. The L^p norm of a vector $\mathbf{v}$ is defined as $\|\mathbf{v}\|_p = (|x|^p + |y|^p + |z|^p)^{1/p}$. Let λ_p denote the Lipschitz constant of the L^p norm, we define super-ellipsoid distance as:

$$f(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}\|_p / \lambda_p - r$$

Using $p \geq 2$ preserves the Euclidean distance property ($\lambda_p = 1$), but this is no longer true for $p < 2$. In this case, the Lipschitz constant is defined as $\lambda_p = 3^{1/p-1/2}$ (see Appendix A for demonstration).

5.4.2 Box

Boxes are essential primitives, particularly useful for defining the 1-Lipschitz signed distance function for a heightfield within a finite domain (Figure 5.4 and Chapter 6), by using the intersection operator (Section 5.5.1). Boxes are also used for generating complex shapes, such as hoodoos and goblins (Chapter 6).

Calculating the signed distance to a box skeleton can be tedious, as 9 different cases arise. However, by exploiting planar symmetries, it is possible to define the function in a compact form. Let $\mathbf{o}$ the origin, $\mathbf{c}$ the center of the box, $\mathbf{h}$ the half diagonal, we define f as:

$$f(\mathbf{p}) = ||\max(\mathbf{q}, \mathbf{o})|| + \min(\max(\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z), 0) \qquad \mathbf{q} = |\mathbf{p} - \mathbf{c}| - \mathbf{h}$$

The definition of $\mathbf{q}$ involves the absolute value of the components of the vector $\mathbf{p} - \mathbf{c}$.

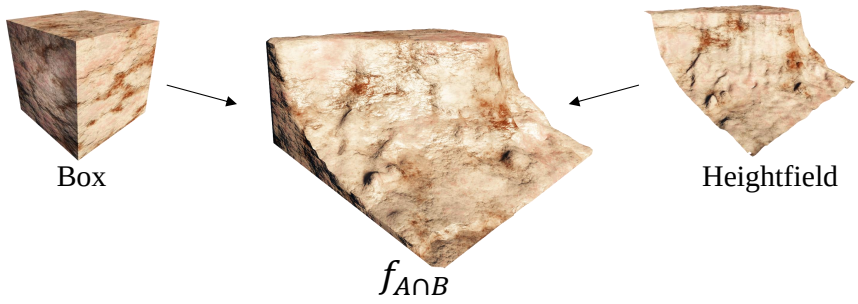


FIGURE 5.4: *Intersection between a box and a heightfield for constructing a 1-Lipschitz continuous signed distance function.*

5.4.3 Segment and curve

Carving overhangs or creating volumetric landforms over large zones may require thousands of spheres, which in turn may be computationally intensive and with a high memory impact. In contrast, a single segment or curve can cover a large zone efficiently (Figure 5.2). We often use segments to model overhangs, and more complex curves to generate arches (Figure 5.5) and karstic tunnels (Chapter 7).

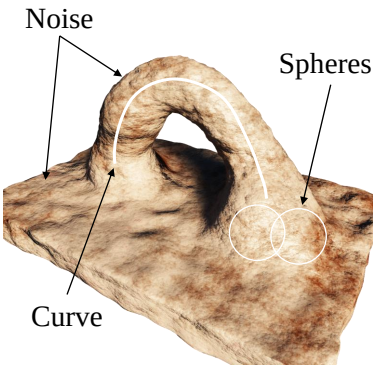


FIGURE 5.5: *Arch made from a curve and spheres.*

Segments Let $\mathbf{a}$ and $\mathbf{b}$ the two endpoints of the segment, $\mathbf{u}$ the normalized direction, and $l_a = (\mathbf{p} - \mathbf{a}) \cdot \mathbf{u}$, $l_b = (\mathbf{p} - \mathbf{b}) \cdot \mathbf{u}$, the signed distance is defined as:

$$f(\mathbf{p}) = \begin{cases} ||(\mathbf{p} - \mathbf{a})|| & \text{if } l_a < 0 \\ ||(\mathbf{p} - \mathbf{b})|| & \text{if } l_b > 0 \\ ||(\mathbf{p} - \mathbf{a})|| - ((\mathbf{p} - \mathbf{a}) \cdot \mathbf{u}) & \text{otherwise.} \end{cases}$$

Curves More complex curves with a higher degree of continuity allow generating landforms on curvilinear trajectories. We aim at computing the minimum distance from $\mathbf{p}$ to a parametric curve Γ of equation $\mathbf{c} : u \in [0, 1] \rightarrow \mathbb{R}^3$:

$$d(\mathbf{p}, \Gamma) = \min_{u \in [0,1]} \|\mathbf{p} - \mathbf{c}(u)\|$$

Finding the minimum distance is equivalent to finding $u \in [0, 1]$ such that the derivative $\|\mathbf{p} - \mathbf{c}(u)\|$ is null, *i.e.* $(\mathbf{p} - \mathbf{c}(u)) \cdot \mathbf{c}'(u) = 0$. If c is a polynomial of degree n, then $(\mathbf{p} - \mathbf{c}(u)) \cdot \mathbf{c}'(u)$ is of degree $2n - 1$.

For quadratic curves, we need to solve a cubic equation for which solutions can be computed analytically. For cubic curves and above, we need to solve quintic, septic and polynomials of higher degree, which can only be done using numerical techniques. One advantage of cubic curves lies in the precise control of the trajectory, making them more intuitive for the user than quadratic curves. Thus, in Chapter 7, we use cubic spline curves for controlling the exact path of karstic tunnels, and approximate the trajectory using piecewise quadratic curves (Truong *et al.* 2020) that provide $10 - 20$ faster distance computation.

5.5 Binary operators

Binary operators include Boolean (union, intersection and difference) (Wyvill *et al.* 1999) and smooth Boolean operators (Barthe *et al.* 2001). In any case, these operators do not preserve the Euclidean distance property: either the interior or the exterior distance is not exact, thus they represent signed distance bounds. In the remainder of this section, we denote A and B two objects with their associated signed distance functions $a : \mathbb{R}^3 \to \mathbb{R}$ and $b : \mathbb{R}^3 \to \mathbb{R}$.

5.5.1 Boolean operators

Boolean operators, *i.e.* union, intersection and difference, are commonly implemented as using $\min$ and $\max$ operations (Wyvill *et al.* 1999):

$$f_{A\cup B} = \min(a, b) \qquad f_{A\cap B} = \max(a, b) \qquad f_{A-B} = \max(a, -b)$$

Alternative techniques have been proposed, in particular in the scope of R-Functions (Pasko *et al.* 1995), where the union and intersection were originally defined as:

$$f_{A\cup B} = (a + b + \sqrt{a^2 + b^2})$$
$$f_{A\cap B} = (a + b - \sqrt{a^2 + b^2})$$

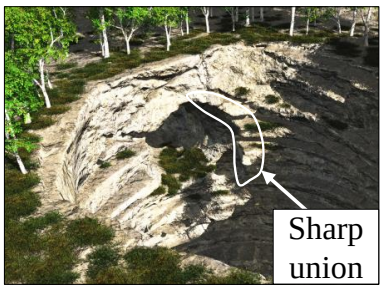


FIGURE 5.6: *Union operator for carving a karstic tunnel.*

The corresponding Lipschitz constant for those R-function operators is $\lambda_R = 2(\lambda_A + \lambda_B)$ (see Appendix A), thus in our context we would define the signed distance functions as $f_{A\cup B}/\lambda_R$ and $f_{A\cap B}/\lambda_R$, respectively. This more complex definition yields a C^1 scalar field f almost everywhere in space except on the surface of A and B, *i.e.* over $\mathbb{R}^3 - (A \cup B)$. While we could have used them for modeling and carving terrains, those operators are more computationally intensive. Moreover, we often need to use smooth Boolean operators that do not produce sharp edges (Figure 5.6) between geometric primitives, as presented in the next section.

5.5.2 Smooth Boolean operators

In the Blob Tree model (Wyvill *et al.* 1999), primitives are defined as the composition of a compactly-supported falloff function g and a distance function d: $f(\mathbf{p}) = g \circ d(\mathbf{p})$. The resulting scalar field allows defining blending as trivially as $f_{A+B} = a + b$. However, this does not generalize to signed distance fields, as summing the two distances does not produce the desired effect. Smooth blending between models has been an active research field, and many operators were defined particularly for Blobs (Wyvill *et al.* 1999) and R-functions (Pasko *et al.* 1995). Here we define smooth Boolean operations following the definition of Pasko *et al.* 1995 and later extended by Barthe *et al.* 2001. Let r denote the control radius, we define the smoothing function $k : \mathbb{R} \times \mathbb{R} \to [0, 1]$ as:

$$k(x, y) = \begin{cases} 1 - |x - y|/r & \text{if } |x - y|/r < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Following this definition, we can define the smooth union $\widetilde{\cup}$, smooth intersection $\widetilde{\cap}$ and smooth differ-

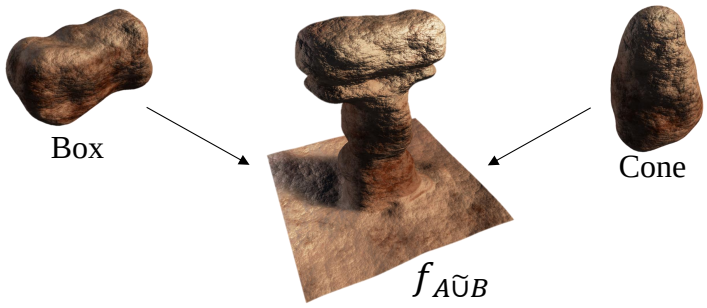


FIGURE 5.7: *Smooth union operator between a box and a cone for modeling a hoodoo.*

ence $\tilde{-}$ operators as:

$$f_{A\tilde{\cup}B} = \min(a,b) - g(a,b) \qquad g(a,b) = rk(a,b)^3/6$$
$$f_{A\tilde{\cap}B} = \max(a,b) + g(a,b)$$
$$f_{A\tilde{-}B} = -\min(-a,b) + g(a,b)$$

Smooth Boolean operators are particularly useful for modeling smooth volumetric landforms (Figure 5.7), such as hoodoos progressively carved by the action of wind and water. They also regularize the gradient of the function, leading to a C^1 continuity on the subtree. Chapter 6 and Chapter 7 use the smooth difference operator to carve overhangs and deep karstic tunnels in the terrain. In Chapter 8, we take advantage of the C^1 continuity of the smooth intersection to define detailed mesoscale block structures. In Appendix A, we demonstrate that these smooth operators are 1-Lipschitz.

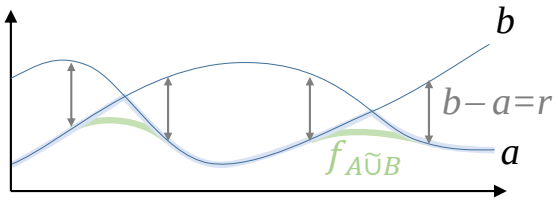


FIGURE 5.8: *Smooth union between two functions a and b.*

5.6 Unary operators

Unary operators include space transformation, such as affine transformations or more complex warping (Barr 1984), displacement, rounding operators as well as replication. In the following sections, we refer to the object as $\mathcal{N}$, with $f_{\mathcal{N}}$ its distance function.

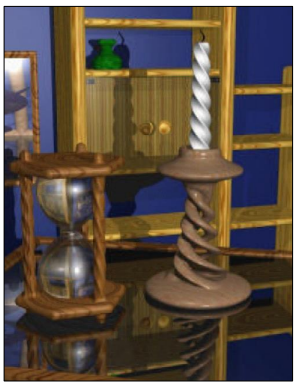


FIGURE 5.9: *Twisting in the Blob Tree (Wyvill* et al. *1999)*

5.6.1 Warping

Warping is a space deformation. It is characterized by a warping function $\omega : \mathbb{R}^3 \to \mathbb{R}^3$ that deforms space or a region of space of the sub tree $\mathcal{N}$. As the definition of the surface is implicit, we need to define the inverse transformation ω^{-1}. The general definition of a warping

node is thus:

$$f_\omega(\mathbf{p}) = f_\mathcal{N} \circ \omega^{-1}(\mathbf{p})$$

The resulting distance estimate is usually a signed distance bound to the surface. To guarantee the 1-Lipschitz property, it is necessary to bound the gradient of the deformation $\nabla f_{\omega^{-1}}$, which involves the computation of the inverse transpose of the 3×3 Jacobian matrix $\mathbf{J}$ of ω^{-1} (Kalra *et al.* 1989):

$$\nabla f_\omega(\mathbf{p}) = (\mathbf{J}^{-1})^t \cdot \nabla f_\mathcal{N} \circ \omega^{-1}(\mathbf{p})$$

The Lipschitz constant of the warping node $\lambda_{\omega^{-1}}$ can then be bounded using the norm of the Jacobian matrix and the Lipschitz constant of the underlying node $\lambda_\mathcal{N}$:

$$\lambda_{\omega^{-1}} \leq ||\mathbf{J}|| \cdot \lambda_\mathcal{N}$$

The Jacobian matrix is difficult to compute in the general case. Compact closed-form expressions exist for tapering, twisting and bending (Barr 1984), and have been used in practice in the Blob Tree (Figure 5.9). Global warping is computationally demanding, and difficult to control for generating details. We prefer either sculpting volumetric terrains with primitives, or using a novel gradient warping method to generate microscale details over the surface of the terrain (Chapter 8).

5.6.2 Affine transformations

Affine transformations are a special case of warping. They include translation, rotation, and scaling. Let t denote a translation vector, s a scaling factor, and $\mathbf{R}$ a 3×3 rotation matrix, the field functions for affine transformations are defined as:

$$f_T(\mathbf{p}) = f_\mathcal{N} \circ (\mathbf{p} - t) \qquad f_R(\mathbf{p}) = f_\mathcal{N} \circ \mathbf{R}^{-1}(\mathbf{p}) \qquad f_S(\mathbf{p}) = s \times f_\mathcal{N} \circ (\mathbf{p}/s)$$

Translations and rotations are 1-Lipschitz and thus conserve the Euclidean distance property. Defining a 1-Lipschitz function for a scaling with a factor s requires multiplying by $1/s$, which is the norm of the Jacobian matrix of transformation (Barr 1984), that can be easily expressed in this case. In Chapter 7, we model tunnel primitives in canonical space, and position them using translations and rotations.

5.6.3 Noise

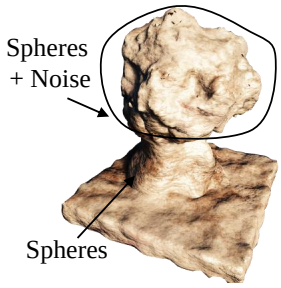


FIGURE 5.10: *Noise displacement.*

Displacement is a common operator for adding details to an implicit surface (Figure 5.10). Let $t : \mathbb{R}^3 \rightarrow \mathbb{R}$ denote a sum of scaled noise, also refereed to as turbulence (Ebert *et al.* 1998), we define the field function as:

$$f(\mathbf{p}) = \frac{f_\mathcal{N}(\mathbf{p}) + t(\mathbf{p})}{1 + \lambda_t}$$

In this example, it is necessary to divide by the Lipschitz constant of the operator $(1 + \lambda_t)$ to preserve a 1-Lipschitz function (see Appendix A for the derivation). The main limitation of this operator is that floating parts can appear due to the turbulence evaluated at a point $\mathbf{p}$. In Chapter 6, we show how to define detailed noised-based primitives without any floating parts, by evaluating the noise on the surface of the underlying skeleton.

5.7 Conclusion

Implicit surfaces provide a powerful framework for modeling a variety of shapes using a compact mathematical expression. The construction tree formalism is useful as it provides a high level paradigm for the user, who manipulates primitives and operators intuitively. However, it is necessary to ensure that the primitives and operators are well defined to guarantee the convergence of fundamental algorithms.

In this thesis, we define 1-Lipschitz continuous signed distance functions that are lower signed distance bounds to the surface. The Lipschitz property ensures that the surface exclusion criteria is correct, and allows for a guaranteed convergence of intersection algorithms such as sphere tracing. In the next chapters, we introduce a fully implicit surface framework for modeling, generating, and authoring volumetric terrains across a range of scales. We show that signed distance functions are able to represent smooth and detailed mesoscale and microscale volumetric terrain landforms efficiently.

Chapter 6

Terrain amplification with implicit 3D features

Contents

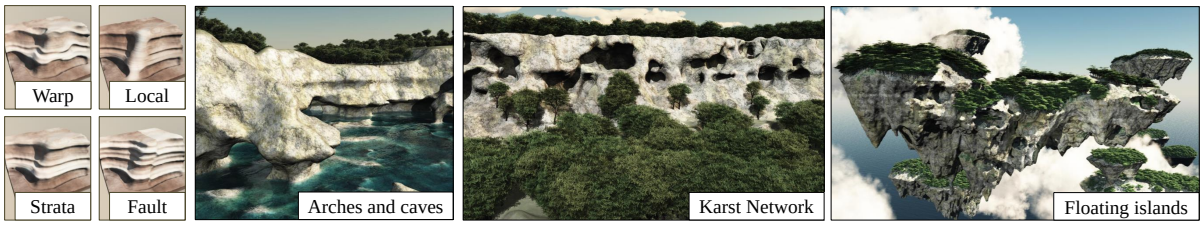


FIGURE 6.1: *From a 2D input heightfield, our method automatically generates an implicit model for representing the terrain, which is augmented with complex volumetric landform features such as caves, overhangs, cliffs, arches or karsts. The model can also represent scenic fictional landscapes such as floating islands, or giant rock spires.*

6.1 Introduction

The sheer variety of shapes and scales of volumetric landforms presents significant modeling challenges. Despite the wide application of digital terrain in games, film, and simulation, and extensive research in this area, effectively representing and generating complex landforms such as caves and overhangs remains an unsolved problem. The very reason for this is that existing techniques mostly address elevation terrains. As a consequence, steep areas, such as cliffs, are generally undersampled, and overhanging features simply cannot be represented. Existing explicit representations are problematic as such structures are memory consuming and, consequently, previous volumetric terrain approaches either focus on small scale isolated landforms (Ito *et al.* 2003; Beardall *et al.* 2007; Jones *et al.* 2010) or represent larger landscapes at a limited sampling resolution (Peytavie *et al.* 2009b; Becher *et al.* 2017; Becher *et al.* 2019).

In this chapter, we provide a conceptually simple solution to the problem of representing and authoring 3D terrain, achieved through a unified *implicit surface model* (Chapter 5). This allows 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 cave complexes to coastal cliffs, incorporates geomorphological effects, and provides extensive user control (Figure 6.1).

The presented implicit model allows the automatic enhancement of terrains with complex volumetric 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 volumetric terrain. In this we are inspired by and extend the notion of construction tree for defining implicit surfaces, as in the Blob Tree introduced by Wyvill *et al.* 1999, signed distance fields (Reiner *et al.* 2011) and feature primitives (Génevaux *et al.* 2015). The pipeline imports a an elevation terrain and converts it into a coherent implicit surface (represented as a signed distance function), identifies feature locations, 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 implicit representation is efficiently polygonized using a novel locally adaptative approach that generates a final mesh amplified with terrain features.

More precisely, the main technical contributions of this chapter include:

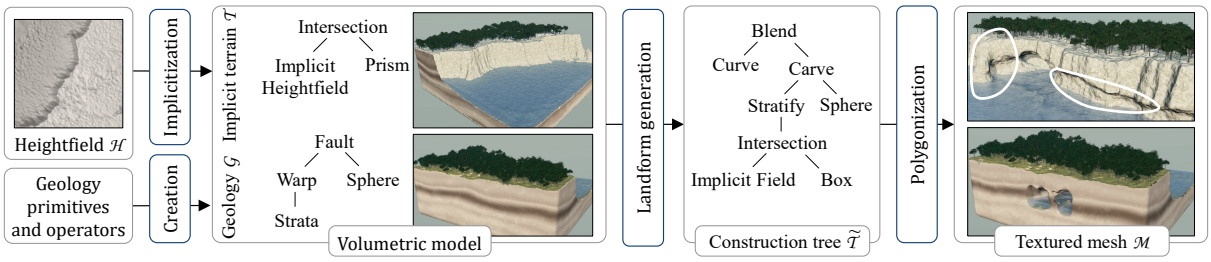


FIGURE 6.2: *Overview of our terrain amplification: Starting from a 2D heightfield $\mathcal{H}$, we first perform an implicitization process to create an implicit terrain model $\mathcal{T}$ suitable for 3D augmentation. At the same time, a model of the underlying geology $\mathcal{G}$ is created by the user. Next, a landform generation process converts $\mathcal{T}$ into an augmented construction tree $\widetilde{\mathcal{T}}$ with sparse volumetric features where required. Efficient polygonization is then used to extract a final mesh.*

- A procedural model for representing the underlying geology of a terrain (Section 6.3) and guiding the generation processes (Section 6.5) in a memory-efficient fashion.
- A coherent implicit surface-based landform construction tree (Section 6.4) that supports the compact encoding of terrains with local volumetric landforms, such as arches and overhanging cliffs.
- 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 6.5).
- An efficient implicit surface polygonization algorithm (Section 6.6) adapted to the sparse amplified terrain data-structure.
- A coherent framework that supports both procedural landforms shaping processes and interactive editing for the creation of complex terrains.

We first give a brief overview of the method (Section 6.2), and explain both geological and terrain models built upon specific implicit primitives and operators (Section 6.3 and Section 6.4). Then, we show how to generate various 3D landforms using different strategies (Section 6.5), and present an efficient space pruning method to polygonize the generated implicit surface (Section 6.6). Finally, we present results and discuss various aspects of our system regarding performance, realism and control (Section 6.7).

The work presented in this chapter was published in Paris *et al.* 2018 and Paris *et al.* 2019a, and received the Replicability Stamp.

6.2 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 volumetric terrain features (see Figure 6.2).

6.2.1 Construction tree models

Two structures are central to volumetric 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 presented in Chapter 5.

In the case of geology, leaves in the construction tree are implicit skeletal primitives that define rock resistance for every point in space. The nodes are either binary operators combining sub-trees or unary operators reproducing folds and faults using various forms of warping. The geology tree defines a resistance function, denoted as ρ. The geology construction tree can be interpreted as a variant of the Blob Tree model (Wyvill *et al.* 1999) extended with new primitives and operators suited for representing geological features.

In the case of the terrain, 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. While the original publication (Paris *et al.* 2019a) presents a terrain model based on the BlobTree, we define the terrain field function f as a signed distance field model (Reiner *et al.* 2011) to be consistent with the following chapters.

6.2.2 Amplification workflow

The stages of the amplification process are depicted in Figure 6.2. To begin with, we automatically convert the representation from a 2D heightfield $\mathcal{H}$, provided as input, to an implicit terrain model $\mathcal{T}$ (Section 6.4). 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, 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, incorporating faults and folds, or re-weighting specific local erosion effects (see Section 6.5).

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.

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

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

6.3 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 includes 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.

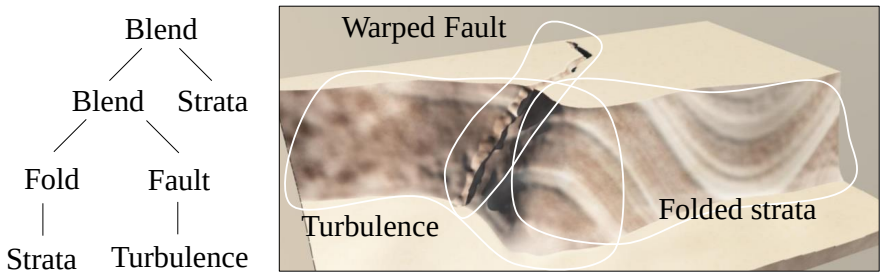


FIGURE 6.3: *An example of the hierarchical construction of a complex geological structure. Horizontal strata representing rock layers at different consistency are folded by warping (on the right) and this is separated off by a fault line from a turbulence function (on the left). Blend nodes combine the subtrees.*

The 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 6.3), with internal nodes that modify or combine resistance values spatially demarcated by the leaf node primitives. We created several specific primitives and warping operators in order to effectively model bedrock strata. The geological construction tree is used as a guide to the generation process (Section 6.5), as landform primitives are placed at locations with minimum resistance in the scene.

6.3.1 Turbulence-based 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 6.3 showcases two kinds of turbulence primitives: noise on the left and a strata obtained from a turbulence combined with a fold on the right. These primitives are often used as a basis for more complex geological settings as they define a global stratification of the scene.

6.3.2 Plane primitives

Turbulence allows us to create stratified geological features easily but lack user control. Thus, we introduce plane primitives, defined by the distance from a plane, which acts as a central core (Figure 6.4 [*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 terrains, the geology tree was first created by blending multiple planes with a turbulence primitive. As in the BlobTree, blended resistance is defined as the sum of the resistance of the sub-trees: $\rho_{A+B} = \rho_A + \rho_B$

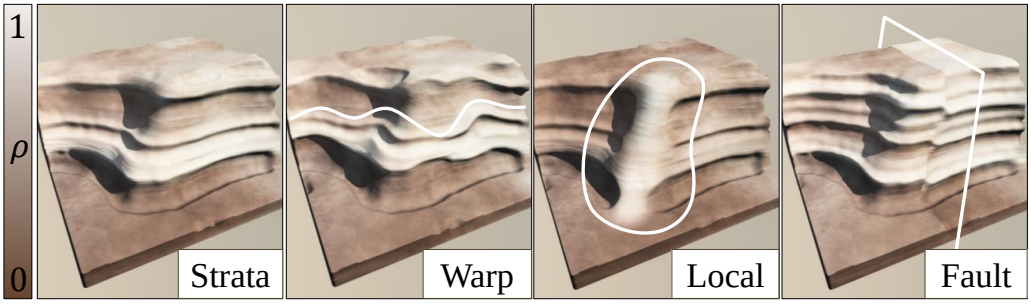


FIGURE 6.4: *Different forms of geology showcased on a simple cliff terrain: simple strata combined with noise, folds produced by a warping operator, a local increase in bedrock resistance produced by spheres, and a fault line. Pale colors map to more resistant and darker to less resistant bedrock, respectively. Note that we applied a small erosion to the cliff to visually differentiate the strata.*

In addition, we enhance user control with skeletal primitives (spheres and curves blended with the construction tree) that locally modify bedrock resistance (Figure 6.4 [*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 6.5).

6.3.3 Fold and deformation operators

Folds and deformations (see Figure 6.4 [*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 turbulence function $t : \mathbb{R}^3 \to \mathbb{R}$ defined as a sum of scaled noise functions 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.

6.3.4 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 6.4[*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.

6.4 Implicit terrain model

The terrain model $\mathcal{T}$ is based on the same underlying hierarchical construction tree as the geology model. The difference is that primitives and their subtrees portray terrain landforms, such as hoodoos (Figure 6.5), rather than strata and bedrock resistance. Crucially, the model must be amenable to the visualization of the underlying surface, e.g using ray tracing or polygonization. To fill this requirement, we associate a signed distance function $f : \mathbb{R}^3 \to \mathbb{R}$ with $\mathcal{T}$ defining the signed distance to the terrain surface. As explained in Chapter 5, the isosurface $\mathcal{S}$ of the terrain is then defined as the set of points where the field function equals 0:

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

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

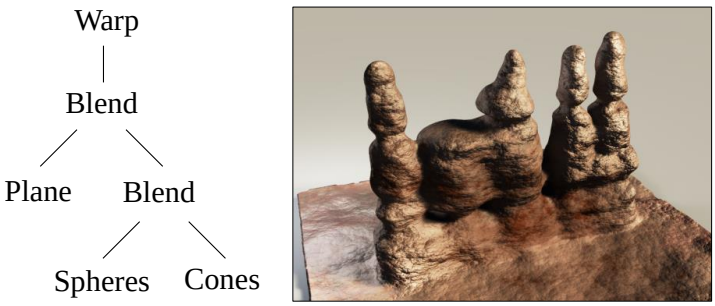


FIGURE 6.5: *In this example the hoodoos were created by blending several perturbed spheres and cones, and merging with the ground.*

6.4.1 Implicitization of elevation terrains

The heightfield representation $h : \mathbb{R}^2 \to R$ is not adapted to modeling volumetric landforms; consequently, transforming input elevation terrains into an implicit construction tree representation is a necessary precursor to any volumetric operations. The challenge is to derive a signed distance bound $f : \mathbb{R}^3 \to \mathbb{R}$ from $\mathcal{H}$, such that the resulting function is 1-Lipschitz. The Lipschitz property is crucial for establishing surface exclusion criteria, with applications in ray tracing and polygonization.

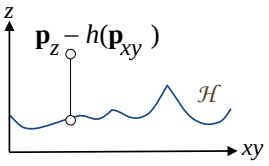


FIGURE 6.6: *Vertical distance to a heightfield.*

Simply using the vertical distance to the terrain $f(\mathbf{p}) = \mathbf{p}_z - h(\mathbf{p}_{xy})$ yields an unbounded signed distance function. To create a signed distance bound from an elevation function, we need to divide by its Lipschitz constant (Chapter 5 Section 5.2.2). Let λ denote the Lipschitz constant of h, the resulting signed distance function

associated with $\mathcal{H}$ is defined as:

$$f_{\mathcal{H}}(\mathbf{p}) = \frac{\mathbf{p}_z - h(\mathbf{p}_{xy})}{\sqrt{1+\lambda^2}}$$

This equations guarantees that f is 1-Lipschitz, *i.e.*, represents a signed distance bound to the surface of the terrain (see Appendix A.2 for a complete derivation). In practice, users may use bounded $\mathcal{H}$ such as heightfield computed from real elevation data. Thus, it is necessary to ensure that the distance is properly defined outside the domain of the elevation function. Let $\mathcal{P}$ denotes the enclosing prism of the $\mathcal{H}$ and $f_{\mathcal{P}}$ its corresponding signed distance function, we define the terrain field function as:

$$f_{\mathcal{T}}(\mathbf{p}) = \max(f_{\mathcal{H}}(\mathbf{p}), f_{\mathcal{P}}(\mathbf{p}))$$

Using an intersection operator ensure that the field function is continuous and 1-Lipschitz everywhere in the domain (Figure 6.7).

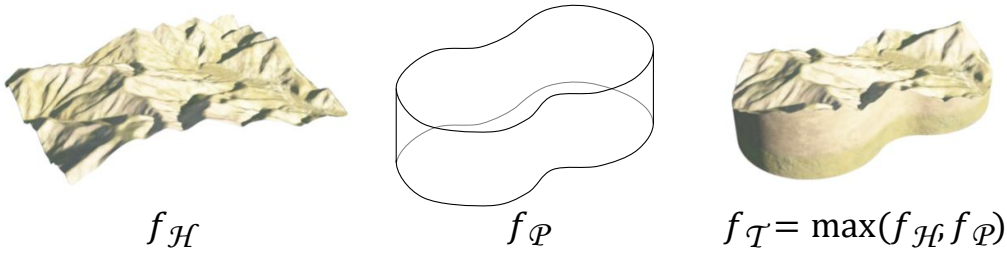


FIGURE 6.7: *By using a prism $\mathcal{P}$ with an associated distance function $f_{\mathcal{P}}$, we define a volumetric terrain primitive $f_{\mathcal{T}}$ from a bounded heightfield $\mathcal{H}$.*

6.4.2 Sculpting primitives

We aim at augmenting an implicit terrain model $\mathcal{T}$ with sculpting primitives to create a range of volumetric features, such as arches and caves. We chose sculpting primitives as a subset of skeletal primitives controlled by a geometric skeleton $\mathcal{S}$ and a radius r controlling the thickness. They are defined using the Euclidean distance $d(\mathbf{p}, \mathcal{S})$ to the skeleton $\mathcal{S}$, which makes their field function a well defined signed distance function.

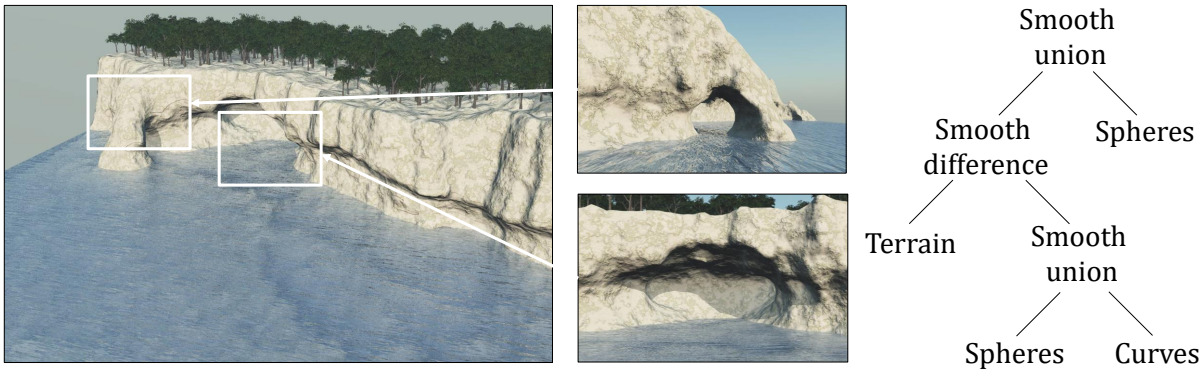


FIGURE 6.8: *We use spheres combined with the terrain using a union operator to create arches. Overhangs and karstic tunnels are created by using a difference operators.*

Spheres and extrusion curves are helpful for delineating linear features, such as stratification and overhangs made by erosion, but also for creating arches and deep karst structures, as described in Section 6.5. 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 *et al.* 2008).

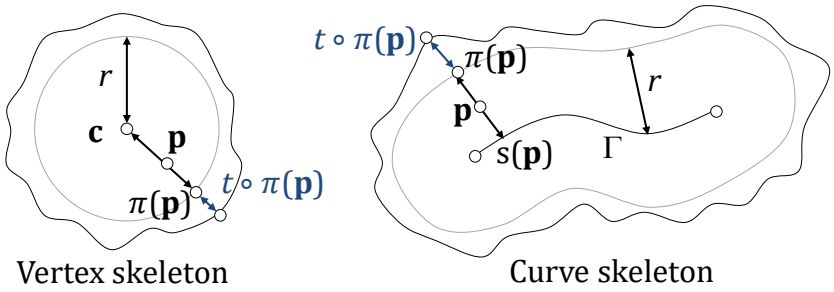


FIGURE 6.9: *Skeletal primitives with anisotropic star-shaped noise displacement for point* $\mathbf{c}$ *and curve* Γ *skeletons.*

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

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

This method can be used to perturb the shape of any skeleton without artefacts, as illustrated in Figure 6.9 for the case of point-based and curve-based anisotropic star-shaped primitives. A 1-Lipschitz signed distance bound can also be defined for these primitives (see Appendix A.5).

6.4.3 Operators

Operators are internal nodes that combine sub-trees and include Boolean and so called smooth Boolean operators (Chapter 5, Section 5.3) to combine the landform construction trees with the base terrain. Union, intersection and difference are helpful for creating sharp transitions between landforms, such as the different strata of a Hoodoo. Smooth Boolean operators create rounded edges and are adapted for creating a more gentle transition between landforms and the terrain, giving a more natural aspect to the result. They are useful for carving deep overhangs on cliffs, and extending the terrain with arches.

While warping is a general useful tool in implicit modeling, we found carving volumetric landforms using arrangements of thousands of skeletal primitives to be more efficient. Recall that a warping operator is defined over a subtree f as $\widetilde{f}(\mathbf{p}) = f \circ \omega^{-1}(\mathbf{p})$ with $\omega^{-1} : \mathbb{R}^3 \to \mathbb{R}^3$. Applying a global warping to the terrain may be computationally expensive. Furthermore, defining a signed distance bound for such

operator involves the derivation of the warping function, which may not be trivial to compute. Thus, warping nodes may be more useful when used on a small sub region of space. Chapter 8 shows how to define a local warping operator for adding microscale details to the surface of the terrain.

6.5 Landform generation

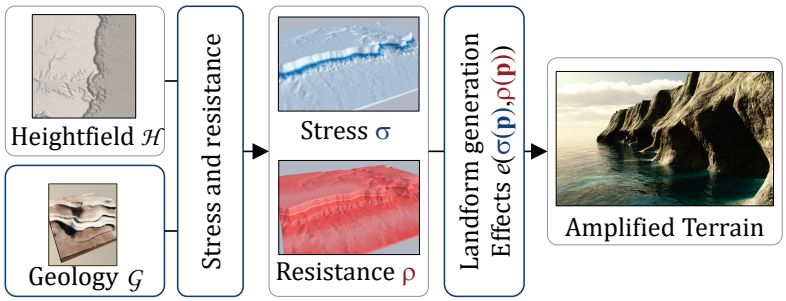


FIGURE 6.10: *An overview of our landforms generation pipeline.*

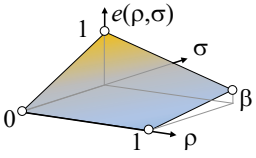
Volumetric landforms are the result of complex erosion processes involving the shape of the terrain, its geology, and the action of environmental erosive agents. Simulating those phenomena would be computationally intensive and would prevent interactive control. We thus avoid physically-based simulation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

Generally, landforms generation algorithms proceed in two phases (Figure 6.10). First, given an input elevation 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 spheres that are used to carve the terrain using difference operators. In contrast, Hoodoo and goblin generation accretes spheres and cones in an additive growth process using a union operator.

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

6.5.1 Shallow procedural erosion

Shallow procedural erosion encompasses erosion processes that impact the terrain to a limited depth. This is the case for sea and stream erosion that produce small overhangs and carved channels in the bedrock. Following the general template for landform generation, we proceed in two steps: we first perform a 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 spheres 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

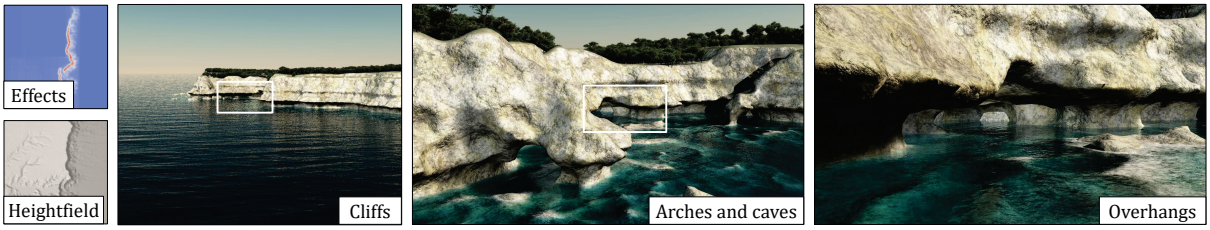


FIGURE 6.11: *An example of sea cliffs produced by our system. Starting from an input synthetic 2D heightfield and geology with horizontal strata, we incrementally applied 3 steps of sea erosion. Sea action was limited to an 8-meter range either side of average sea level, leading to strong overhangs 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.*

as sea elevation range, shown in Figure 6.10). We chose to define e as 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 radius of spheres is proportional to the erosion energy computed as $e(\rho(\mathbf{p}_k), \sigma(\mathbf{p}_k))$. Note that we discard samples with energy below a user-defined threshold, since the associated primitives would have negligible influence.

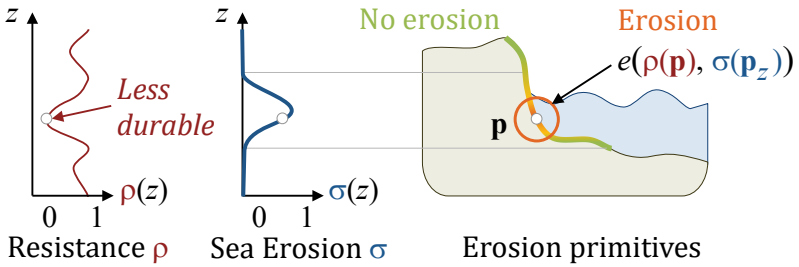


FIGURE 6.12: *Sea erosion impact e combines geology resistance ρ, sea erosion stress σ and accessibility (not illustrated). Spheres are seeded over the surface with a radius derived from $e(\rho(\mathbf{p}), \sigma(\mathbf{p}_z))$.*

Sea cliffs and arches are formed by the erosive action of the sea on coastal geology (refer to Figure 6.11). Sample points $\mathbf{p}_k$ are generated on the initial terrain around sea level (Figure 6.12). 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 are the result of the erosive action of strong rivers in narrow confines, which often creates scenic overhangs. One option for computing water impact is to run a fluid simulation on

the 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 6.13 shows a comparison between an original 2D terrain and the result of the amplification process.

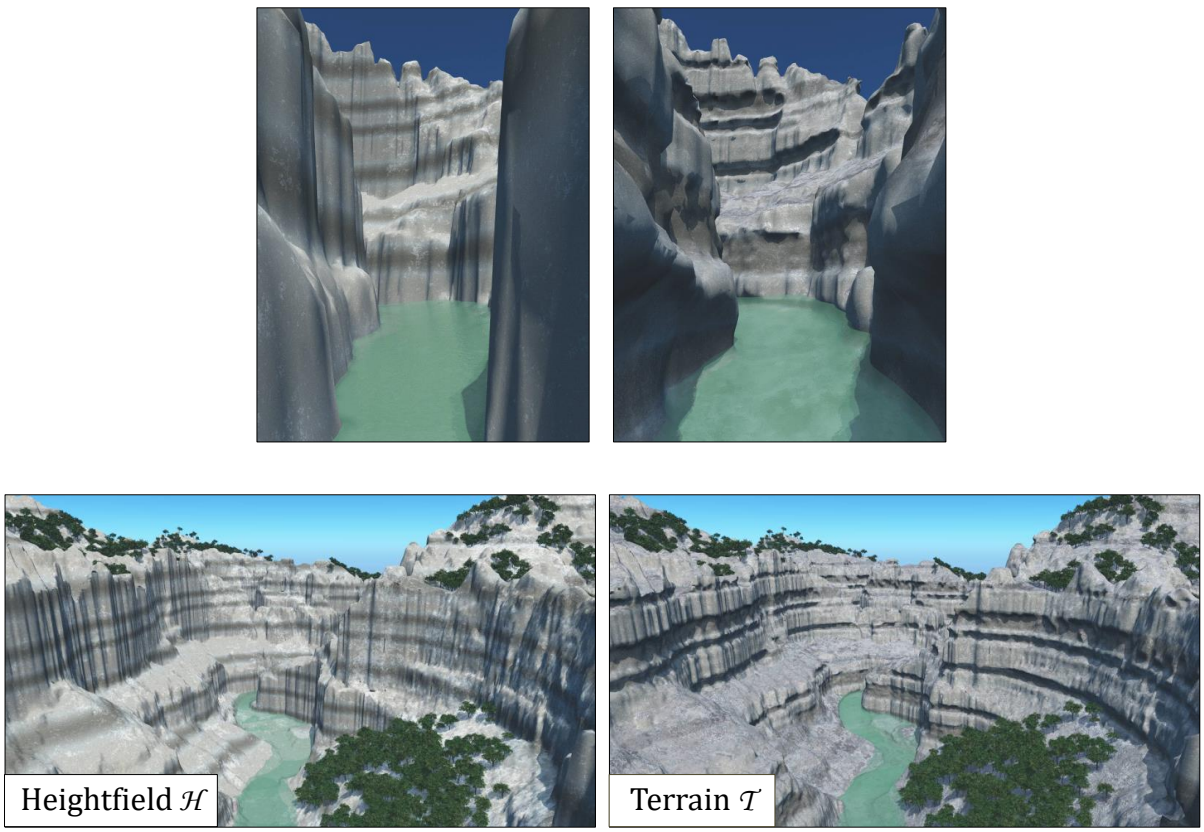


FIGURE 6.13: *A comparison between raw data (with a resolution of* $1\mathrm{m}$ *per pixel) of The Mystery Canyon in the Zion National Park, Utah, and the outcome of the amplification process. Volumetric features occupy* 20% *of the terrain's surface area.*

6.5.2 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.

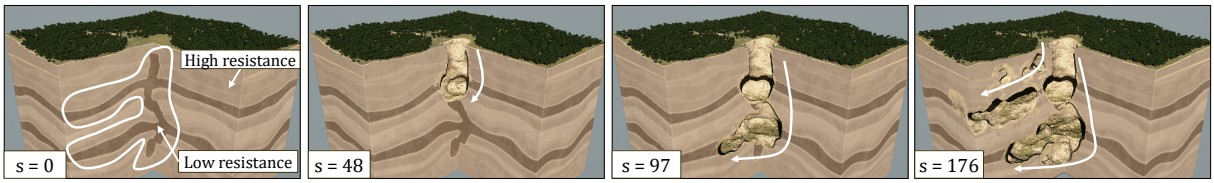


FIGURE 6.14: *Four steps in the generation of a cave system using a modified Invasion-Percolation algorithm. Starting from three sinks in the 2D heightfield, invasion percolation progressively carves the subsurface of the terrain by following the least resistant layers of bedrock. Poisson Sphere sampling allows the creation of multiple tunnels.*

We propose an original method for generating karsts, taking inspiration from the *invasion percolation* simulation (Wilkinson *et al.* 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 $\mathcal{Q}$ 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.

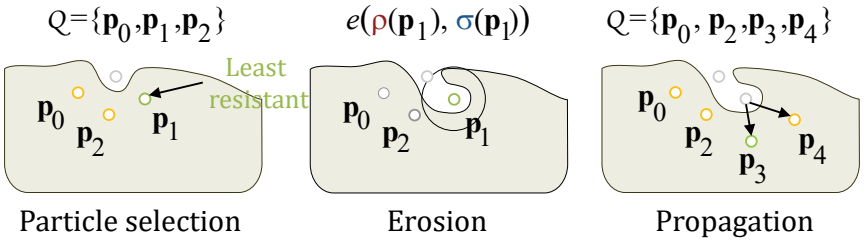


FIGURE 6.15: *Overview of the modified invasion percolation algorithm: after selecting the candidate point with least resistance ρ, the terrain is carved by generating with a sphere, and new candidate points are added to the queue.*

First, the queue of candidate points $\mathcal{Q}$ is initialized with the sinkholes of the input terrain. 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 generated landforms.

While the queue $\mathcal{Q}$ has candidate points whose resistance is below a user-defined threshold, we perform the following steps (Figure 6.15): 1) Find the point $\mathbf{p}_k$ in $\mathcal{Q}$ with the least resistance $\rho(\mathbf{p}_k)$ and remove it from $\mathcal{Q}$; 2) Locally carve the bedrock with spheres; 3) Propagate percolation by finding new points in the lower hemisphere at $\mathbf{p}_k$ and add them to $\mathcal{Q}$. These steps are repeated until $\mathcal{Q}$ is empty.

FIGURE 6.16: *This example showcases a complex topography with sinkholes, tunnels and caves formed by our invasion percolation algorithm. The portion of terrain extends over* $5.2 \times 5.2\text{km}$ *and the underground network of tunnels covers* 2% *of the surface.* $165\,773$ *spheres were generated to create the caves.*

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 radius of the primitive according to the erosion effect, taking into account the stress and rock resistance $e(\rho, \sigma)$ (see Section 6.5.1). 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 due to gravity. New samples are added to the queue $\mathcal{Q}$ 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 6.14 illustrates this phenomenon: we used a set of parallel horizontal strata with some turbulence to produce layered and connected tunnel structures. Figure 6.16 shows an example where sinks were computed automatically on the plateau, leading to a complex set of tunnels emerging on the cliff. While the generated networks conform to the geology of the terrain, important phenomena such as fractures and inception horizons are currently neglected. Furthermore, user-control is only indirect through the use of the geology tree (Section 6.3) and the placement of sink points. Chapter 7 shows how to solve these limitations by generating karstic networks through an anisotropic shortest path method, with a cost function built as a sum of multiple terms, each taking into account a different geological parameter.

6.5.3 Hoodoos and goblins

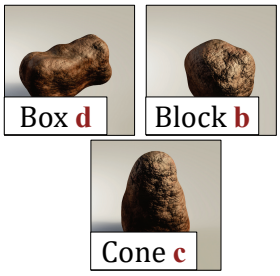
FIGURE 6.17: *Examples of hoodoos generated with different symbols and production rules.*

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 *et al.* 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. Figure 6.17 shows different types of hoodoos produced by grammar rules.

FIGURE 6.18: *A more complex landscape constructed with multiple hoodoo blocks.*

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



Terminal Symbols

FIGURE 6.19: *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.*

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 6.19). 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 6.18 shows a more complex terrain composed of multiple Hoodoo blocks.

6.6 Efficient polygonization

Terrains are defined as implicit surfaces generated by evaluating a construction tree $\mathcal{T}$. Although implicit surface visualization can be achieved both directly using ray tracing, typically with interval arithmetic (Mitchell 1990) or Lipschitz (Kalra *et al.* 1989; Hart 1996) techniques, and indirectly by first extracting a mesh (Wyvill *et al.* 1986; Lorensen *et al.* 1987), 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 elevation 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.

In our case, construction trees consist of thousands of complex skeletal primitives (Table 6.1). 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.

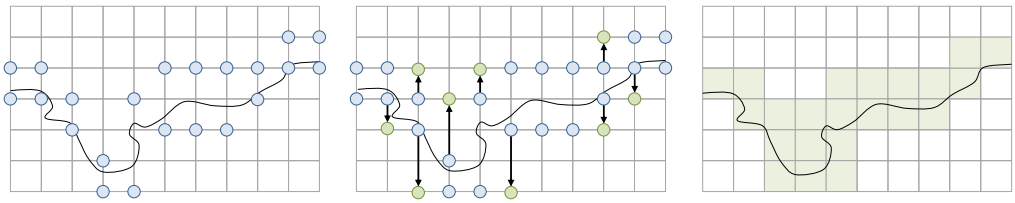


FIGURE 6.20: *A side-view summary of our polygonization algorithm: (1) We compute altitude bounds $\bar{z}$ for each grid vertex by querying the construction tree; (2) these bounds are dilated to ensure continuity in the output mesh, leading to extended bounds $\tilde{z}$, thereby (3) defining a minimal zone for cube traversal.*

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 *et al.* 2003). Our goal is to extract a C^0 surface from the 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 volumetric 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 6.1 indicates for more extensive scenes (such as in Figures 6.11 and 6.16) this proportion tends to be small.

The original Marching Cubes algorithm is a continuation method (Wyvill *et al.* 1986) that 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 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 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 6.20):

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 $\mathbb{R}^2 \to \mathbb{Z}^2$ 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: $\bar{z}_{ij} = \cup(\bar{z}_1, \bar{z}_2)$.

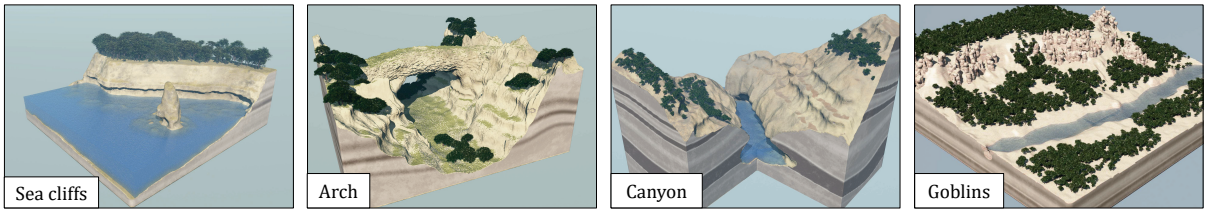


FIGURE 6.21: *Landforms generated using our implicit model on small terrains* ($500 \times 500\,\mathrm{m}^2$)*: a coastal cliff with three sea erosion steps applied (left), a large arch forming a bridge (center-left), a canyon where river erosion has sculpted deep overhangs (center-right) and goblins placed along the banks of a river (right).*

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 6.1.

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 2D 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 6.1 reports statistics regarding visualization. Particularly, it shows the considerable reduction in the computationally demanding field function calls ($\#\mathcal{C}$ vs. $\#\mathcal{C}_0$) and consequent acceleration by up to a factor of 12 (t vs. t_0).

The optimized version benefits 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 6.24), where the ratio a is atypically high and the speedup is only 1.8. Thus, the improved version is most efficient in the context of realistic terrains with localized volumetric features.

6.7 Results and discussion

We implemented our system in C++. Experiments were performed on a desktop computer equipped with Intel® Core $i7$, clocked at 4 GHz with 16GB of RAM, and an NVidia GTX 970 graphics card. The output was streamed into Vue Xstream® to produce photorealistic landscapes with local volumetric features (Figures 6.1, 6.11, 6.13, 6.16, 6.22, 6.23, 6.24). Figures 6.11 and 6.8 show coastal cliffs procedurally eroded by sea, as well as complex features such as arches and caves created by interactive editing. The code for generating the results is available at:

`https://github.com/aparis69/Implicit-Volumetric-Terrains`

Figures 6.16 and 6.14 depict procedural invasion-percolation simulations leading to the evolution of caves and tunnels deep below the surface. Figure 6.18 and 6.19 show hoodoos created with an open shape

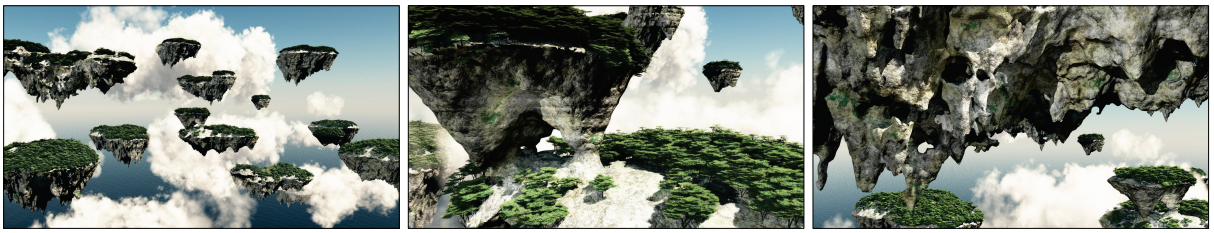


FIGURE 6.22: *The floating islands were created by combining implicitized heightfields; an erosion operator was then added to the construction tree to produce a precise stratification, and we finally carved some tunnels and caves into the bedrock of one island by applying the Invasion-Percolation algorithm. Individual islands are between* 300*m and* 600*m wide and were placed manually in the scene, for a total of* 6*mb in memory.*

FIGURE 6.23: *A comparison between real (left) and synthetic karsts (right). From an initial* 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.*

grammar based on the geology $\mathcal{G}$. Finally, Figure 6.22 demonstrates the capability of our framework in creating and authoring fantastical terrains.

6.7.1 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 results match corresponding effects in nature and so we rely on visual inspection.

Figure 6.23 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 6.24 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.

FIGURE 6.24: *Comparison between a real (left) and synthetic cave (right) in Benagil, Portugal. The terrain was made by an experienced user in less than* 10 *minutes using skeletal brushes (spheres and curves) as sculpting tools.*

6.7.2 Control

Scene	Size	a	$\#\mathcal{N}$	$\mathcal{T}_G$	$\#\mathcal{E}$	Memory		Meshing				
						Ours	Arches	Grid	t	t_0	$\#\mathcal{C}$	$\#\mathcal{C}_0$
Sea (6.11)	6.0^2	0.01	51k	5.5	156	3.0	300	$2000^2 \times 33$	13.3	91.5	9	210
Karst (6.16)	5.2^2	0.02	166k	6.2	43	9.9	140	$1520^2 \times 447$	14.2	188.4	14	1000
Canyon (6.13)	1.1^2	0.20	137k	6.8	0	9.3	110	$1000^2 \times 546$	32.6	159.0	30	569
Hoodoo (6.18)	0.35^2	0.05	56k	2.1	0	5.5	10.2	$650^2 \times 245$	2.9	15.8	4	111
Benagil (6.24)	0.4^2	0.35	13k	-	551	1.3	2.1	$550^2 \times 87$	2.7	3.9	8	20

TABLE 6.1: *Statistics for different amplified terrains: size* $[\mathrm{km}^2]$*, percentage of* 2D *to* 3D *surface area* a*, number of nodes in the construction tree* $\#\mathcal{N}$ *in thousands, generation time* $\mathcal{T}_G$ $[s]$*, 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. Peytavie* et al. *2009b, meshing grid resolution, optimized and standard polygonization time* t $[s]$ *and* t_0 $[s]$*, number of calls to* f *with our optimized algorithm* $\#\mathcal{C}$ *and with the standard algorithm* $\#\mathcal{C}_0$ *(in millions). Benagil was entirely authored by an experienced user using skeletal brushes in less than* 10 *minutes and therefore has no generation time.*

In this work we tried to provide 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 6.1 reports the number of editing clicks $\#\mathcal{E}$ required to produce the different figures: Figures 6.11 and 6.16 were edited in less than five minutes and after multiple procedural erosion steps (Section 6.5). The user then placed spheres to better sculpt the arches and the caves. Figures 6.13 and 6.18 were fully procedural. Figure 6.24 was entirely authored by an experienced user who interactively hollowed out the cave and sculpted the arches with spheres and extrusion curves to match the reference picture; the scene was completed in approximately 10 minutes. A key benefit of our framework is that the implicit terrain model offers a single consistent global scene structure. 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 parts of the terrain.

6.7.3 Performance

Table 6.1 reports the following statistics for the landscapes portrayed in our results: the extent of the input terrain (in $[\mathrm{km}^2]$), 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 model (Peytavie *et al.* 2009b).

Speed The amplification methods generate the construction tree representing complex landforms at a precision of $\approx 1\mathrm{m}$ in less than 7 seconds for terrains that extend over $5\mathrm{km}^2$. 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 The hierarchical implicit construction tree is space efficient in modeling volumetric landforms. One important aspect of our approach is that primitives are only located where required. Thus, most terrains have a low occupancy ratio a compared to their extent. Exceptions are the Zion Canyon (Figure 6.13), which exhibits extensive overhangs resulting from hydraulic erosion, and the Benagil Cave (Figure 6.24), which is a small scene dominated by a sea cave. The implicit representation 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 6.1 we developed an instancing system that effectively halves the memory cost by avoiding replication of primitives.

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 elevation terrains with a wide range of volumetric landforms. This extensibility is depicted in Figure 6.21, which shows the outcomes of a variety of processes. As we shall see in Chapter 7 and 8, the implicit model can be extended with new primitives and operators for representing mesoscale and microscale volumetric features. We also believe that the model is suited to physical simulation, but testing this is left as future work.

6.7.4 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.* 2009b), which rely on voxels or material stacks. Table 6.1 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 volumetric primitives where needed, and resort to more memory efficient implicitized heightfield representations elsewhere.

6.8 Conclusion

We have introduced a novel method for augmenting heightfields with landforms, such as sea cliffs, canyons with overhangs, network of caves and tunnels, hoodoos and goblins, and even floating islands, which are essential scenic elements in synthetic environments. Our compact hierarchical primitive-based implicit representation captures 3D features over large terrains up to $5\,\mathrm{km}^2$ in extent with a memory footprint of at most a few megabytes. 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.

One limitation of our system is user control. While sculpting precise landforms using skeletal brushes is incredibly efficient for the user, modeling large-scale *structured* landforms such as cave networks using invasion percolation can be rather difficult to control. In Chapter 7, we investigate the generation of realistic karstic networks with an emphasis on control and influence of geological settings, thus solving some limitations of the method presented in this chapter.

In this chapter, we investigated the generation of *smooth* volumetric features. Detailed mesoscale and microscale scale features, such as cracks in granite or thin seams of limestone, currently require seeding thousands of sphere primitives using dedicated algorithms, with a concomitant increase in both memory and computation. Chapter 8 introduces a new method for generating such detailed structures in volumetric terrains by introducing new implicit primitives and operators suited for this task.

Chapter 7

Synthesizing geologically-coherent karstic networks

Contents

7.1 Introduction

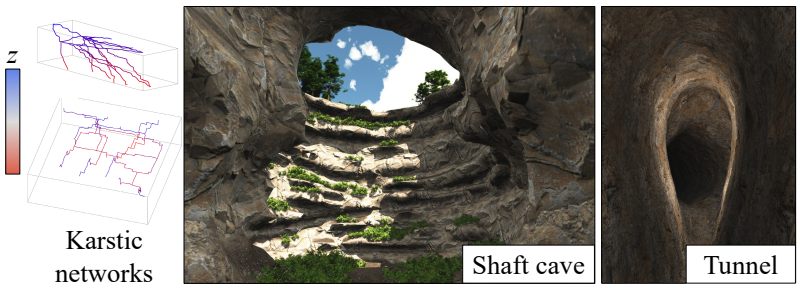


FIGURE 7.1: *Given an input relief, geological characteristics, and user-defined key points, our method generates complex karstic networks and the corresponding detailed geometric model.*

Karstic systems are underground networks composed of conduits and caves that have grown by the dissolution of the host bedrock, generally limestone. They are characterized by complex underground networks made of a variety of tunnels and breakout chambers with stalagmites and stalactites. Although karstic landscapes cover around 12% of the planet's continental surface (Hartmann *et al.* 2014), modeling cave networks has not received much attention from the Computer Graphics community. Their volumetric nature, combined with the winding paths of the network and complex geometry of the conduits make it a challenging problem. Recent volumetric modeling techniques (Peytavie *et al.* 2009b; Becher *et al.* 2019) are not suited for large-scale karstic networks consistent with geological information such as inception features or permeability.

The difficulties stem from the fact that karstic networks are formed by multiple, interconnected geological processes (rock fracturing, percolation) operating at different time scales (from a few years to hundreds of thousands of years) and resulting from various geological settings (fracture distributions, inception features) and hydrogeological time-varying conditions. Karstic networks can be interpreted at multiple spatial scales: macroscale refers to the global topology of the network, whereas mesoscale refers to the shape of the conduits.

We propose a geologically-based framework for modeling karstic networks. We focus on user-control and the generation of realistic and detailed conduit shapes. Given an input relief, we automatically compute a three-dimensional geometric graph connecting control points corresponding to sinks (inlets), springs (outlets), and known passages inside the bedrock. The skeleton of the karstic network is constructed by applying a gridless anisotropic shortest path taking into account geological parameters such as the permeability contrasts of the bedrock, fracture orientations, and inception horizons (geological surfaces particularly prone to karstification (Filipponi *et al.* 2009)). Around this skeleton, a detailed volumetric geometry of the conduits is defined as a signed distance function. We extend the implicit modeling framework introduced in Chapter 6 with sweeping primitives and specific warping operators to represent the shape of the tunnels. Our framework provides multiple levels of control: the user may define inlets, outlets, and waypoints as passages to constrain the construction of the karstic network, prescribe the different volumetric geological parameters, and adjust the paths of the tunnels inside the bedrock. As opposed to the Invasion-percolation approach presented in Chapter 6, this work aims at bridging the gap between realism in Geomorphology and effective authoring in Computer Graphics. The main contributions of this chapter are as follows:

1. A geological framework for modeling karstic systems, taking into account known inlets, outlets, and underground passages, as well as parameters such as inception horizons locations, fracture orientations, and bedrock permeability contrasts.

2. An implicit modeling approach with new primitives and operators organized in a construction tree for creating the geometry and details of the caves around the simulated skeletons.

3. An interactive authoring framework providing multiple direct controls to the user for tuning the network as well as the final geometry of the cave.

4. A detailed quantitative comparison of the generated networks with real karstic systems.

We first introduce the geological background to the reader (Section 7.1, Section 7.2) and present an overview of the method (Section 7.3). The following sections explain the solution in details, including shortest path computation (Section 7.4), generalization to a complete network (Section 7.5), and generation of the terrain geometry (Section 7.6). We finally compare results against real karstic networks and discuss control as well as limitations (Section 7.7).

The work presented in this chapter was published in Paris *et al.* 2021 and received the Replicability Stamp.

7.2 Geomorphology background

Karstic systems are underground networks of conduits and caves that have grown by the dissolution of the host rocks, generally limestone. In epigenic systems, the most common and documented ones, meteoric water penetrates the ground through diffuse infiltration or point sources of recharge. During this process, the water progressively carves different tunnel shapes such as canyon, keyhole, or tube passages based on various geological and physical conditions such as permeability, pressure, and water velocity.

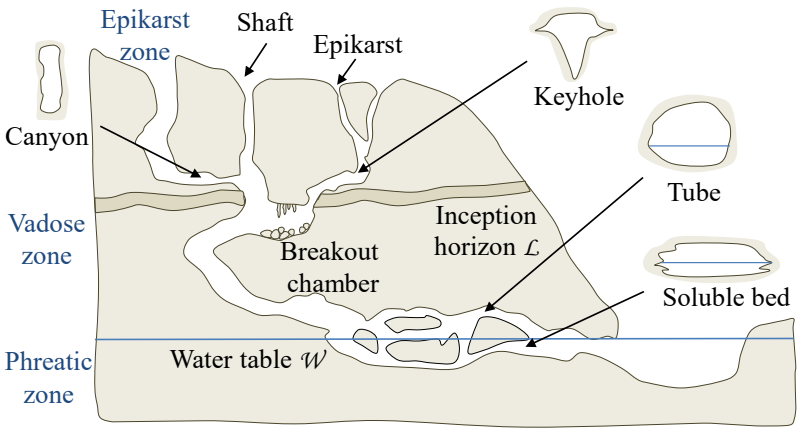


FIGURE 7.2: *Idealized cross-section through a complex underground cave network.*

The water-table $\mathcal{W}$ position delimits the upper unsaturated zone (also called vadose) and the lower saturated zone (also called phreatic). The phreatic zone is generally directly connected to a spring (or the base-level), which constitutes the output of the karstic network. Note that some systems have several connected outputs. In the vadose zone (above the phreatic level), the conduits develop preferentially vertically along fractures, with canyon passages linking them along inception horizons (Figure 7.2). When approaching the phreatic zone, the development becomes progressively more horizontal-dominant. The conduits develop preferentially at or below the water-table in case of long-time steady-state base-level (Jouves *et al.* 2017). If the water-table fluctuates highly with seasons, the networks develop more in maze patterns. As the base-level can change over geological times, today networks can show several levels of horizontal-dominant drains that witness the past water-table positions.

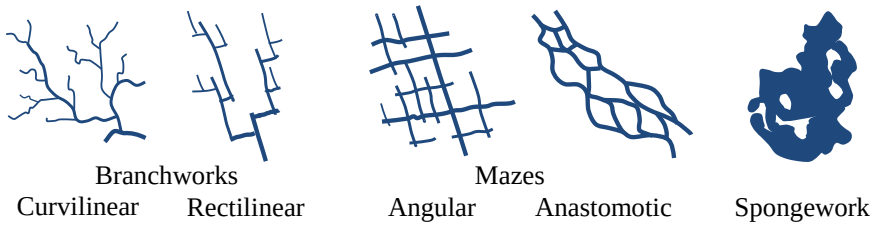


FIGURE 7.3: *Top-view classification of archetypes of cave patterns as proposed by Palmer 2003.*

Prior work on karst simulation. Karsts are an ongoing subject of research in Geomorphology and Hydrology. We distinguish two categories of methods: the ones focusing on the network skeletons (macroscale) and the ones focusing on the conduit shape geometry (mesoscale).

At the macroscale, karstic networks feature a variety of shapes. Palmer 2003 proposed a classification of common patterns found in solutional caves, such as curvilinear and rectilinear branchworks, anastomotic and angular mazes, spongework, or ramiform caves (Figure 7.3). Jouves *et al.* 2017 adapted this classification to account for updated speleological observations, which includes the dimensions and shape of caves. Considering skeletons of karstic networks, we identify the following properties: 1) real systems range between elongated hierarchical branchwork and more anarchic anastomotic mazes; 2) the bedrock fracturing degree is linked to the alignments (rectilinear and curvilinear) of conduits; 3) in the vadose zone (above the phreatic level), conduits have a vertical-dominant development. Several approaches have been proposed to simulate these different types of cave networks based on geostatistical methods (Pardo-Iguzquiza *et al.* 2012; Viseur *et al.* 2014), anisotropic shortest paths (Borghi *et al.* 2012; Collon *et al.* 2012) or more recently percolation clusters (Hendrick *et al.* 2016). Most of these techniques rely on a pre-defined grid, which considerably slows down the simulation when dealing with large systems, and generates stair-step conduits due to aliasing when not aligned with the grid topology.

At the mesoscale, conduits often feature shape variations such as abrupt narrowing or enlargements, which are known to play a fundamental role in fluid flows. With the generic Object-Distance Simulation Method, Henrion *et al.* 2010 proposed to model conduit envelopes by combining an Euclidean distance field around a skeleton with a random threshold. The resulting cross-sections are irregular at the microscale but globally cylindrical at the mesoscale. Rongier *et al.* 2014 described various shapes of karstic conduits and built on previous works to account for perturbations along weakness planes such as fractures, inception horizons, or faults, allowing one to generate a larger diversity of conduit shapes. Overall, few works focus on tunnel geometry because their reconstruction is currently limited by the field acquisition process: the captured data by speleologists is often sparse, incomplete, or incorrect.

7.3 Overview

The geological constraints and key points $\mathcal{P}$ corresponding to sinks (inlets), springs (outlets), and known passages inside the bedrock (waypoints) are at the heart of the proposed procedural approach (Figure 7.4). We focus on the generation of a geometric graph connecting a set of key points $\mathcal{P}$ using a non-Euclidean metric.

The user first provides an initial elevation terrain $\mathcal{H}$ and specifies the geological parameters such as the active water-table $\mathcal{W}$, inception horizons $\mathcal{L}$, permeability π, and preferred conduit orientations which are linked to fracturation. All these parameters may be interactively edited or chosen among a variety of template presets. Then, the user may prescribe key points $\mathcal{P}$ and label geometric nodes: points may be sinks (inlets), springs (outlets), or known passages inside the bedrock, and influence the karstic structure

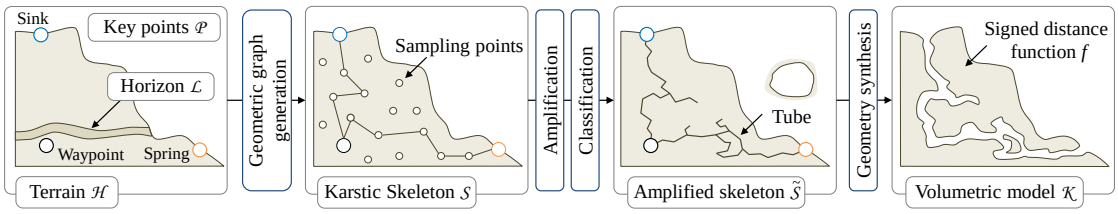


FIGURE 7.4: *Given an input terrain $\mathcal{H}$, geological parameters such as inception horizons $\mathcal{L}$ and control key points $\mathcal{P}$, we synthesize the skeleton of the karstic system $\tilde{\mathcal{S}}$ using an anisotropic shortest path operating on an off-lattice geometric graph in $\mathbb{R}^3$ combined with a γ–skeleton geometric graph construction. We then convert the skeleton in a karstic volumetric terrain model $\mathcal{K}$ defined as a procedural signed distance function obtained by combining implicit primitives with blending operators.*

generation. The user may also force points to be linked together, thus creating a path that will guide the karstic structure.

From these inputs, the network generation proceeds in three steps (Figure 7.4). Starting from key points and geological constraints, we first generate an off-lattice geometric graph connecting those key points using an anisotropic shortest path algorithm based on a specific cost function (Section 7.4), taking into account multiple geological parameters such as conduit orientations, horizons and bedrock permeability. This geometric graph is then simplified using a γ-skeleton approach, leading to a large-scale skeleton $\mathcal{S}$ of the karstic network (Section 7.5).

In a second step, tunnel paths are labeled into different categories according to their geometrical and geomorphological parameters, and the network is finally amplified with small tunnels or mesoscale dendritic structures to obtain an augmented skeleton $\widetilde{\mathcal{S}}$ representing the final network (Section 7.5.2 and Section 7.5.3).

The third step consists of synthesizing the volumetric model $\mathcal{K}$ (Section 7.6). The mesoscale geometry $\mathcal{K}$ of the karstic system is generated from $\widetilde{\mathcal{S}}$ by constructing a hierarchical primitive-based signed distance function $f : \mathbb{R}^3 \to \mathbb{R}$ representing a signed distance bound to the surface. This approach extends the construction tree presented in Chapter 6 with sweep primitives to synthesize the variety of conduit shapes, and creates microscale details and irregularities of the bedrock using noise. The final carved terrain model is directly obtained as the difference between the terrain $\mathcal{H}$ and the volumetric model of the karstic system $\mathcal{K}$.

7.4 Tunnel path computation

In this section we address the construction of a path ρ connecting two key points $\mathbf{a}$ and $\mathbf{b}$ in the bedrock. The path should minimize the line integral over the path of a cost weighting function $c(\mathbf{p}, \dot{\mathbf{p}})$ representing the characteristics of the bedrock at a given position $\mathbf{p}$ and in a given direction $\dot{\mathbf{p}}$.

Let $\mathcal{C}$ denote the set of all continuous paths inside the bedrock $\mathcal{H} \subset \mathbb{R}^3$ from $\mathbf{a}$ to $\mathbf{b}$ that are piecewise continuously differentiable, *i.e.*, the set of paths $\rho : [0, 1] \to \mathcal{H}$, for which $\rho(0) = \mathbf{a}$ and $\rho(1) = \mathbf{b}$. Let $\chi : \rho \to [0, \infty($ denote the function characterizing the cost of a path $\rho \in \mathcal{C}$:

$$\chi(\rho) = \int_0^1 c(\mathbf{p}(t), \dot{\mathbf{p}}(t))\, \mathrm{d}t$$

The continuous anisotropic shortest path problem consists in finding a path ρ^* that minimizes the functional $\chi(\rho)$:

$$\rho^* = \arg\min_{\rho \in \mathcal{C}} \chi(\rho)$$

The solution is approximated by adaptively sampling the bedrock $\mathcal{H}$ with a set of points $\mathcal{Q}$. We define the path as a set of segments connecting points, which converts the continuous shortest-path problem into a discrete shortest-path problem on a finite geometric graph $\mathcal{G}$ whose edges store the approximation of the line integral of the cost function c. We proceed in two steps. First, the bedrock is adaptively sampled according to the geological characteristics, and samples are connected using a relative neighborhood approach. Then, the cost function is evaluated over each edge of the graph. The discrete anisotropic shortest path is finally directly computed using an A* algorithm.

7.4.1 Sampling

Sampling is a crucial step for approximating the continuous anisotropic shortest path problem. Uniform grid sampling sustains two important limitations. First, it often produces regular axis-aligned patterns and yields unrealistic large-scale networks. Moreover, it does not conform to the geological characteristics of the bedrock unless utilizing a dense sampling, which dramatically increases computations. Increasing the neighborhood distance between samples to introduce more directions and enhance the angle accuracy between path segments (Galin *et al.* 2010) only partially alleviates the problem.

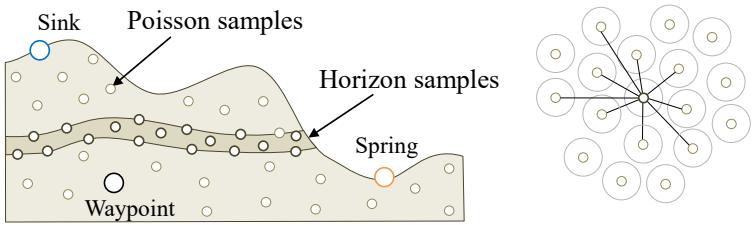


FIGURE 7.5: *The adaptive sampling of $\mathcal{H}$ allows the generated network to precisely follow features such as inception horizons; empty remaining space is filled with a Poisson sphere distribution.*

Instead, we adaptively sample the bedrock $\mathcal{H}$ according to its geological characteristics (Figure 7.5). The set of points $\mathcal{Q}$ is initialized with key points $\mathcal{P}$ representing sinks (inlets), springs (outlets) and interior points in the terrain. Samples are then procedurally-generated for every geological feature, such as horizons, current water table $\mathcal{W}$ and permeability volumes (Section 7.4.2). By using an importance function derived from the prescribed geological features, one can resort to importance sampling (Kahn *et al.* 1953) with a dart-throwing approach as a generic sampling solution. However, this technique can be slow depending on the extents of the domain and the desired sampling density. Instead, we use a more sophisticated approach: each geological feature is sampled independently, and the remaining empty space is finally filled with another distribution.

Horizons and water table are modeled as a function of altitude, which means that geological samples are distributed at these elevations using a Poisson disk distributions (Cook 1986) with a small radius $r = 8\text{m}$ (Figure 7.6, left). Sometimes, precise topological data is available for the horizons, and the chosen representation is a 3D mesh rather than an elevation surface (Collon *et al.* 2012). These can be sampled efficiently using techniques such as constrained Poisson disk sampling (Corsini *et al.* 2012) or Poisson disk sample elimination (Yuksel 2015).

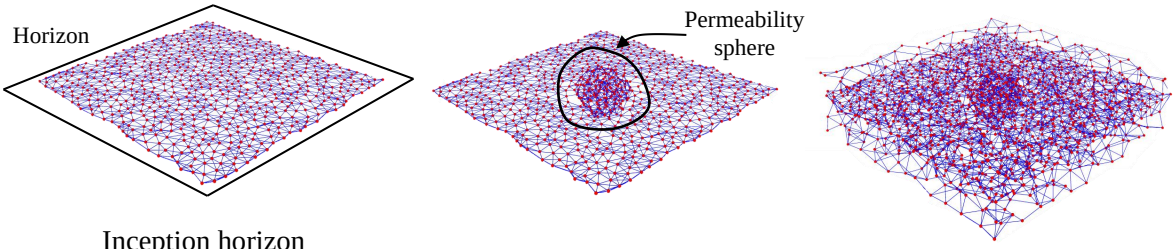


FIGURE 7.6: *Geological features such as inception horizons and permeability volumes are first sampled using a Poisson disk or Poisson sphere distribution process with a small radius (left, center). The rest of the domain is filled with samples from another distribution with a larger radius (right), leading to the complete set of points $\mathcal{Q}$. Connections between samples are shown in blue.*

Permeability volumes are sampled using importance sampling combined with a Poisson sphere criteria (Lagae *et al.* 2006a), using the same radius as for horizons and the water table. The importance sampling algorithm allows to distribute more samples where permeability is high (Figure 7.6, center). Empty space is finally filled with another Poisson sphere distribution (Lagae *et al.* 2006a), but with a larger radius $r = 16\text{m}$ (Figure 7.6, right). Independently of the sampling techniques, geological features should be sampled more densely than empty space. This step ensures that the generated network accords with the structures observed in real karstic systems.

The graph $\mathcal{G}$ is constructed as the nearest neighbor graph connecting n samples (in our experiments, we used $n = 32$) to obtain sufficient angle accuracy between the directions of the edges of the graph (see Figure 7.7). Using a smaller value for n reduces the memory footprint of the graph but can lead to unrealistic or even unfeasible paths in practice.

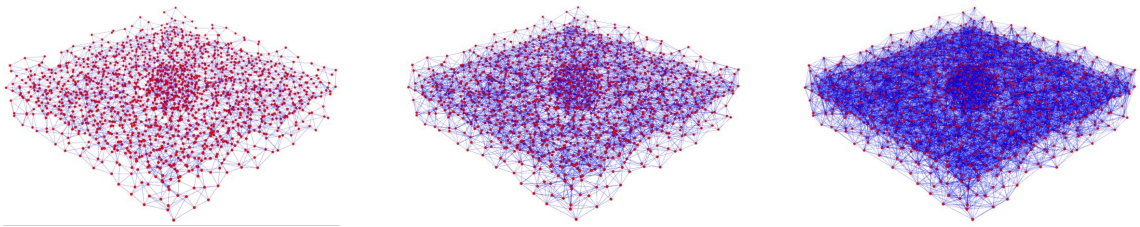


FIGURE 7.7: *Influence of the neighborhood size n over the topology of te graph.*

7.4.2 Geology-based cost functions

The cost function c defined as a weighted sum of different terms, each representing the influence of a geological factor:

$$c(\mathbf{p}) = w_{\mathcal{L}} c_{\mathcal{L}}(\mathbf{p}) + w_{\mathcal{F}} c_{\mathcal{F}}(\mathbf{p}) + w_{\pi} c_{\pi}(\mathbf{p})$$

The user-editable weights (namely $w_{\mathcal{L}}$, $w_{\mathcal{F}}$ and w_{π}) control the influence of the geological characteristics, which include the inception horizons $\mathcal{L}$, fracturing orientations $\mathcal{F}$, and permeability of the bedrock π, whose corresponding cost functions are denoted as $c_{\mathcal{L}}$, $c_{\mathcal{F}}$, and c_{π} respectively. In addition, the user may prescribe specific paths to influence the generation by assigning a small cost to some of the edges of the graph. The following paragraphs detail each of these term and show how it affects the computation of a path.

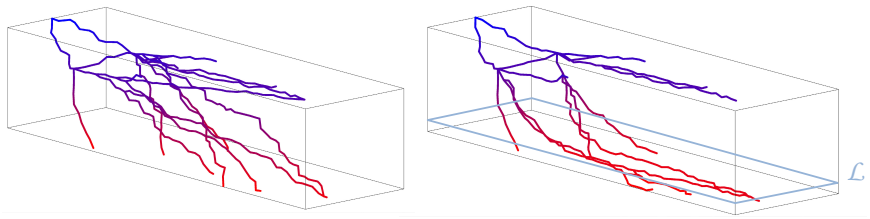


FIGURE 7.8: *Influence of an inception horizon on the karstic network: conduits tend to follow the inception horizon elevation. The blue plane indicates the elevation of the horizon.*

Horizons refer to the bedding surfaces that mark a particular change in the lithology of rocks. They generally influence the karstification process as karstic systems tend to develop along particular *inception horizons* (Filipponi *et al.* 2009). In monoclinal contexts where the underlying geology is organized into multiple, almost parallel strata, they can be modeled as a function of altitude. The corresponding cost function is defined as a function of the vertical distance to the nearest horizon $d(\mathbf{p}, \mathcal{L}) = |\mathbf{p}_z - \mathcal{L}_z|$:

$$c_{\mathcal{L}}(\mathbf{p}) = g(d(\mathbf{p}, \mathcal{L})/r)$$

The smooth step function $g(x) = 3x^2 - 2x^3$, for $x \in [0, 1]$, limits the influence of the horizon beyond a distance r. Figure 7.8 shows the impact of an inception horizon over the generation process.

Fractures play an important part in the development of karstic systems. A fracture is a break of continuity in the bedrock resulting from tectonics. They have different orientations, such as axis-aligned or diagonal distributions (Palmstrom 2001).

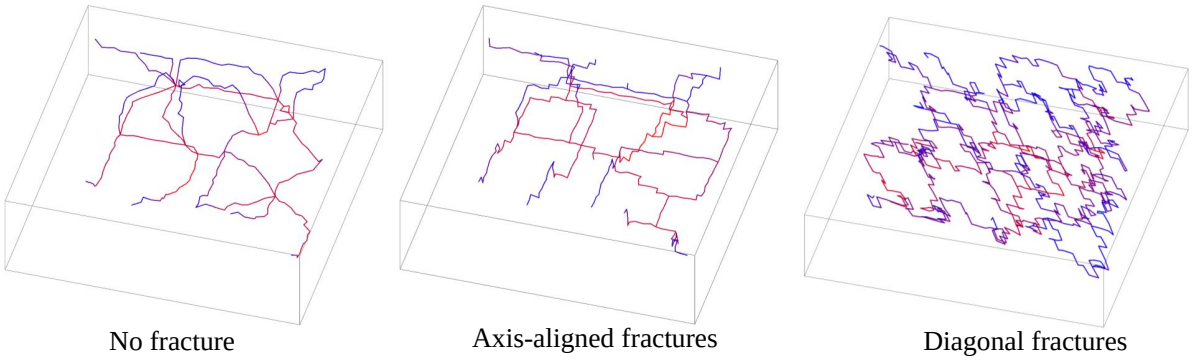


FIGURE 7.9: *Influence of an axis-aligned (center) and diagonal (right) fracture orientation distribution on the karstic network.*

At the mesoscale, karstic conduits tend to follow local fracture plane orientations. At large-scale (a few kilometers), fracture directions remain almost constant, therefore we represent every fracture as a normalized direction $\mathbf{n}_k$ representing the orthogonal vector to the fracture plane, and a weight w_k. Let $\mathcal{F}$ denote the set of fractures in the bedrock, we define the corresponding cost function $c_{\mathcal{F}}$ as:

$$c_{\mathcal{F}}(\mathbf{p}, \dot{\mathbf{p}}) = \sum_{k=0}^{n} w_k \left(1 - (\mathbf{n}_k \cdot \hat{\dot{\mathbf{p}}})^2\right)$$

The term $\hat{\mathbf{p}} = \dot{\mathbf{p}}/\|\dot{\mathbf{p}}\|$ represents the unit direction of the path at vertex $\mathbf{p}$. Edges with a direction parallel to a fracture plane have a smaller cost value than orthogonal ones and are therefore preferred as illustrated in Figure 7.9.

Permeability is an important geological parameter as it expresses the capacity of a rock to let the fluids circulate through its pores. Permeable bedrock lets more water flow, and is thus more prone to chemical dissolution, *i.e.*, karstification. We do not model directly this parameter, but a qualitative parameter expressing a normalized relative resistance to flow passage (the inverse of permeability) using a function $\pi : \mathbb{R}^3 \to [0, 1]$ defined at each point $\mathbf{p}$. The least permeable rocks have thus the highest value 1, and the most permeable rocks have the lowest value 0. The permeability cost is then defined as $c_\pi(\mathbf{p}) = \pi(\mathbf{p})$, with π defined as a construction tree similar to the one described in Chapter 6 for the geology tree. The leaves of the construction tree are skeletal primitives that define the local permeability; internal nodes combine primitives to define the permeability function over the entire domain. Depending on the requirements, different primitives and operators such as strata or faults can be used. Our system implements sphere and strata primitives that provide control to the user. Figure 7.10 illustrates the influence of permeability: the karstic network expands in the most permeable region determined by two user-defined sphere primitives with a high permeability coefficient.

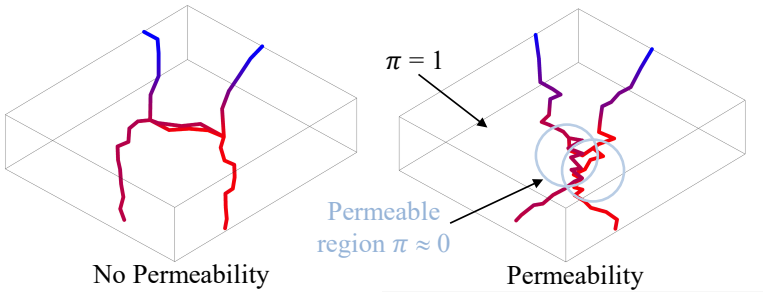


FIGURE 7.10: *Influence of permeability on the karstic network: conduits are generated in the most permeable ($\pi \approx 0$) regions.*

7.5 Network generation

We generate the skeleton of the karstic system in two steps. First, we build a large-scale skeleton $\mathcal{S}$ by computing the 3D γ-skeleton of the key points $\mathcal{P}$. Then, we amplify the network by applying a stochastic subdivision and generate ramifications to obtain a mesoscale skeleton $\widetilde{\mathcal{S}}$.

7.5.1 Large-scale network

We address the generation of a geometric skeleton connecting the set of key-points $\mathcal{P}$ using a non-Euclidean metric. The distance metric is directly drawn from the computation of the cost between two points, and defined as $d(\mathbf{a}, \mathbf{b}) = \chi(\rho^*(\mathbf{a}, \mathbf{b}))$. The method takes its inspiration from Galin *et al.* 2011, with the difference that we operate on a volumetric domain and incorporate several types of key points to constrain the graph generation.

We first compute the complete graph formed by the set of paths connecting $\mathcal{P}$ by applying multiple anisotropic shortest path algorithms between all pairs of key points as described in Section 7.4. We then

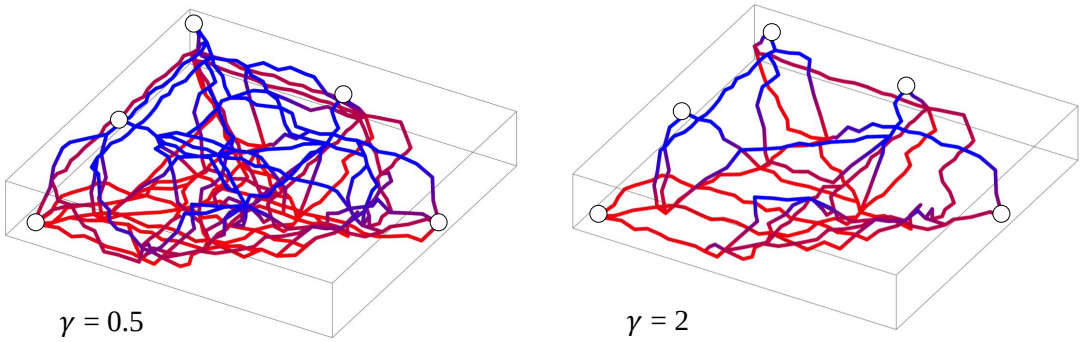


FIGURE 7.11: *Influence of the parameter γ: lower values produce denser networks (key-points are represented with a white circle).*

prune the edges using an empty region criterion to create a proximity graph based on the metric d. Proximity graphs, also referred to as neighborhood graphs, are defined on a finite set of vertices such that there exists an edge between any two vertices if they satisfy proximity conditions in the context of other connections in the graph.
To account for geological features and thus obtain a 3D karstic network, we use a gamma skeleton formalism using this non-Euclidean metric d. Let $\mathbf{a}$ and $\mathbf{b}$ two key points in $\mathcal{P}$, and $\gamma \in]0, +\infty($, the parameterized neighborhood region of $(\mathbf{a}, \mathbf{b})$ is defined as:

$$\Omega_\gamma(\mathbf{a}, \mathbf{b}) = \{\mathbf{p} \in \Omega,\ d(\mathbf{a}, \mathbf{b})^\gamma < d(\mathbf{a}, \mathbf{p})^\gamma + d(\mathbf{p}, \mathbf{b})^\gamma\}$$

The neighborhood graph of $\mathcal{P}$ is created from this definition of the region $\Omega_\gamma(\mathbf{a}, \mathbf{b})$ which is associated to candidate paths: $(\mathbf{a}, \mathbf{b})$ forms a path in the graph if and only if $\Omega_\gamma(\mathbf{a}, \mathbf{b}) \cap \mathcal{P} = \varnothing$. More precisely, a path connecting two key points $\mathbf{a}$ and $\mathbf{b}$ of distance $d(\mathbf{a}, \mathbf{b})$ is kept if and only if there is no path connecting $\mathbf{a}$ to $\mathbf{b}$ passing through another key point $\mathbf{k}$ having $d(\mathbf{a}, \mathbf{k})^\gamma + d(\mathbf{k}, \mathbf{b})^\gamma < d(\mathbf{a}, \mathbf{b})^\gamma$.

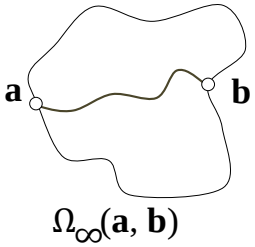


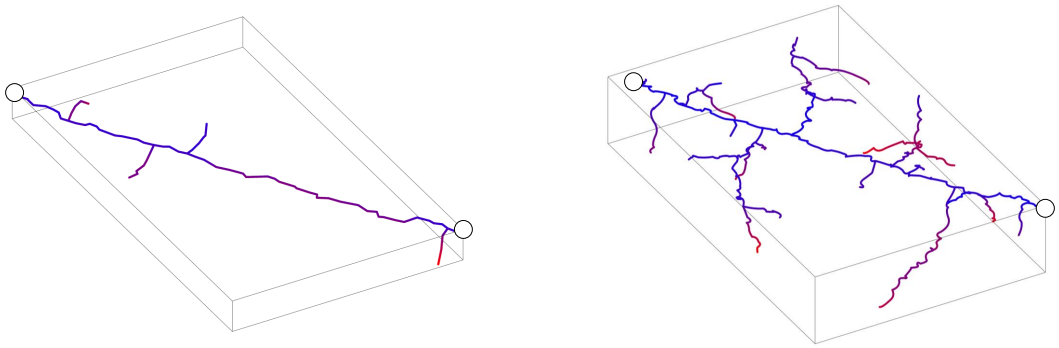
FIGURE 7.12: *Geometric representation of Ω_∞.*

When the value of γ decreases, the neighborhood region shrinks and fewer edges are pruned by the process, which leads to denser skeletons. The parameter γ creates a variety of skeletons connecting the points in $\mathcal{P}$ and provides a simple and global control to the user over the density of the karstic network (Figure 7.11). Particularly, note that as $\gamma \to \infty$, the generated graph is equivalent to the relative neighbor graph, and using $\gamma = 2$ creates the same graph as the Gabriel graph created with a non-Euclidean metric. Using $\gamma = 0$ would lead to the complete graph between all key points.

7.5.2 Network amplification

The amplification step generates a refined mesoscale skeleton featuring dendritic conduits and more tortuous paths (Figure 7.13). We improve the large-scale skeleton into a mesoscale skeleton by adding secondary branches and tributaries. We then refine the trajectories of the paths by adding stochastic displacement.

The ramification process starts by distributing new key points randomly in the bounding box of $\mathcal{S}$, inside the bedrock $\mathcal{H}$, and connecting them to $\mathcal{S}$ using the procedure described in Section 7.4. Points may be of two different types: interior or dead-end nodes. Interior nodes are not constrained regarding



Large scale skeleton $\mathcal{S}$ Mesoscale skeleton $\widetilde{\mathcal{S}}$

FIGURE 7.13: *Procedural amplification of a large-scale skeleton $\mathcal{S}$ with new tunnels into a refined structure $\widetilde{\mathcal{S}}$.*

their degree in the graph (they can be linked to an infinite number of paths), so they tend to increase the overall complexity of the network, possibly leading to mazes. On the other end, dead-end points are limited to a degree of 1 (only one path can be linked to these nodes) and tend to produce branchwork structures (Figure 7.13). The number of new points distributed in the amplification process is controlled by the user.

As real karstic systems exhibit tortuous trajectories, we finally refine edges using a stochastic midpoint displacement parameterized by a tortuosity factor θ. In Geomorphology, tortuosity (also referred to as sinuosity) has been proposed to characterize karsts (Collon *et al.* 2017), even if it is heavily dependent on the data acquisition strategy. Consequently, we model tortuosity as a qualitative parameter expressing the maximum displacement factor for the tunnels, ranging from a few centimeters ($\theta = 0$) to 4 meters ($\theta = 1$).

7.5.3 Classification strategy and parameter computation

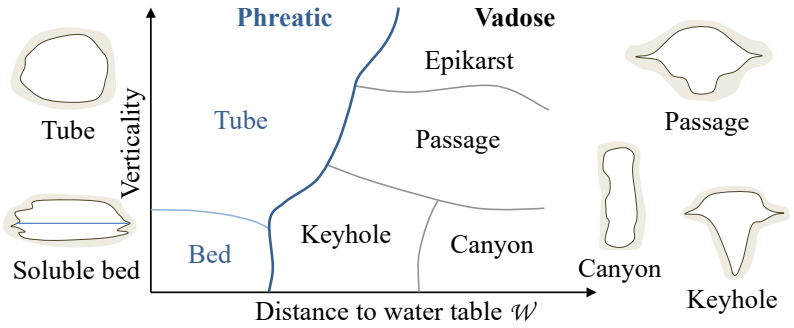


FIGURE 7.14: *Classification of tunnel types according to their verticality and the distance to the water table.*

In order to construct the detailed geometry from the mesoscale skeleton $\widetilde{\mathcal{S}}$ (section 7.6), we compute the type of tunnels for all the graph edges and their corresponding geometric parameters such as tunnel diameters. Edges are labeled as *keyhole*, *passage*, *epikarst*, *tube*, *canyon*, *bed*, or *chamber*, depending on the local geological characteristics. Current knowledge in Geomorphology does not allow the identification of precise criteria for the placement of tunnel types. Therefore, we propose a simple labeling

strategy (Figure 7.14) correlating the type to the distance to the current water table $\mathcal{W}$ and the verticality of the trajectory. A notable advantage of this approach is that it can be adapted to account for a more accurate classification as knowledge in Geomorphology evolves.

Vertical tunnels, such as shafts or pits, expose roughly circular shapes (tubular vertical shafts), whereas horizontal tunnels reveal a variety of types such as canyon-shaped tunnels, keyhole-shaped (in vadose zones), or soluble bed conduits (in phreatic zones). Epikarst tunnels are narrow vertical conduits that start from the surface to reach the vadose zone before morphing into a different configuration. Breakout chambers are large caves in the karst. They are created when a tunnel traverses a ghost rock zone – a region where permeability is particularly high ($c_\pi(\mathbf{p}) \approx 0$) and where large collapses are more likely to happen. Figure 7.15 shows an example of classification applied to a generated network.

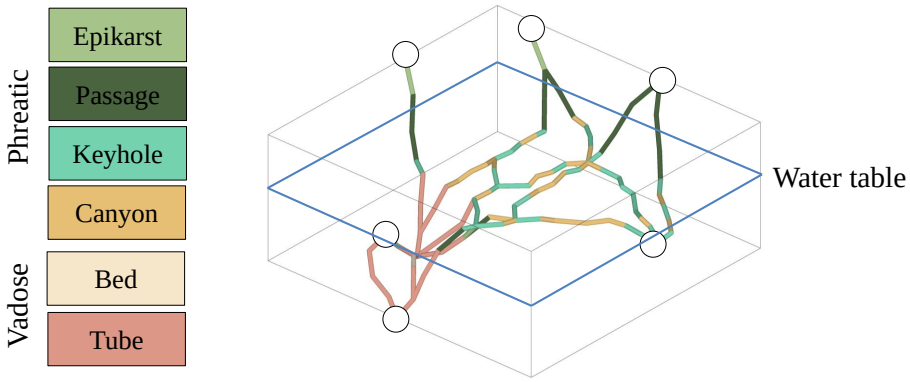


FIGURE 7.15: *Classification of tunnel types on a generated network.*

Concerning the equivalent radius of the karstic conduits, speleological observations do not allow to identify any specific rule of size distribution. A recent statistical study performed on 49 real networks was unable to show any hierarchy in size distribution, or relation with the relative position of the springs (Frantz *et al.* 2021). With a variographic analysis, Frantz *et al.* 2021 identified a spatial correlation along the networks, with empirical semi-variogram ranges ranging between 50 to 200m. This means that two points along a conduit separated by a distance superior to that range have no correlation.

Epikarst tunnels typically have a fixed radius of 50 cm as they start from the surface and are usually small. For other types, the radius is computed randomly in a $0.5 - 4.0\,\mathrm{m}$ range. These values can be adjusted by the user, or computed by more realistic heuristics as knowledge in Geomorphology evolves.

7.6 Implicit cave modeling

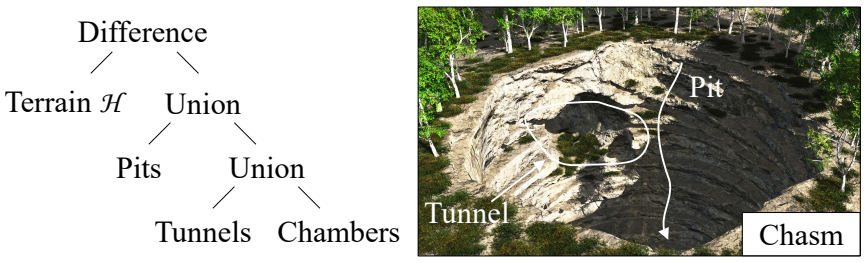


FIGURE 7.16: *Synthetic representation of the construction tree of a chasm carved into an elevation terrain.*

The generation of the mesoscale geometry addresses two complementary challenges: creating the walls of conduits, chambers, and pits that accord to the cross-sections observed in geology, and generating a sufficiently compact volumetric model allowing to represent karstic networks of a few kilometers with a high level of detail.

Building on a hierarchical construction tree (Figure 7.16, Chapter 5.3), we define a signed distance function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, taking positive values outside of the bedrock and negative values inside. We ensure that the implicit surface is defined by a 1-Lipschitz lower signed distance bound to the surface. We rely on the base terrain primitive presented in Chapter 6 Section 6.4.1, generate the geometry of the karstic conduits (Section 7.6.1), and finally add breakout chambers, large pits, and small-scale geometric details such as rock irregularities and stalactites (Section 7.6.2).

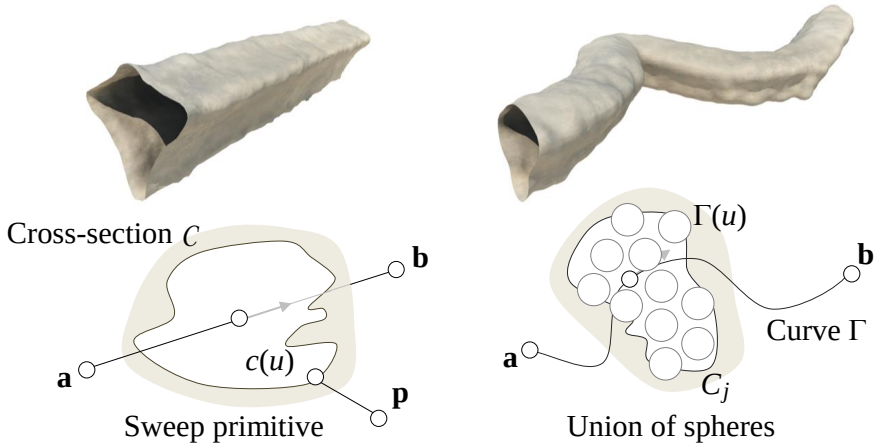


FIGURE 7.17: *Two different strategies are used to model volumetric conduits, based on sinuosity: sweep primitives or unions of spheres.*

7.6.1 Mesoscale geometry of tunnels

Tunnels identified and referenced in geology have cross-sections that may have irregular and asymmetric silhouettes with folds, thus there is a need for a general sweep primitive along a curvilinear trajectory. Accelerated primitives introduced in Crespin *et al.* 1996 are limited to star-shaped cross-sections. The swept volumes described in Sellán *et al.* 2021 could generate conduits, however the tunnels may result from the interpolation of different cross-sections. Additionally, our approach creates a procedurally-defined signed distance field, whereas Sellán *et al.* 2021 uses a discrete representation using a sparse voxel grid.

However, defining a continuous signed distance function for a sweep object with cross-sections swept along a curve Γ is not possible in the general case, in particular when the curve has a high curvature because it creates $\mathcal{C}^1$ or even $\mathcal{C}^0$ discontinuities in the distance function. To solve this issue, we propose a two-fold strategy for modeling volumetric conduits based on the sinuosity of the conduit. Sinuosity σ is defined as the ratio between the length of the curve l and the distance between the first and last point of the section:

$$\sigma = \frac{l}{||\mathbf{p}(0) - \mathbf{p}(1)||}$$

Depending on the value of σ, we model tunnels either as *sweep primitives*, or as *unions of spheres* sampled along the trajectory. This scheme guarantees that the resulting signed distance function is continuous

FIGURE 7.18: *Interior of a vadose cave with canyon, bed and tube tunnels, and a phreatic tunnel partially overflowed with water.*

and 1-Lipschitz. Sweep primitives are compact in memory but are restricted to straight paths, whereas unions of spheres are adapted to tortuous paths but are more memory consuming.

In practice, a user-defined threshold T determines the path generation (see Figure 7.17). When $\sigma < T$, we approximate the trajectory as a straight sweep primitive. Tortuous path, with $\sigma > T$, are modelled using unions of carefully placed spheres along the trajectories.

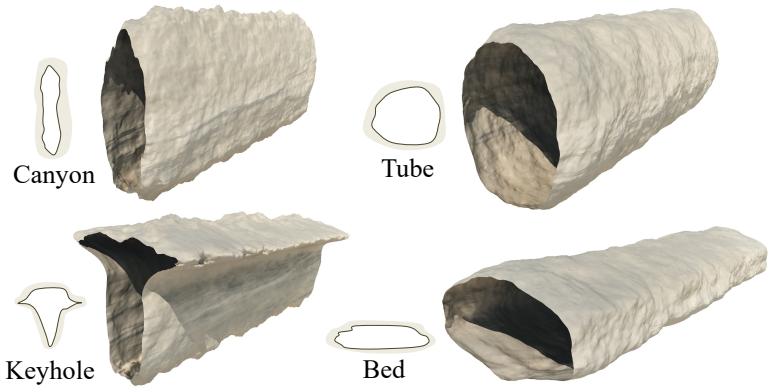


FIGURE 7.19: *Tunnels generated with sweep primitives.*

Straight paths are defined by sweeping a closed piecewise quadratic curve $\mathcal{C}$ representing the cross-section along a line segment $\mathbf{ab}$. This representation is compact in memory, but the evaluation of the distance is computationally intensive. The signed distance function to the cross-section is defined as $f(\mathbf{p},\mathcal{C}) = d(\mathbf{p},\mathcal{C})\,\delta(\mathbf{p},\mathcal{C})$, where $d(\mathbf{p},\mathcal{C})$ denotes the (positive) Euclidean distance to $\mathcal{C}$ and $\delta(\mathbf{p},\mathcal{C})$ is the sign function. The 3D distance is then derived from $f_{\mathcal{C}}(\mathbf{p})$ according to the segment $[\mathbf{a},\mathbf{b}]$ (see Appendix A.7). As the cross-section is extruded on a segment and not a curve, the resulting signed distance function is 1-Lipschitz and continuous.

Tortuous paths are modelled using unions of spheres. First, we compute a Poisson sphere distribution $\mathcal{D} = \{\mathcal{D}_i\}$ inside the bounding box of the curve Γ extended by a radius R. We then generate n interpolating cross-sections $\mathcal{C}_j$ along the curve at regular intervals taking into account the local frame of the curve. Then, we select the Poisson spheres whose minimum distance to the cross sections is lower than $R/2$ (Figure 7.17):

$$\widetilde{\mathcal{D}} = \left\{\mathcal{D}_i \mid \min_{j=0}^{n} d(\mathcal{D}_i,\mathcal{C}_j) < R/2\right\}$$

Using a small radius $R \approx 20\,\mathrm{cm}$ provides a good approximation of the extruded volume, at the expense of memory. Typically, modeling a 10 meters long straight tunnel using a single sweep primitive takes

less than 1kb of memory, whereas the same tunnel modeled with thousands of spheres is $\approx$ 100 times more expensive (see Figure 7.20).

FIGURE 7.20: *Tunnels generated with different amount of spheres. Using a small amount of spheres is more compact in memory but can lead to unwanted bumps and defects in the approximation of the tunnel.*

7.6.2 Volumetric terrain decoration

While tunnels are the main characteristic landforms of karstic networks, large-scale features such as breakout chambers and chasms also have a large impact on the visual representation of the scene and cannot be modeled using extruded profile primitive. Instead, in the spirit of the invasion-percolation method (Chapter 6 Section 6.5), we detect the location of these features and use procedural arrangements of skeletal primitives (Chapter 5 Section 5.4) to generate them. This approach guarantees a 1-Lipschitz function and provides precise control to the user over the location and shape of the features.

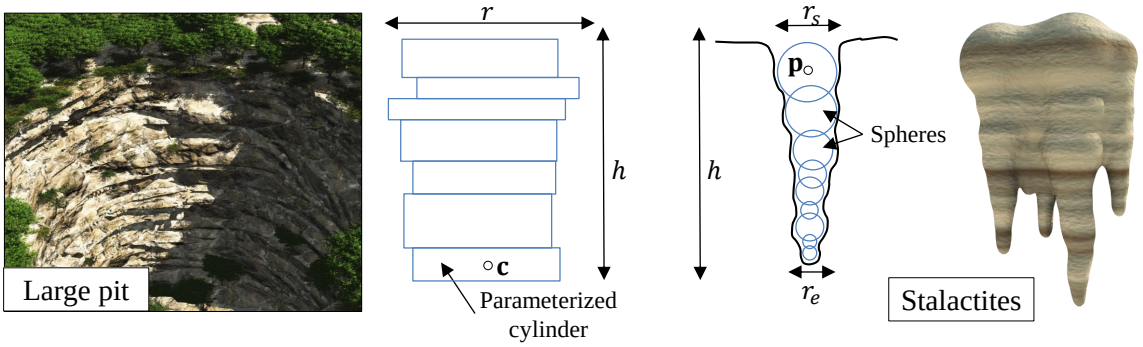


FIGURE 7.21: *We model punctual karstic features as arrangements of skeletal primitives, such as cylinders (for pits and chambers) and spheres (for stalactites).*

Breakout chambers and large pits are placed by the user using specific control points (see Section 7.5) in the generated network. The geometry is modeled using arrangements of parameterized cylinder primitives. Using piles of cylinders allows for reproducing stratification of different lengths efficiently with very few primitives. Starting from an initial position $\mathbf{p}$ in the bedrock, we procedurally generate primitives in contact with varying radii and slightly displaced centers (Figure 7.21, left) until a target vertical size h is reached. The amount of variation of the different parameters is controlled by the user. Optionally, a radial turbulence can be added to provide small-scale details without introducing floating parts in the terrain. Cylinders are combined together in a single subtree using a union operator,

which allows conserving sharp transitions between the different strata (as opposed to the smooth union operator). Finally, the terrain is carved with the pit (or chamber) subtree using a difference operator.

Stalactites and stalagmites are usually located on planar zones in caves. We detect feature locations by sampling the terrain using Poisson sampling, and keep sufficiently planar positions verifying $|\nabla f(\mathbf{p}) \cdot \mathbf{z}| > T$ with T a user-defined threshold. The generation of the primitives follows a similar procedure as chambers and pits, using sphere primitives. Starting from a position $\mathbf{p}$, a sphere of varying radius is generated. An initial and end radii r_s and r_e control the size of the spheres in contact (Figure 7.21, right). The procedure is repeated until the stalactite reaches a target vertical size h. Spheres are combined using a smooth union operator, and finally blended with the terrain.

7.7 Results

We implemented the proposed method in C++. The karstic systems depicted in this chapter were generated on a desktop computer equipped with Intel® Core $i7$, clocked at $4\,\mathrm{GHz}$ with $16\mathrm{GB}$ of RAM, and an Nvidia GTX 1080ti graphics card. The implicit surface representing the final terrain was polygonized (Wyvill *et al.* 1986) and the resulting mesh directly streamed and procedurally textured into Vue Xstream® to render the final images (Figure 7.1, 7.18, 7.16, 7.19). The code for synthesizing the karstic systems is available at:

`https://github.com/aparis69/Karst-Synthesis`

The proposed implementation is capable of generating a variety of karstic networks as well as the detailed geometry of the tunnels. Figure 7.22 shows complex networks generated according to different geological parameters such as inception horizons, orientations and permeability of the bedrock. Figures 7.18 and 7.19 show the capabilities of the implicit modeling approach for synthesizing the detailed geometry of the tunnels, capturing different shapes such as keyhole, canyon and bed tunnels.

7.7.1 Performance

The computation of the anisotropic shortest path combined with the three dimensional gamma skeleton geometric graph construction completes in less than a second, which allows for interactive authoring even for relatively large networks featuring thousands of nodes and dozens of key-points (Table 7.1). The adaptive sampling and generation of the nearest neighbor graph between all sample points $\mathcal{Q}$ is the most computationally intensive step and takes up to a few seconds in the most complex geological configurations.

One significant advantage of the signed distance function construction tree is the reduced memory footprint compared to hybrid (Peytavie *et al.* 2009b) and voxel-based models (Becher *et al.* 2019). The tortuosity factor θ directly relates to the construction tree generation time: trajectories with a high coefficient θ require instantiating a high number of spheres. Still, the generation of the construction tree is completed in a few seconds in the worst case scenario (Table 7.1).

A typical editing session takes up to a few minutes: the user inserts points in the domain, sets the weights of the cost function, adjusts geological parameters (such as fracture orientations and horizon elevations), and triggers the computation of the network. During user interaction, visualization uses a symbolic representation of the caves and tunnels. Those steps are repeated until the user is satisfied

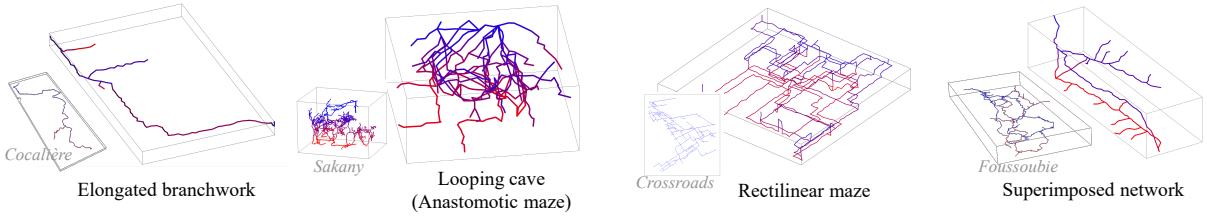


FIGURE 7.22: *A variety of computer-generated karstic networks compared to real networks.*

with the generated network. The detailed geometry of the karstic network is finally computed for final visualization and rendering.

Karstic system	Extent		Topography		Skeleton				Geometry				
	Size	$\mathcal{L}$	$\#\mathcal{Q}$	$\#\mathcal{P}$	$\#\mathcal{K}$	$\mathcal{T}_\mathcal{S}$	θ	γ	$\#\mathcal{N}$	$\#\mathcal{S}_w$	$\#\mathcal{S}$	$\mathcal{M}$	$\mathcal{T}_\mathcal{G}$
Branchwork	$0.4 \times 0.6 \times 0.1$	2.3	25k	17	154	0.2	0.1	2.0	31k	59	16k	2.9	2.9
Looping cave	$0.5 \times 0.7 \times 0.2$	22.1	60k	47	668	0.5	0.5	0.8	67k	809	33k	7.4	7.5
Rectilinear	$0.9 \times 1.1 \times 0.1$	19.6	65k	26	830	0.5	0.4	0.8	30k	1076	14k	3.3	3.0
Superimposed	$0.2 \times 0.7 \times 0.2$	2.9	92k	17	209	0.9	0.1	2.0	43k	189	21k	4.7	5.2

TABLE 7.1: *Karstic system extent* $[\mathrm{km}^3]$*, skeleton length* $\mathcal{L}$ *in* $[\mathrm{km}]$*, number of point samples* $\#\mathcal{Q}$*, and number of key points* $\#\mathcal{P}$ *placed by the user, karstic skeleton node count* $\#\mathcal{K}$*, skeleton generation time* $\mathcal{T}_\mathcal{S}$ *in* $[s]$*, tortuosity factor* θ *given by the user, neighborhood parameter* γ *given by the user, number of node in the construction tree* $\#\mathcal{N}$*, number of sweep* ($\#\mathcal{S}_w$) *and sphere* ($\#\mathcal{S}$) *primitives, memory footprint* $\mathcal{M}$ *of the construction tree* $[\mathrm{Mb}]$ *and construction tree generation time* $[s]$.

7.7.2 Control

Our approach combines the placement of key points, the definition of geological features (inception horizons, orientation distributions), the tuning of the cost function as well as setting the γ coefficient. These parameters provide user control and allow the reproduction of identified patterns found in real karstic systems. Figure 7.22 shows synthesized networks compared to real ones displayed in insets. Typically, rectilinear and anastomotic mazes are created by setting $\gamma = 0.8$ and tuning orientation distributions (three axis-aligned directions for rectilinear, and 8 directions on the sphere for the looping cave). In contrast, an elongated branchwork results from the placement of a single sink and a spring. A superimposed network results from two inception horizons and several amplification steps (3 in the presented models) adding dendritic tunnels to the network.

The proposed architecture allows the interactive authoring of complex karstic networks spanning over several kilometers (see Figure 7.22). Labeled key points $\mathcal{P}$ not only control the location of sinks, springs, or breakout chambers inside the bedrock but also allow the user to prescribe tunnel sub-paths that are directly taken into account in the network generation and locally influence the structure of the skeleton. The cost function may also be tuned with interactive visual feedback. The geological parameters and control waypoints provide user-control over the density and structure of the resulting network. At the mesoscale, sweep primitives allow for a detailed and varied reconstruction of the conduits (Figure 7.19). A high tortuosity parameter θ significantly impacts the complexity of the construction tree by producing tortuous trajectories as observed in real karstic systems. The looping cave example (Figure 7.22) was generated with the highest tortuosity ($\theta = 0.5$) with a memory consuming construction tree featuring 32 809 sphere primitives (see Section 7.6.1). In contrast, the elongated branchwork or superimposed networks with $\theta = 0.1$ have a reduced memory footprint.

7.7.3 Comparison with real karstic networks

Collon *et al.* 2017 presented and discussed a set of metrics to describe, compare and quantify karstic networks. Among the 21 tested metrics, they recommend computing 6 of them to identify the geometry and the topology of a network. The open-source Karstnet code implements them in Python and was used to compute the metrics for the karstic networks presented in Figure 7.22 so as to compare them with a dataset of 34 real caves (Collon *et al.* 2017). The code is available at:

https://github.com/karstnet/karstnet

Karstic system	$\bar{l}$	H_O	$\overline{k}$	r_k	$\overline{SPL}$	CPD
Branchwork	96	0.9	2.0	−0.4	4.1	0.6
Looping cave	79	0.9	3.2	0.04	5.0	0.2
Rectilinear maze	121	0.6	3.0	0.0	5.6	0.1
Superimposed net	101	0.8	2.1	−0.3	4.9	0.4
Observed range	8	0.8	1.8	−0.6	2.3	0.0
	331	1.0	2.6	−0.2	55.7	0.6

TABLE 7.2: *Average branch length $\bar{l}$ [m], entropy of orientation H_O, average vertex degree $\overline{k}$, correlation of vertex degrees r_k, average shortest path length $\overline{SPL}$, central point dominance CPD.*

Table 7.2 reports statistics and compares to a range of values computed on a dataset of 34 real karstic networks (Collon *et al.* 2017). The average branch length $\bar{l}$ and the entropy of orientation H_O describe the geometry of the network. The simulated networks show similar values to the karsts of the database, except for the entropy of orientation H_O of the *rectilinear maze* which is slightly lower than those of studied networks. This is, however, not surprising as we voluntarily restricted the influence of fracture orientation to generate this *stereotyped* network. In the available database, there was no 3D network with such marked orientation. The data of the *crossroads* karst, presented in Figure 7.22, was only available as a 2D map projection (Palmer 1991). Computing this metric on *crossroads* data yields $H_O = 0.46$, demonstrating that the value $H_O = 0.62$ obtained on the *rectilinear maze* is acceptable for this kind of network.

The computed values also remain consistent for the topological metrics. Average shortest path length $\overline{SPL}$ and central point dominance CPD fall both into the observed range of values for all simulated networks, as well as average vertex degree $\overline{k}$ and correlation of vertex degrees r_k for the *elongated branchwork* and *superimposed* networks. The maze-like networks *looping cave* and *rectilinear maze* have $\overline{k}$ values slightly superior to what was observed. It is admittedly rare that real karstic networks have such a high amount of crossing points where more than 3 conduits meet, and breakout chambers with more than 4 conduits (and consequently node degree superior or equal to 4) are scarce. This is again a consequence of our intent to generate stereotyped networks, which also explains the corresponding $r_k \simeq 0$ while natural systems tend to be slightly disassortative.

7.7.4 Comparison with other techniques

Few methods for modeling complex karstic networks exist. The Arches model (Peytavie *et al.* 2009b) requires manual editing for authoring terrains and does not automatically generate tunnels or caves. While invasion-percolation (Chapter 6) can synthesize complex karstic structures, the user has no control over

the generated network and the resulting tunnels are carved using simple spheres or curves, leading to unrealistic tunnel shapes. Voxel-based approaches (Pytel *et al.* 2015; Franke *et al.* 2022) analyze parameters such as pressure and fractures, but those simulations are computationally intensive and do not lend themselves to synthesizing large karstic networks. Our approach extends the grid-based method presented in Galin *et al.* 2011 and proposes a grid-less technique that can adaptively sample three-dimensional domains. In particular, the gridless shortest path solves geometric aliasing issues resulting in unrealistic tunnel paths, which occur when using regular or adaptive grids. To the best of our knowledge, this is the first controllable procedural method that captures the complex structure of karstic networks both at large scale and mesoscale.

Considering the geometry of the tunnels, our two-fold strategy for modeling tunnels proves to be versatile and exhibits good properties. Tunnels modeled with sweep primitives (or sampled cross-sections) allows for representing a wide variety of tunnel shapes (see Figure 7.19). Compared to data-oriented or hybrid models, the construction tree is compact in memory and efficient for modeling large karstic networks.

7.7.5 Limitations

We rely on a simple strategy for the classification of tunnel types and parameters computation, as knowledge in Geomorphology is sparse on this topic. However, the proposed framework provides effective user-control and processing parts could be easily replaced with more accurate computations in the future.

Although implicit surfaces allows modeling of the highly detailed geometry of karstic conduits, visualization remains expensive as the signed distance function f is defined by a hierarchical combination of thousands of nodes. Polygonizing a $1 \times 1 \times 0.4\text{km}^3$ volume at 0.5m precision takes up to two minutes for a dense complex karstic model featuring $30\,000$ nodes. The construction tree may be amenable to simplification; primitives and operators could be further optimized and field function queries performed in parallel using compute shaders on graphics hardware.

7.8 Conclusion

We have introduced a geologically-based framework for modeling karstic networks while retaining user control. Given an input terrain, our method computes a three-dimensional geometric graph connecting key points corresponding to sinks (inlets), springs (outlets), and known passages inside the bedrock. The paths connecting control points are constructed by using a gridless anisotropic shortest path taking into account geological parameters such as the permeability of the bedrock, fracture orientations, and inception features. The geometry of the conduits is finally constructed by a signed distance function defined as a construction tree combining volumetric primitives. The synthesized karstic networks accord with real data obtained from geological observations.

Our approach allows for modeling large scale karstic networks as well as their mesoscale geometry. Small scale details such as stalactites and stalagmites can be added by blending skeletal primitives (Chapter 6), at the expense of a more complex construction tree. However, it still relies on noise-based primitives for meso and microscale details, whereas real tunnels exhibit more complex structures which require thousands of primitives in our system. In the next chapter, we show how to efficiently model such landforms by introducing new primitives and operators suited for this task, along with a procedural algorithm to generate them.

Chapter 8

Modelling rocky scenery using implicit blocks

Contents

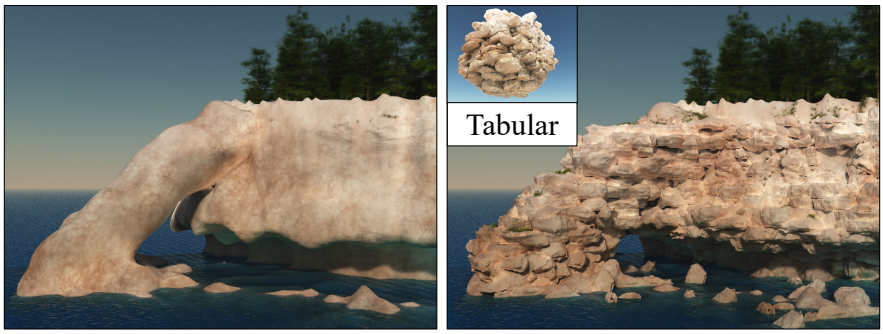


FIGURE 8.1: *Given a low resolution input terrain $\mathcal{T}$, which may be either an implicitized heightfield (See Chapter 6 Section 6.4.1) or a volumetric representation, our method generates fractured mesoscale blocks featuring small-scale details inside a cubic tile, and replicates them realistically in the terrain according to the geology of the different strata.*

8.1 Introduction

Three-dimensional and vertical landforms such as cliffs, steep-walled canyons, crags, promontories, or overhangs are fundamental visual elements of scenic terrains. Despite the wide application of artificial terrains in the entertainment industry and in simulations, modeling truly volumetric landforms with highly detailed rocky overhangs and cliffs with bare rock strata remains an unsolved problem.

Challenge The vast majority of existing techniques addresses only heightfield terrains that do not allow for an accurate representation of vertical landforms. Even though recent advanced techniques for modeling truly volumetric terrains have been proposed (Peytavie *et al.* 2009b; Becher *et al.* 2019), most of them have a limited resolution. Hyper-textures (Perlin *et al.* 1989) can synthesize fractal mesoscale and microscale details over the surface of an otherwise low-resolution terrain, however, the self-similar appearance of the resulting bedrock, often corresponding to sandstone, lacks structure. Generating the mesoscale block structures and small-scale patterns of bedrock that appear on bare rocky terrain and the vertical walls of canyons, steep-walled cliffs or promontories, has received little attention. While the method presented in Chapter 6 is capable of generating large-scale volumetric features such as deep overhangs and arches, but the results still lack mesoscale structure on the vertical parts of the terrain.

The challenge stems from the fact that the geometry of rocky surfaces results from different physical processes (including fracturing, percolation, and erosion), depends on the materials involved (such as limestone, dolomite, sandstone or basalt), and shows in a variety of forms (from regular hexagonal prisms to seemingly chaotic polyhedral shapes) and scales (from a few decimeters to meters).

Contributions In this chapter, we extend the implicit surface modelling framework introduced in previous chapters with new primitives and operators capable of representing detailed mesoscale volumetric features, such as vertical walls of rocky cliffs, crags or promontories. We also propose an amplification method that enhances an otherwise smooth input terrain by automatically carving geomorphologically consistent volumetric details over cliffs and overhangs. Our method consists in spawning fractures which are used to generate blocks that will be visible on the vertical walls of cliffs (Figure 8.1). More precisely, the main contributions of this work are as follows:

1. We present an original geologically-based fracturing process to generate realistic blocks of rock separated by fractures tiling space.

2. We introduce a novel gradient-based warping operator for adding surface details to implicit primitives which allows us to carve small-scale rock patterns from synthetic or real images over blocks.

3. We define a field function node compatible with any hierarchical implicit surface modeling framework for replicating the field functions characterizing the blocks, therefore implicitly replicating them over a volumetric terrain or a heightfield.

Altogether, we provide a unified implicit framework for the representation of mesoscale ($\approx 1\,\mathrm{m}$) and small-scale ($\approx 1\,\mathrm{cm}$) bedrock details allowing to reproduce complex bedrock patterns and shapes. This approach provides multiple levels of control, allowing the user to author blocks and tune the placement rules.

After an overview of our approach (Section 8.2), we detail how to compute block tiles for different geological shapes (Section 8.3) and show how to amplify an input terrain with these created blocks (Section 8.4). We finally discuss performance, limitations and compare our work with existing techniques (Section 8.5).

The contributions presented in this chapter were published in Paris *et al.* 2020 and the code for reproducing the results is available on Github.

8.2 Overview

Real terrains often feature complex rock formations in areas such as cliffs or caves. These landforms are the result of complex, interconnected physical processes that include glacial erosion, catastrophic rock collapses due to instability, or aeolian erosion. To avoid computationally intensive simulations, we propose a procedural approach to create block formations based on classifications used in geomorphology.

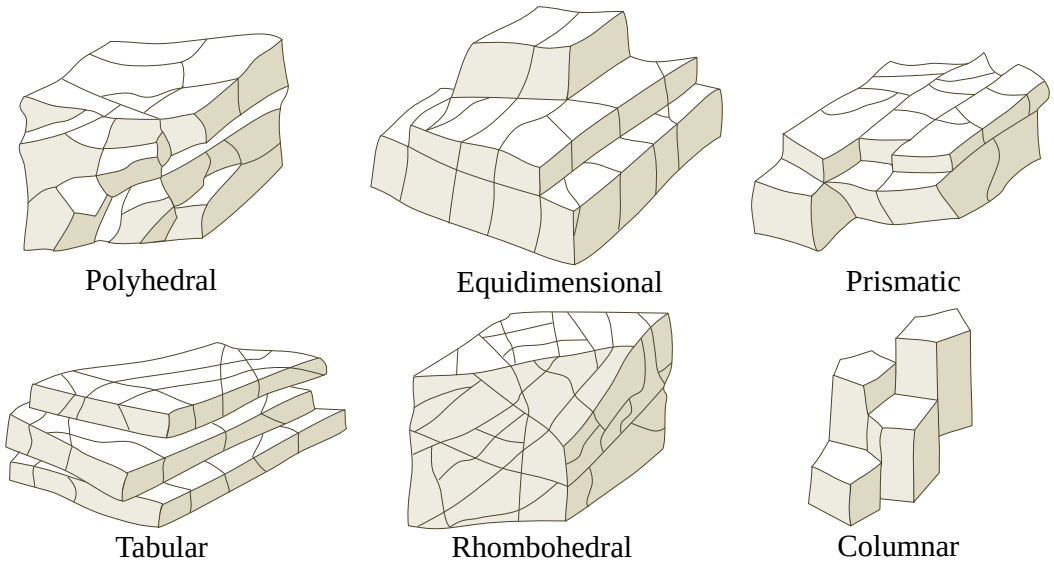


FIGURE 8.2: *Categories of blocks identified in geomorphology corresponding to various fracture distributions.*

This work comes from the observation that fractures in bedrock form blocks exhibiting a variety of shapes of different sizes. Archetype structures such as prismatic, equidimensional or rhombohedral (Figure 8.2) have been identified in geomorphology (Palmstrom 2001; Dearman 1991). A fracture is a break

of continuity in the body of rock whose origin is natural. It most frequently occurs as *fracture sets* that are defined as a family of parallel, evenly spaced broken fractures that can be identified by analyzing their orientations, spacing, and physical properties. Regular and periodic distributions of fractures result in regular columnar blocks, whereas distributions with three major orthogonal directions result in equidimensional blocks; randomly distributed fractures produce polyhedral type.

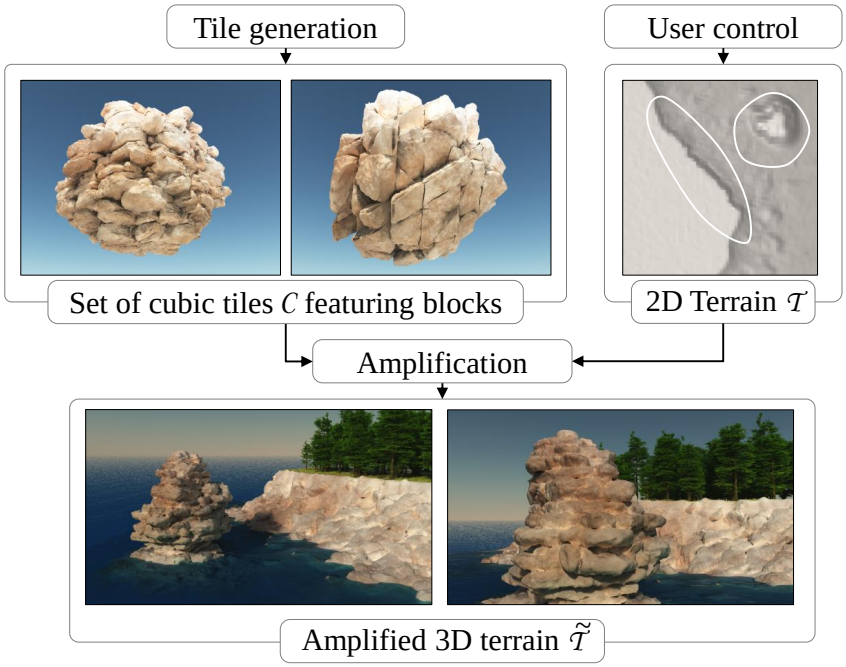


FIGURE 8.3: *Overview of the algorithm: during a pre-processing step, we generate cubic tiles containing blocks of distinct types according to different fracture distributions; then, given an initial low-resolution heightfield, strata definition and control regions, we automatically generate an implicit model defined as a field function replicating the blocks for creating mesoscale and small-scale details.*

Fracturing the entire input terrain $\mathcal{T}$ would be computationally and memory intensive. Therefore, we propose a procedural tiling method whose goal is to compute cubic tiles containing geologically correct blocks that will be placed over the vertical parts of bare bedrock.

The overall terrain amplification process is composed of two steps (Figure 8.3). We first generate a set of blocks $\mathcal{B}_i$ organized into a cubic tile denoted as $\mathcal{C}$ using a procedural fracturing approach based on a geomorphological classification (Section 8.3). These blocks $\mathcal{B}_i$ are implicitly defined by scalar functions $b_i : \mathbb{R}^3 \rightarrow \mathbb{R}$, which allows to obtain a 3D volumetric representation of the mesoscale patterns and small-scale details using a new gradient-based warping operator.

The second step consists in amplifying a smooth input terrain $\mathcal{T}$ by replicating blocks over the bare vertical parts of the bedrock (Section 8.4). The input terrain can be either an elevation terrain such as a heightfield, or a volumetric terrain model such as the ones produced by a construction tree (see Chapter 6 and Chapter 7). The new detailed volumetric terrain model $\widetilde{\mathcal{T}}$ is defined as an implicit surface whose scalar field function $\widetilde{f}$ is a construction tree combining the field function of the initial terrain f and the scalar functions of the blocks b_i.

The cubic tile $\mathcal{C}$ is virtually tiling $\mathbb{R}^3$, and only the blocks $\mathcal{B}_i$ that straddle the vertical parts of the terrain are replicated. To avoid explicit instantiation, we propose an original selective field function replication operator inspired from Stolte 2002 that allows to virtually replicate the field functions b_i of the blocks only over the vertical parts of the input terrain.

8.3 Block tile generation

The key process in the formation of blocks is *fracturing*. In geomorphology, fractures are represented and simulated using 3D discs that define the formation of blocks by breaking the continuity of the bedrock.

From this observation, we propose a procedural and controllable method to simulate the different types of block formations as found in nature. We address the generation of blocks tiling space. In the following section, we consider periodic tiling out of clarity, the generation of aperiodic tiling using Wang Cubes (Culík 1996) or Corner Cubes (Lagae *et al.* 2006a; Peytavie *et al.* 2009a) is a direct technical generalization of this work. Therefore, we address the generation of blocks in a cubic tile $\mathcal{C}$ of size s (the implementation uses a cube with size $s \approx 20\,\mathrm{m}$). The algorithm proceeds in two steps as depicted in Figure 8.4.

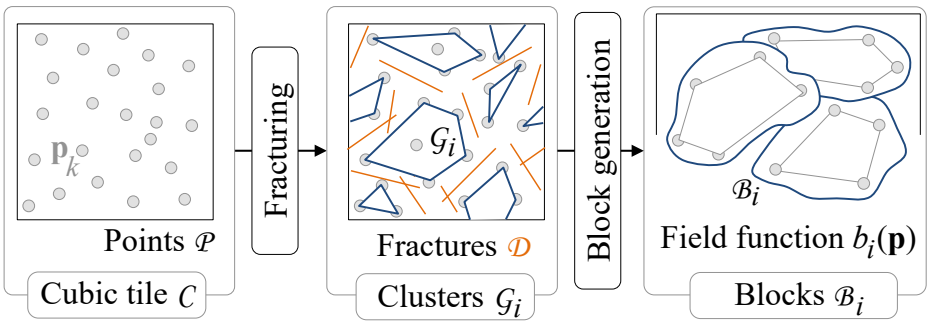


FIGURE 8.4: *Overview of the blocks generation pipeline inside a cubic tile $\mathcal{C}$. We first compute a nearest neighbor graph $\mathcal{G}$ over a set of sample points $\mathcal{P}$ inside the tile. Fracturing discs $\mathcal{D}$ remove edges crossing fractures. We extract clusters $\mathcal{G}_i$ as the set of disconnected sub-graphs, and generate implicit primitives $\mathcal{B}_i$ for each cluster. For the sake of clarity, the method is depicted in 2D and edges of $\mathcal{G}$ are not represented.*

Fracturing. Starting from an initial set of points $\mathcal{P} = \{\mathbf{p}_k\}$ sampling the cubic tile, we compute the geometric nearest neighbor graph $\mathcal{G}$ over this set. A set of fractures $\mathcal{D} = \{\mathcal{D}_i\}$ where $\mathcal{D}_i$ are discs is then generated using procedural rules. These fractures cut edges from the graph, thus creating connected sub-graphs called clusters and denoted as $\mathcal{G}_i$ that will finally form the blocks.

Implicit primitive generation. The low-resolution geometry of the blocks is defined as the convex hull of the point sets for every cluster $\mathcal{G}_i$. For every block, we define a scalar function b_i that computes a signed distance bound to the surface of the block. Its corresponding construction tree combines the planes of the convex hull using a smooth intersection operator. The convex blocks are responsible for the mesoscale details of the bare rock. We finally apply a new gradient-based warping operator for generating the small-scale volumetric details.

8.3.1 Fracturing

The fracturing process starts by generating a set of sample points $\mathbf{p}_k$ inside the cube, using a Poisson sphere distribution. Experiments demonstrated that a regular sampling leads to 3D aliasing with unnatural axis-aligned fractured shapes.

The Poisson radius r influences the size and shape of the blocks. Higher radius values yield tiles with fewer points inside, which in turn leads to blocks with convex shapes made of fewer polygonal faces after the fracturing process. For a cubic tile size of $s \approx 20\,\mathrm{m}$, we generate points with a Poisson radius $r \approx 10\,\mathrm{cm}$ which leads to $\approx 14\,000$ points inside the tile. After fracturing, each individual block contains between 80 and 150 sample points (see Table 8.1). We then compute the nearest neighbor graph $\mathcal{G}$ connecting the set of points $\mathcal{P}$ using r as the neighboring distance threshold, *i.e.* an edge between points $\mathbf{p}_i$ and $\mathbf{p}_j$ exists if $\|\mathbf{p}_i - \mathbf{p}_j\| < r$.

Fractures $\mathcal{D}_i(\mathbf{c}_i, r_i, \mathbf{n}_i)$ are defined as discs in space characterized by their centers $\mathbf{c}_i$, radii r_i and normal orientation $\mathbf{n}_i$. Geomorphological types are distinguished by the way fractures are distributed in the cube, *i.e.* by the following parameters: number of fractures, average radius size, distribution of disc centers, and relative orientations. Without loss of generality, discs centers $\mathbf{c}_i$ are randomly generated in the cube using a Poisson sphere sampling with an average radius of $\approx 2\,\mathrm{m}$. Both r_i and $\mathbf{n}_i$ distributions are tuned according to the block type (see Figure 8.5).

The user may control the fracturing process either by tuning the disc distribution parameters, or by placing specific fractures in the cubic tile, or by directly placing several authored blocks in the tile (see Figure 8.6), and the system will adapt. It is also possible to increase the overall amount of fracture by decreasing the Poisson radius of the fracture centers distribution. Other disc sampling strategies could be used, however we found Poisson sampling practical for obtaining regular-pattern-free distributions with a practical control over the disc centers spacing.

The following paragraphs explain the different fracture distributions, as described in Palmstrom 2001 and their control parameters.

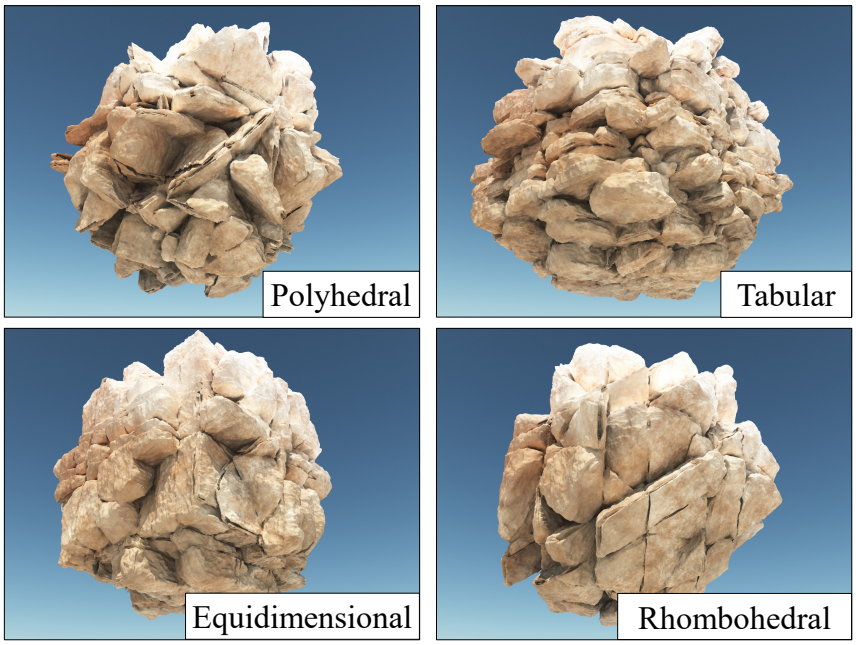


FIGURE 8.5: *Different types of blocks generated with different disc distributions.*

Equidimensional block type has three dominant sets of fractures, approximately orthogonal, with occasional irregular fractures, giving almost cubic-shaped blocks. Prismatic blocks are similar in shape but are formed under slightly more irregular fracture distributions. For these types, normals $\mathbf{n}_i$ are

determined using a random axis-aligned direction. Radii r_i are randomly chosen in an interval given by a user parameter. In our experiments, we set the radii to $\approx 5\mathrm{m}$.

Polyhedral type shows irregular small blocks without explicit arrangement into distinct sets. This type requires more fractures due to the completely stochastic essence of the distribution, therefore we set the radius of the fracture centers distribution to $\approx 1\,\mathrm{m}$ and create more fractures.

Rhombohedral blocks have three (or more) dominant mutually oblique sets of fractures, giving oblique-shaped blocks. The parameters are the same as for equidimensional blocks, but with tilted axis-aligned directions to obtain diagonal orientations.

Tabular block type has one dominant set of parallel fractures orthogonal to a dominant axis direction, for example bedding planes, with other non-persistent fractures; thickness of blocks is much lower than length or width. Discs orthogonal to the dominant axis orthogonal have a large radius equal to the size s of the cubic tile $\mathcal{C}$, whereas the other discs parallel to the two orthogonal axes occur less frequently, and have with small radii $r_i \approx s/10$.

Columnar type is composed of several (usually more than three) sets of continuous, parallel fractures. The length is much greater than other dimensions.

Type	$\#\mathcal{P}$	$\#\mathcal{D}$	$\#\mathcal{B}$	$\#\overline{\mathcal{P}}_i$	Time
Equidimensional	14 524	15	125	110	6.9
Rhombohedral	14 589	24	78	161	12.0
Polyhedral	14 532	44	100	140	25.4
Tabular	14 554	30	82	143	16.3

TABLE 8.1: *Statistics for the generation tiles: number of points $\#\mathcal{P}$, number of fracture discs $\#\mathcal{D}$, number of blocks $\#\mathcal{B}$, average number of points $\#\overline{\mathcal{P}}_i$ inside a block $\mathcal{B}_i$, and generation time (in seconds).*

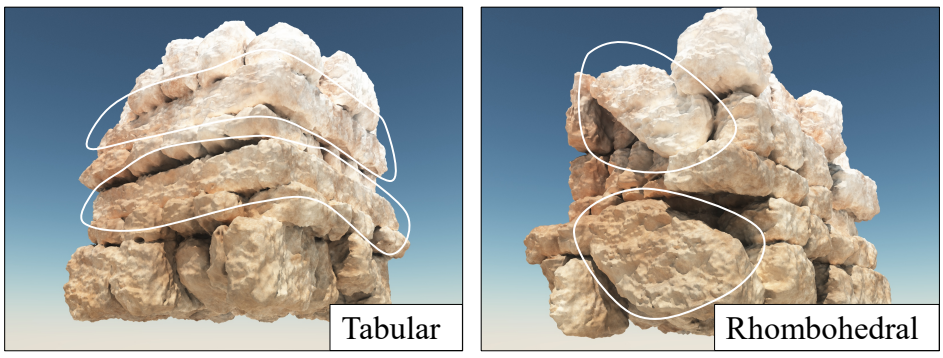


FIGURE 8.6: *Example of user-control during the fracturing step: the user authored specific tabular blocks (left), and rhombohedral blocks (right) inside an equidimensional block tile.*

The next step consists in removing edges cut by a disc $\mathcal{D}_i$ and then create clusters $\mathcal{G}_i$ of connected nodes $\mathbf{p}_k$. This is performed by using a greedy algorithm: starting from a random point, aggregation propagates through the graph until no more points can be connected. Formally, an edge $\mathbf{p}_j\mathbf{p}_k$ is kept if it

does not intersect any fracture:

$$\forall\, \mathcal{D}_i \in \mathcal{D},\ \mathcal{D}_i \cap \mathbf{p}_j\mathbf{p}_k = \varnothing$$

Finally, we check that all the points in a given cluster are visible to each other, *i.e.* that fractures do not cut an edge connecting any pair of points in the cluster. This guarantees that clusters should not spread around fracture discs, which would create blocks that do respect the constraints.

The complexity of the fracturing step depends on the number of fractures $\#\mathcal{D}$: a highly fractured cube will require more intersection tests between edges of the graph and the discs. Table 8.1 reports some statistics for the different types illustrated in Figure 8.5. Taken individually, blocks have relatively simple shapes; a key aspect is the coherency of the generation process, which takes into account the fracture distribution and creates blocks in contact.

8.3.2 Implicit block generation

Recall that we aim at generating an amplified terrain model $\widetilde{\mathcal{T}}$ as an implicit surface. We extend the signed distance field representation introduced in Chapter 7 and 6 with new primitives and operators to represent the mesoscale blocks along with their small-scale details. We create the 1-Lipschitz signed distance function $b_i : \mathbb{R}^3 \to \mathbb{R}$ from the previously computed clusters $\mathcal{G}_i$ as a construction tree defined as the smooth-intersection of half-spaces forming a base convex shape, and then deformed using a gradient-based warping operator (Figure 8.7).

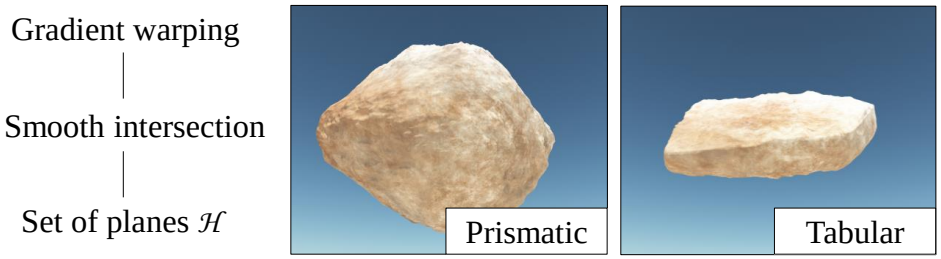


FIGURE 8.7: *Simplified hierarchical representation of blocks $\mathcal{B}_i$: the base convex shape is defined as the smooth intersection of a set of plane primitives $\mathcal{H}$, and procedurally warped.*

We first compute the convex hull of each cluster and extract a corresponding polygonal mesh $\mathcal{M}_i$. We compute the planes $\mathcal{H}_k = (\mathbf{o}_k, \mathbf{u}_k)$ associated to every polygon of $\mathcal{M}_i$ such that their normal $\mathbf{u}_k$ should be oriented towards the exterior of $\mathcal{M}_i$. The corresponding scalar function of the convex base c_i is defined as the smooth intersection of the half-spaces functions h_k associated to planes $\mathcal{H}_k$. Every half-space is defined by the signed distance to the plane $h_k(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_k) \cdot \mathbf{u}_k$. The final implicit convex blocks $\mathcal{B}_i$ are finally defined as:

$$b_i(\mathbf{p}) = c_i \circ \omega^{-1}(\mathbf{p})$$

where $c_i(\mathbf{p})$ denotes the function associated to the block $\mathcal{B}_i$, and ω^{-1} denotes a gradient-based warping operator modeling small-scale details. Should the points of the cluster $\mathcal{G}_i$ be coplanar, or contain less than 3 points, we define the base geometry as a thick polygon, line segment or point primitive.

Smooth convex base. Using a traditional intersection operator (Wyvill *et al.* 1999) defined as $f_{A\cap B} = \max(\mathrm{f_A}, \mathrm{f_B})$ creates gradient discontinuities (Figure 8.8) which prevents the use of gradient-based warping as the resulting field function would no longer be continuous. Therefore, we use the smooth intersection operator introduced in Barthe *et al.* 1998 over the n planes of the block $\mathcal{B}_i$. Without loss of

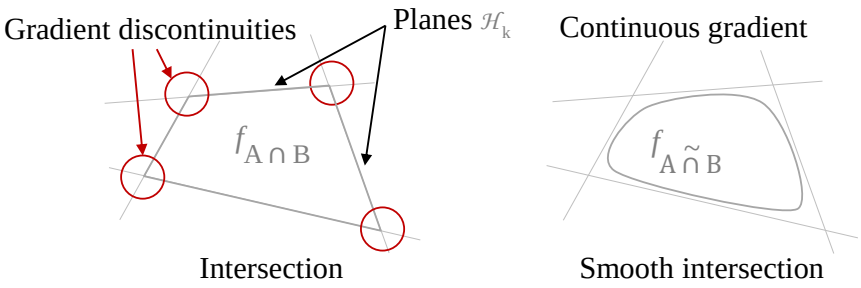


FIGURE 8.8: *Smooth intersection generates a convex block shape with rounded edges and preserves the continuity of the gradient ∇c_i, therefore allowing a correct gradient-based warping.*

generality, we consider the intersection in the case of two half-spaces A and B. The field function of the smooth intersection $A \widetilde{\cap} B$ is parameterized by a radius R and defined as:

$$f_{A \widetilde{\cap} B} = \max(f_A, f_B) - R\, k(f_A, f_B)^3/6$$

$$k(f_A, f_B) = \max(1 - |f_A - f_B|/R, 0)$$

Higher values for R lead to smoother shapes, whereas smaller values preserve sharp features (we set $R = 25\,\mathrm{cm}$). Since the smooth intersection operator is not associative, the order in which intersections are performed may influence the final scalar field. In practice, the impact of the ordering is limited and appears not to have any visual impact over the block shape. Different operators could also be used (Gourmel *et al.* 2013), as long as they do not introduce gradient discontinuities. This operator provides a C^0 gradient for the convex primitive and is also 1-Lipschitz (see Appendix A.1 for a demonstration).

Surface details. Adding surface details to implicit surfaces is a challenging problem as implicit surfaces do not provide an explicit parameterization. Existing methods rely either on interactive authoring (Sugihara *et al.* 2010) or require an explicit parameterization of the implicit surface (Zanni *et al.* 2012), and do not lend themselves for generating the surface details of blocks. We introduce a new gradient-based warping operator (Chapter 5 Section 5.6.1) for adding details to any implicit surfaces, taking inspiration from tri-planar projection (Geiss 2008). This warping, applied for every block, does not require an explicit parameterization of the surface, and introduces small-scale volumetric details as displacements that enhance the model. Figure 8.9 shows the effect of warping with different relief functions d encoded as images. The operator is defined as:

$$\omega^{-1}(\mathbf{p}) = \mathbf{p} + \delta(\mathbf{p})$$

Let $\hat{g}(\mathbf{p}) = \nabla b(\mathbf{p})/\|\nabla b(\mathbf{p})\|$ denote the normalized gradient. Let $\gamma_i(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^2$ denote the projection of $\mathbf{p}$ on the i-th plane, namely xy, xz and yz. The function $\delta(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^3$ computes the 3D warping of $\mathbf{p}$ according to a 2D displacement function $d : \mathbb{R}^2 \to \mathbb{R}$:

$$\delta(\mathbf{p}) = \hat{g}(\mathbf{p}) \sum_{i=0}^{3} (\alpha_i \circ \hat{g}(\mathbf{p})) \cdot d \circ \gamma_i(\mathbf{p})$$

The weighting function $\alpha_i : \mathbb{R}^3 \to \mathbb{R}$ weights the contributions of the three displacements $d \circ \gamma_i(\mathbf{p})$ according to the scalar product between the normalized gradient and the unit axis-aligned vectors $\mathbf{u}_i$: $\alpha_i(\mathbf{p}) = |g(\mathbf{p}) \cdot \mathbf{u}_i|$.

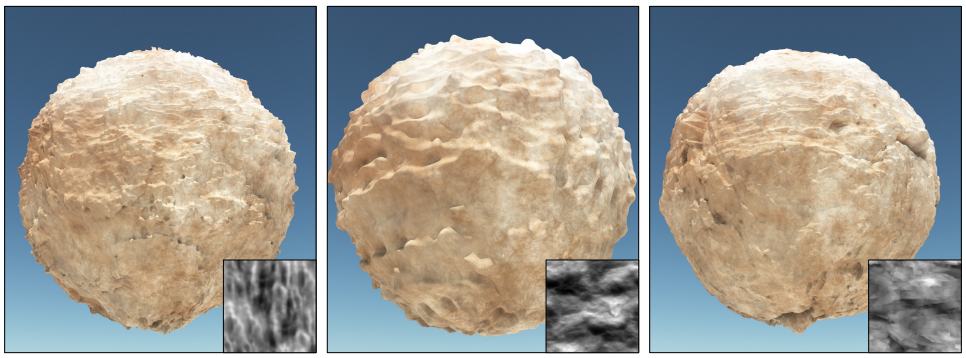


FIGURE 8.9: *Examples of gradient-based warping with different reliefs applied to a sphere primitive.*

The function $d : \mathbb{R}^2 \to \mathbb{R}$ computes the displacement distance and can be effectively defined either as a procedurally defined turbulence, or from real displacement images. It is computed for every projection of the point $\mathbf{p}$, thus three times. Figure 8.10 shows the effect of gradient warping for the tabular type. This gradient-method warping can be applied to other implicit surface models such as the Blob Tree (Wyvill *et al.* 1999).

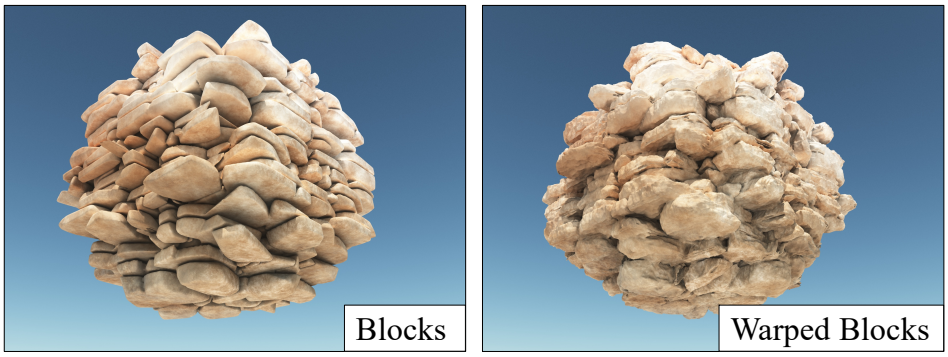


FIGURE 8.10: *Tabular block tile without warping (left) and after gradient-based warping (right): this new operator allows us to create highly detailed blocks without holes.*

8.4 Terrain amplification

The amplification process aims at generating a modified terrain $\widetilde{\mathcal{T}} = \mathcal{T} \cup \mathcal{R}$ defined as the union of $\mathcal{T}$ and the replication $\mathcal{R}$ of some of the blocks $\mathcal{B}_i$ from the tile $\mathcal{C}$ over the bare vertical walls of the cliffs. Let f denote the scalar field of the initial terrain model, the replication operator transforms f into an amplified model $\widetilde{f}$.

Infinite replication of a scalar field b_i over the entire space, first addressed in Stolte 2002, can be obtained by directly computing b_i with the modulo between the argument point $\mathbf{p}$ and the size s of the tiling cube: $b_i(\mathbf{p} \bmod s)$ with $\mathbf{p} \bmod s = \mathbf{p} - s \lfloor \mathbf{p}/s \rfloor$. The challenge consists of computing the scalar fields b_i only for the blocks $\mathcal{B}_i$ that straddle the terrain at certain positions, therefore according to the field function representing the terrain f. We employ a presence function $e : \mathbb{R}^3 \to \{0, 1\}$ to evaluate whether a block should be replicated or not.

For every block $\mathcal{B}_i$, we define a corresponding anchor point $\mathbf{a}_i$ that will be used to evaluate its presence. Here we present the concept with only one anchor point per block out of clarity, but the method

FIGURE 8.11: *A canyon amplified with equidimensional blocks located at the bottom of the canyon, and tabular blocks placed on the higher parts of the cliff walls.*

can be easily generalized for a set of anchor points. Recall that $\lfloor \mathbf{p}/s \rfloor$ denote the integer coordinates of the virtual cell containing $\mathbf{p}$. Blocks are selectively replicated by computing the presence e function at the virtual anchor point in world space $\mathbf{a}_i + s \lfloor \mathbf{p}/s \rfloor$. The replication operator is a function t which defines its field function as the union of all block field function b_i times their presence function e_i.

$$t(\mathbf{p}) = \max_i b_i(\mathbf{p} \bmod s)\, e_i(\mathbf{a}_i + s \lfloor \mathbf{p}/s \rfloor)$$

The final function $\tilde{f}$ of the terrain is defined as $\tilde{f} = \max(f, t)$. More precisely, the evaluation of the replication operator at a point $\mathbf{p}$ is performed as follows. We evaluate in which cell of the grid lies $\mathbf{p}$, using a modulo operation on the floating point coordinates of $\mathbf{p}$. We compute the contribution of each block function b_i, virtually translated in the cell for the point $\mathbf{p}$.

If the distance from $\mathbf{p}$ to a border of a cell is less than a given threshold we compute the contribution in neighboring cells to account for blocks straddling $\mathcal{C}$. Therefore, we compute the total contribution as the union between blocks in the current cell and 2, 4 or 8 neighboring cells.

In the case of a set $\mathcal{A} = \{\mathbf{a}_k\}$ of several anchor points per individual block $\mathcal{B}_i$, the presence function e_i computes the percentage of anchor points $\mathbf{a}_k$ of $\mathcal{B}_i$ which satisfy geometric criteria. In our implementation, we replicate a block if more than half, *i.e.* $\#\mathcal{A}/2$, of the anchor points satisfy $e(\mathbf{a}_k) = 1$.

The presence function can be any combination of a variety of criteria; here we briefly review some important ones that allow for the automatic placement of blocks over the vertical walls, and that provide control by prescribing a geological strata definition as presented in Chapter 6.

We first define criteria based on the distance to the surface: a block can be replicated only if enough anchor points $\mathbf{a}_k$ are in a given distance range $[v_a, v_b]$ to the surface, i.e $f(\mathbf{a}_k) \in [v_a, v_b]$. To only replicate blocks on steep slopes and vertical walls, we compare the direction of the gradient with the up direction u_z; recall that g denotes the normalized gradient of the terrain, we set $e(\mathbf{a}_k) = 1$ if $|g(\mathbf{p}) \cdot u_z| < \varepsilon$, with ε the maximum slope parameter. Finally, we extend on the implicit geological function γ defined in Chapter 6 and define different bedrock material at different location in the domain. Therefore, an anchor point $\mathbf{a}_k$ satisfies the replication criteria if the material $\gamma(\mathbf{a}_k)$ corresponds to the material of the underlying block $\mathcal{B}_i$.

Note that the presence function is generic and can be easily extended to account for other criteria such as the presence of a water level, trees and obstacles.

8.5 Results

We implemented the block generation and replication algorithms in C++ and all the terrains were generated on a desktop computer equipped with Intel® Core $i7$, clocked at 4 GHz with 16GB of RAM. The

implicit surface representing the amplified terrain was polygonized (Wyvill *et al.* 1986) and the resulting mesh directly streamed into Vue Xstream® to produce the final images (Figures 8.5, 8.6, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14). The code is available at:

`https://github.com/aparis69/Rock-fracturing`

Table 8.2 reports some statistics about the different scenes. Contrary to noise-based hyper-textures that only require a few dozens of parameters, the implicit representation needs to store the hierarchical models for the different blocks; the required amount of memory remains small (less than a megabyte).

Scene	Figure	Size	$\#\mathcal{R}$	Memory
Sea cliff	8.1, 8.14	100×100	959	0.57
Canyon	8.11	100×50	1041	0.54
Sea spire	8.3	150×150	1680	0.53

TABLE 8.2: *Statistics for the terrains: size (in meters), number of replicated blocks $\#\mathcal{R}$, and amount of memory (in megabytes) needed to store the field functions b_i representing the blocks $\mathcal{B}_i$ in the cubic tiles $\mathcal{C}$.*

8.5.1 Control

Figure 8.5 shows a variety of types of blocks: different fractures distributions were prescribed (Section 8.3) and lead to different shapes such as polyhedral, rhombohedral or tabular blocks. The user can tune the parameters of different fractures within a tile, and our method automatically computes block shapes that adapt to these constraints; it is also possible to sculpt specific blocks and let the system automatically adapt and generate the remaining ones, as demonstrated by Figure 8.6, where the user placed specific tabular or rhombohedral blocks at the desired locations in the tiles.

Figure 8.11 illustrates user-control over the block placement: an equidimensional bedrock strata was defined by the user at the bottom of the cliff resulting in regular cliff walls. A second strata with tabular type was prescribed above the previous one and tabular blocks were automatically added to account for the specified material.

In practice, it takes a few iterations for the user to tune the placement rules of the different block tiles for a specific scene; the main limitation remains the computationally intensive visualization of the final implicit surface (see Section 8.5.4). Figure 8.11 also demonstrates the effectiveness of control based on the computation of the gradient ∇f for detecting the vertical parts of the terrain: no blocks were generated on the top of the plateau, thus keeping the ground flat where trees appear. Figure 8.14 shows different styles of blocks applied to a sea cliff, completely changing the overall mesoscale geometry of the final landscape.

Microscale details are defined by using the gradient-based warping operator combined with different displacement maps as showcased in Figure 8.9. The relief function d can be painted by the user; procedurally defined by combining noise functions in construction tree of scalar primitives (Génevaux *et al.* 2015) or scanned from real rock surfaces.

8.5.2 Comparison with other methods

While the sum of scaled-noise functions (Ebert *et al.* 1998) can, in theory, produce an infinite amount of details, the self-similar geometries resulting from the fractal process do not capture the characteristic

structures and patterns observed on real terrains. Figure 8.12 shows a comparison of different methods used to add details onto a sphere. Hyper-textures (Perlin *et al.* 1989; Ebert *et al.* 1998) may generate holes and disconnected parts, star-shaped primitives with a radial turbulence (Chapter 6) avoids artifacts but both methods lack geological structure. In contrast, fracturing generates geomorphologically consistent blocks, and gradient-based warping allows for the generation of small-scale details captured from real rocks. Hyper-textures (Perlin *et al.* 1989) could also be used to add small-scale details, however gradient-based warping allows for better control over the displacement by using reliefs from real rocks.

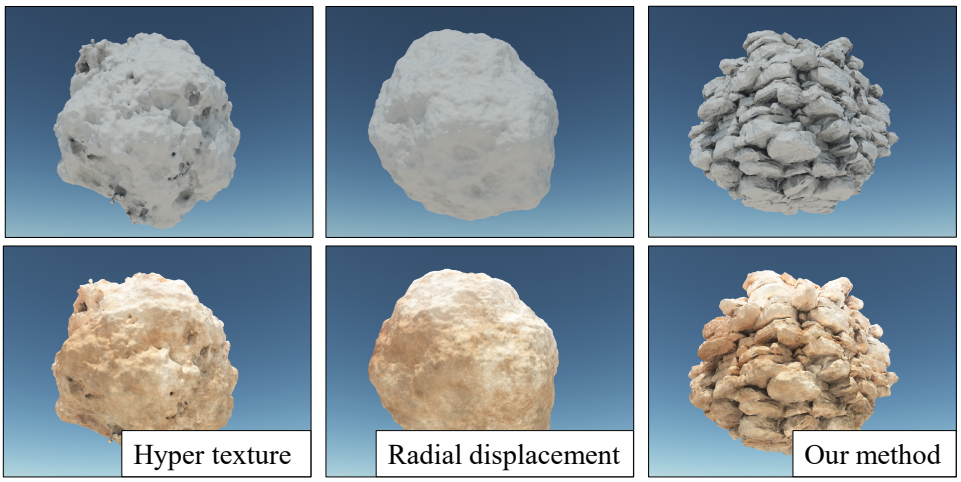


FIGURE 8.12: *Comparison of different methods used to add details onto an implicit sphere, with an ambient shading (top) and a high resolution texture (bottom).*

In terms of mesoscale details, implicit blocks improve existing volumetric terrain models. Global warping operators (Gamito *et al.* 2001) do not provide sufficient accuracy to reproduce the complex geometry of the vertical parts of the bedrock. Voxels and features curves-based approaches (Becher *et al.* 2019) are limited by the grid resolution and generate smooth large-scale terrains.

While the system described in Chapter 6 can generate large scale landforms such as arches, overhangs, and simulate large scale erosion effects, the vertical surface of cliffs lacks geometrical patterns and bedrock details. Although primitive-based implicit surfaces can theoretically reproduce small-scale details, their modeling comes at the price of defining a huge number of small primitives in the construction tree, a memory-intensive process. In contrast, the memory-efficient tiling and replication function implicitly tile space with a union of blocks to amplify and add details to cliffs.

In spirit, generating and replicating instances resembles the Ghost Tiling approach (Guérin *et al.* 2016), and the rock pile generation based on aperiodic tiling (Peytavie *et al.* 2009a) that instantiate rock meshes. Our approach differs in the sense that blocks are defined as implicit primitives and virtually replicated by using a specific operator combined with a presence function, which allows us to combine them to produce a consistent volumetric model. Moreover, these methods would need to be improved to account for fracture distributions to guarantee the replicability of certain block types.

8.5.3 Compatibility with other techniques

An important aspect of this work is that it is compatible with other terrain modeling techniques. The primitives and operators extend the implicit surface model presented in previous chapters (Chapter 7 and

Chapter 6) for modeling volumetric terrains. Terrains modeled using the hybrid layer-stack convolution-surface framework described in Peytavie *et al.* 2009b can also be amplified with mesoscale details, as demonstrated in Figure 8.13.

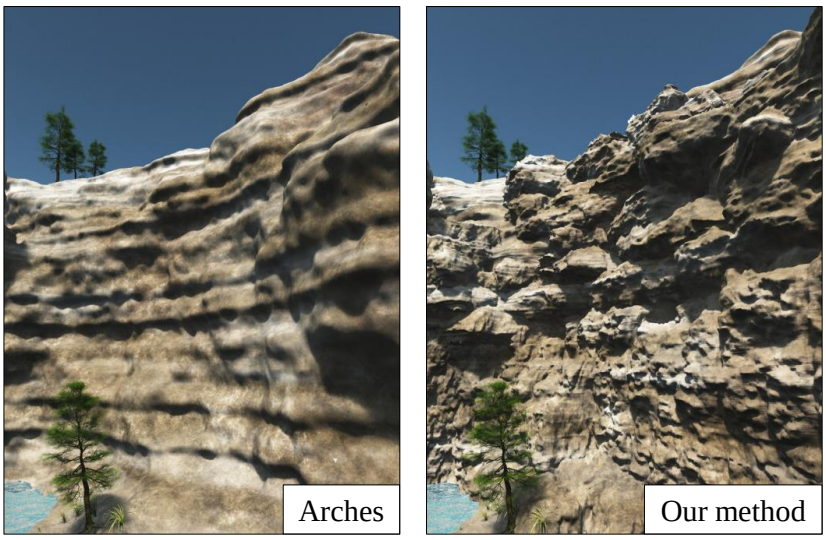


FIGURE 8.13: *Side by side comparison showing a canyon with smooth overhangs produced by the layer-stack model, and the rocky amplified vertical-wall generated by blocks.*

Finally, although we used implicit surfaces for modeling the complex mesoscale features of terrains, the block generation algorithm (Section 8.3) can also produce a mesh representation $\mathcal{M}_k$ for every block $\mathcal{B}_k$ in the tile. In this context, our method can be streamed in production environments using mesh representations and is compatible with standard instantiation techniques such as the ones described in Peytavie *et al.* 2009a or Guérin *et al.* 2016 to distribute block meshes over the vertical parts of the input terrain.

8.5.4 Limitations

The block tile generation process (Section 8.3) can take up to 30s because of the many intersection tests between segments (linking two points in the tile) and fractures discs. However, time was not spent on optimizing this step of this pipeline, since it is done once in pre processing. A more significant limitation is that block primitives can only model convex shapes. This could be resolved by developing primitives more suited for non-convex shapes, but it is beyond the scope of this research and is left as future work.

Although implicit surfaces provide a powerful, sparse, and compact framework for generating complex volumetric structures, their visualization remains computationally intensive. In particular, the computation of the field function b_i for one single block $\mathcal{B}_i$ requires many half space distance computations, combined with a warping operator: blocks could be optimized with a limited number of primitives to reduce the overall number of field function queries.

Using a limited set of block tiles sometimes yields visible repetitions when applied to flat cliff walls parallel to the tiling directions. Aperiodic tiling using Wang cubes (Culík 1996) or Corner Cubes (Lagae *et al.* 2006b; Peytavie *et al.* 2009b) would reduce visible artifacts, yet at the price of a larger number of pre-computed tiles and, therefore, a more memory demanding implementation. Finally, the procedural fracturing tool only provides an indirect control over the distribution of fractures, and thus the shape

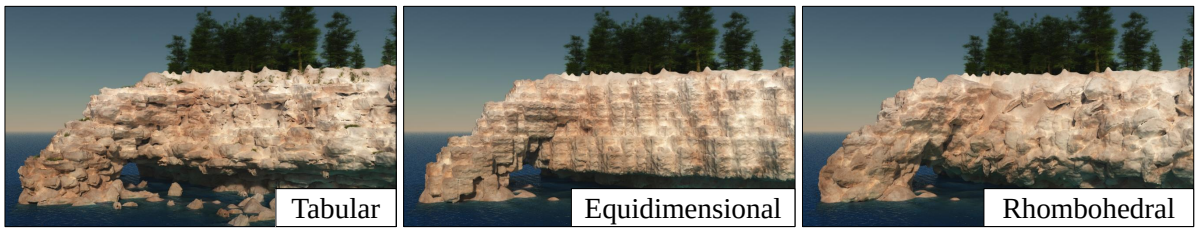


FIGURE 8.14: *Different styles of blocks generated on a cliff and arches. From left to right, tabular block style, equidimensional blocks and finally rhombohedral block style.*

of the blocks: a better interactive material editing tool would allow the user to author a wider range of shapes conforming to his intent.

8.6 Conclusion

We have introduced an implicit representation of blocks for automatically generating mesoscale and microscale details on the vertical walls of rocky cliffs, crags, or promontories. The implicit surface description allows generating realistic shapes enhanced with details organized into patterns and structures as observed in geomorphology. We rely on the decomposition of a bedrock tile into blocks using a fracturing approach based on classifications in geomorphology. Moreover, it is compatible with other terrain modeling techniques: the block generation algorithm can additionally produce triangle meshes, which in turn can be instantiated to enhance heightfields.

A direct extension of this work would consist of generalizing to different block types, particularly prismatic and columnar configurations that require specific fractures distributions. Investigating the application of a gradient-based warping operator for small scale details to other implicit surface models would be another interesting research topic.

Chapter 9

Conclusion

We introduced novel contributions to the field of terrain modeling and generation. Our work has led to four publications in international journals (Paris *et al.* 2019b; Paris *et al.* 2019a; Paris *et al.* 2020; Paris *et al.* 2021) (including a best paper award), and one publication in a national conference awarded with a second best paper award (Paris *et al.* 2018). The research presented in Chapter 3 is a work in progress and has not yet been published. We also collaborated to other works published in international journals (Argudo *et al.* 2019; Argudo *et al.* 2020; Galin *et al.* 2020). In the following sections, we summarize our contributions regarding simulation of time-evolving macroscale landforms and volumetric terrain modeling, and finally conclude by outlining future research perspectives.

9.1 Summary

This thesis started by proposing a new classification of terrain generation methods based on the spatial scale at which they operate and the landforms that they generate. Terrain features were separated in three categories: microscale, mesoscale and macroscale landforms. This classification helped us build a better understanding of previous work and allowed to identify several neglected phenomena, especially in *macroscale landform generation* and *volumetric terrain modeling*, which were the subject of our contributions.

Macroscale simulations. In the first part of this thesis, we focused on the simulation of complex macroscale phenomena on terrains. We first studied meandering rivers (Chapter 3) and introduced a new simulation for reproducing the complex time-evolving behavior of meanders. The presented curvature-based approach allows for generating oxbow lakes, crevasses and deviated trajectories by avulsion events over a large river network. We incorporate several levels of control: the user can prescribe trajectories, place control points, or modify the terrain topography in real time, and let the simulation adapt. This work could be easily integrated in existing terrain modeling softwares, where artists may want to generate river networks with realistic meander trajectories.

In Chapter 4, we tackled the problem of simulating desert landscapes. The presented method works on a layerfield representation with bedrock, sand and vegetation stored as elevation models on regular grids. We first proposed a procedural wind computation that takes into account terrain obstacles and acceleration effects. We then described how to transport sand across the terrain to simulate important phenomena such as saltation, reptation, avalanching and abrasion. We are able to reproduce different types of dunes, such as barchan and transverse dunes, but also yardangs created from the abrasive action of the wind. Our method provides interactive feedback, which allows the user to add or remove sand, modify the wind regime, or place vegetation. This research may also be integrated in modeling softwares to generate desert landscapes, especially if using the optimized algorithm that we described.

Volumetric terrain modeling. The second part of this thesis was dedicated to volumetric terrain modeling and generation. We identified that existing models (voxels and layer stacks) were memory intensive and could be used in the context of large terrains. Instead, we developed a complete framework to model, generate, and author volumetric terrains, based on signed distance functions. We showed that implicit surfaces are an expressive modeling framework that can be used for terrain modeling across all scales (microscale, mesoscale, and macroscale).

In Chapter 6, we developed new techniques for procedurally generating large-scale volumetric landforms such as arches, coastal overhangs and Hoodoos, as arrangements of skeletal primitives such as spheres and curves. The erosion is guided by a 3D geology function that associates a resistance value to every point in space. The resulting smooth volumetric landforms can be generated over several kilometers efficiently due to the compact nature of the representation. However, structured large-scale landforms, such as karstic network are tedious to generate using the presented techniques, and the generated features lack details.

In Chapter 7, we focused on the generation of karstic networks made of tunnels and chambers. Our method relies on an anisotropic shortest path computation that takes into account geological parameters (such as inception horizons, permeability, and fractures) and is capable of reproducing different types of network identified in geomorphology. The varied geometry of tunnels is then generated using dedicated signed distance primitives, and finally carved in the terrain.

Finally, Chapter 8 tackled the problem of generating detailed volumetric terrain structures using signed distance functions. By identifying fracture distribution archetypes in geomorphology, we were able to generate different types of block structures that exhibit mesoscale and microscale details. These blocks, encoded as signed distance functions, were used to amplify the terrains generated in previous chapters.

Research replicability. is a crucial subject in all research domains. The ability to reproduce the results of an experiment of another team is key in advancing the state of the art. In Computer Graphics, this may be done by shading the code necessary to reproduce the results shown in the associated paper. In this thesis, each chapter (except for Chapter 3, which is still a work in progress) has led to a public code release on Github. The published code has been reworked to minimize dependencies, and is runnable on both Windows and Linux systems. When possible, we also submitted the repository to the Replicability Stamp initiative.

9.2 Future work

The work presented in this thesis open several new interesting avenue of future work. We detail some of these perspective in the field of terrain modeling and implicit surfaces below.

9.2.1 Regarding terrain modeling

Simulating complex landforms. In this thesis, we took inspiration from classifications and methods from geomorphology, and presented new simulations for generating desert landscapes (Chapter 4 and meandering rivers (3). Other complex phenomena such as braided rivers (**???**) may require similar techniques. More generally, we think that collaborations between geomorphologists and computer scientists may lead to interesting results and ideas for both fields of research.

Multi-materials volumetric terrains. Cordonnier *et al.* 2017 used a multi-layer planar representation for simulating both terrain erosion and ecosystem evolution. This allows the generation of complex interconnected effects, and leads to more varied terrains in output. The same principle could be applied to our implicit surface model for terrains. The presented framework could be extended to account for multiple materials on top of the bedrock, such as sand, small rocks, and water, also defined as an implicit surface. Having this multi-material volumetric definition would give interesting possibilities to designers and generation algorithms. For instance, sand and small rocks could be deposited at the bottom of overhangs, where coastal erosion occurs (Chapter 6). The water surface in phreatic caves could be modeled, and vegetation may also be generated.

9.2.2 Regarding implicit surfaces

Global Lipschitz bounds. In this thesis, we focused on modeling 1-Lipschitz signed distance functions. This requires the computation of the *global* Lipschitz constant λ of the underlying node. The resulting field function is defined as $f(\mathbf{p})/\lambda$, which in turn provides guarantees regarding the convergence of fundamental algorithms, such as sphere tracing (Hart 1996). We compute such Lipschitz constants for several primitive and operators in Appendix A. It would be interesting to extend this work to other nodes, such as replications, symmetries or complex warping.

Local Lipschitz bounds. Another approach introduced by Galin *et al.* 2020 would be to compute local Lipschitz bounds over limited domain, such as the extent of a segment. These tighter, more local bounds are generally significantly lower than the global Lipschitz constant, which lead to less field function evaluation and better computation time. This new segment tracing algorithm has been applied to Blob Trees models (Wyvill *et al.* 1999), and investigating its application to signed distance functions is an interesting research idea that we plan to investigate. This would require the computation of local Lipschitz bounds for signed distance primitives and operators.

Bibliography

Araújo, B. R. de, Daniel S. Lopes, Pauline Jepp, Joaquim A. Jorge, and Brian Wyvill (2015). "A Survey on Implicit Surface Polygonization". In: *ACM Comput. Surv.* 47.4, 60:1–60:39. ISSN: 0360-0300.

Argudo, Oscar, Carlos Andujar, Antonio Chica, Eric Guérin, Julie Digne, Adrien Peytavie, and Eric Galin (2017). "Coherent multi-layer landscape synthesis". In: *The Visual Computer* 33.6, pp. 1005–1015.

Argudo, Oscar, Eric Galin, Adrien Peytavie, Axel Paris, James Gain, and Eric Guérin (2019). "Orometry-based Terrain Analysis and Synthesis". In: *ACM Transactions on Graphics (SIGGRAPH Asia 2019)* 38.6, 199:1–199:12.

Argudo, Oscar, Eric Galin, Adrien Peytavie, Axel Paris, and Eric Guérin (2020). "Simulation, Modeling and Authoring of Glaciers". In: *ACM Transactions on Graphics (SIGGRAPH Asia 2020)* 39.6, 177:1–177:14.

Baas, Andreas C.W. (2002). "Chaos, fractals and self-organization in coastal geomorphology: simulating dune landscapes in vegetated environments". In: *Geomorphology*, pp. 309 –328.

Barbier, Aurelien and Eric Galin (2004). "Fast Distance Computation Between a Point and Cylinders, Cones, Line-Swept Spheres and Cone-Spheres". In: *Journal of Graphics Tools* 9.2, pp. 11–19.

Barnes, Richard, Clarence Lehman, and David Mulla (2014). "Priority-flood: An Optimal Depression-filling and Watershed-labeling Algorithm for Digital Elevation Models". In: *Computers & Geosciences* 62, pp. 117–127. ISSN: 0098-3004.

Barr, Alan H. (1984). "Global and Local Deformations of Solid Primitives". In: *SIGGRAPH Computer Graphics* 18.3, pp. 21–30. ISSN: 0097-8930.

Barthe, Loïc, Véronique Gaildrat, and Rene Caubet (2001). "Extrusion of 1D implicit profiles: Theory and first application". In: *International Journal of Shape Modeling* 7, pp. 179–198.

Barthe, Loïc, Véronique Gaildrat, and René. Caubet (Apr. 1998). "Combining Implicit Surfaces with soft blending in a CSG tree". In: *CSG Conference Series*, pp. 17–31.

Beardall, M., 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*, pp. 7–14.

Becher, Michael, 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.

— (2019). "Feature-Based Volumetric Terrain Generation and Decoration". In: *IEEE Transactions on Visualization and Computer Graphics* 25.2, pp. 1283–1296.

Belhadj, Farès 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, pp. 447–450.

Benes, B. and X. Arriaga (2005). "Table Mountains by Virtual Erosion". In: *Eurographics Workshop on Natural Phenomena*. The Eurographics Association.

Benes, Bedrich (2007). "Real-Time Erosion Using Shallow Water Simulation". In: *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2007)*. The Eurographics Association.

Beneš, Bedřich and Rafael Forsbach (2001). "Layered Data Representation for Visual Simulation of Terrain Erosion". In: *Proc. Spring Conference on Computer Graphics*, pp. 80–85.

Beneš, Bedřich and Toney Roa (2004). "Simulating Desert Scenery". In: *WSCG Proceedings of the 12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pp. 12–18.

Beneš, Bedřich, Václav Těšínský, Jan Hornyš, and Sanjiv K. Bhatia (2006). "Hydraulic erosion". In: *Computer Animation and Virtual Worlds* 17.2, pp. 99–108.

Borghi, Andrea, Philippe Renard, and Sandra Jenni (2012). "A pseudo-genetic stochastic model to generate karstic networks". In: *Journal of Hydrology* 414-415, pp. 516–529.

Bridson, Robert, Jim Houriham, and Marcus Nordenstam (July 2007). "Curl-noise for procedural fluid flow". In: *ACM Transactions on Graphics* 26.3, 46:1–46:3.

Brosz, John, Faramarz F. Samavati, and Mario Costa Sousa (2007). "Terrain Synthesis By-Example". In: *Advances in Computer Graphics and Computer Vision*. Ed. by José Braz, Alpesh Ranchordas, Helder Araújo, and Joaquim Jorge, pp. 58–77.

Carli, Daniel Michelon De, Cesar Tadeu Pozzer, Fernando Bevilacqua, and Victor Schetinger (2014). "Procedural Generation of 3D Canyons". In: *2014 27th SIBGRAPI Conference on Graphics, Patterns and Images*, pp. 103–110.

Carpentier, Giliam J. P. de and Rafael Bidarra (2009). "Interactive GPU-Based Procedural Heightfield Brushes". In: *Proceedings of the 4th International Conference on Foundations of Digital Games*, pp. 55–62.

Chiba, Norishige, Kazunobu Muraoka, and K. Fujita (1998). "An erosion model based on velocity fields for the visual simulation of mountain scenery". In: *The Journal of Visualization and Computer Animation* 9.4, pp. 185–194.

Cojan, Isabelle, Olivier Fouché, Simon Lopéz, and Jacques Rivoirard (2005). "Process-based Reservoir Modelling in the Example of Meandering Channel". In: *Geostatistics Banff 2004*. Ed. by Oy Leuangthong and Clayton V. Deutsch. Springer Netherlands, pp. 611–619.

Collon, Pauline, David Bernasconi, Cécile Vuilleumier, and Philippe Renard (2017). "Statistical metrics for the characterization of karst network geometry and topology". In: *Geomorphology* 283, pp. 122 –142.

Collon, Pauline, Vincent Henrion, and Jeanne Pellerin (Nov. 2012). "An algorithm for 3D simulation of branchwork karst networks using Horton parameters and A*: Application to a synthetic case". In: *Geological Society of London – Special Publications* 370 (1).

Cook, Robert L. (1986). "Stochastic Sampling in Computer Graphics". In: *ACM Transactions on Graphics* 5.1, pp. 51–72.

Cooke, R.U., A. Warren, and A.S. Goudie (2006). *Desert Geomorphology*.

Cordonnier, Guillaume, 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". In: *Computer Graphics Forum* 35.2, pp. 165–175.

Cordonnier, Guillaume, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin (2018a). "Sculpting Mountains: Interactive Terrain Modeling Based on Subsurface Geology". In: *IEEE Transactions on Visualization and Computer Graphics* 24.5, pp. 1756–1769.

Cordonnier, Guillaume, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani (2018b). "Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches". In: *Computer Graphics Forum* 37.2, pp. 497–509.

Cordonnier, Guillaume, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani (2017). "Authoring Landscapes by Combining Ecosystem and Terrain Erosion Simulation". In: *ACM Transactions on Graphics* 36.4.

Corsini, Massimiliano, Paolo Cignoni, and Roberto Scopigno (2012). "Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes". In: *IEEE Transactions on Visualization and Computer Graphics* 18.6, pp. 914–924.

Cortial, Yann, Eric Guérin, Adrien Peytavie, and Eric Galin (2020). "Real-Time Hyper-Amplification of Planets". In: *The Visual Computer*, pp. 2273–2284.

Cortial, Yann, Adrien Peytavie, Eric Galin, and Eric Guérin (2019). "Procedural Tectonic Planets". In: *Computer Graphics Forum* 38.2, pp. 1–11.

Crespin, Benoît, Carole Blanc, and Christophe Schlick (1996). "Implicit Sweep Objects". In: *Computer Graphics Forum* 15.3, pp. 165–174.

Crespin, Benoit, Richard Bézin, Xavier Skapin, Olivier Terraz, and Philippe Meseure (2014). "Generalized maps for erosion and sedimentation simulation". In: *Computers & Graphics* 45, pp. 1–16.

Cui, Juncheng, Yang-Wai Chow, and Minjie Zhang (2011). "Procedural generation of 3D cave models with stalactites and stalagmites". In: *International Journal of Computer Science and Network Security* 11, pp. 94–101.

Culík, Karel (1996). "An aperiodic set of 13 Wang tiles." In: *Discrete Mathematics* 160.1-3, pp. 245–251.

Damiand, Guillaume and Pascal Lienhardt (2014). *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. A K Peters/CRC Press.

Daviet, Gilles and Florence Bertails-Descoubes (2016). "A Semi-Implicit Material Point Method for the Continuum Simulation of Granular Materials". In: *ACM Transactions on Graphics* 35.4, 102:1–102:13.

Dearman, W.R. (1991). *Engineering geological mapping*. Butterworths advanced series in geotechnical engineering. Butterworth-Heinemann.

Derzapf, Evgenij, Björn Ganster, Michael Guthe, and Reinhard Klein (2011). "River Networks for Instant Procedural Planets". In: *Computer Graphics Forum* 30.7, pp. 2031–2040.

Desbrun, Mathieu and Marie-Paule Gascuel (1995). "Animating Soft Substances with Implicit Surfaces". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 287–290.

Dey, Rahul, Jason G. Doig, and Christos Gatzidis (2018). "Procedural feature generation for volumetric terrains using voxel grammars". In: *Entertainment Computing* 27, pp. 128–136.

Dixon, A. R., G. H. Kirby, and D. P. M. Wills (1994). "A Data Structure for Artificial Terrain Generation". In: *Computer Graphics Forum* 13.1, pp. 37–48.

Dong, Junyu, Jun Liu, Kang Yao, Mike Chantler, Lin Qi, Hui Yu, and Muwei Jian (2020). "Survey of Procedural Methods for Two-Dimensional Texture Generation". In: *Sensors* 20.4.

Dunne, Thomas (1978). "Field studies of hillslope flow processes". In: *Hillslope hydrology*, pp. 389–227.

Ebert, David S., Forest Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley (1998). *Texturing and Modeling: A Procedural Approach*. 3rd. The Morgan Kaufmann Series in Computer Graphics. Elsevier.

Emilien, Arnaud, Pierre Poulin, Marie-Paule Cani, and Ulysse Vimont (2015). "Interactive Procedural Modelling of Coherent Waterfall Scenes". In: *Computer Graphics Forum* 34.6, pp. 22–35.

Filipponi, Marco, Pierre-Yves Jeannin, and Laurent Tacher (2009). "Evidence of inception horizons in karst conduit networks". In: *Geomorphology* 106.1, pp. 86–99.

Fournier, Alain, Don Fussell, and Loren Carpenter (1982a). "Computer Rendering of Stochastic Models". In: *Commun. ACM* 25.6, 371–384.

— (1982b). "Computer Rendering of Stochastic Models". In: *Commun. ACM* 25.6, 371–384.

Franke, Kai and Heinrich Müller (2022). "Procedural generation of 3D karst caves with speleothems". In: *Computers & Graphics* 102, pp. 533–545.

Frantz, Yves, Pauline Collon, Philippe Renard, and Sophie Viseur (2021). "Analysis and stochastic simulation of geometrical properties of conduits in karstic networks". In: *Geomorphology* 377, p. 107480.

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

Frisken, Sarah F., Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones (2000). "Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH'00, pp. 249–254. ISBN: 1-58113-208-5.

Gain, James, H. Long, Guillaume Cordonnier, and Marie-Paule Cani (2017). "EcoBrush: Interactive Control of Visually Consistent Large-Scale Ecosystems". In: *Comput. Graph. Forum* 36.2, pp. 63–73.

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

Gain, James E., Bruce Merry, and Patrick Marais (2015). "Parallel, Realistic and Controllable Terrain Synthesis". In: *Computer Graphics Forum* 34.2, pp. 105–116.

Galin, Eric, Eric Guérin, Axel Paris, and Adrien Peytavie (2020). "Segment Tracing Using Local Lipschitz Bounds". In: *Computer Graphics Forum* 39.2, pp. 545–554.

Galin, Eric, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain (2019). "A Review of Digital Terrain Modeling". In: *Computer Graphics Forum (proceedings of Eurographics 2019 STAR)* 38.2, pp. 553–577.

Galin, Eric, Adrien Peytavie, Eric Guérin, and Bedřich Beneš (2011). "Authoring Hierarchical Road Networks". In: *Computer Graphics Forum* 30.7, pp. 2021–2030.

Galin, Eric, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin (2010). "Procedural Generation of Roads". In: *Computer Graphics Forum*. 2nd ser. 29, pp. 429–438.

Gamito, Manuel and Steve Maddock (2008). "Topological Correction of Hypertextured Implicit Surfaces for Ray Casting". In: *The Visual Computer* 24.6, pp. 397–409.

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

Geiss, Ryan (2008). "Generating Complex Procedural Terrains Using the GPU". In: *GPU Gems 3*. Addison-Wesley. Chap. 1, pp. 7–37.

Génevaux, Jean-David, Éric Galin, Éric Guérin, Adrien Peytavie, and Bedřich Beneš (2013). "Terrain Generation Using Procedural Models Based on Hydrology". In: *ACM Transaction on Graphics* 32.4, 143:1–143:13.

Génevaux, Jean-David, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes (2015). "Terrain Modelling from Feature Primitives". In: *Computer Graphics Forum* 34.6, pp. 198–210. ISSN: 1467-8659.

Gourmel, Olivier, Loïc Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger (2013). "A Gradient-Based Implicit Blend". In: *ACM Transactions on Graphics* 32.2.

Groh, Christopher, Andreas Wierschem, Nuri Aksel, Ingo Rehberg, and Christof A Kruelle (2008). "Barchan dunes in two dimensions: Experimental tests for minimal models". In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 78.

Grosbellet, François, Adrien Peytavie, Eric Guérin, Eric Galin, Stéphane Mérillou, and Bedrich Benes (2016). "Environmental Objects for Authoring Procedural Scenes". In: *Computer Graphics Forum* 35.1, pp. 296–308.

Guérin, Eric, Julie Digne, Eric Galin, and Adrien Peytavie (2016). "Sparse representation of terrains for procedural modeling". In: *Computer Graphics Forum (Proceedings of Eurographics)* 35.2, pp. 177–187.

Guérin, Eric, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoit Martinez (2017). "Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks". In: *ACM Transactions on Graphics (proceedings of Siggraph Asia 2017)* 36.6, 228:1–228:13.

Guérin, Eric, Eric Galin, François Grosbellet, Adrien Peytavie, and Jean-David Génevaux (2016). "Efficient modeling of entangled details for natural scenes". In: *Computer Graphics Forum (Proceedings of Pacific Graphics 2016)* 35.7, pp. 257–267.

Guérin, Eric, Adrien Peytavie, Simon Masnou, Julie Digne, Basile Sauvage, James Gain, and Eric Galin (2022). "Gradient Terrain Authoring". In: *Computer Graphics Forum* 41.2, pp. 85–95.

Guillén-Ludeña, S., M.J. Franca, A.H. Cardoso, and A.J. Schleiss (2016). "Evolution of the hydromorphodynamics of mountain river confluences for varying discharge ratios and junction angles". In: *Geomorphology* 255, pp. 1–15.

Hart, John C. (1996). "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces". In: *The Visual Computer* 12.10, pp. 527–545.

Hartmann, A., N. Goldscheider, T. Wagener, J. Lange, and M. Weiler (2014). "Karst water resources in a changing world: Review of hydrological modeling approaches". In: *Reviews of Geophysics* 52.3, pp. 218–242. ISSN: 87551209.

Hendrick, Martin and Philippe Renard (2016). "Subnetworks of Percolation Backbones to Model Karst Systems Around Tulum, Mexico". In: *Frontiers in Physics* 4, p. 43.

Henrion, Vincent, Guillaume Caumon, and Nicolas Cherpeau (2010). "ODSIM: An Object-Distance Simulation method for Conditioning Complex Natural Structures". In: *Mathematical Geosciences* 42.8, pp. 911–924.

Hnaidi, Houssam, Éric Guérin, Samir Akkouche, Adrien Peytavie, and Éric Galin (2010). "Feature based terrain generation using diffusion equation". In: *Computer Graphics Forum* 29.7, pp. 2179–2186.

Hooshyar, Milad, Arvind Singh, and Dingbao Wang (2017). "Hydrologic controls on junction angle of river networks". In: *Water Resources Research* 53.5, pp. 4073–4083.

Howard, Alan and Thomas Knutson (Nov. 1984). "Sufficient Conditions for River Meandering: A Simulation Approach". In: *Water Resources Research* 20, pp. 1659–1667.

Huggett, R.J. (2003). *Fundamentals of Geomorphology*. Fundamentals of Geomorphology. Routledge. ISBN: 9780415241465.

Ikeda, Syunsuke, Gary Parker, and Kenji Sawai (1981). "Bend theory of river meanders. Part 1. Linear development". In: *Journal of Fluid Mechanics* 112, pp. 363–377.

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

Jones, M., M. Farlay, M. Butler, and M. Beardall (2010). "Directable Weathering of Concave Rock using Curvature Estimation". In: *IEEE Transactions on Visualization and Computer Graphic* 16.1, pp. 81–97.

Jouves, Johan, Sophie Viseur, Bruno Arfib, Cécile Baudement, Hubert Camus, Pauline Collon, and Yves Guglielmi (2017). "Speleogenesis, geometry and topology of caves: a quantitative study of 3D karst conduits". In: *Geomorphology* 298, pp. 86–106.

Jákó, Balázs and Balázs Tóth (2011). "Fast Hydraulic and Thermal Erosion on GPU". In: *Eurographics 2011 - Short Papers*. The Eurographics Association.

Kahn, Herman and Andy W. Marshall (1953). *Methods of Reducing Sample Size in Monte Carlo Computations*. Santa Monica, CA: RAND Corporation.

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

Kamal, K. Raiyan and Yusuf Sarwar Uddin (2007). "Parametrically Controlled Terrain Generation". In: *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. GRAPHITE '07, pp. 17–23.

Kämpe, Viktor, Erik Sintorn, and Ulf Assarsson (2013). "High Resolution Sparse Voxel DAGs". In: *ACM Transactions on Graphics* 32.4, pp. 1–13.

Kelley, Alex D., Michael C. Malin, and Gregory M. Nielson (1988). "Terrain simulation using a model of stream erosion". In: *Computer Graphics* 22.4, pp. 263–268.

Krištof, Peter, Bedřich Beneš, Jaroslav Křivánek, and Ondřej Šťava (2009). "Hydraulic Erosion Using Smoothed Particle Hydrodynamics". In: *Computer Graphics Forum* 28.2, pp. 219–228.

Kruger, J. and R. Westermann (2003). "Acceleration Techniques for GPU-based Volume Rendering". In: *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. VIS '03. Washington, DC, USA: IEEE Computer Society, pp. 38–. ISBN: 0-7695-2030-8. URL: https://doi.org/10.1109/VIS.2003.10001.

Lagae, A. and P. Dutré (2006a). "Poisson Sphere Distributions". In: *Vision, Modeling, and Visualization*, pp. 373–379.

Lagae, Ares and Philip Dutré (2006b). "An Alternative for Wang Tiles: Colored Edges versus Colored Corners". In: *ACM Transactions on Graphics* 25.4, pp. 1442–1459.

Lagae, Ares, Sylvain Lefebvre, George Drettakis, and Philip Dutré (2009). "Procedural Noise using Sparse Gabor Convolution". In: *ACM Transactions on Graphics* 28.3, pp. 54–64.

Laine, Samuli and Tero Karras (2010). "Efficient Sparse Voxel Octrees". In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Association for Computing Machinery, pp. 55–63.

Lancaster, Nicholas, Andreas Baas, and Douglas Sherman (2013). "11.1 Aeolian Geomorphology: Introduction". In: *Treatise on Geomorphology*. Vol. 11. Elsevier Academic Press Inc, pp. 1–6.

Leopold, Luna B. and M. Gordon Wolman (1960). "River Meanders". In: *GSA Bulletin* 71.6, pp. 769–793.

Lorensen, William E. and Harvey E. Cline (1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *SIGGRAPH Comput. Graph.* 21.4, pp. 163–169.

Mandelbrot, B. B. (1983). *The fractal geometry of nature*. New York: W. H. Freeman and Comp.

Mark, Benjamin, Tudor Berechet, Tobias Mahlmann, and Julian Togelius (2015). "Procedural Generation of 3D Caves for Games on the GPU". In: *Foundations of Digital Games*.

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

Mieloszyk, Krzysztof (2017). "Terrain generation using glacial and tectonic models". In: *25th International Conference on Computer Graphics, Visualization and Computer Vision 2017*, pp. 1–7.

Miller, Gavin (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, pp. 319–326.

Mitchell, Don P. (1990). "Robust ray intersection with interval arithmetic". In: *Proceedings on Graphics interface '90*, pp. 68–74.

Momiji, Hiroshi, Ricardo Carretero-Gonzalez, Steven Bishop, and Andrew Warren (2000). "Simulation of the effect of wind speedup in the formation of transverse dune fields". In: *Earth Surface Processes and Landforms* 25.

Musgrave, Forest Kenton, Craig E. Kolb, and Robert S. Mace (1989). "The synthesis and rendering of eroded fractal terrains". In: *Computer Graphics* 23.3, pp. 41–50.

Měch, Radomír and Przemyslaw Prusinkiewicz (1996). "Visual Models of Plants Interacting with Their Environment". In: SIGGRAPH'96. ACM, pp. 397–410. ISBN: 0-89791-746-4.

Nakajima, Takeshi, Jeffrey Peakall, William McCaffrey, Douglas Paton, and Philip Thompson (Nov. 2009). "Outer-Bank Bars: A New Intra-Channel Architectural Element within Sinuous Submarine Slope Channels". In: *Journal of Sedimentary Research* 79, pp. 872–886.

Narteau, C., D. Zhang, O. Rozier, and P. Claudin (2009). "Setting the length and time scales of a cellular automaton dune model from the analysis of superimposed bed forms". In: *Journal of Geophysical Research: Earth Surface* 114.F3.

Neidhold, B., M. Wacker, and O. Deussen (2005). "Interactive physically based Fluid and Erosion Simulation". In: *Eurographics Workshop on Natural Phenomena*.

Nickling, W. G. (1978). "Eolian sediment transport during dust storms: Slims River Valley, Yukon Territory". In: *Canadian Journal of Earth Sciences* 15.7, pp. 1069–1084.

Onoue, Koichi and Tomoyuki Nishita (2000). "A method for modeling and rendering dunes with wind-ripples". In: *Proceedings of the Pacific Conference on Computer Graphics and Applications*. IEEE, pp. 427–428.

Palmer, A.N. (1991). "Origin and morphology of limestone caves". In: *Geol. Soc. Am. Bulleetin* 103, pp. 1–21.

Palmer, Arthur (Jan. 2003). "Speleogenesis in carbonate rocks". In: *Speleogenesis and Evolution of Karst Aquifers*, p. 11.

Palmstrom, Arild (2001). "Measurement and characterization of rock mass jointing". In: *In-situ characterization of rocks*. Rotterdam, pp. 49–97.

Parberry, Ian (2015). "Modeling Real-World Terrain with Exponentially Distributed Noise". In: *Journal of Computer Graphics Techniques (JCGT)* 4.2, pp. 1–9.

Pardo-Iguzquiza, E.a, Peter A. Dowd, Xu Chaoshui, J. J. Duran-Valsero, Eulogio Pardo-Igúzquiza, Chaoshui Xu, and Juan José Durán-Valsero (2012). "Stochastic simulation of karst conduit networks". In: *Advances in Water Resources* 35, pp. 141–150. ISSN: 03091708.

Paris, Axel, Eric Galin, Adrien Peytavie, Eric Guérin, and James Gain (2018). "Amplification de Terrain avec des caractéristiques implicites 3D". In: *JFIG*.

— (2019a). "Terrain Amplification with Implicit 3D Features". In: *ACM Transactions on Graphics* 38.5, 147:1–147:15.

Paris, Axel, Eric Guérin, Adrien Peytavie, Pauline Collon, and Eric Galin (2021). "Synthesizing Geologically Coherent Cave Networks". In: *Computer Graphics Forum* 40.7, pp. 277–287.

Paris, Axel, Adrien Peytavie, Eric Guérin, Oscar Argudo, and Eric Galin (2019b). "Desertscape Simulation". In: *Computer Graphics Forum* 38.7, pp. 47–55.

Paris, Axel, Adrien Peytavie, Eric Guérin, Jean-Michel Dischler, and Eric Galin (2020). "Modeling Rocky Scenery using Implicit Blocks". In: *The Visual Computer* 36.10, pp. 2251–2261.

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Parulek, Julius and Ivan Viola (2012). "Implicit representation of molecular surfaces". In: *2012 IEEE Pacific Visualization Symposium*, pp. 217–224.

Pasko, A., V. Adzhiev, A. Sourin, and V. Savchenko (1995). "Function representation in geometric modeling: concepts, implementation and applications". In: *The Visual Computer* 11.8, pp. 429–446. ISSN: 1432-2315.

Passos, Vladimir Alves dos and Takeo Igarashi (2013). "LandSketch: A First Person Point-of-View Example-Based Terrain Modeling Approach". In: *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. ACM.

Peakall, J., Bridget Mccaffrey, and B. Kneller (2000). "A Process Model for the Evolution, Morphology, and Architecture of Sinuous Submarine Channels". In: *Journal of Sedimentary Research* 70, pp. 434–448.

Perlin, K. and E. M. Hoffert (1989). “Hypertexture”. In: *SIGGRAPH Computer Graphics* 23.3, pp. 253–262.

Peytavie, A., E. Galin, J. Grosjean, and S. Merillou (2009a). “Procedural Generation of Rock Piles using Aperiodic Tiling”. In: *Computer Graphics Forum* 28.7, pp. 1801–1809.

Peytavie, Adrien, Thibault Dupont, Eric Guérin, Yann Cortial, Benes Benes, James Gain, and Eric Galin (2019). “Procedural Riverscapes”. In: *Computer Graphics Forum* 38.7, pp. 35–46.

Peytavie, Adrien, Éric Galin, Stephane Mérillou, and Jérôme Grosjean (2009b). “Arches: A Framework for modeling complex terrains”. In: *Computer Graphics Forum* 28.2, pp. 457–467.

Pirk, Sören, Till Niese, Torsten Hädrich, Bedrich Benes, and Oliver Deussen (2014). “Windy Trees: Computing Stress Response for Developmental Tree Models”. In: *ACM Transactions on Graphics* 33.6, 204:1–204:11.

Posamentier, Henry W. and Venkatarathnan Kolla (2003). “Seismic Geomorphology and Stratigraphy of Depositional Elements in Deep-Water Settings”. In: *Journal of Sedimentary Research* 73.3, pp. 367–388.

Pyrcz, M.J., J.B. Boisvert, and C.V. Deutsch (2009). “ALLUVSIM: A program for event-based stochastic modeling of fluvial depositional systems”. In: *Computers and Geosciences* 35.8, pp. 1671–1685.

Pytel, Alex and Stephen Mann (2015). “Procedural Modeling of Cave-like Channels”. In: *Journal of Computer Graphics Techniques* 4.2, pp. 10–29.

Reiner, Tim, Gregor Mückl, and Carsten Dachsbacher (2011). “Interactive Modeling of Implicit Surfaces Using a Direct Visualization Approach with Signed Distance Functions”. In: *Computer & Graphics, Proceedings of Shape Modeling International* 35.3, pp. 596–603. ISSN: 0097-8493.

Reinfelds, Ivars and Paul Bishop (Jan. 1998). “Palaeohydrology, palaeodischarges and palaeochannel dimensions: Research strategies for meandering alluvial rivers”. In: pp. 27–42.

Rongier, Guillaume, Pauline Collon, and Philippe Renard (2017). “A geostatistical approach to the simulation of stacked channels”. In: *Marine and Petroleum Geology* 82, pp. 318–335.

Rongier, Guillaume, Pauline Collon-Drouaillet, and Marco Filipponi (2014). “Simulation of 3D karst conduits with an object-distance based method integrating geological knowledge”. In: *Geomorphology* 217, pp. 152–164.

Rosgen, D. L. (1994). “A classification of natural rivers”. In: *Catena* 22, pp. 169–199.

Roudier, P., B. Peroche, and M. Perrin (1993). “Landscapes Synthesis Achieved through Erosion and Deposition Process Simulation”. In: *Computer Graphics Forum* 12.3, pp. 375–383.

Schmidt, R., B. Wyvill, M. C. Sousa, and J. A. Jorge (2006). “ShapeShop: Sketch-Based Solid Modeling with BlobTrees”. In: SIGGRAPH ’06.

Scott, Joshua J. and Neil A. Dodgson. (2021). “Example-based terrain synthesis with pit removal”. In: *Computers and Graphics* 99, pp. 43–53.

Sellán, Silvia, Noam Aigerman, and Alec Jacobson (2021). “Swept Volumes via Spacetime Numerical Continuation”. In: *ACM Transactions on Graphics* 40.4, 55:1–11.

Sitzmann, Vincent, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein (2020). “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 7462–7473.

Skorkovská, Věra, Ivana Kolingerová, and Bedrich Benes (2015). “Hydraulic Erosion Modeling on a Triangular Mesh”. In: *Surface Models for Geosciences*. Springer International Publishing, pp. 237–247.

Skorkovská, Vera, Ivana Kolingerová, and Petr Vanecek (2019). “A Unified Curvature-driven Approach for Weathering and Hydraulic Erosion Simulation on Triangular Meshes”. In: *VISIGRAPP*.

Slingerland, R. and Norman Smith (Apr. 2004). “River avulsions and deposits”. In: *Annual Review of Earth and Planetary Sciences* 32, pp. 257–285.

Stam, Jos and Ryan Schmidt (2011). “On the Velocity of an Implicit Surface”. In: *ACM Transactions on Graphics* 30.3.

Stolte, Nilo (2002). "Infinite Implicit Replication: Case Study for Voxelizing and Representing Cyclical Parametric Surfaces Implicitly." In: Shape Modeling International'02, pp. 105–111.

Sugihara, Masamichi, Brian Wyvill, and Ryan Schmidt (2010). "WarpCurves: A tool for explicit manipulation of implicit surfaces". In: *Computers & Graphics* 34.3, pp. 282–291.

Šťava, Ondřej, Bedřich Beneš, Matthew Brisbin, and Jaroslav Křivánek (2008). "Interactive Terrain Modeling Using Hydraulic Erosion". In: *Proc. Symposium on Computer Animation*, pp. 201–210.

Sylvester, Zoltán, Paul Durkin, and Jacob A. Covault (Feb. 2019). "High curvatures drive river meandering". In: *Geology* 47.3, pp. 263–266.

Tasse, Flora Ponjou, Arnaud Emilien, Marie-Paule Cani, Stefanie Hahmann, and Adrien Bernhardt (2014). "First Person Sketch-based Terrain Editing". In: *Proceedings of Graphics Interface*, pp. 217–224.

Tasse, Flora Ponjou, James Gain, and Patrick Marais (2012). "Enhanced Texture-Based Terrain Synthesis on Graphics Hardware". In: *Computer Graphics Forum* 31.6, pp. 1959–1972.

Teoh, Soon Tee (2009). "RiverLand: An Efficient Procedural Modeling System for Creating Realistic-Looking Terrains". In: *Proc. International Symposium on Advances in Visual Computing*. Las Vegas, USA: Springer, pp. 468–479.

Tricard, Thibault, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre (July 2019). "Procedural Phasor Noise". In: *ACM Transactions on Graphics* 38.4, Article No. 57:1–13.

Truong, Nghia, Cem Yuksel, and Larry Seiler (2020). "Quadratic Approximation of Cubic Curves". In: *Proc. ACM Comput. Graph. Interact. Tech.* 3.2.

Tsoar, Haim (1983). "Wind Tunnel Modeling of Echo and Climbing Dunes". In: *Eolian Sediments and Processes*. Vol. 38. Elsevier, pp. 247–259.

Vanek, Juraj, Bedřich Beneš, Adam Herout, and Ondřej Šťava (2011). "Large-Scale Physics-Based Terrain Editing Using Adaptive Tiles on the GPU". In: *Computer Graphics and Applications* 31.6, pp. 35–44.

Villanueva, Alberto Jaspe, Fabio Marton, and Enrico Gobetti (2017). "Symmetry-aware Sparse Voxel DAGs (SSVDAGs) for compression-domain tracing of high-resolution geometric scenes". In: *Journal of Computer Graphics Techniques (JCGT)* 6.2, pp. 1–30.

Viseur, Sophie, Johan Jouves, Arnaud Fournillon, Bruno Arfib, and Yves Guglielmi (2014). "3D stochastic simulation of caves : application to Saint-Sébastien case study (SE , France)". In: *Karstologia* 64, pp. 17–24.

Ward, A. W. and R. Greeley (1984). "Evolution of the Yardangs at Rogers Lake, California". In: *GSA Bulletin* 95, pp. 829 –837.

Wei, Li-Yi, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk (2009). "State of the Art in Example-based Texture Synthesis". In: *Eurographics 2009 - State of the Art Reports*, pp. 1–25.

Werner, B. T. (1995). "Eolian dunes: Computer simulations and attractor interpretation". In: *Geology*.

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

Williams, Garnett P. (1986). "River meanders and channel size". In: *Journal of Hydrology* 88.1, pp. 147–164.

Worley, Steven (1996). "A Cellular Texture Basis Function". In: SIGGRAPH '96, pp. 291–294.

Wyvill, Brian, Andrew Guy, and Éric Galin (1999). "Extending the CSG Tree - Warping, Blending and Boolean Operations in an Implicit Surface Modeling System". In: *Computer Graphics Forum* 18.2, pp. 149–158.

Wyvill, Geoff, Craig McPheeters, and Brian Wyvill (Aug. 1986). "Data Structure for Soft Objects". In: 2, pp. 227–234.

Yan, Xiao, Yun-Tao Jiang, Chen-Feng Li, Ralph R. Martin, and Shi-Min Hu (2016). "Multiphase SPH Simulation for Interactive Fluids and Solids". In: *ACM Transactions on Graphics* 35.4, 79:1–79:11.

Yuksel, Cem (2015). “Sample Elimination for Generating Poisson Disk Sample Sets”. In: *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)* 34.2, pp. 25–32. ISSN: 0167-7055.

Zanni, Cédric, Paul Bares, Ares Lagae, Maxime Quiblier, and Marie-Paule Cani (2012). “Geometric Details on Skeleton-based Implicit Surfaces”. In: *Eurographics Short Papers*. Cagliari, Italy, pp. 49–52.

Zhang, Deguo, Clement Narteau, Olivier Rozier, and Sylvain Courrech du Pont (2012). “Morphology and dynamics of star dunes from numerical modelling”. In: *Nature Geoscience* 5, pp. 463–467.

Zhang, Jian, Chen Li, Peichi Zhou, Changbo Wang, Gaoqi He, and Hong Qin (2022). “Authoring multi-style terrain with global-to-local control”. In: *Graphical Models* 119, p. 101122.

Zhao, Yiwei, Han Liu, Igor Borovikov, Ahmad Beirami, Maziar Sanjabi, and Kazi Zaman (2019). “Multi-Theme Generative Adversarial Terrain Amplification”. In: *ACM Transactions on Graphics* 38.6, pp. 1–14.

Zhou, Howard, Jie Sun, Greg Turk, and James M. Rehg (2007). “Terrain Synthesis from Digital Elevation Models”. In: *Transactions on Visualization and Computer Graphics* 13.4, pp. 834–848.

Part III

Appendix

Appendix A

Signed distance fields

In this section, we address the computation of Lipschitz constants (or bounds), denoted as λ, for primitives and operators. This allows for defining 1-Lipschitz signed distance functions, which in turn provides guarantees regarding fundamental queries such as sphere tracing. Recall that it is possible to derive a signed distance bound (Hart 1996) from any scalar function f through the use of its Lipschitz bound λ, by using $f(\mathbf{p})/\lambda$ as the field function.

A.1 Smooth union

In this part, we first aim at computing the Lipschitz constant of the smooth union operator. Let a, b denote two signed distance functions $\mathbb{R}^3 \to \mathbb{R}$. Let $d(a,b) = 1 - |b-a|/r$ if $|b-a| < r$ and 0 otherwise:

$$s(a,b) = \begin{cases} a & \text{if } a - b \leq r \\ b & \text{if } b - a \geq r \\ \min(a,b) - k\, r\, d(a,b)^3 & \text{otherwise} \end{cases}$$

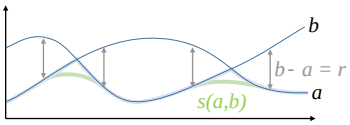


FIGURE A.1: *Smooth union.*

Consider the left and right derivatives around $b - a = r$, on the right $s'(a,b) = a'$ and on the left:

$$s'(a,b) = a' + 3\, k\, d^2(a,b)(b' - a')$$

The second derivative writes:

$$s''(a,b) = a'' - 6\, k\, d(a,b)(b' - a')^2/r + 3\, k\, d^2(a,b)(b'' - a'')$$

Therefore both left and right limits of s' and s'' when $b - a \to r$ are equal to a' and a'' respectively, and thus s is C^2 at this limit.

We also need to check differentiability when $a = b$. On the left where $a < b$, $s'(a,b) = a' + 3kd^2(b' - a')$ and on the right for $a > b$, $s'(a,b) = b' + 3kd^2(a' - b')$. If $k = 1/6$, as $d \to 0$ both derivatives tend to $(a' + b')/2$. Therefore, s is C^1. Moreover, we can bound the derivative. Let $\alpha = d^2/2$; if $a < b$ then

$s'(a,b) = (1-\alpha)b' + \alpha a'$ and if $a > b$ then $s'(a,b) = (1-\alpha)a' + \alpha b'$, with $d(a,b) < 1$. Therefore s' interpolates a' and b' and $\boxed{\lambda = \max(\lambda_a, \lambda_b).}$

The limits for the second derivative s'' are not the same. Therefore, the smooth union operator is only C^1. The same demonstration holds for the smooth intersection and difference operators.

A.2 Heightfield

Let $h : \mathbb{R}^2 \to \mathbb{R}$ denote an elevation function, we aim at constructing a signed distance bound $f_{\mathcal{H}}$ from h. Let λ denote the Lipschitz constant of h, which represents the maximum slope of the terrain, *i.e.* the upper bound of the norm of the gradient $\|\nabla h\|$. Then, the squared gradient of the vertical signed distance function $f(\mathbf{p}) = \mathbf{p}_z - h(\mathbf{p}_{xy})$ is:

$$\begin{aligned}\|\nabla f\|^2 &= 1 + (\partial h/\partial x)^2 + (\partial h/\partial y)^2 \\ &= 1 + \|\nabla h\|^2\end{aligned}$$

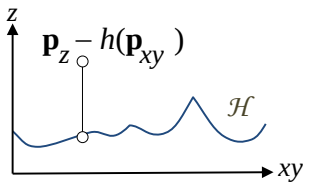


FIGURE A.2: *Vertical distance to a heightfield.*

Thus we have $\|\nabla f\|^2 < 1+\lambda^2$ and the bound $\|\nabla f\| < \sqrt{1+\lambda^2}$. Therefore, we define the 1-Lipschitz signed distance function $f_{\mathcal{H}}$ as:

$$\boxed{f_{\mathcal{H}}(\mathbf{p}) = \frac{\mathbf{p}_z - h(\mathbf{p}_{xy})}{\sqrt{1+\lambda^2}}}$$

This equation guarantees that $f_{\mathcal{H}}$ represents a signed distance bound to the surface of the terrain.

A.3 Turbulence

Let $n : \mathbb{R}^3 \to \mathbb{R}$ denote a noise function (such as gradient or value noise), we define the turbulence $t : \mathbb{R}^3 \to \mathbb{R}$ as:

$$t(\mathbf{p}) = \sum_{i=1}^{o} a_i n(\mathbf{p}/s_i)$$

We denote o the octave count, and a_i, s_i the amplitudes and scales at each octave, respectively. We aim at computing the Lipschitz constant of t, thus we define its gradient:

$$\nabla t(\mathbf{p}) = \sum_{i=1}^{o} \frac{a_i}{s_i} \nabla n(\mathbf{p}/s_i)$$

Let λ_n the Lipschitz constant of the noise, we can bound the norm of ∇t as:

$$\boxed{\|\nabla t\| \leq \lambda_n \sum_{i=1}^{o} \frac{a_i}{s_i}}$$

A.4 Noise displacement

Let f denote the 1-Lipschitz distance function for a subtree, we define the unary noise displacement operator with associated distance function $\tilde{f}$ as:

$$\tilde{f}(\mathbf{p}) = f(\mathbf{p}) + t(\mathbf{p})$$

With $t : \mathbb{R}^3 \to \mathbb{R}$ a turbulence. As f is 1-Lipschitz, a simple bound can be derived from the Lipschitz constant λ_t of the turbulence (see Appendix A.3):

$$\boxed{\|\nabla \tilde{f}\| \leq 1 + \lambda_t}$$

A.5 Perturbed skeletal primitives

Volumetric skeletal primitives are based on the Euclidean distance function $e(\mathbf{p})$ to a skeleton such as a point or a curve, and a radius r. The signed distance function is defined as:

$$f(\mathbf{p}) = e(\mathbf{p}) - r$$

In the case of perturbed noised-based skeletal primitives, we replace r by a function $\tilde{r} : \mathbb{R}^3 \to \mathbb{R}$ which computes the modified radius from the projection of $\pi : \mathbb{R}^3 \to \mathbb{R}^3$ on the skeleton and a noise:

$$\tilde{r}(\mathbf{p}) = r + a\, n(\pi(\mathbf{p})/l)$$

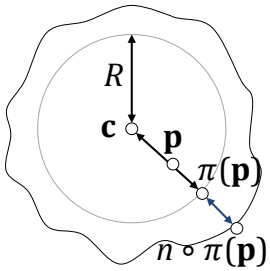


FIGURE A.3: *Perturbed sphere primitive.*

With $n : \mathbb{R}^3 \to \mathbb{R}$ the noise function, a the amplitude, and l the wavelength. The perturbed distance function $\tilde{f} = e(\mathbf{p}) - \tilde{r}(\mathbf{p})$ computes an anisotropic distance, thus there is a need to compute its Lipschitz constant λ to define a proper signed distance bound. Let us first compute the gradient of $\tilde{f}$:

$$\nabla \tilde{f}(\mathbf{p}) = \nabla e(\mathbf{p}) - \nabla \tilde{r}(\mathbf{p})$$

The norm of $\nabla e(\mathbf{p})$ is bounded by 1 (Euclidean distance). The gradient $\nabla \tilde{r}$ involves the Jacobian matrix $\mathbf{J}_\pi$ of the π:

$$\nabla \tilde{r}(\mathbf{p}) = \frac{a}{l} \mathbf{J}_\pi \nabla n(\pi(\mathbf{p})/l)$$

As demonstrated below, the norm of the Jacobian is bounded by 1 ($\|\mathbf{J}_\pi\| < 1$), thus the bound of the gradient is simply written as:

$$\|\nabla \tilde{r}\| < \frac{a}{l} \lambda_n$$

Finally, we can compute the Lipschitz bound of $\tilde{f}$:

$$\boxed{\|\nabla \tilde{f}\| < 1 + \frac{a}{l} \lambda_n}$$

As a corollary, we demonstrate that the Jacobian of the projection onto unit sphere has a norm bounded by one. Let $\pi : \mathbb{R}^3 - \{0\} \to \mathbb{R}^3 - \{0\}$ the projection on the unit sphere, *i.e.*, $\pi(\mathbf{p}) = \hat{\mathbf{p}} = \mathbf{p}/\|\mathbf{p}\|$. We want to compute the norm of the Jacobian $\|\mathbf{J}_\pi\|$. Noting that $\partial(x/\sqrt{x^2 + y^2 + z^2})/\partial x = (y^2 +$

$z^2)/(x^2+y^2+z^2)^{3/2}$ and $\partial(y/\sqrt{x^2+y^2+z^2})/\partial x = -xy/(x^2+y^2+z^2)^{3/2}$, the Jacobian matrix is defined as:

$$\mathbf{J}_\pi(x,y,z) = \frac{1}{(x^2+y^2+z^2)^{3/2}} \begin{pmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{pmatrix}$$

The norm of a matrix $\mathbf{A}$ is defined as the square root of the spectral radius of $\mathbf{A} \cdot \mathbf{A}^T$. Since $\mathbf{J}_\pi$ is symmetric, then $\mathbf{J}_\pi \cdot {\mathbf{J}_\pi}^T = {\mathbf{J}_\pi}^2$ is symmetric and all its eigenvalues are real, and the maximum of $\|\mathbf{A}^2\mathbf{u}\|$ will be achieved on the unit ball $\|\mathbf{u}\| = 1$.

We need to find the maximum absolute eigenvalue of $\mathbf{J}_\pi$. The characteristic polynomial is:

$$-\lambda^3 + 2\lambda^2(x^2+y^2+z^2) - \lambda(2x^2y^2 + 2x^2z^2 + y^4 + x^4 + 2y^2z^2 + z^4)$$

Factoring λ, noting that the second and last terms are $\|\mathbf{u}\|^2$ and $\|\mathbf{u}\|^4$ respectively and since $\|\mathbf{u}\| = 1$, we have:

$$-\lambda(\lambda^2 - 2\lambda + 1) = -\lambda(\lambda - 1)^2$$

Therefore 1 is the maximum root, and $\boxed{\|\mathbf{J}_\pi\| \leq 1.}$

A.6 L^p norm primitives

Let $p \geq 1$ and $f : \mathbb{R}^n \to \mathbb{R}$ defined as $f(\mathbf{x}) = \|\mathbf{x}\|_p$. From the triangle inequality and in finite space, we have:

$$\sup_{\mathbf{x}\neq\mathbf{y}} \frac{|\|\mathbf{x}\|_p - \|\mathbf{y}\|_p|}{\|\mathbf{x}-\mathbf{y}\|_2} \leq \sup_{\mathbf{x}\neq\mathbf{y}} \frac{\|\mathbf{x}-\mathbf{y}\|_p}{\|\mathbf{x}-\mathbf{y}\|_2} = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{x}\|_p}{\|\mathbf{x}\|_2} = \sup_{\|\mathbf{x}\|_2=1} \|\mathbf{x}\|_p$$

For $p = 2$ we get the Lipschitz bound 1. For $p > 2$, since $|\mathbf{x}_i|^p \leq |\mathbf{x}_i|^2$, we deduce $\sup_{\|\mathbf{x}\|_2=1} \|\mathbf{x}\|_p \leq 1$ thus the Lipschitz constant is also 1. For $p < 2$, by using Holder's inequality:

$$\left(\frac{|x_0|^p + ... + |x_n|^p}{n}\right)^{1/p} \leq \left(\frac{|x_0|^2 + ... + |x_n|^2}{n}\right)^{1/2}$$

If $\|x\|_2 = 1$, then $\|x\|_p \leq n^{1/p-1/2}$ (equality obtained when $\forall i \in [1,n]$, $x_i = 1/\sqrt{n}$). In the case $n = 3$ (three dimensional space), the Lipschitz constant is $3^{1/p-1/2}$. Thus, in conclusion:

$$\boxed{\text{If } p \geq 2,\ \|\nabla\|\mathbf{x}\|_p\| < 1 \qquad \text{If } p < 2,\ \|\nabla\|\mathbf{x}\|_p\| < n^{1/p-1/2}}$$

A.7 Sweep primitive

A 3D sweep primitive is defined as a general contour in the xy plane, that is then extruded along the z axis. The signed distance function to the contour is defined as $f(\mathbf{p},\mathcal{C}) = d(\mathbf{p},\mathcal{C})\,\delta(\mathbf{p},\mathcal{C})$, where $d(\mathbf{p},\mathcal{C})$ denote the (positive) Euclidean distance to $\mathcal{C}$ and $\delta(\mathbf{p},\mathcal{C})$ is the sign function. More precisely, δ computes the number of intersections between a line Δ passing through $\mathbf{p}$, and $\mathcal{C}$. The sign is finally computed depending on the parity of the number

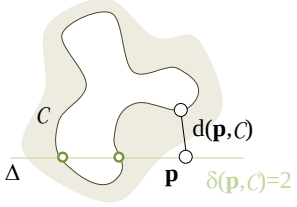


FIGURE A.4: *Signed distance to a contour* $d(\mathbf{p},\mathcal{C})$.

of intersections:

$$\delta(\mathbf{p}, \mathcal{C}) = \begin{cases} -1 & \text{if } \#\Delta \cap \mathcal{C} \text{ odd} \\ +1 & \text{otherwise.} \end{cases}$$

The 3D distance for a sweep primitive is then derived from $f(\mathbf{p}, \mathcal{C})$ according to the segment $\mathbf{ab}$. Let $\mathbf{u} = (\mathbf{b} - \mathbf{a})/\|\mathbf{b} - \mathbf{a}\|$ the unit vector of the segment $\mathbf{ab}$. Let $l = (\mathbf{p} - \mathbf{a}) \cdot \mathbf{u}$ denote the distance of the projection of $\mathbf{p}$ on the segment to $\mathbf{a}$. The signed distance to the sweep primitive is defined as:

$$f_{\mathcal{S}}(\mathbf{p}) = \sqrt{s(\mathbf{p})^2 + f(\mathbf{p}, \mathcal{C})^2} \quad s(\mathbf{p}) = \begin{cases} -l & \text{if } l < 0 \\ l - \|\mathbf{b} - \mathbf{a}\| & \text{if } l > \|\mathbf{b} - \mathbf{a}\| \\ 0 & \text{otherwise.} \end{cases}$$

As the contour is extruded on a segment and not a curve, the resulting signed distance function is 1-Lipschitz and continuous.

Appendix B

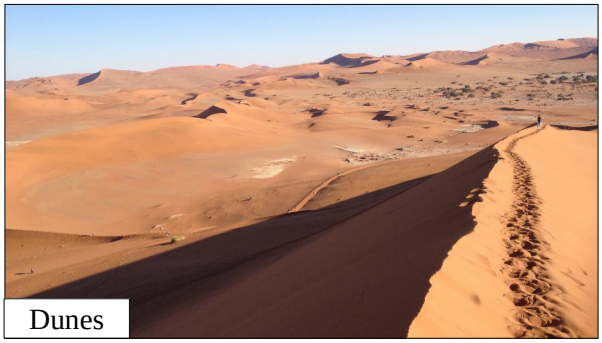
A visual dictionary of terrain landforms

Overhang

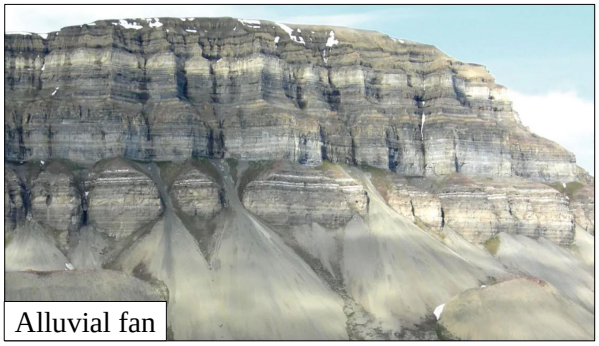
Sand ripples

Arche

River delta



Dunes

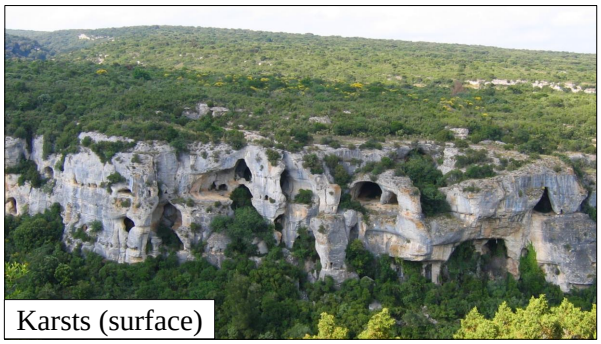


Alluvial fan


Mesa

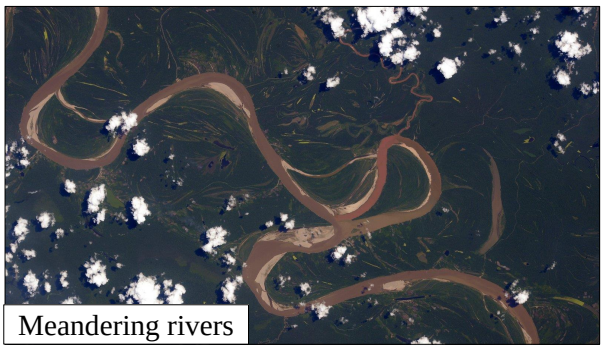

Hoodoos

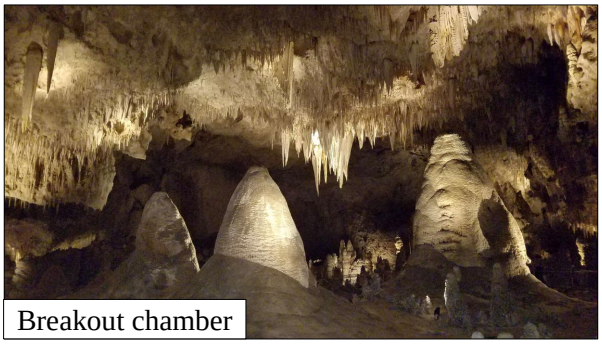

Rocky cliffs


Karsts (surface)


Gullies and ravines


Canyon

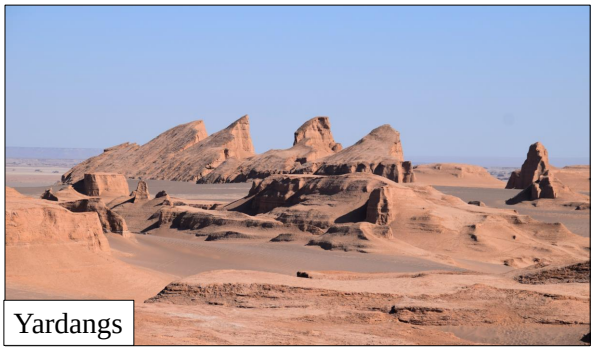

Meandering rivers


Breakout chamber

Ventifacts

Ventifacts

Reg (rock desert)



Yardangs

Fjord

Bastal columns

Overhangs

Cliff with tabular strata

Type	Source
River delta	Source
Arche	Source
Alluvial fan	Source
Mesa	Source
Hoodoos	Source
Hoodoos	Source
Rocky cliffs	Source
Karsts (surface)	Source
Gullies and ravines	Source
Canyon	Source
Meandering rivers	Source
Breakout chambers	Source
Ventifacts	Source
Reg (rock desert)	Source
Yardangs	Source
Fjord	Source

TABLE B.1: *Links to the source photographs of the different terrain landforms.*