HAL
open science

Layered Models for Large Scale Time-Evolving Landscapes

Guillaume Cordonnier

To cite this version:

Guillaume Cordonnier. Layered Models for Large Scale Time-Evolving Landscapes. Computer Science [cs]. Université Grenoble Alpes (France), 2018. English. NNT : . tel-01971947v1

HAL Id: tel-01971947

https://theses.hal.science/tel-01971947v1

Submitted on 7 Jan 2019 (v1), last revised 3 Apr 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Guillaume CORDONNIER

Thèse dirigée par **Marie-Paule CANI,** Professeur des universités, Ecole Polytechnique,
et codirigée par **Éric GALIN,** Professeur des universités, Université de Lyon

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Modèles à couches pour simuler l'évolution de paysages à grande échelle

Layered Models for Large Scale Time-Evolving Landscapes

Thèse soutenue publiquement le **6 décembre 2018,**
devant le jury composé de :

Monsieur François SILLION
Directeur de Recherche, Inria, Président
Monsieur Pierre POULIN
Professeur, Université de Montréal, Rapporteur
Monsieur Jean-Michel DISCHLER
Professeur, Université de Strasbourg, Rapporteur
Monsieur Bedrich BENES
Professeur, Purdue University, Examinateur
Monsieur Jean BRAUN
Professeur, GFZ German Research Center for Geosciences, Examinateur
Madame Marie-Paule CANI
Professeur, École Polytechnique, Directrice de thèse
Monsieur Eric GALIN
Professeur, Université de Lyon, Co-directeur de thèse

A ma famille.

Résumé

Le développement des nouvelles technologies permet la visualisation interactive de mondes virtuels de plus en plus vastes et complexes. La production de paysages plausibles au sein de ces mondes devient un défi majeur, en raison de l'importance des éléments de terrain et des écosystèmes dans la qualité et le réalisme du résultat. S'y ajoute la difficulté d'éditer de tels éléments sur des échelles spatiales et temporelles aussi vastes que peuvent l'être celles des chaînes de montagnes. Cette édition se fait souvent en couplant des méthodes manuelles et de longues simulations numériques dont le calibrage est complexifié par le nombre des paramètres et leur caractère peu intuitif.

Cette thèse propose d'explorer de nouvelles méthodes de simulation de paysages à grande échelle, avec pour objectif d'améliorer le contrôle et le réalisme des scènes obtenues. Notre stratégie est de fonder nos méthodes sur des lois éprouvées dans différents domaines scientifiques, ce qui permet de renforcer la plausibilité des résultats, tout en construisant des outils de résolution efficaces et des leviers de contrôles intuitifs.

En observant des phénomènes liés aux zones de compression de la croûte terrestre, nous proposons une méthode de contrôle intuitif de la surrection à l'aide d'une métaphore de sculpture des plaques tectoniques. Combinée avec de nouvelles méthodes efficaces d'érosion fluviale et glaciaire, celle-ci permet de sculpter rapidement de vastes chaînes de montagnes. Pour visualiser les paysages obtenus à échelle humaine, nous démontrons le besoin de combiner la simulation de phénomènes variés et de temporalités différentes, et nous proposons une méthode de simulation stochastique pour résoudre cette difficile cohabitation, que nous appliquons à la simulation de processus géologiques tels que l'érosion, jointe à la formation d'écosystèmes. Cette méthode est déclinée sur GPU et appliquée à la formation du manteau neigeux, en combinant des aspects au long cours (précipitations, changements d'état de l'eau) et des aspects dynamiques (avalanches, impact des skieurs).

Les différentes méthodes proposées permettent de simuler l'évolution de paysages à grande échelle, tout en accordant une attention particulière au contrôle. Ces aspects sont validés par des études utilisateur et des comparaisons avec des données issues de paysages réels.

Mots-Clés

Modélisation - Simulation - Phénomènes Naturels - Paysages - Terrains - Ecosystèmes

Abstract

The development of new technologies and algorithms allows the interactive visualization of virtual worlds showing an increasing amount of details and spatial extent. The production of plausible landscapes within these worlds becomes a major challenge, not only because the important part that terrain features and ecosystems play in the quality and realism of 3D sceneries, but also from the editing complexity of large landforms at mountain range scales. Interactive authoring is often achieved by coupling editing techniques with computationally and time demanding numerical simulation, whose calibration is harder as the number of non-intuitive parameters increases.

This thesis develops new methods for the simulation of large-scale landscapes. Our goal is to improve both the control and the realism of the synthetic scenes. Our strategy to increase the plausibility consists of building our methods on physically and geomorphologically-inspired laws: we develop new numerical methods, which, combined with intuitive control tools, improve user experience.

By observing phenomena triggered by compression areas within the Earth's crust, we propose a method for the intuitive control of the uplift based on a metaphor on the sculpting of the tectonic plates. Combined with new efficient methods for fluvial and glacial erosion, this allows for the fast sculpting of large mountain ranges. In order to visualize the resulting landscapes withing human sight, we demonstrate the need of combining the simulation of various phenomena with different time spans, and we propose a stochastic simulation technique to solve this complex cohabitation. This methodology is applied to the simulation of geological processes such as erosion interleaved with ecosystems formation. This method is then implemented on the GPU, combining long term effects (snowfall, phase changes of water) with highly dynamics ones (avalanches, skiers impact).

Our methods allow the simulation of the evolution of large scale, visually plausible landscapes, while accounting for user control. These results were validated by user studies as well as comparisons with data obtained from real landscapes.

Keywords

Modeling - Simulation - Natural Phenomena - Landscapes - Terrains - Ecosystems

Thanks

I first want to address a sincere thank to my supervisors, Marie-Paule Cani and Eric Galin, for their support and advises during these three years (and more!). You contributed in a large part to the successful completion of my PhD. I specially want to thanks Marie-Paule for opening me the opportunity to visit several international groups and create very interesting collaborations.

A special thanks too, for Jean Braun. Your advices and expertise where highly valuable for my work. Thanks for your invitation to visit your geomorphology group in Potsdam for two months. This was an amazing experience, and I believe that this visit, along with all our interactions, opened me new horizons and will continue to bring fruitful results.

I also want to thanks my collaborators, James, Bedrich, Pierre, Eric and Adrien. It was great working with you and I look forward our next collaborations.

I want to thanks my teammates, for patiently listening me exposing my scientific issues, for all our discussions and the good time we had together. Thanks Ulysse, Grégoire, Even, Maxime, Ameya, Pierre, Amélie, Sandra, Robin, Youna, Maguelonne, Julien, Antoine, Aarohi, Pierre-Luc, Tibor, Camille (Grenoble). Thanks Thomas, Pierre, Thibault, Pauline, Corentin, Dorian, Marie-Julie, Robin (Polytechnique). I also want to thanks the permanent researchers for their advises and fruitful discussions. Thanks Damien, Pooran, François, Mélina, Rémi, Stefanie, Jean-Claude, George-Pierre, Nicolas.

I have a special thought for all those who supported me and directed me during these three years and before. My friends, my former teachers, and all of you who had a considerable impact on my life.

A huge thanks to my family, my parents, Oliver and Clotilde. You built around me an amazing place to live and a taste for what I work for.

A last but definitively not least thank to my wife, Guillemette, for all your support, patience, and everything else. This PhD was a lot more comfortable for me than it was for you, and still, you were always here for me. Thanks to my children, François and Agathe, and you who I hope to meet someday, for all the happiness you are giving me.

Contents

1 Introduction **1**

1.1 Evolving landscapes in virtual environments . . . 1

1.1.1 Virtual environments . . . 2

1.1.2 Landscapes . . . 2

1.1.3 Large scale mountains evolution . . . 3

1.1.4 Authoring . . . 4

1.1.5 Previous work in landscape generation . . . 4

1.2 General overview . . . 5

1.2.1 Contributions . . . 5

1.2.2 Outline . . . 6

1.2.3 Publications . . . 7

2 State of the art on landscape modeling **9**

2.1 Terrain representation . . . 10

2.2 Procedural terrain generation: modeling the effects . . . 12

2.2.1 Fractal and noise-based terrains . . . 13

2.2.2 Focus on terrain features . . . 13

2.2.3 By example . . . 15

2.2.4 Methods from artificial intelligence . . . 16

2.3 Simulation of terrain evolution . . . 17

2.3.1 Small scale features . . . 17

2.3.2 Hydraulic erosion . . . 18

2.3.3 Thermal erosion . . . 19

2.3.4 Geologically based simulation . . . 20

2.3.5 Plate tectonics . . . 20

2.4 Ecosystems . . . 21

2.4.1 Modeling individual plants . . . 21

2.4.2 Methods from Ecology . . . 22
2.4.3 Lagrangian simulation . . . 23
2.4.4 Statistical synthesis . . . 24
2.5 Snow . . . 25
2.5.1 Lagrangian snow simulation . . . 25
2.5.2 Physically-based Eulerian heat transfer . . . 26
2.5.3 Procedural surface displacement . . . 27
2.5.4 Avalanches . . . 27
2.6 Full landscape authoring . . . 28
2.7 Conclusion . . . 29

I Large scale mountain formation 31

3 Combining uplift and fluvial erosion 33
3.1 Background and overview . . . 35
3.1.1 Geological background . . . 35
3.1.2 Algorithm overview . . . 37
3.2 Stream generation . . . 38
3.2.1 Stream graph initialization . . . 38
3.2.2 Stream tree computation . . . 39
3.2.3 Lake overflow . . . 39
3.3 Erosion . . . 43
3.4 Results . . . 44
3.4.1 Visual realism . . . 45
3.4.2 Rendering . . . 45
3.4.3 Performance . . . 46
3.4.4 Lake overflow . . . 47
3.4.5 Stream power erosion . . . 51
3.5 Conclusion . . . 56

4 Interactive manipulation of tectonically driven uplift 57
4.1 Overview . . . 60
4.1.1 Plate tectonics in geology . . . 60
4.1.2 Geologically-inspired interactive simulation . . . 61
4.2 Earth crust as a viscous material . . . 63
4.2.1 Moving plates creation . . . 63
4.2.2 Viscous compression . . . 64
4.2.3 Uplift from thickness changes . . . 65
4.3 Earth crust as layered sheets . . . 65

4.3.1 Folding of layered materials . . . 66
4.3.2 Procedural fold generation . . . 66
4.3.3 Uplift update from folds . . . 69
4.4 Terrain surface generation . . . 69
4.4.1 Interactive terrain generation . . . 69
4.4.2 Rock layers at the surface . . . 70
4.5 Implementation, results and discussion . . . 72
4.5.1 Architecture . . . 72
4.5.2 Qualitative and quantitative results . . . 74
4.5.3 Validation and discussion . . . 76
4.5.4 User study . . . 80
4.6 Conclusion . . . 80

5 Glacial erosion 83
5.1 Overview . . . 85
5.1.1 Glacial erosion in Geology . . . 85
5.1.2 Governing equations for glaciers . . . 86
5.1.3 Efficient simulation of glacial erosion . . . 87
5.1.4 Secondary erosion . . . 88
5.1.5 Main algorithm . . . 88
5.2 Ice flux propagation over the terrain . . . 89
5.2.1 Path graph computation . . . 90
5.2.2 Ice flux propagation . . . 90
5.3 Steady-state and erosion . . . 91
5.3.1 Computations at each iteration . . . 92
5.3.2 Convergence . . . 94
5.4 Debris flow, fluvial and hill slope erosion . . . 95
5.4.1 Debris flow and fluvial erosion . . . 95
5.4.2 Hill-slope erosion . . . 95
5.4.3 Interactions with glacial erosion . . . 97
5.5 Results and discussion . . . 97
5.5.1 Validation experiments . . . 98
5.5.2 Efficiency and speed . . . 103
5.5.3 Limitations . . . 105
5.6 Conclusion . . . 106

II Combining landscape simulation with medium scale phenomena 107

6 Joint simulation of vegetation and erosion 109
6.1 Method overview . . . 111

6.1.1 Layered landscape model . . . 111
6.1.2 Simulation . . . 112
6.1.3 Control . . . 114
6.2 Geomorphological events . . . 115
6.2.1 Rainfall and running water . . . 115
6.2.2 Temperature . . . 116
6.2.3 Lightning . . . 117
6.2.4 Gravity . . . 118
6.2.5 Fire . . . 118
6.3 Ecosystem events . . . 119
6.4 Implementation . . . 122
6.5 Results and discussion . . . 122
6.6 Conclusion . . . 132

7 Dynamic snow cover evolution 133
7.1 Overview . . . 135
7.1.1 Simulation method . . . 136
7.1.2 Categories of events . . . 138
7.2 Environmental conditions . . . 138
7.2.1 Temperature . . . 138
7.2.2 Wind . . . 140
7.3 Snow cover . . . 142
7.3.1 Snowfall . . . 142
7.3.2 Snow state changes . . . 143
7.3.3 Diffusion of powdery snow . . . 144
7.3.4 Wind transport . . . 144
7.4 Interactive phenomena . . . 145
7.4.1 Avalanches . . . 146
7.4.2 Ski tracks . . . 147
7.5 Implementation . . . 149
7.6 Results and discussion . . . 152
7.7 Conclusion . . . 156

8 Conclusion 159
8.1 Summary . . . 159
8.2 Future work . . . 160

Chapter 1

Introduction

Contents

1.1 Evolving landscapes in virtual environments 1
1.1.1 Virtual environments 2
1.1.2 Landscapes 2
1.1.3 Large scale mountains evolution 3
1.1.4 Authoring 4
1.1.5 Previous work in landscape generation 4
1.2 General overview 5
1.2.1 Contributions 5
1.2.2 Outline 6
1.2.3 Publications 7

Exploring and dreaming - these two major engines that propelled mankind to the top of evolution have reached another dimension within our new digital age. The generation and manipulation of huge virtual environments became possible with the increase of computational power and the discovery of new algorithms, while the development of new visualization technologies and of virtual reality makes them more accessible to both experts and the general public. This leverages our ability to both explore and manipulate imaginary worlds, which is needed in the entertainment industry, and also to study simplified versions of our own world at any scale, which considerably extends the possible research outreach in natural sciences. The temporal evolution of a virtual environment can have significant importance in both the generation of its current state and in the study of relationships between its successive states. In this thesis, we use the simulation of this temporal evolution as a key element to improve both the plausibility of large scale landscapes and the efficiency of authoring tools.

1.1 Evolving landscapes in virtual environments

Landscapes form the foundations of natural virtual environments. In this thesis, we explore new solutions for creating and authoring landscapes and their evolution over large time periods.

1.1.1 Virtual environments

With the increasing power of computers and the never ending demand of end-consumers for digital content, the need for virtual environments has dramatically increased. These are used as a context for the exploration and the manipulation of virtual objects. In the entertainment industry, virtual environments can be used as a background for a story, and in particular be instrumented to direct the context of that story: time period, mood, places, characters, etc. This environment has a fundamental role in defining limits and points of interest in interactive stories, as well as orienting the training proposed by serious games. In the manufacturing industry or architecture, virtual environments are often used to highlight a product, for commercial or informational purposes. Natural sciences generate approximated versions of our world to validate their model for natural laws, and human sciences use virtual environments as a research context, for example to understand the panel of behaviors observed in reaction to emergency situations. In this thesis, we tackle the problem of the efficient and controllable simulation of the temporal evolution of consistent large scale mountainous landscapes for virtual environments. In the following, we will define these notions and the subsequent challenges.

1.1.2 Landscapes

Landscapes play a fundamental role in virtual environments. Although sometimes considered as a background element, they usually cover most of the sight space, and have a strong influence on the general mood of the virtual world. The geometry of landscapes sets the limits of the environment, and directs the explorer's movements. It also influences the placement of objects, which is crucial for both storytelling applications and for the commercial promotion of products.

The term *virtual landscape* is generic and encompasses a wide range of definitions. In this thesis, we restrict our scope to elements and inanimate life forms. A landscape can be separated into several entities:

- **The terrain**, which is an essential part of the landscape as it is used to ground virtual objects. Several representations exist for modeling the terrain and its surface is used for visualization and as a support of the virtual environment. It can be internally segmented into several groups of different materials (usually some variety of rock types).
- **Water bodies** are then placed in accordance to the terrain surface. These are large areas of connected regions filled with water, either in a relative rest state (lakes, seas), either in a dynamic state (rivers). Resulting from the advection of precipitations, they also result in solid snow cover and glaciers in cold climates.
- **Vegetation and rocks** are separated into multiple plants (trees, shrubs, grass) or mineral elements (stones and pebbles) that are scattered over the terrain surface. These objects are important for the visual quality of the generated landscapes. We also show that they exhibit deep and effective interactions with many of the natural events responsible for the transformation of a landscape.

Challenge: A perennial challenge in the generation of landscapes is its realism, which is a key ingredient to the immersion required by virtual environments. Many features are hard to

generate accurately: dendritic erosion channels, vegetation and rock placements with respect to slope and environmental resources, or large scale shapes such as the distribution of valleys and ridges. We separate two factors that impact this realism. *Visual plausibility* is directly evaluated by the human visual system and may depend on the observer's experience. The underlying neurological mechanisms are under debate, which makes visual plausibility hard to quantify objectively. *Physical consistency* is generally not detected at first sight but has a specific importance as it impacts the plausibility perceived in accordance with the user's knowledge: the water flows downstream, deciduous and conifer trees are sorted by increasing altitude, etc.

1.1.3 Large scale mountains evolution

Landscapes show a wide range of pattern sizes; for some viewpoints, large mountain ranges may even cover the same area on the screen than small details. Therefore, when considering the targeted space and time scale, most generations methods have to choose a trade-off between the precision of details and the extent of the generated landscape. Interestingly, there is a correlation between longevity and size of natural features. This duality between spatial and temporal scales induces a natural choice of time windows when simulating landscape formation given the targeted spatial extent. This observation allows us to consider both space and time indistinguishably in the following, in a unified scale concept. In this thesis, we propose a multi-resolution approach, dividing the scale range into three main categories, related to the lifetime of the observed landscape features.

At *large scale*, geological measures are considered. Landscapes result from millions of years of earth deformation: the spatial domain encompasses the whole mountain range, where the main features have a characteristic scale of several kilometers. Typical features include large mountains or valleys carved by glaciers or by the long action of the water flow. The second category encompasses a *medium* scale, where features are the distributions of rocks, plants or snow cover, in a maximum extent of 10 km, and where the minimal detail size is above 1 to 10 m. This scale is accurate enough to show the main rock structures and to define plant densities, shaped by events occurring over a few centuries. We must mention a *smaller scale* with centimeter wide details. This includes the precise positioning and details of vegetation, ground elements, or rock texturing. This scale is generally handled when rendering the landscape, and thus is out of the scope of this thesis.

Tackling landscape generation at large scales open the possibility for modeling the natural forces resulting in the formation mountainous ranges. We orient our study to the solution of this particular problem, because the visual appeal of mountainous regions is sought for in virtual worlds, and because the complexity of the underlying physical events makes it an interesting open problem to investigate.

Since we do not consider human and animal activities in our definition, landscapes are the result of slow natural processes, hardly observable in a human life span. Still, modeling the evolution of the landscape over large, geological time scales have some important use cases. In particular, a landscape at present day is the result of its past evolution, and plausibly reproducing it can be achieved by accurately simulating the underlying natural phenomena. Simulation approaches for landscape generation use the succession of natural events to progressively shape the landscape. The resulting versions of the same landscapes at different state of evolution can be used in many applications, for example to show the

outcome of different climatic or geological conditions on similar initial settings.

Challenge: The problem of simulating large scale landscape evolution on geological times has received little attention in Computer Graphics. First, given the large temporal scale, it is hard to obtain a simulation interactive enough to enable parameters exploration and feature placement. Second, a large range of natural processes are responsible of landscape evolution, it is necessary to carefully select which of them have physical meaning at the considered scales and how they interact.

1.1.4 Authoring

Many applications require a fine tuning of the virtual environment to fulfill the desired purpose. Without any automatic generation tool for both a plausible storytelling and the corresponding virtual world, virtual landscapes have to be carefully sculpted by artists. This is still done exhaustively by hand, a workload that increases dramatically with the growth of virtual contents. In production pipelines, the user starts to sculpt the terrain surface with low level primitives (usually noise). Then, a complex simulation adds physically based enhancements such as erosion and can last for hours for large $10,000^2$ cells terrains. Vegetation and natural details are coarsely placed on the ground with respect to simple surface properties such as slope and altitude, where they need to be manually adjusted to some more plausible and interesting locations. Although providing maximal control, this work-flow comes at the cost of a very low productivity. Thus, any new landscape generation technique should incorporate higher lever user control that both allows for fast and intuitive editing and guarantee the plausibility of the final result.

Challenge: Authoring terrains brings the same difficulty than many other applications in Computer Graphics. Users are usually seeking for a tool that enables them to quickly shape what they have in mind, and in the meantime allows them to edit even the more detailed features. Authoring also gains in complexity when coupled with the two other challenges: the tools must enforce the desired (preferably tunable) level of plausibility, and be compatible with large scale settings.

1.1.5 Previous work in landscape generation

Two main categories of generation methods have coexisted in the landscape generation literature. They targeted either the generation of the *effects*, *i.e.,* the observed features of the final landscape, either the generation of the *causes*, *i.e.,* the forces that trigger the long term evolution of the landscape. All these methods have tried to solve some of the challenges mentioned above: large scale, controllability and plausibility of both visual and physical features.

Modeling the effects of processes that generate landscapes can be achieved by reproducing particular patterns or features observed in our environment. This method has a long history in Computer Graphics, with various approaches. Fractal noise combination (Ebert et al. 2002) instantaneously generates infinite terrain surface at the cost of realism and controllability. Some methods consider the edition of specific terrain features (Génevaux et al. 2015; Hnaidi et al. 2010). Observations from hydrology (Emilien, Poulin, et al. 2015; Génevaux et al. 2013) are used to build consistent terrain surface and water bodies. Designed for

controllability and sometime large scale applications, these methods often suffer from their specific focus, which sacrifice the overall plausibility. Statistical analysis on landscape object distributions (Emilien, Vimont, et al. 2015) or on existing terrain data (Gain et al. 2015; Guérin et al. 2017), uses the complexity of real world data to reproduce plausible and controllable landscapes. The main issues come at larger scales, where it is hard to enforce physical consistency.

Modeling the causes of landscape formation is inspired from simulation methods commonly used in Natural Sciences, where specific forces and laws are used to transform an initial setting. In Natural Sciences, the goal is to validate the laws or to obtain parameters from acquired data, whereas the goal in Computer Graphics is to generate plausible data to augment the user's immersion in a virtual environment. Simulation techniques for ecosystem synthesis take inspiration from ecology and model plants as a resource-dependent density distribution (Lane et al. 2002), or as competitive individual agents (Bradbury et al. 2015). Terrains are also used as input of simulation methods that mainly address the problem of surface enhancement from erosion. Two main types of erosion are usually considered. Cliff weathering is averaged into a *thermal erosion* formulation on large scale landscapes (Musgrave et al. 1989) or detailed on a rock by rock basis (Ito et al. 2003). Various formulations of *hydraulic erosion* have been used, simulating water and sediment flow by using either Eulerian (Beneš et al. 2002; Benes et al. 2006; Musgrave et al. 1989; Pytel et al. 2013) or Lagrangian (Krištof et al. 2009; Kurowski 2012) fluid simulation, which triggers a coupled bedrock erosion.

To our knowledge, no attempt was previously made to simulate the formation of mountain ranges. Controllability is often the weakest point of simulation methods, although some efforts has been made to improve it. Indeed, these methods usually involve many parameters which are tuned through several trials and errors. This process is tedious and demanding, because of the large computation times needed by the simulation or because of the large number of time steps. But when the right set of parameters has eventually been found, the results are both visually plausible and physically consistent. Simulation techniques, especially of hydraulic erosion, generally fail to generalize at large scales because of the precise physical simulations used to model the water flow, which induce the need for a fine simulation grid leading to unpractical computation times.

1.2 General overview

This thesis proposes a series of methods for simulating the temporal evolution of large scale multi-layered landscapes. We cover several complementary problems, from the generation of large scale mountain ranges to the interleaving of smaller scale natural processes.

1.2.1 Contributions

Our contribution addresses the main challenges listed above: large scale landscape simulation, plausibility and control of the result.

Landscape simulation. A large part of this research was conducted in collaboration with geomorphologists, specialized in the simulation of *large scale* earth surface processes. These

collaborations enabled us to understand and reuse state of the art knowledge about the geological laws, which are both specifically designed to the considered scale (mountain ranges), and adapted to the needs of Computer Graphics. We also address the cohabitation issue between events in a large range of time scales, such as erosional effects and vegetation lifetime. We improve the *plausibility* of our results by rooting each of the simulated effect in laws from Natural Sciences, even if procedurally simplified for efficiency purpose. We also validate our results through user studies and through a phenomenological verification: we select some of the patterns which are known to be the result of the modeled process, and we check if it emerges from our simulations. Lastly, we propose to use tectonic forces as a high level *user control* tool for sculpting mountains. We also investigate the use of an editable timeline, where both materials and events could be input at any point in space and time, and later refined.

Multi-layered representation. These contributions benefit from the use of a multi-layered representation of landscapes. At the large considered scales, fully volumetric structures such as cliffs, overhangs or arches are barely visible, which implies that our landscapes can benefit from a planar parametrization: the different properties of the landscape are regularly sampled on a 2D plane. The sampling is usually performed over a regular grid, although other distributions are possible such as triangular irregular network, commonly used in geolomorphology and geography. Many different properties of the landscapes are stored at the observed points, the most important being the surface elevation (*heightmap*). We introduce other elements such as ordered layers of rocks, vegetation density, glacier thickness, or any space varying simulation properties. This representation serves several purposes: the multiple layers embed a part of volumetric information without paying the cost of a complete 3D data structure. The fixed, regular sampling allows for a fast access of the different properties at a given location, and is by definition, directly compatible with Eulerian physical simulation methods. When the sampling is performed over a regular grid, the output is also straightforward to plug in existing rendering software or game engines.

1.2.2 Outline

After visiting some of the methods previously introduced in Computer Graphics for landscape simulation, we organize the contributions of this thesis in two parts, separated by the scales at which we consider landscapes.

Large scale mountain formation. In a first part, landscapes are considered at geological scales, enabling the formation of whole mountain ranges from the combination of fluvial erosion and tectonic uplift (Chapter 3). Fluvial erosion is modeled thanks to the Stream Power Law, borrowed from Geomorphology literature. We use an efficient implementation that allows for large time steps. The computation of space varying uplift is then refined according to geological knowledge on plate tectonics, considering successive layers at progressively refined scales: global crust compression, subsequent folding and erosion. This results in a sculpting tool that allows users to interactively shape mountain ranges in reaction to tactile gestures mimicking tectonic forces (Chapter 4). In Chapter 5 we introduce another erosion law that takes into account the abrasion of glaciers, fundamental at low latitude or

high altitude mountain ranges. We propose a new simulation method for the combination of the steady state of glaciers and the implied erosion on large time steps.

Combining landscape simulation with medium scale phenomena. In a second part, we narrow down the scale to consider the interleaving of various medium scale phenomena. We propose a new simulation method for the interactive authoring of landscape through the combination of many natural processes: vegetation life, fire, lightnings, hydraulic and thermal erosion (Chapter 6). This stochastic simulation is based on a segmentation of the different considered processes into atomic events, which are triggered randomly and are simulated along a simple path. This framework is easy to implement, largely extensible, and proposes a user control in space and time of both the events and the processed materials thanks to a timeline. We extend this method with a GPU simulation and adapt it to the generation of the snow cover (Chapter 7). This includes the handling of snow phase, wind, and measures of snow stability. The simulation is augmented with shorter term dynamic effects: avalanches and skiers. Chapter 8 concludes this thesis and explores some perspectives opened by the presented research.

1.2.3 Publications

The work presented in this thesis has been subject to related publications:

Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Eric Guérin (2016). "Large scale terrain generation from tectonic uplift and fluvial erosion". In: *Computer Graphics Forum* 35.2, pp. 165–175 (Chapter 3)

Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani (2017). "Authoring landscapes by combining ecosystem and terrain erosion simulation". In: *ACM Transactions on Graphics* 36.4, pp. 134:1–134:12 (Chapter 6)

Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin (2018). "Sculpting mountains: Interactive terrain modeling based on subsurface geology". In: *IEEE Transactions on Visualization and Computer Graphics* 24.5, pp. 1756–1769 (Chapter 4)

Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani (2018). "Interactive generation of time-evolving, snow-covered landscapes with avalanches". In: *Computer Graphics Forum* 37.2, pp. 497–509 (Chapter 7)

Two publication are under review, one related to Chapter 5 on glacial erosion, and an extension of the local minima problem for flow distribution mentioned in Chapter 3.

We collaborated to another work, that greatly inspired the vegetation part of Chapter 6, but is not detailed in this thesis: James Gain, Harry Long, Guillaume Cordonnier, and Marie-Paule Cani (2017). "EcoBrush: Interactive control of visually consistent large-scale ecosystems". In: *Computer Graphics Forum* 36.2, pp. 63–73.

Chapter 2

State of the art on landscape modeling

Contents

2.1 Terrain representation **10**
2.2 Procedural terrain generation: modeling the effects **12**
2.2.1 Fractal and noise-based terrains 13
2.2.2 Focus on terrain features 13
2.2.3 By example 15
2.2.4 Methods from artificial intelligence 16
2.3 Simulation of terrain evolution **17**
2.3.1 Small scale features 17
2.3.2 Hydraulic erosion 18
2.3.3 Thermal erosion 19
2.3.4 Geologically based simulation 20
2.3.5 Plate tectonics 20
2.4 Ecosystems **21**
2.4.1 Modeling individual plants 21
2.4.2 Methods from Ecology 22
2.4.3 Lagrangian simulation 23
2.4.4 Statistical synthesis 24
2.5 Snow **25**
2.5.1 Lagrangian snow simulation 25
2.5.2 Physically-based Eulerian heat transfer 26
2.5.3 Procedural surface displacement 27
2.5.4 Avalanches 27
2.6 Full landscape authoring **28**
2.7 Conclusion **29**

Virtual landscapes have been studied extensively in the last decades (Natali et al. 2013; Smelik et al. 2014). In this chapter, we present an overview of these researches and we qualify them with respect to the three main challenges raised by this problem: plausibility, scale, and user control. First, we consider terrain generation, which has received the most attention in Computer Graphics. We present the different options for terrain representation, which condition the data storage for the whole landscape. We then separate methods that directly generate the *effects* of geomorphologic processes in the form of specific terrain features (Section 2.2), and the methods that simulate the terrain evolution after modeling the *causes* responsible for present day topography (Section 2.3). Subsequent sections focus on the generation of ecosystems (Section 2.4) and snow cover (Section 2.5). Only few methods address the problem of combining multiple of these aspects to improve the plausibility of complete landscapes, as described in Section 2.6.

2.1 Terrain representation

Several strategies are used to represent terrains in computer memory, so that they can be handled by generation algorithms. Choosing among one of these representation depends on the applications and more importantly on the targeted scale.

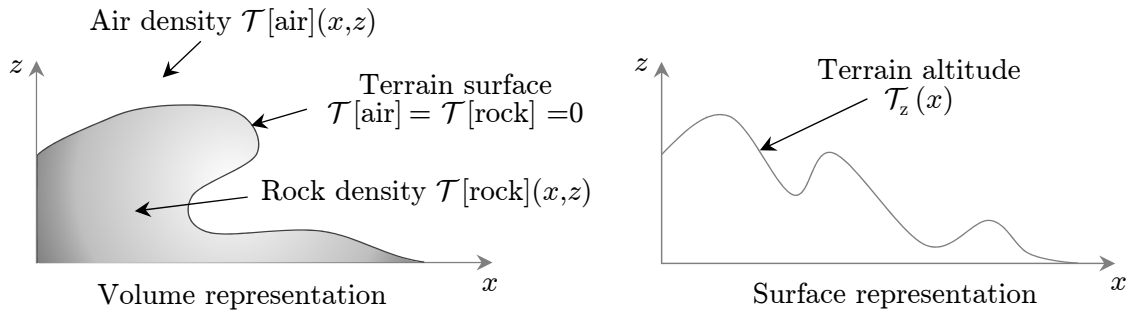


Figure 2.1: Side view (xz) of volume and surface representations. Volume representation (left): the terrain is defined by a density function for the different materials considered. Terrain surface is obtained implicitly at the junction of an interior material (here rock) and an exterior material (here air). When the surface is uniquely defined for each x, a surface representation can be extracted (right).

We introduce a formal definition for a terrain, from which be can derive the representation generally used in Computer Graphics. A terrain is characterized by a surface, but can also embed multiple materials, as well as complex topological structures as caves and overhangs. To account for this different volumetric aspect, we define a terrain thanks to a density function. For n different types of materials, including at least a terrain material and air, the function representing a terrain maps a position to a vector describing the density of each of the materials: $\mathcal{T} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$. Multiple definitions are valid for the density, a possibility is to associate values, ranging continuously from negative (absence of material) toward positive (presence of material). By assuming that the materials can not be mixed, impose the resulting density function to outputs a vector with a single non-zero element. The boundaries of a material with identifier k is then given by the solution of an implicit equation $\mathcal{T}[k] = 0$, and

the interface between two materials, especially the terrain surface in contact with a material representing the exterior (air, void, *etc.*) is obtained by the intersection of two such functions (Figure 2.1, left). This technique is an application of the general formulation of *implicit surfaces*, widely used in Computer Graphics to represent objects boundaries. The visualization of the terrain is generally obtained thanks to ray tracing or by converting it to a mesh representation (Lorensen et al. 1987; Wyvill et al. 1986). In some cases, an interface does not self-overlap when vertically projected, which means that the equation $\mathcal{T}_i(x, z) = \mathcal{T}_j(x, z) = 0$ solved for z has a unique solution, for all x in the horizontal terrain boundary. The surface thus follows an explicit representation, mapping a 2D horizontal position to an altitude: $\mathcal{T}_z : \mathbb{R}^2 \to \mathbb{R}$ (Figure 2.1, right).

Although some analytical formulations exist for this equation, terrains are generally shaped thanks to numerical computations, especially when simulations or user control are considered. To store the altitudes in computer memory, a discrete terrain representation is used: $(\mathcal{X}, \mathcal{T}_z(\mathcal{X}))$, where $\mathcal{X}$ is a set of horizontal 2D positions, and the altitude is interpolated between the discrete points. In many applications, $\mathcal{X}$ is chosen as the nodes of a regular grid $\mathcal{X} = \{(i\,dx, j\,dx), (i, j) \in [0, n-1] \times [0, m-1]\}$, where dx is the cell spacing and (n, m) the number of sampling points per dimension. This discretization, along with the associated elevation, is called *heighfield*. It has a small memory footprint because $\mathcal{X}$ can be deduced easily from dx, mn and n, and neighborhood lookup is straightforward.

Triangular Irregular Networks (Peucker et al. 1978) is an alternate way of storing altitude data at the nodes of a planar graph. The locations of the nodes are distributed randomly with some regularity in their spacing (blue noise), and the edges are chosen to form only triangles between nodes. At the cost of a higher memory consumption, this representation is straightforward to convert to a mesh and thus easily pluggable in any rendering or visualization engine. The irregularity of the sampled points makes it an interesting choice when distinct features have to be represented at various scales (especially in the case of highly varying surface curvature), and hides artifacts which are often observed in regular grids.

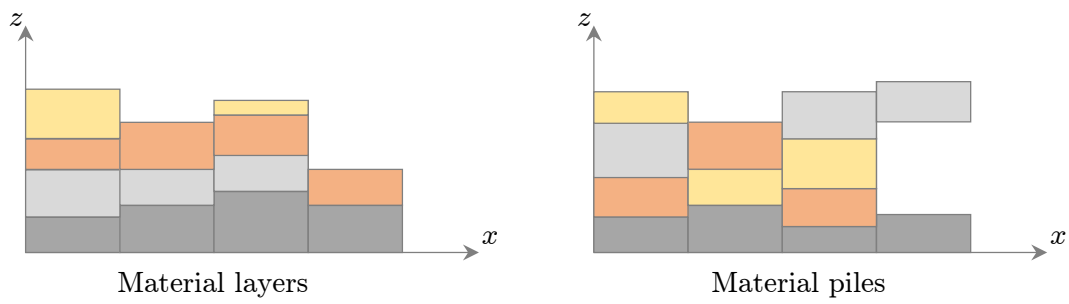


Figure 2.2: The thickness of multiple materials is stored in each cell in *layered* (left) or *pile* (right) representations. The main difference is that the ordering of the materials is shared by all cells in a layered representation, whereas it is free within the piles. Additional air blocks can be interleaved in the piles, enabling the representation of caves to caves and overhangs.

A semi volumetric extension was introduced by Benes et al. 2001 to handle multiple material layers thanks to a planar data structure (usually heightfields) for each interface between materials: $\mathcal{T}_z(\mathcal{X}) \in \mathbb{R}^l$, where l is the number of layers (Figure 2.2, left). The altitude values of a material can be stored relatively to the previous layer and thus embeds

a local thickness. We mainly use this representation in our methods, because of the large scale possibilities offered by planar representations and because the static setting of layers is well suited to represent geological materials. We extend on this by adding other types of layers, such as resources (moisture and illumination), or object densities (vegetation, rocks). A limitation of that representation is that the order of materials in a cell is prescribed by the global ordering of the layers.

Figure 2.3: Volumetric representation of terrain through piles of materials. Void layers are used to model arches and overhangs. (Peytavie et al. 2009).

More accurate volumetric solutions are also considered in the literature, generally only applicable to lower scale sceneries. Gamito et al. 2001 applies a 3D displacement $\mathcal{W} : \mathcal{X} \rightarrow \mathbb{R}^3$ on heightfields to enable overhangs. The resulting terrain is obtained by $\mathcal{T}_\mathcal{W} = \{(x, y, \mathcal{T}_z(x, y) + \mathcal{W}(x, y)), (x, y) \in \mathcal{X}\}$. This method is efficient in modeling precise cliffs, as long as the targeted result is topologically equivalent to a surface. An approach dual to the previously described layered representation consists in vertically stacking a pile of materials at each grid point (Peytavie et al. 2009): $\mathcal{T}_z(x, y) \in (\mathbb{R}, \mathcal{N})^{l(x,y)}, (x, y) \in \mathcal{X}$, where l(x, y) is the sample dependent number of elements in a pile and $\mathcal{N}$ is a set of material types. By using a specific void (or air) material, this method can represent arches and overhangs (Figure 2.2, left and Figure 2.3). Fully volumetric solutions require to associate materials to 3D positions, usually in *voxels* (cells of a regular 3D grid). Some optimizations alleviate the large storage cost of such structures, for example by using Sparse Voxel Octrees Laine et al. 2011. Although fundamental for the accurate representation of many close view natural features, we do not consider such precise representations. By considering terrain at large scale, we can neglect volumetric features, which enable us to capture mountain formation with the maximum possible horizontal extent.

Recent works propose an alternative to discrete representation. Génevaux et al. 2015 segment different atomic terrain features, called primitives, as leaves of a mathematical tree. Junctions in the tree embeds operators, explaining how the features combines, for example by blending, carving, or warping. Parsing the tree leads to an implicit function defining the terrain surface, which enables interactive edition and visualization thanks to ray-tracing. Editing the primitive tree is done either by hand, either by analyzing real data to produce both the primitives and the operator tree as proposed by Guérin, Digne, Peytavie, et al. 2016. These approaches are very convenient for generating large scale landscapes with a small memory footprint. As we demonstrate in Chapter 3, this representation can be coupled with a discrete one to add local details.

In this thesis, we choose a discrete planar layered representation. Planarity is justified by the large considered scale, discretization (with TINs or regular grid) is needed by the numerical simulations and the layers nicely fits to the natural ordering of geological structures.

2.2 Procedural terrain generation: modeling the effects

Several algorithms specialized at the generation of terrains are designed by observing terrain features as they are now, regardless of their past evolution. We call this approach *modeling the effects*. In particular, many methods use fractals to mimic the self-repeatability of nature

(Section 2.2.1), focus on the reproduction of specific terrain features (Section 2.2.2), copy existing terrain data by example (Section 2.2.3), or rely on artificial intelligence methods (Section 2.2.4).

2.2.1 Fractal and noise-based terrains

Fractals (Mandelbrot et al. 1983; Voss 1991) are important mathematical tools for representing natural scenes, from the observation that many features show some similarity at various scales. Many approaches have been designed to use fractal generation for terrains (Ebert et al. 2002)).

Subdivision schemes are used to progressively refine a terrain. Starting from an initial coarse state, the terrain is progressively subdivided, and the altitude values for the new points are interpolated with respect to their neighbors (Fournier et al. 1982). Simple rules are used to choose the interpolation neighbors and randomly displace the newly added point. The whole class of algorithms is called *midpoint displacement*. The interpolation rules vary from the classical diamond-square algorithm (Miller 1986) where new points are computed with respect to four neighbors, successively chosen among the axis-aligned and diagonal neighbors, to more complex methods by Lewis 1987 who advocate for the use of a small linear system, achieving less artifacts and a richer variety of resulting terrains. A second category of techniques, called *multi-fractal* combines different noise functions (Lagae et al. 2010), progressively refining the scale. This falls in the more general formulation of fractional Brownian motion (Mandelbrot et al. 1968). The use of a limited number of simple equations allows for the fast creation of nearly infinite terrains, easily rendered on the GPU Schneider et al. 2006.

Figure 2.4: A fractal terrain made by combining noise functions (Ebert et al. 2002).

A common problem among these approaches is the lack of direct control. Many attempts were made to translate the user's intent into fractal parameterizations. By analyzing real terrains, Lawick et al. 1995 and Belhadj 2007 extract a limited number of noise parameters, that are locally tuned by the user to generate the desired terrain. Sketches (Kamal et al. 2007; Talgorn et al. 2018), or brushes (Carpentier et al. 2009) are the preferred control tool to locally parametrize generation algorithms.

These methods are very efficient at generating extremely large landscapes, sometimes within milliseconds. On the other hand, both plausibility and control are hard to achieve, in spite of the decades of research on the subject. We use these techniques, especially multi-fractal noise, mainly to introduce natural irregularities in the parameters of our simulations (*e.g.,* erosion strength, tectonic forces, *etc.*).

2.2.2 Focus on terrain features

Several methods try to tackle the problem of terrain generation by reproducing specific, isolated terrain features. We separate two main categories of landforms: general landform

features and hydrology network. Visualizing terrain features generally benefits from vectorial representations, as demonstrated by Bruneton et al. 2008.

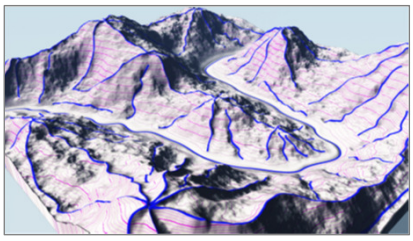


Figure 2.5: Terrain edited through feature curves (Hnaidi et al. 2010).

Landform features. Rusnell et al. 2009 focus on the profile of featured. The user sketches the profiles of generators: mountains, hills or craters, and place them in the terrain. Then, a least cost path is computed in a weighted graph formed by connecting these generators to compute the network of secondary ridges, leading to the generation of various and easily controllable landscapes. Ridges and profiles of cliffs can also be used as the main control tool, as proposed by Hnaidi et al. 2010, where the user's strokes representing preeminent features are interpolated by using a diffusion equation, augmented with some noise. This is extended by Bernhardt et al. 2011 who introduces a vectorial representation of terrain features and copy-paste tools inspired from vector-graphics softwares. Thanks to a coupled CPU-GPU computation, the user can observe the results of the edition in real time. Another possibility is to interpolate the terrain between constraints induced by a ridge network (Ariyan et al. 2015) drawn in 2D by the user, and where elevation are automatically deduced thanks to random walk with controllable probability distributions. Recent work includes the generalization to fully volumetric terrains, enabling the sketching of overhangs and arches (Becher et al. 2017). These methods provide a favorable amount of user control, but the plausibility is limited by the user's skills. Furthermore, although the user can generally edit any specific detail of the generated terrain, this process can be demanding if the features where not designed for these particular cases. With optimizations, these methods adapt generally well to large scale applications. In this thesis, we chose brushes and sculpting metaphor to guide simulations as these can be used to paint parameters map and define physical forces, but we could inspire from sketch based approaches to define the target of inverse simulation methods.

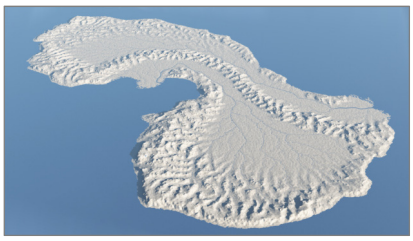


Figure 2.6: Large scale terrain modeled from an hydrology map (Génevaux et al. 2013).

Hydrology. A common observation in terrain modeling is that valleys are carved by rivers; thus hydrology is a major factor directing the distribution of terrain features. Early works propose to apply L-systems to guide the subdivision of a terrain mesh while ensuring the emergence of a river network (Prusinkiewicz et al. 1993). This idea has been recently extended to planet-scale river networks thanks to a GPU accelerated adaptive refinement of the terrain at the camera location (Derzapf et al. 2011). Other generation methods first compute a hydrology network and interpolate the terrain surface between river curves. Belhadj et al. 2005 introduce an *inverse midpoint displacement* method for the interpolation, while Génevaux et al. 2013 automatically generate a large scale hydrologically plausible river network from a user given hydrology map and coarse terrain elevations, before generating the terrain surface to primitive trees. A precursor work from Kelley et al. 1988 use steady state observation for stream

erosion to produce a terrain shaped by fluvial erosion, and the results are similar the ones we present in Chapter 3, the main difference being that their river network is computed geometrically while ours is induced by the simulation, thus leading to more diverse results. Some approaches specifically target the generation of rivers and waterfalls Emilien, Poulin, et al. 2015 or canyons De Carli et al. 2014. While hydrology based methods are generally well adapted to large scale simulation and the river graph is a powerful control tool, the interpolation of plausible landforms between rivers is usually their main weakness, especially where terrain features does not result from the action of water, for example when glaciers or wind events actively shape the terrain.

The strength of feature-based terrain generation methods comes from the visual preeminence of these features, inducing a natural artistic control on the generated result. But these methods generally fail at reproducing plausible terrains, because of the lack of geological knowledge embedded in the underlying models. Embedding feature-based control of simulation parameters could be a solution to take benefit of both methods. We explore this possibility for the local control of sculpted mountain ranges in Chapter 4, and further research would leverage the possibilities induced by this combination.

2.2.3 By example

Inspired from texture synthesis techniques, a solution for improving the plausibility of terrains is to generate them from real data. These *by example* methods are separated in to too broad categories, the ones that aim to deform a given topography, and the ones that generate a new terrain by combining data from multiple sources.

Terrain deformation approaches use a single DEM, modified to match the user's intents. Passos et al. 2013 compute silhouettes of a terrain from different viewpoints, and try to find the best match between one of these silhouettes and a sketch drawn by the user from a first person view. Subsequent terrain additions due to new sketches are blended thanks to a gaussian convolution kernel. An alternative consists in deforming the example terrain to match the input sketches. By using the unknown depth of the silhouette as an optimization variable, Tasse et al. 2014 minimize the amount of deformation needed to reach the target stroke. In addition, this method uses T-junctions of the sketch to infer an ordering of the silhouettes. These approaches generally work well near the user's original viewpoint and where the amount of deformation is small.

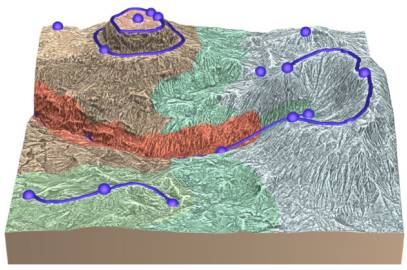


Figure 2.7: Brushes and strokes used to control example based terrain synthesis (Gain et al. 2015).

Terrain synthesis, inspired by texture synthesis techniques, uses many input data acquired from real terrains, and blend them together to achieve both controllable and plausible results. Classical patch based texture synthesis is used by Zhou et al. 2007: overlapping patches of terrains are cut and combined along seams. The placement of patches is guided by terrain features such as crest lines and valley profiles, which can be edited by the user. The seams are directed by a graph cut algorithm and hidden by a smoothing procedure. This is

improved by Tasse et al. 2012 who parallelize parts of the algorithm on the GPU and use a more accurate method for merging seams by assuring a continuity of the gradient. Control is also improved by adding height information to the user input strokes. Patch-based terrain synthesis is replaced by a pixel based approach by Gain et al. 2015: terrain pixels are chosen by comparing their neighborhood with pixels in the input exemplar. Beside improving the quality of results, this method achieves interactive performances and provides an important addition in the terrain control tools. Another solution (Guérin, Digne, Peytavie, et al. 2016) takes advantage of the hierarchical primitive terrain representation to learn and synthesize specific patterns, enabling application ranging from terrain synthesis to detail enhancements.

Figure 2.8: Terrain generated by a generative Adverserial Network fed with a simple user-drawn sketch (Guérin et al. 2017).

Deep learning has also been used to learn the correspondences between terrains and network of ridges or rivers (Guérin et al. 2017). A Generative Adversarial Network is trained on these features, automatically extracted from real landscapes, and applied to user drawn curves representing ridges or rivers. Several additional control elements are added, such as level sets for elevation or altitude clues. They also propose to learn the results of erosion to accelerate the application of erosional post process.

These methods show a paradox in the plausibility of their results. While by-example techniques inherit the local visual plausibility of the exemplar they are built on, the physical plausibility is generally lost: terrain features could still be placed in locations where they could not physically appear. In particular, using a water flow algorithm to add water bodies in example-based terrains may fail because of inconsistencies in river flow directions. This specific issue is solved thanks to our simulations, but we lose some of the detailed visual accuracy captured from real data by using example based techniques. An interesting junction between both approaches would be learn simulation parameters. A second issue prevents these methods to be applied in more complex sceneries where multiple geological layers must be synthesized (for example to render sedimentary fold lines as explained in Chapter 4): whereas real terrain elevations are available on the whole Earth at a 30m sampling, such data is less accessible for stratigraphic information or specific landscape data (plants, rocks, *etc.*).

2.2.4 Methods from artificial intelligence

Some authors inspired from techniques commonly used in artificial intelligence to generate terrains. For example, genetic algorithms are used to combine terrain features while both preserving the user's intent and gameplay considerations such as accessibility (Frade et al. 2010). This method does not focus on plausibility but provides original control tools and automatically produce various yet similar outputs. Doran et al. 2010 use software agents, each of them with a particular purpose: shaping mountains, hills or coastline. These agents randomly walk on the terrain and shape specific features based on the neighboring terrain geometry. A focus is made on the control parameters of these agents, as well as on the variety of landform they are able to produce. We use a similar idea in our stochastic framework pre-

sented in Chapter 6, although we also provide more guaranties on plausibility by specifically considering geological and ecosystem related phenomena.

2.3 Simulation of terrain evolution

Simulation-based approaches model the temporal evolution of the terrain subject to geomorphological agents, in many cases by solving systems of partial derivative equations. The goal is generally to improve the plausibility of existing terrains, although some techniques aim at creating terrains from scratch. This subject has been extensively studied in Geology, although not with the same goal: the physical accuracy is of critical importance when considering measurable spatial and temporal patterns, although visual plausibility and control are generally less considered. A large range of scales is encompassed in geological simulations, from microscopic chemical processes to earth-wide tectonics. It is well known that mountain ranges are formed from the compression of tectonic plates, moved by the convective process of fluid rocks inside Earth mantle (McClay 1992), which inspired our work in Chapter 4. The induced uplift, or mountain elevation rate, is combined with erosion laws, leading to present day valleys profile and mountains distribution (Tucker et al. 2010). We specifically considered fluvial erosion in Chapter 3, modeled with the widely adopted *Stream Power Law* (Lague 2014a), and we addressed the issue of glacial erosion (Montgomery 2002) in Chapter 5. More information on the geological foundations will be given in each chapters.

In the following, we describe simulation techniques used in Computer Graphics. Note that no method in this field has achieved to model the effects of glacial erosion, although leading to extremely preeminent features on the majority of high mountain ranges. We tackle this issue and propose a new efficient method for glacial erosion in Chapter 5.

2.3.1 Small scale features

Although the research in this thesis targets large scale sceneries, reviewing simulation methods specifically designed for small scale features (from a few meters to a few centimeters) is still important, for three reasons. First, the underlying simulations schemes can be shared between different ranges of scales, because of fundamental similarities in the nature of the physical processes involved and the representation of the data. Second, large scale simulation generally averages the effects of smaller one, an observation that could result in a validation mechanism. Third, several small scale phenomena have a strong impact on larger setting. In particular, phenomena at different scales could be interleaved as we propose for temporal scales in Chapters 6 and 7, or parameters of phenomena acting at a large spatial extent could be derived from finer simulations. Furthermore, when generating a close view of a large landscape, small features becomes visible, which augment the needed range of scales needed by a complete simulation.

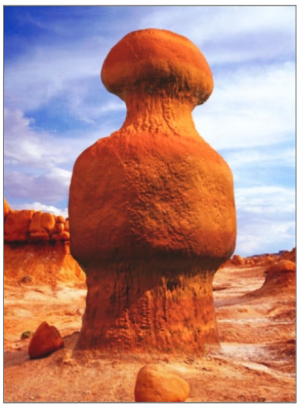


Figure 2.9: A *goblin*, generated by spheroidal weathering (Beardall et al. 2007).

Weathering is the first stage of erosion. It is widely used in Computer Graphics (Mérillou et al. 2008) to enhance the plausibility of any aging 3D model. This has been applied to stone by

considering a slab data structure: a small volumetric region around a surface (Dorsey et al. 1999). Flow of moisture, dissolution, transport and deposition are considered in this region. Interestingly, this erosion method is similar to larger-scale hydraulic erosion. Although addressing the weathering of statues, this technique is quite general and a similar idea is used to compute volumetric *goblins* by spherical weathering (Beardall et al. 2007): starting from a pile of rock layers with various resistance to erosion, the algorithm places a bubble at the location of voxels in contact with the air and erodes them in function of the amount of air in the bubble. Different results are obtained by varying the initial shape of the eroded column and of the erosion bubble. Erosion and corrosion where further simulated by Wojtan et al. 2007 at the junction between solid objects modeled thanks to level sets and a particle-based fluid. Erosion is modeled by considering the shear stress at the solid-fluid interface, and the eroded sediments are progressively deposited along the path of the fluid. Hydraulic erosion and spherical weathering are both considered by Tychonievich et al. 2010 who used a volumetric data structure to store the terrain: a Delaunay deformable mesh, resulting in small scale yet plausible eroded terrains.

2.3.2 Hydraulic erosion

Several methods argue that the most preeminent landform features result from the action of water on the terrain, especially on steep ranges under temperate climatic conditions. Hydraulic erosion results in deep channels joining in dendritic patterns, and is widely used to improve the visual plausibility of virtual mountains. Furthermore, it is generally linked with a simulation of water flow, that can be used to automatically generate consistent water bodies on top of the generated terrain. Several erosive forces are considered, from chemical dissolution which depends on the amount of infiltrated water, to mechanical abrasion often modeled as a function of the height of running water, sometimes dampened by the load of suspended sediment.

An Eulerian representation for a flow field is a representation where the flow quantities are sampled at fixed locations. It is usually preferred, because the sampling can follow the same discretization as the altitude value of the terrain. Musgrave et al. 1989 introduce the concept of hydraulic erosion, applied to fractal terrains generated by a fractional Brownian method. In their erosion model, erosion happens in function of the amount of water, creating suspended sediments. The excess of water is propagated toward neighboring cells carrying the sediments which are progressively deposited. Alternatively, water flow can be computed thanks to hydrology networks, for example obtained with a midpoint displacement method by Nagashima 1998, on top of which the erosion and deposition processes are simulated. Subsequent works introduce more accurate physics to enhance the plausibility of the water motion, by using shallow water simulations (Beneš et al. 2002) or Navier-Stokers equations on a 3D voxel grid (Benes et al. 2006). The visual accuracy of these methods benefits from the physicality of the underlying simulations, but the choice of a regular grid as a support for the simulation sometimes comes with visible artifacts, which we reduce by using a Triangular Irregular Network in Chapter 3.

A lagrangian representation conserves the flow properties along the trajectory of flow particles. Chiba et al. 1998 assumes that the amount of erosion depends on the speed of the water particles, which are physically modeled with no interactions between particles. Advances in fluid simulation has been incorporated by Krištof et al. 2009, considering Smoothed Particle Hydrodynamics (SPH) for water flow and shear stress based erosion. The terrain itself has also been considered with particles of granular material Hudák et al. 2011, where more precise water-soil interactions such as the proportion of water absorbed drives the simulation of mass movement erosion events. SPH are also used to model the formation of meanders due to alluvial deposition (Kurowski 2012), controlled through an exaggerated Coriolis force. The main advantage of Lagrangian over Eulerian representation lies in the extended plausibility of the result, because the fluid simulation is not constrained by a regular grid, that often leads to directional artifacts. Conversely, these methods require more computational time for neighborhood lookup needed by particles-particles and terrain-particles interactions.

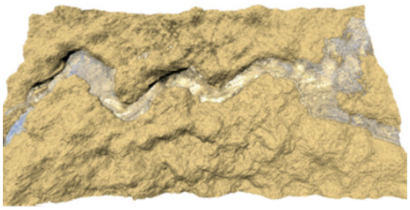


Figure 2.10: Hydraulic erosion by a river, using Smoothed Particles Hydrodynamics (Krištof et al. 2009).

Erosion control and efficiency have for long been the stumbling point of erosion techniques. Some methods try to reduce the cost of the physical simulation while conserving its validity (Benes 2007; Neidhold et al. 2005), while several authors use parallel and GPU implementations (Beneš et al. 2001; Jákó et al. 2011; Mei et al. 2007). This allows the introduction of erosion brushes (Šťava et al. 2008) or strokes (Vanek et al. 2011). These tools are powerful for setting the erosion parameters, this is why we use as similar control methodology in Chapters 3, 6 and 7.

We see a common drawback of erosion methods in the targeted scale. They are very efficient at modeling terrains at small to medium scale, when water flow algorithms are valid. When targeting larger scale applications, the width of rivers is often below the size of a discrete grid cell. This is why we consider an integrated drainage information in Chapter 3 in a new fluvial erosion technique.

2.3.3 Thermal erosion

Another main erosional feature comes from the combination of rock weathering and gravity driven landslide events, generally called *thermal erosion* in Computer Graphics, because of the weathering power of the abrupt day-night changes of temperature in high altitudes.

Thermal erosion is inseparable from hydraulic erosion, because it reduces the high slopes generated by the water induced carving of deep and thin trenches. This have been observed since the early introduction of both hydraulic and thermal erosion by Musgrave et al. 1989. Thermal erosion is often approximated at large scale by removing rocks above a given critical slope (from the assumption that mountains are made of granular materials). Broken rocks are shifted and deposited in the neighboring cells. Different critical slopes can be given to bedrock and eroded materials, as simulated by Benes et al. 2001 while using the first layered model for terrain erosion. Several landforms are explained mainly by this process, as table mountains (Beneš et al. 2005) where hydrology has less importance. Several approaches also

try to model rock erosion with more precision, for example by considering blocks of rocks virtually separated by directional faults (Ito et al. 2003). The simulation consists in removing joints between blocks chosen by considering gravity forces, resulting in more plausible rock cliffs.

We extensively use thermal erosion to model small landslide effects, and we improve it on Chapter 5 by incorporating the erosive contribution of the falling rocks themselves when they impact the ground, a process called *debris flow erosion*. We also mention an efficient implementation of *hill slope erosion*, which models weathering effects thanks to a diffusion equation.

2.3.4 Geologically based simulation

Although many studies root themselves in technique from fluid mechanics, this is less the case when considering geomorphological laws. An early work from Roudier et al. 1993 tries to follow this direction. Although the general idea is quite similar to classical Eulerian hydraulic erosion, they propose two important modifications. First, they integrate the amount of rainwater to compute the abrasive runoff, which is common in geology to model the shear stress applied by water on the ground on large spatial and temporal scale. We use this idea in our fluvial erosion model (Chapter) 3), where we call the outcome of this integration the *drainage area*. They also use several other geological processes: alluvial deposition, gravity creep or rock dissolution, characterized by parameters dependent on the rock type in a volumetric representation. We automatize the formation of folded rock layers with varying rock competence in Chapter 4.

2.3.5 Plate tectonics

Although responsible for the formation of the vast majority of mountain ranges, plate tectonics have only been used by Michel et al. 2015 to generate a procedural terrain from a 2D vector map of mountain summits. There, the terrain is divided into tectonic plates by using Voronoi-based tessellation and an image processing method is used to generate simple multi-scale folds along plate borders, without taking any layered earth-crust model into account. In contrast, in Chapter 4, we introduce the first 3D interactive simulation of plate tectonics, thanks to a deformable approach that couples a viscous model with multi-scale folding of multi-layered rocks. Our model provides a more accurate simulation of plate collision and folding processes.

The simulation of terrain formation has been widely studied because of the implied visual plausibility and physical consistency. It has mainly been used as a post-process to improve the realism of handcrafted or noise-based mountains. Hydraulic erosion is generally preferred for the induced very specific dendritic features, and is combined with thermal erosion that soften the slopes. The main drawback of simulations is generally the lack of control, and in particular the impossibility of authoring features that cannot be explained by the underlying equations. In the first part of this thesis, we extend on these methods by providing more geological foundations, which enable the fast creation of complete mountain ranges. We also use geological knowledge on plate tectonics to propose new control tools.

2.4 Ecosystems

Inspired from Botanic and Ecology, several methods aim at representing vegetation at different level of details. Although we give a slight overview of the modeling of individual plant geometry, our focus is on the generation of large scale ecosystems, either through simulation or statistical synthesis. More information on vegetation synthesis can be found on the book of Deussen et al. 2006.

2.4.1 Modeling individual plants

Even when targeting large scale landscapes, the precise modeling of individual plants can have some benefit for visualization and rendering, or because the deep interaction between the finest details of the plant and surrounding context may have some important visual effects in the overall landscape.

Early work (Bloomenthal 1985) proposes simple models for modeling plants with generalized cylinders to model trunk and branches. The distributions of branches follows simple stochastic patterns, which have latter been extended to give a generic algorithmic definition trough L-systems (Lindenmayer 1968; Prusinkiewicz et al. 2012), a formalism that allows simple formulations of a large variety of branching systems.

Several methods specifically target the design of plants. Prusinkiewicz et al. 2001 use artist-sketched silhouette coupled with botanical knowledge to enable the creation of realistic plants. Although mainly controllable through sketches, some expert knowledge is needed for more advanced operations, such as the definition of a new specie. This is the cost for achieving a large variety of extremely plausible vegetation instances. A similar idea is used by Wither et al. 2009 where the silhouette of a plant is sketched from coarse to fine, first giving the overall geometry and progressively refining the details of branches and leaves. This control tool allows for an intuitive, fast and expressive authoring of the plant, but gives less guaranties to the plausibility of the generated result. Another point of view is followed by Benes et al. 2009, in an approach where the simulation of growing plants is contained by a user-specified 3D envelop. As with many controlled simulation techniques, the plausibility is balanced with a limited control, here on the overall geometry of the plant.

Figure 2.11: Diamant plant interactively created from artist sketch and feature knowledge (Prusinkiewicz et al. 2001).

One advantage observed when modeling each plant instance is the ability to adapt the result to the surrounding environment. Měch et al. 1996 extend the L-system formalism to incorporate bi-directional exchanges with the environment into a new framework called *open L-systems*. Soler et al. 2001 accurately simulate the radiant energy transfer in a model where plants compete both for space and for light. A recent work (Hädrich et al. 2017) uses simulation on the GPU to interactively grow climbing plants. The realism of the results is increased by the synergy with the environment and the GPU implementation enable an interactive feedback from user inputs.

Overall, these methods aim at precisely modeling of a handful of plants, and applying them to large scale landscapes would result in an extremely realistic vista. Unfortunately, it would require enormous amounts of generation time, often wasted because the user will never see the details of the majority of the plants in dense ecosystems. An original approach is proposed by Bornhofen et al. 2009, who show synergies between plant growth and landscape scale forest modeling. Plants grow and evolve with respect to environmental and competition conditions, and ecosystems are generated with the new plants parameters. This approach allows to generalize ecosystem generation to weakly documented species or virtual settings. In the following, we will consider a more classical approach to build large scale ecosystems, where the plants are abstracted behind a density map or a list of their positions. In particular, we will first see how this approach is rooted in specialized literature.

2.4.2 Methods from Ecology

A large amount of ecosystem simulations in Computer Graphics is rooted into experiments from Ecology and Botanic. An extensive survey on specialized ecosystem modeling would be out of the scope of this thesis. Therefore, we propose only some examples of the different research directions pursued in these fields. More details can be found in (Shifley et al. 2017, focused on forest dynamics) and (Wullschleger et al. 2014, about *Plant Functional Types* formalism).

Due to the strong interactions between large scale vegetation coverage and climate change, many work use Eulerian simulations to understand the response of forests after changes in climatic conditions. For example, an early work from Prentice et al. 1993 takes into account many characteristics of different species, such as limiting seasonal temperatures, foliage assimilation or swapwood transpiration, in function of environmental conditions, such as temperature, moisture or CO_2 levels to phenomenologically grow the vegetation. A single tree is modeled per grid cell, and competes with its neighbors for light and resources. This simple representation allows to simulate large landscapes during centuries, and to have a broad idea of the impact of climatic changes on ecosystems.

Large scale Dynamic Global Vegetation Models extend this approach to Earth sized landscapes. Foley et al. 1996 discretizes the planet into 2° (~ 200 km) cells and use a multi-level method to simulate the ecosystem behavior. Environmental conditions (water, carbon, energy) are simulated on an hourly basis, and the result is integrated other a year, as an input for a global ecosystem simulation. A synthesis of different kind of Dynamic Global Vegetation Models is shown by Sitch et al. 2008, which compares their predictions after dramatic changes in climate to show the possible outcomes and their uncertainty. As an exception in this range of methods, Sato et al. 2007 do not use an Eulerian simulation but instead propose to model precisely plant position and geometry to more accurately capture light competition in 30×30 m areas. These areas are grouped in larger regular grid to reach the global scale.

When lower scales are considered, it is common in ecology to use statistics based on field observations to accuracy sample plants. Law et al. 2009 explore different measures applied to spatial points theory to compute a plant distribution from various observations in field work. This distribution is used to extract important ecological information. A possible outcome in Computer Graphics is the automatic synthesis of similar landscapes through computed plants distribution. This will be further discussed in Section 2.4.4.

These methods specifically target large scale ecosystems. Their purpose is to provide

a high biological plausibility and are validated by comparing biomass evolution with real world-data. However, it is not straightforward to apply them in Computer Graphics because of their induced complexity which precludes user control and artistic beautification.

2.4.3 Lagrangian simulation

Since the early days of virtual ecosystems, a Lagrangian trend has been preferred, where each plant is represented by a particle. By using a simulation approach, the plants are seeded at random locations, propagate, compete for resources, and eventually die. The simulations also account for external resources (temperature, sun, moisture, slope, wind, *etc.*) and for biological models for plant growth and competitions.

A seminal work by Deussen et al. 1998 introduced this Lagrangian representation for plants in Computer Graphics. This simulation method takes as input a user painted density map or randomly seeds the plants. The competition is only considered for space, in an approach called self-thinning where collisions between plants slow down the growth and smaller plants die. Vigorous plants scatter new seeds, which are positioned thanks to an open L-system formulation. A similar simulation approach is used by Lane et al. 2002, with an emphasis on the combination of simulation at different scales (multilevel communities) in particular, an extension is proposed to the L-systems methodology, called *multi-set L-systems*, which is applied to a set of strings representing multiple trees. This work also investigates the generation of tree samples from density map (see below). Asymmetric competition, where larger plants steal resources from weaker ones, is introduced by Alsweis et al. 2005. The plants grow in accordance to the disposable amount of resource from a user given map. Both symmetric and asymmetric competitions are considered based on a simple radial interaction (resources are shared in the overlapping area of disks representing the extent of the plants). This single resource map is extended by Ch'Ng 2011 to handle variations in soil, monthly sunlight and temperature, with extremal conditions for these variables, out of which the plant cannot survive. This is the strategy adopted by Gain et al. 2017, who use four extremal variables for the fitness to environmental conditions: the extremes at which the plant dies, and inner values surrounding the optimal conditions.

Figure 2.12: Asymmetric competition prevent small species to grow under larger ones (Alsweis et al. 2005).

Control is provided only through resource maps and competition parameters, which makes the precise definition of the user's intent a hard task. Recent work (Bradbury et al. 2015) tried to alleviate this issue by providing higher level controls such as copy paste, typify or density change. Conditions are automatically updated to account for the user's instructions, and the simulation is continued to eventually reach a consistent state closer to the artistic intent.

It must be noted that very few methods incorporate human intervention in the development of ecosystems. Virtual agents have been considered to improve user control thanks to a set of *habits* (Benes et al. 2003), continuous functions that define the per-species behavior that an agent must follow when stochastically walking in the virtual environment. Another

work considered ecosystems in urban environments, where previous simulations are used in wild regions and procedural rules guides plants gardening in urban areas.

The plausibility of such simulations is generally balanced by their complexity, which makes large scales ecosystems hard to generate. We do not adopt these approaches in our landscape generation model. We preferred data structures more easily pluggable with classical terrain representations, because we focus on the interactions between the different landscape elements. In the following, we will see how statistical methods can help to generate large scale ecosystems.

2.4.4 Statistical synthesis

Larger scale scenes can be populated thanks to the synthesis of statistical distribution. Rather than simulating the progressive growth of plants with all the implied interactions and natural pruning, these methods aims at generating an instant landscape from statistical knowledge of the distributions of plant properties (mainly position and size) for different species, depending on environmental conditions and neighboring vegetation.

Simple techniques are proposed by Deussen et al. 1998, who use half toning on a density image as a starting condition for a Lagrangian simulation (see above), or Andújar et al. 2014 who use dart throwing techniques to prevent trees from being sampled below a critical distance to their neighbors.

Figure 2.13: Result of condition dependent disk based vegetation synthesis on Grand Canyon (Gain et al. 2017).

More control on the per species interactions is proposed by Lane et al. 2002, where the plants are progressively sampled on a probability map. Each new plant updates the map thanks to a deformation kernel, which can have different formulation depending of the plant type. The kernel can, for example, reduce or augment the seeding probability of another plant being seeded depending on its distance. A typical kernel would reduce the probability near a sampled plant, while increasing it in a small ring at a given radius. This kernel favors groves or clumps of vegetation.

Wang tiles are aperiodic tiles with a colored border, such that a tiling is made by associating tiles with the same border color. Alsweis et al. 2006 fills them randomly by using a Field-of-Neighbourhood plant distribution model, where each plant influences its neighboring region. Several versions of a tile are given to match different resource level, which enable the tiling to generate areas with different densities according to the user's intent.

Statistical synthesis is efficient at quickly populating large scale landscapes, at the cost of a reduced plausibility and hard user control. We can think of two solutions for improving both. A first approach is proposed by Gain et al. 2017[1], built on previous simulations methods to compute accurate simulations on 100×100 m sandboxes with various environmental conditions (sun, moisture, slope, temperature). Statistical point process (Emilien, Vimont, et

[1]While I co-authored this work, it will not be presented on the contribution part of this manuscript; it is therefore summarized here.

al. 2015; Hurtut et al. 2009) are extended to handle interactions between the canopy and the roots of the different plants and is used to analyze the result of these simulations and deduce conditions dependent distributions. Plants are then synthesized on large 10×10 km terrains depending on local conditions. Control is proposed by interactively exploring the space of conditions, leading to high level brushes similar to the one proposed by Bradbury et al. 2015, while providing instant feedback. First, complex coarse simulation can be held on discrete grids, and the results of the simulation are forwarded as input to synthesis techniques, which compute the actual tree placement. We will follow this lead in Chapter 6.

in summary, ecosystem modeling has been explored in Computer Graphics for decades, leading to the development of two main trends: the simulation of competing plants in a Lagrangian representation, and the synthesis of statistical distributions. The choice between both mainly depends on the targeted scale: simulation based methods generally capture more realistic vistas at the cost of a reduced scale. Filing the gap between the two approaches, we prefer a coarse simulation on a regular grid, used as the input of a statistical synthesis. Furthermore, except for rare methods that will be detailed in Section 2.6, ecosystem simulation does not take into account other natural processes such as rock falls. We propose to extend the plausibility of the generated scenes by considering how ecosystems interact with erosional processes in Chapter 6. In the next section, we will climb to higher altitudes to find another important feature of mountain ranges: the snow cover.

2.5 Snow

Snow is a major element on high mountains. Accurately modeling snow distribution greatly improves the temporal variability of a landscape between seasons, as well as the spatial variability of the winter sceneries. Indeed, although snow can be perceived as a binary mask, the presence of snow is driven by competing physical phenomena such as heat transfer, mass movements or wind effects on a steep topography.

Existing methods for the modeling of snow cover fall into three broad categories: particle-based, physically-based heat transfer, and procedural surface displacement.

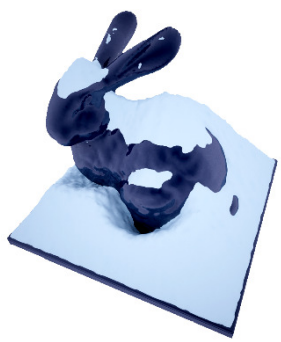


Figure 2.14: Stanford Bunny covered with a simulation wind blown snow and accumulated thanks to a level set approach (Hinks et al. 2009).

2.5.1 Lagrangian snow simulation

Many works discretize the snow into particles and simulate their temporal evolution subject to wind and scene geometry. Early work (Nishita et al. 1997) proposed to consider snow as skeletal points (blobs). The induced implicit surface defines a smooth upper boundary for snow cover. A probabilistic snow cover is introduced by Fearing 2000 who samples snow elements on the surfaces, and compute inverse trajectories to the source clouds. Particle likelihood is obtained by accounting for collision, and the snow surface is generated using importance sampling. A final stabilization step is performed to account for a natural angle of repose for the snow, by simulating micro avalanches. This approach has been later extended by considering a 3D fluid simulation of

the wind, which transports and deposits snow elements to eventually form snowdrifts and depressions.

Wind is a major factor for the buildup of the snow cover and has been considered by many modeling strategies. Early work use cellular automata-based approximations Masselot et al. 1995. Latter studies (Moeslund et al. 2005) simplify the Navier-Stokes equations (incompressible Euler equations) on a 3D regular grid (voxels) to compute the wind speed field, responsible for the advection of snow particles. Wind and gravity also move snow in the work from Wang et al. 2006, who chose to model snow by following the Boltzmann law of statistical physics. A probability distribution of wind speed is stored in a voxel grid where this equation is solved, and instantiated to model the actual wind field. A parallelization of the simplified Navier-Stokes equations is proposed by Saltvik et al. 2007 to achieve real-time feedback on large scenes, and a level set approach (Hinks et al. 2009) improves the plausibility of the accumulation patterns.

Material Points Methods (MPM) have been introduced to Computer Graphics by Stomakhin et al. 2013 to simulate snow dynamics. Snow particles are represented in a classical Lagrangian way, accompanied by a static Eulerian implicitly handling of fractures and self-collisions. The complexity of snow rheology is handled thanks to an elasto-plastic model.

There are also benefits in a hybrid treatment of snow that separate static structure from dynamic elements implemented with particles. Sai-Keung et al. 2015 use voxels and spring-connected particles, while Dagenais et al. 2016 use an heightfield and two type of snow particles: dense snow modeled as granular materials, and volatile snow mist simulated as an incompressible fluid. This representation allows for rigid objects to interact with the snow.

Particle based methods achieve astonishing plausibility, but are limited to small scale scenes with an upper limit of about 100×100m. They also follow the curse of simulation methods, where user control is limited to the setting of external conditions and scene geometry, and to the complex and time consuming tuning of parameters.

2.5.2 Physically-based Eulerian heat transfer

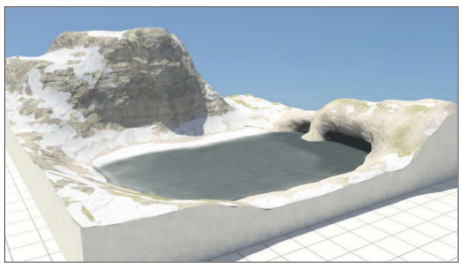


Figure 2.15: A partially frozen lake obtained with voxel-based heat transfer (Maréchal et al. 2010).

Physical phenomena such as phase change and heat transfer are of paramount importance when considering snow on longer term seasonal periods. Vortex driven snow falls (Muraoka et al. 2000) build a snow layer on small scenes. Sun and thermal radiations are accounted for in a heat transfer simulation responsible to the irregular melting of snow. Different thermal behaviors are observed depending on the flake type, itself built in function of the different possible environmental conditions.

Maréchal et al. 2010 go further by simulating snowfall altogether with conductive, convective and radiative thermal transfers using a finite volume method over a voxel grid. This accounts for the complexity of phase changes where snow melts to become water or water freezes into ice, as well as a variety of environmental conditions such as cloud cover, day night cycles or variations of air temperature.

The simulation of heat transfer increases the realism of dynamic landscapes, but comes at a high computational cost, since it requires to solve complex radiative transfer equations on a memory-heavy volumetric representation of the scene. Therefore, only small scenes are considered and user control is weakened. In Chapter 7, we also consider this dependence of snow cover upon radiative transfer and state change, and we extend it to larger scale sceneries thanks to a planar layered representation.

2.5.3 Procedural surface displacement

Other phenomenological approaches achieve a faster generation of the snow cover by applying a small offset on top of the surface of objects. Shadow buffer techniques are used to build a snow map probability on complex objects (Tokoi 2006). The implied distribution is used to progressively offset the snow cover, which undergoes small displacements to model lateral snow stabilization. A similar method is proposed by Reynolds et al. 2015 who use accumulation buffers on dynamic scenes. Height span maps are used to convert complex geometry onto vertically stacked layers(Festenberg et al. 2009): surfaces are classified as air-solid transition if their normals point toward the sky direction, or as solid-air transition otherwise. The snow cover evolves interactively on the air-solid layers based on statistical considerations, later improved with a diffusion process (Festenberg et al. 2011).

These methods are targeted at small scale sceneries, and few approaches consider large scales terrains. In an attempt to model mountain sceneries from aerial images, Premože et al. 1999 proposed a simple model of snow cover taking into account temperature varying with altitude, sun occlusion and vegetation. Lighting considerations are improved by a model from (Foldes et al. 2007), where snow is dissipated thanks to combinations of ambient occlusion and direct sunlight. Wind can be approximated by using a pre-computed Radiance Transfer scheme to back up the effects of directional wind on the scene geometry and induce the snow cover (Moriya et al. 2010).

Only a few methods focused on improving the direct generation of the surface of the snow cover, and approaches targeting the extent of a whole landscape are limited either by a small targeted scale, either by the small number of snow related phenomena. Furthermore, they do not take into account large scale dynamic effects such as avalanches.

2.5.4 Avalanches

While some models exist in the specialized literature to explain snow motion in avalanches, for example by simulating powder-snow in 2D vertical cuts (Étienne et al. 2004; Rastello et al. 2004), this problem has been weakly studied in Computer Graphics. Tsuda et al. 2010 propose one of the very few methods for simulating avalanches. The authors particularly target mixed-motion avalanches, made from an upper powdery suspension layer and a lower dense-flow layer which behaves similarly to fluids. The upper layer is modeled on a density grid, while the lower one is handled by particles. A third layer corresponding to the snow cover (called accumulated snow layer in the paper) is

Figure 2.16: A slab snow avalanche modeled with Material Points Method parametrized with field observation (Gaume et al. 2018).

also modeled with particles. Both these three layers interact with each other, capturing the complex behavior of avalanche motion.

In a recent work, Gaume et al. 2018 use the Material Points Method previously considered for simulating snow (Stomakhin et al. 2013). The plastic behavior of snow is parameterized thanks to field observations, especially the stress threshold above which the snow fractures, they are able to model slab snow avalanches formed by the rupture of a weak snow layer below a strong one. Their method does not only provide a strong advance in the physical modeling avalanches, which is an important step towards the difficult prediction of slab snow avalanches, it also results in a visually interesting simulation of avalanches that could be used by the Graphics industry.

These methods strongly rely on simulations, which relegates them to narrow scale sceneries. We propose a simpler avalanche model in Chapter 7, enabling the simulation of avalanches on large terrains. Therefore, our avalanche model benefits from the computation of a stability map, jointly evolving with the snow cover throughout the season. The stability map influences both the starting probability of an avalanche and the amount of snow it can carry away. In the next section, we will consider methods that go deeper in the modeling of the interactions between the different aspects of a landscapes.

2.6 Full landscape authoring

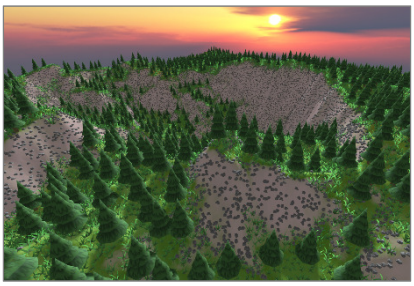


Figure 2.17: Distributions of trees, rocks and grass are learned from a user specified example and adapt to the terrain slopes. (Worldbrush Emilien, Vimont, et al. 2015).

Only a handful of techniques consider several aspects of a landscape together in an unified modeling framework. Capturing the interactions between these aspects allows an improved plausibility as well as an easier control, because it reduces the needs for back and forth iterations between the different layers of a landscape: a modification that happens on one layer, *e.g.,* the vegetation, can have cascading influence on the other layers, *e.g.,* terrain, rocks, snow, etc.

The interactions between water bodies such as rivers, waterfalls or lakes, and the underlying terrain where accounted for in the work of Emilien, Poulin, et al. 2015. The user paints the trajectories of rivers with different tools for each of the considered water body. The river network is computed automatically by following hydrological knowledge and the terrain is locally adapted to ensure the physical consistency of the water flow.

Emilien, Vimont, et al. 2015 use statistical analysis to train brushes that holds stochastic distributions of features, such as trees, rocks, houses and roads in a set of user specified palettes. Statistics can be correlated to wider scale phenomena such as the local slope of the terrain, proximity to a river, etc. Several tools inspired from painting software are proposed to edit a target landscape. In particular, distribution can be painted and regions deformed, the underlying objects are updated while preserving the statistical distribution thanks to histograms of pairwise interactions. A histogram is computed for each pair of types of object and embeds the distribution of distances between objects. This

method enhances user control of both 2D artwork maps and 3D virtual worlds (taking into account terrain surface), with a method able to shape large scale landscapes. A limitation lies in the lack of plausibility constraints: the distributions are learned from user painted maps, and the plausibility of the results is only ensured by the artistic skills of the user.

Another example-based method (Argudo et al. 2017) targets large scale sceneries by storing detailed real landscapes features in a dictionary. A high resolution landscape including detailed terrain features and vegetation is then synthesized on top of a coarse terrain representation. By including multiple layers of slope, sun exposure and water drainage, this method enables the user controlled generation of landscapes on multiple biomes, such as various forests, deserts and mountains. The by-example approach results in strong local visual plausibility, but in a weaker global physical consistency, in particular where the river networks are considered. Furthermore, this approach needs to be coupled with some efficient methods to generate the coarse terrain.

In a more recent work,Grosbellet et al. 2016 provide a general architecture for detailing small but complex scenes, by allowing objects in the environment (such as lampposts, trees, and fountains) to affect scalar parameter fields (such as temperature, fallen-leaf density and humidity), which ultimately dictate the decoration of the objects themselves with snow, ice and fallen leaves. The use of procedural laws for the hierarchical interactions between the scene objects and the environment eases user control thanks to local environment parameters and allows near interactive performance. The decoration of small scales sceneries is also addressed by Guérin, Galin, et al. 2016, in an approach where small details (leaves, rocks, dead branches) are instantiated without collision following user painted density fields.

The existing landscape generation methods suffer either from a reduced physical consistency, either from a small targeted scale. The second part of this thesis tackles these issues: thanks to a 2D planar representation, we are able to focus on larger scales. By simulating the interleaved phenomena responsible from the shaping of landscapes, we improve both visual plausibility and physical soundness.

2.7 Conclusion

Landscape generation encompasses a broad range of methods, ranging from the generation of terrains to the modeling of ecosystems and snow cover, but several directions are yet to be explored. By tackling the issue of simulating the temporal evolution of landscapes, we try to improve the plausibility of large scale landscapes while accounting for user control. In particular, to the best of our knowledge, no previous work in Computer Graphics tried to model the formation of whole mountain ranges. We build on methods from Geology to achieve this, by combining tectonically driven uplift (Chapter 4) with fluvial (Chapter 3) and glacial (Chapter 5) erosion. We also observe that many solutions exist to individually model the phenomena responsible for landscape features, but few attempts were made to simulate them jointly. We propose a new method to achieve this interleaved simulation, applied to the combination of geological effects and vegetation (Chapter 6) and of the different events involved in the evolution of snow cover (Chapter 7).

Part I

Large scale mountain formation

Chapter 3

Combining uplift and fluvial erosion

Contents

3.1 Background and overview **35**

3.1.1 Geological background 35

3.1.2 Algorithm overview 37

3.2 Stream generation **38**

3.2.1 Stream graph initialization 38

3.2.2 Stream tree computation 39

3.2.3 Lake overflow 39

3.3 Erosion **43**

3.4 Results **44**
3.4.1 Visual realism 45
3.4.2 Rendering 45
3.4.3 Performance 46
3.4.4 Lake overflow 47
3.4.5 Stream power erosion 51
3.5 Conclusion **56**

Mountain ranges are major constituents of virtual landscapes. Recent 3D applications are surrounded by extremely large virtual sceneries, increasing the need for the modeling of plausible mountains at large scale. This chapter presents the generation of large scale terrains (at the order of 100 km large ranges). It extends our work on fluvial erosion, an improvement from a first version published at Eurographics in 2016 (Cordonnier et al. 2016). While the first version promoted fluvial erosion as one of the major natural processes that sculpts mountain ranges, we propose a more efficient implementation of geological equations, enabling for simple, yet effective user control.

Previous work in Computer Graphics mainly considered simulation of hydraulic erosion as a decoration tool, a post processing steps in a landscape modeling workflow (see Chapter 2, Section 2.3.2). This process is well recognized for the realism of the induced landform features, especially the dendritic distribution of channels. In spite of the large amount of research on the subject, hydraulic erosion has never been used to fully generate mountain ranges. This can be explained by the complexity of the underlying flow simulations that prevented large time steps, therefore precluding the use of temporal scales at the extent of mountains formation. We use a simple integrated formulation of fluvial erosion that alleviates this issue and ables us to compute the mountain elevation from a flat terrain to a fully grown mountain range.

We observe that erosion processes do not only impact the aspect of the terrain surface, but are also combined with uprising tectonic forces to deeply carve valleys, resulting in the shaping of mountain ranges at large scale. These tectonic forces are grouped in a common phenomenon, called *uplift* (Beaumont et al. 1992). The role of the interaction between the uplift and the erosion has been extensively studied in geomorphology and expressed through different models, such as the *stream power law* (Whipple et al. 1999). We adapt this theory to Computer Graphics needs, by providing an easily controllable mechanism that generates large scale realistic terrains conforming to a global geomorphological process.

Our method specifically simulates the generation of both visually plausible and physically consistent large scale landform features and patterns. The input to our algorithm is an uplift map painted by the user that locally defines the speed at which mountains are lifted. From this input, and a random planar graph covering the region on the map, we iterate through elevation updates for graph nodes, using the stream power equation to simulate the interaction between tectonic uplift and fluvial erosion processes. The original implicit implementation from Braun et al. 2013 is extended to efficiently model water flowing from lakes. This simulation process produces a stream graph derived from the initial graph. The graph is augmented with stream directions along edges and elevation information at the nodes. The graph can either be converted into an elevation map by interpolating the elevation information between streams for real-time visualization, or converted into a primitive-based terrain model with a high level of detail embedding riverbeds, ridges and valleys, using a

combination of parameterized terrain primitives introduced by Génevaux et al. 2015 and automatic terrain amplification (Guérin, Digne, Peytavie, et al. 2016).

Our contributions are:

- A new geologically inspired method for efficient erosion simulation
- A user controlled uplift map that drives the generation of complete, large scale mountain ranges.
- A new algorithm to compute water path out of topographic depressions (lakes).

This last contribution is an addition to the original article (Cordonnier et al. 2016), and will be published in a specialized journal.

3.1 Background and overview

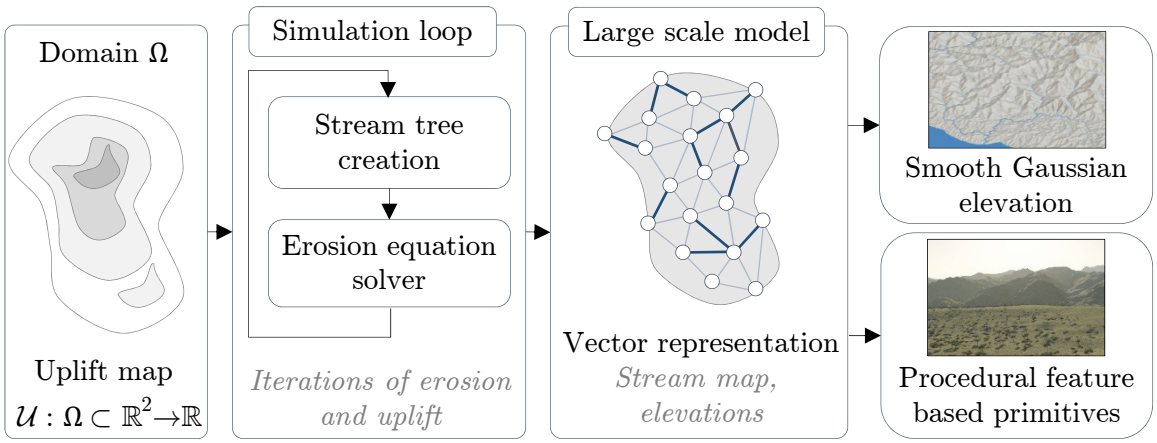


Figure 3.1: Our algorithm takes the uplift map as input and repeatedly applies stream tree creation and erosion. The output is encoded in a vector-based representation which is then converted into a procedural terrain construction tree.

We extend the way hydraulic erosion is modeled in Computer Graphics to capture its action at large spatial and temporal scales by including *fluvial erosion*. At such scales, erosion due to streams cannot be considered without also taking into account mountain development. This section recalls geological background that provides a basis for the following overview of our method.

3.1.1 Geological background

Uplift and faults. Terrains result from the combined action of tectonic uplift of the Earth's surface and erosion. Collisions between continental plates, as well as subduction of ocean plates under continental ones, cause the continental crust to shorten and thicken. This results in the growth of mountains along the main boundaries between plates. Faults and folds appear in regions where the crust undergoes the highest stress (Willett et al. 1993).

In geology, the term *uplift* is used to denote the local speed at which a mountain grows. Because growth occurs between the series of parallel faults, considering the uplift as locally uniform between these faults is a legitimate approximation. The complex landform features found in nature are mainly the result of the interaction between the uplift factor and fluvial erosion, *i.e.,* the action of water forming streams that carve the terrain while it grows.

Fluvial erosion is the erosion of the bedrock material and its transportation downhill by streams. It is caused by the shear stress exerted by running water and the sediment it contains onto the bed of a stream. The interaction between the fluvial erosion and the tectonic uplift has been studied for many years in geology and is often modeled by *the stream power equation* Whipple et al. 1999:

$$\frac{\partial\, h(\mathbf{p})}{\partial t} = u(\mathbf{p}) - k\, A(\mathbf{p})^m\, s(\mathbf{p})^n \tag{3.1}$$

The stream power equation states that the rate of change of surface topography $h(\mathbf{p})$ at a position $\mathbf{p}$ is controlled by the balance between the surface uplift $u(\mathbf{p})$ and the fluvial erosion, which is a function of the local slope $s(\mathbf{p})$ and the drainage area $A(\mathbf{p})$. The local slope $s(\mathbf{p})$ is defined as the surface topographic gradient:

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

The constants m and n depend on rock strength, climate, and the topology of river networks. While the values of those parameters are poorly understood, the ratio m/n is constrained by the shape of the stream profiles and is thought of being $m/n \approx 0.5$ Whipple et al. 1999. As in most geomorphological studies, we use $n = 1$ and $m = 0.5$. Moreover, some geological studies attempt to tune these parameters by example Croissant et al. 2014 and a recent survey Lague 2014b studies the limit of geological knowledge regarding the parameters of the stream power equation.

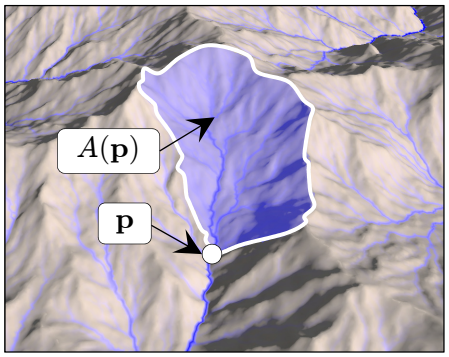


Figure 3.2: The drainage area $A(\mathbf{p})$ is defined as the planar region where streams flow to point $\mathbf{p}$.

The drainage area $A(\mathbf{p})$ is the upstream area draining through point $\mathbf{p}$, assuming that water flows along the topographic gradient (Figure 3.2). In our implementation, the terrain is represented by a geometric graph $\mathcal{G}$ connecting points sampled over the terrain domain. The drainage area $A(\mathbf{p})$ is the area associated to the set of points $\{\mathbf{q} \in \mathcal{G}\}$ strictly above $\mathbf{p}$ such that there exists one path of strictly increasing height starting form $\mathbf{p}$ and ending at $\mathbf{q}$. The factor k is an erosion constant that depends on many factors, such as lithology (the composition of the soil/bedrock), vegetation, climate, and climate variability.

Note that, when applied at the right temporal and spatial scales (typically between 10^5 and 10^7 years and a few tens to hundreds of kilometers), the stream power equation does not only model erosion, but also captures the way a complex relief emerges from a supposedly flat part of the continental crust Howard 1994.

3.1.2 Algorithm overview

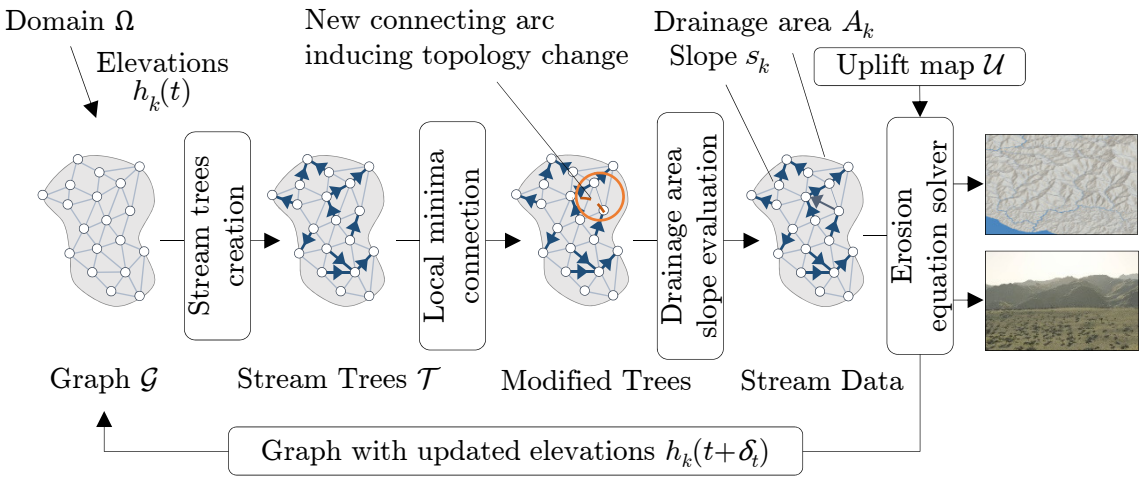


Figure 3.3: Overview of the stream power resolution algorithm: given an initial domain Ω and a random planar graph $\mathcal{G}$ over Ω, we build the set of stream trees representing the drainage structure of Ω and solve the stream power equation to update the elevation $h_k(t)$ from the drainage and the uplift map $\mathcal{U}$.

The input to our algorithm (Figure 3.1) is the uplift map $\mathcal{U}$ defining the speed at which the terrain is elevated by tectonics. We define it as a piece-wise uniform distribution of values over the domain Ω. It is given by the user through gray-scales images. The output is a vector-based representation of the terrain resulting from the interaction between the uplift and the fluvial erosion, and in the case of the final high quality rendering, a set of procedural feature elements.

Our algorithm proceeds in two main steps: erosion simulation computed on planar graph $\mathcal{G}$ embedding elevation and flow information, called the *stream graph*, and conversion of this graph into an elevation model $\mathcal{M}$ representing the terrain.

Erosion simulation. Starting from the input domain Ω where the uplift $\mathcal{U} \neq 0$, we initialize the stream-graph $\mathcal{G}$ as a random planar graph defined by triangulating uniformly distributed terrain sample points $\mathbf{p}_k$ in Ω. We set the initial elevation of the nodes h_k of $\mathcal{G}$ to zero. We then iterate until we get plausible elevation information (or water flow directions) associated to each node (or arc) of $\mathcal{G}$. This is done by iterating the following steps as long as the system has not reached steady state (Figure 3.3):

1. A set of oriented stream trees $\mathcal{T}$ covering the graph $\mathcal{G}$ is extracted according to the current elevations $h_k(t)$ of the nodes to model the direction of running water.

2. The set of stream trees $\mathcal{T}$ is augmented by adding new arcs modeling water from lakes overflowing into streams, yielding a modified stream tree $\widetilde{\mathcal{T}}$.

3. The drainage area $A_k(t)$ is computed for every node in the trees $\widetilde{\mathcal{T}}$, and the local slope $s_k(t)$ is evaluated from the current elevations $h_k(t)$.

4. The stream power equation (3.1) is solved to compute the new elevation values $h_k(t+\delta_t)$ from the uplift map $\mathcal{U}$ and the new values of $A_k(t)$ and $s_k(t)$.

This iterative process stops when a stabilization criterion is met, *i.e.,* when the changes in elevation $|h_k(t+\delta_t) - h_k(t)|$ are below a given threshold. In practice, we observe that the obtained steady state is often approached in real mountains. This criterion ensure that the generated terrain embeds a fully growth mountain range. The initial state can be different from a flat terrain, in which case few iterations with a low time step can also be used as other previous erosion techniques for improving the plausibility of terrains generated by noise. The resulting stream graph $\mathcal{G}$ has the same topology as the initial graph, but embeds a map of streams and elevation information for all nodes.

Conversion of the stream graph into a terrain model. We propose two methods for generating the terrain model as a global elevation function $h : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ from the stream graph $\mathcal{G}$. The first, which can be used to visualize the simulation at interactive rates, uses a simple interpolation scheme to define h from the elevations h_k.

The second method targets off-line high-quality rendering and generates a detailed terrain from the data embedded in the stream graph. This is achieved by combining procedural function-based primitives representing landform features as described in Génevaux et al. 2015, which enables us to add visual details such as ridges, valleys, and river beds.

3.2 Stream generation

In this section, we describe how the stream graph is generated and how stream trees are extracted from it (Figure 3.3).

3.2.1 Stream graph initialization

We use a coarse triangulation of the domain Ω to initialize the (undirected) stream graph $\mathcal{G}$. More precisely, points $\mathbf{p}_k$ are generated by using a Poisson distribution over Ω and the edges of the graph are created by computing a constrained Delaunay triangulation of $\mathbf{p}_k$, where constraints are used to fit the borders of the domain. Although our algorithm could also be applied to a regular grid, the randomized triangular graph generates more plausible results because the edges of the graph better represent possible directions for local streams. Moreover, Poisson sampling ensures the coverage of the domain and guarantees a minimum distance between points. Its association with a Delaunay triangulation provides edges of quasi-uniform length, enabling us to set the level of detail at which terrain features will be generated.

In addition to its position $\mathbf{p}_k$, each graph node $\mathcal{N}_k$ holds an initial height $h_k = 0$, an uplift value $u_k = \mathcal{U}(\mathbf{p}_k)$, and local values derived from the erosion parameters k, m and n. Note that h_k is set constant for nodes lying on the border of the domain Ω. These nodes are tagged as external nodes and serve as river mouths, *i.e.,* points at the sea level, and are the outflows of Ω.

Finally, we compute a Voronoï tessellation of Ω and assign an area value a_k to the nodes $\mathcal{N}_k$, defined as the area of the Voronoï cell surrounding $\mathcal{N}_k$. The area a_k is used to evaluate the amount of rain directly received by the node $\mathcal{N}_k$ in the computation of the drainage area (Figure 3.2).

3.2.2 Stream tree computation

Computing a set of (directed) stream trees that cover the graph (the first step of the simulation loop in Figure 3.3) and updating them at each iteration is the key for efficiently computing the drainage areas needed for solving in the stream power Equation (3.1).

We define the set of directed stream trees $\mathcal{T}$ as follows. For each node $\mathcal{N}_k$, considering that water only flows from $\mathcal{N}_k$ to its neighbor $\mathcal{N}_l$ with the steepest downward slope, which we call the *receiver* of $\mathcal{N}_k$, we connect $\mathcal{N}_k$ to $\mathcal{N}_l$ by an arc (a directed edge). Since these connections cannot create loops, they result in a set of trees that are oriented from leaves (*i.e.,* the nodes that are not receivers for any other node) to the root nodes (*i.e.,* lakes or outflows from the Ω). By construction, the set $\mathcal{T}$ covers $\mathcal{G}$, in the sense that all the nodes of $\mathcal{G}$ are included, although only a subset of the edges from $\mathcal{G}$ is represented by arcs in $\mathcal{T}$. Note that during the first iteration, when all nodes are of height zero, no arc is created and each node is initialized by a tree with only a root node.

3.2.3 Lake overflow

The stream trees $\mathcal{T}$ cannot be used directly to simulate water flow over Ω because these trees are not connected, and the water would stop at internal root nodes that represent lakes.

This water routing problem has been studied in Geography and Hydrology (Barnes et al. 2014; Jenson et al. 1988; Lindsay 2016; Zhou et al. 2017), where heightfields are used to compute drainage networks. Holes appear on the terrain, either coming from natural depressions, either from the artifacts captured during the acquisition of altitude values (often by satellite imagery). These holes trap the flow, which yields inconsistent and resolution dependent drainage. Several methods try to solve the problem by modifying the heightfield and filling the holes (Depression Filling, Barnes et al. 2014) or carving breaches to allow for the water to flow out (Depression Carving, Lindsay 2016).

Recent work (Barnes et al. 2014) uses a priority queue to virtually flood the terrain. The algorithm starts at the boundaries of the domain Ω, and progressively parses the nodes in the order given by the priority queue, which is filled by the neighbors of the parsed nodes. When unparsed nodes have lower elevation than parsed ones, a depression is detected and filled. This method was recently improved by considering less nodes in the priority queue or with parallel implementation (Barnes 2016; Wei et al. 2018; Zhou et al. 2017), but is still bounded by a $O(n \log n)$ complexity.

Braun et al. 2013 proposed a $O(n\sqrt{n})$ step to achieve correct drainage computation. Although less efficient, this algorithm is straightforward to plug in the stream graph formulation, as opposed to the state of the art algorithms which need to be applied as an invasive preprocess to the elevations. We built on both approaches and developed a linear time algorithm, adapted to the stream graph.

This algorithm takes advantage of the fact that the graph formed by connecting all neighboring basins is planar. Computing water outflow can be translated to a minimum spanning tree problem on this graph, whose planarity allows for a linear time solution.

Abstraction of the lake flows. Lakes are located at the root nodes $\mathcal{N}_l \in \mathcal{T}$ that are not on the boundary of Ω (local minima). We assign the unique identifier $L(\mathcal{N}_k) = l$ to $\mathcal{N}_l$ and to all the nodes n_k belonging to the same tree (*i.e.,* in the drainage area of $\mathcal{N}_l$), to represent

water passing through these nodes and flowing to the lake n_l. We call this set of nodes a *basin*. As the water level rises, some of these upper nodes will be absorbed by the lake before water overflows.

Our goal is to model overflow between these different lakes down to the river mouths. We create a directed super graph of lakes $\mathcal{G}_L$, where each basin (group of nodes in $\mathcal{G}$ with the same identifier $L(\mathcal{N}) = l$) is represented by a single node named from its identifier l. Extra nodes are created to represent river mouths (root nodes at the border of Ω) and their stream tree.

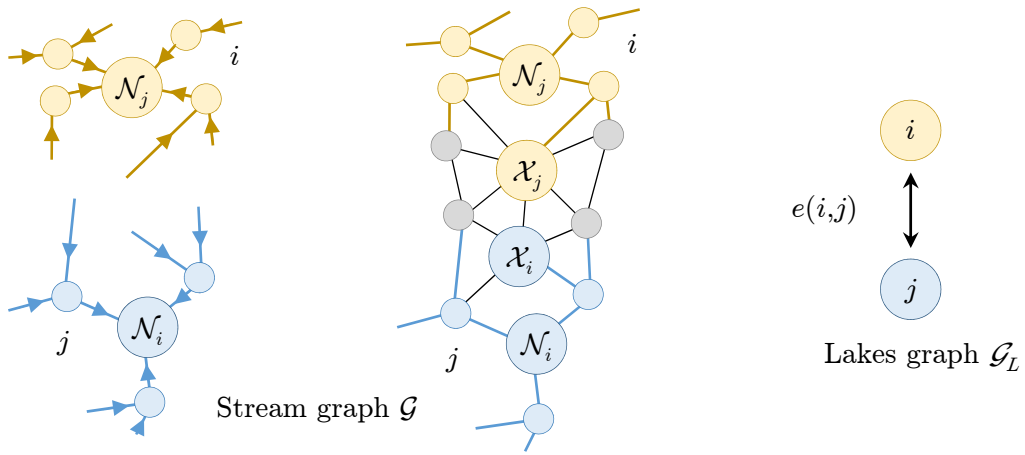


Figure 3.4: Left: nodes $\mathcal{N}_k$ belonging to the stream tree of root $\mathcal{N}_i$ (yellow) share the basin identifier $L(\mathcal{N}_k) = i$. Middle: a pass, defined as the lowest pair $(\mathcal{X}_i, \mathcal{X}_j)$ connecting two lakes in $\mathcal{G}$. Right: Lakes are abstracted as nodes of G_L, connected by an undirected edge if there exists a pass between them.

We call a *pass* the lowest pair $(\mathcal{X}, \mathcal{Y}) \in \mathcal{G}$ of two neighboring nodes $\mathcal{X}$ and $\mathcal{Y}$ belonging to different basins $L(\mathcal{X}) \neq L(\mathcal{Y})$. Formally, a pass $(\mathcal{N}_i, \mathcal{N}_j)$ between two basins identified by l_0 and l_1 needs the height of its nodes to satisfies: $\forall k, l$ such that $L(\mathcal{X}_k) = l_0$ and $L(\mathcal{X}_l) = l_1$, $\max(h_i, h_j) \leq \max(h_k, h_l)$. This $\max(h_i, h_j)$ is called the pass height. The pass is a connection between both basins; it corresponds to a saddle point in terms of elevation in the stream graph $\mathcal{G}$. Basins are connected by adding an arc $(L(\mathcal{X}), L(\mathcal{Y}))$ for each pass $(\mathcal{X}, \mathcal{Y})$ as shown in Figure 3.4. Because it is not yet clear in which direction the water will flow, the arc is undirected.

Extraction of the lake connections. We solve the problem of flow routing by selecting and orienting some of the edges in $\mathcal{G}_L$. Without loss of generality, we assume the existence of a single boundary basin (if more exist, they can be connected together with a elevation pass of altitude $-\infty$).

The key observation is that the water will use a path of minimum energy to flow from any basin to the boundary. Thus, all the edges followed by the water flow are connected, and form no loop: these edges can be regrouped in a tree $\mathcal{T}_L$ covering $\mathcal{G}_L$. The potential energy needed for the water to flow from a basin into another one is proportional to the altitude value of the pass between both basins. Minimizing the total energy of the system is then

equivalent to finding $\mathcal{T}_L$, the minimum spanning tree of $\mathcal{G}_L$ (Figure 3.5), where the weight of the edges of $\mathcal{G}_L$ are the altitude of the associated passes.

Figure 3.5: Illustration of the water flow resolution on a synthetic case. The initial terrain is computed thanks to a *Voronoi* diagram: a set of points at random locations are set to altitude 0, and the elevation is obtained by locally computing the distance to the nearest sample point (left). The basins are first segmented and each pair of basins are connected through a single pass (middle). Then, the minimum spanning tree algorithm is used to select some of these connections, giving the final flow direction. In the rightmost figure, the basins undergo a moderate erosion and we show the water level in blue.

Several algorithms exist to find a minimum spanning tree. In (Cordonnier et al. 2016), we used a modification of *Prim's algorithm*, but without properly introducing the minimum spanning tree formalism. Following *Kruskal's algorithm*, it is also possible to order the edges of $\mathcal{G}_L$ based on their weight, and parse them thought an *Union-Find* data structure to select the branches of the tree by progressively growing connected components.

We note that $\mathcal{G}_L$ is planar by construction: each basin of $\mathcal{G}_L$ corresponds to a connected region of nodes of $\mathcal{G}$, and the edges are dual to the borders between neighboring basins. Mareš 2002 proposes an algorithm to computed minimum spanning trees of planar graphs in linear time. The idea is that half of the nodes in a planar graph have at most 8 neighbors. *Boruvka's algorithm* is then modified to computed the minimum spanning tree (Algorithm 3.1).

Algorithm 3.1: Minimum Spanning Trees on planar graphs (Mareš 2002).

```
while There remains nodes in G_L do
    while There is a basin B that has less than 8 neighbors do
        Add the edge with the lowest neighboring pass to T_L;
        Contract the edge (if the edge links the basins b and c, remove b and append
            all remaining neighbors of b to c);
    end
    Clean the graph: bucket sort all edges lexicographically to remove parallel edges;
end
```

Note that the obtained tree is undirected, *i.e.,* it shows where the water flows but not the flow direction. This information can be obtained by *rooting* the tree: we start at the outflow, and then parse the tree, progressively setting parsed node as outflows of their unparsed neighbors.

Stream tree correction. We generate a new version of $\mathcal{T}$, denoted as $\widetilde{\mathcal{T}}$, which includes the pass information from the previous step. Recall that a basin of $\mathcal{T}_L$ is associated to a local minimum $\mathcal{N}_i$ in $\mathcal{G}$. Moreover, an arc $e(i,j)$ of G_L is associated to a pass $(\mathcal{X}_{in}, \mathcal{X}_{out})$, where $\mathcal{X}_{out}$ is a node of $\mathcal{G}$ in the direction of the outflow of lake i (toward lake j).

The simplest correction that can be applied to $\mathcal{T}$ is the addition of a new directed arc $e(\mathcal{N}_i, \mathcal{X}_{out}) \in \mathcal{G}$ for each arc $e(i,j) \in \mathcal{G}_L$ (Cordonnier et al. 2016, Figure 3.6). Note that extra care must be taken when $\mathcal{X}_{in}$ is higher than $\mathcal{X}_{out}$: in that case, $\mathcal{N}_i$ is set to flow in $\mathcal{X}_{in}$ and $\mathcal{X}_{in}$ flow in $\mathcal{X}_{out}$.

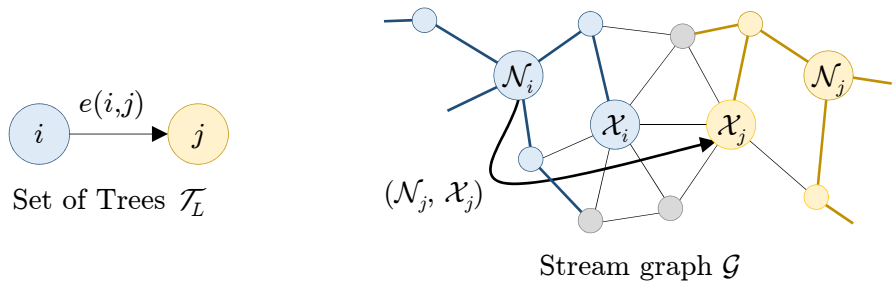


Figure 3.6: An arc $e(i,j)$ in $\mathcal{T}_L$ (left) is converted into an arc in $\widetilde{\mathcal{T}}$ by connecting the bottom of the lake i ($\mathcal{N}_i$ in $\mathcal{G}$) to the pass belonging to lake j ($\mathcal{X}_j$ in $\mathcal{G}$).

This simple solution has an important drawback. The water directly jumps from the local minimum to the pass, which breaks the connectivity of the stream tree. This impacts the implicit solution of the stream power equation: if a time step of large erosion is applied, only the pass node is eroded (and the water level decreases), while we would want a succession of neighbors of the pass nodes to be eroded as the water level decreases. For that, we propose two other algorithms, one parented to *depression carving* (Lindsay 2016) and another to *depression filling* (Barnes et al. 2014).

The idea behind the depression carving strategy is to force the flow to follow the shortest path from the local minimum to the pass. The *carving* metaphor comes from the fact that applying a large time step of erosion would carve a thin, deep trench at the ouflow of the lake. This shortest path is already defined by the receivers, but in the reverse order: from the pass to the bottom of the lake. The correction algorithm requests a parse of the successive receivers between the pass and the local minimum, progressively reverting the receivers.

The depression filling strategy (Algorithm 3.2) makes the assumption that the depression is filled by some material. We compute receivers by filling the depression, starting at a pass and progressively connecting all the unparsed neighbors to the parsed ones, in a breadth first order, as long as the nodes are below water level (pass altitude). The parsing of the basins is ordered from sea to crest, to ensure that the water level is accurate. The receiver of a parsed node is chosen with respect to a cost function, which we define as the minimal distance between a node and the pass. This choice does not give the perfect ordering that would be given by an Euclidean distance function with obstacles on a regular grid, but is

simple and accurate enough.

Algorithm 3.2: Depression filling.

```
receiver(N_in) = N out ;
queue.append(N_in) ;
while queue not empty do
    N = queue.pop_front() ;
    cost(receiver(N_in) = ∞ ;
    foreach neighbors N_nb of N in the basin, such as h_N_nb ≤ h_N do
        if N_nb is not in queue then
            queue.append(N_nb);
        else if N_nb has already been parsed and cost(N_nb) < cost(receiver(N)) then
            receiver(N) = n_nb;
    end
end
```

In practice, we prefer the filling strategy as the resulting erosion patterns more easily leads to natural mountain features. The carving strategy would be beneficial in situations where the flow information below water level needs to be preserved, for instance if the algorithm was to be improved with sedimentation. The effects of these strategies are shown in Figure 3.12.

3.3 Erosion

This section describes the last two steps of our iterative algorithm: drainage and slope computation, and solving the stream power Equation (3.1).

Drainage and slope. Let $\mathcal{N}_k$ denote a node of the graph-covering stream trees $\widetilde{\mathcal{T}}$ and $C(\mathcal{N}_k)$ the set of its children nodes. The drainage area A_k can be computed using the recursive formula:

$$A_k = a_k + \sum_{x_l \in C(\mathcal{N}_k)} A_l.$$

To compute A_k efficiently, we perform a breadth first traversal of the tree, storing each node in parsing order in a set P. Then we compute the drainage area for each node of P, parsed in the reverse order, *i.e.,* from leaves to the root (Figure 3.7). This enables us to compute A in linear time.

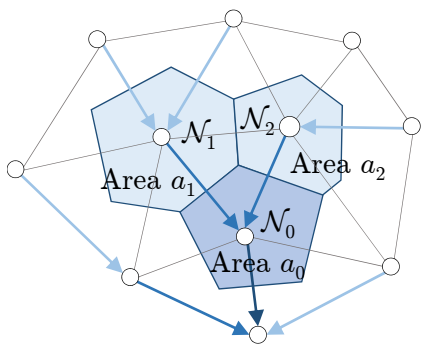


Figure 3.7: Drainage area of the leaf node $\mathcal{N}_1$ is $A_1 = a_1$, where a_1 is the area of the cell surrounding the vertex. For the node $\mathcal{N}_0$ of lower elevation, $A_0 = a_0 + A_1 + A_2$.

Given a node $\mathcal{X}_k$, we use its receiver $\mathcal{X}_l$ stored in $\widetilde{\mathcal{T}}$ to compute the slope. Let $\|\mathbf{p}_k - \mathbf{p}_l\|$ denote the distance between nodes $\mathcal{X}_k$ and $\mathcal{X}_l$ located at points $\mathbf{p}_k$ and $\mathbf{p}_l$ in the horizontal plane, we have:

$$s(\mathbf{p}_k) = \frac{h_k - h_l}{\|\mathbf{p}_k - \mathbf{p}_l\|}$$

Solving the stream power equation. The stream power Equation (3.1) can be solved efficiently by using an implicit scheme. For a node $\mathcal{X}_i$ of receiver $\mathcal{X}_j$, it can be rewritten as:

$$\frac{h_i(t+\delta t) - h_i(t)}{\delta t} = u_i - kA_i^m \left(\frac{h_i(t+\delta t) - h_j(t+\delta t)}{\|\mathbf{p}_i - \mathbf{p}_j\|} \right)^n$$

Assuming $n = 1$ (Section 3.1.1), ans setting $K = \dfrac{\delta t\, k\, A_i^m}{\|\mathbf{p}_i - \mathbf{p}_j\|}$ this equation can be solved as follows:

$$h_i(t + \mathrm{d}t) = \frac{h_i(t) + \delta t\, u_i + K\, h_j(t+\delta t)}{1+K} \tag{3.2}$$

This scheme requires that $h_j(t+\delta t)$ should be computed before $h_i(t+\delta t)$, which is made possible by parsing the previously computed trees from root to leaves. Thus, the implicit solver has an $O(n)$ complexity.

Correction based on thermal erosion. While the simulation of the stream power equation works efficiently for carving the bottom of the rivers, other phenomena may be predominant in some cases, in particular for low drainage areas. In such cases, the stream erosion equation produces unrealistic sharp and high peaks.

We correct this effect by using a thermal erosion mechanism (Musgrave et al. 1989). Thermal erosion embeds the set of processes that causes rocks to break because of the thermal shocks caused by the infiltrated water and changes in temperature. We take this process into account by limiting the slopes to a prescribed value (usually 30°).

Stability and Convergence. The implicit solver ensures unconditional stability, enabling very large time steps. But the maximal value for a time step is bounded by the accuracy of the solution, because of the changes of connectivity of the stream tree. In practice, values up to 10^5 years work fine, for grid containing up to $1,000^2$ cells.

We aim to reach topographic steady state, as this results into plausible fully grown mountain ranges. While tracking changes in the topology of the stream graph is an accurate predictor, we often cut down the number of iterations (in practice to $100-300$ steps) because the system quickly gets in a state visually similar to the steady state. Future work is needed to study local steady state (in the extent of the valley), as a visual steady state criterion.

3.4 Results

Our system is developed in C++ and uses OpenGL and GLSL for rendering. High quality image output were directly streamed to Vue 2015® (http://www.e-onsoftware.com). All examples in this paper were created on a desktop computer equipped with an Intel Core i7 CPU, clocked at 3GHz with 16GB of RAM.

In all experiments, unless stated differently, we use a terrain size of $50 \times 50\,\mathrm{km}^2$. We set the maximum tectonic uplift to $\mathcal{U} = 5.0\,10^{-4}\,\mathrm{my}^{-1}$ (meters per year), which is the average uplift among earth mountains. The erosion rate depends on many factors, such as precipitation and rock strength. In order to get a more intuitive setting, we follow the relationship between height, uplift, and erosion detailed in Section 3.4.5. We set the erosion rate to

$k = 5.61\,10^{-7}\,y^{-1}$ for mountains to culminate at about 2 000 m. We set the time step at the geological scale $\delta t = 2.5\,10^5$ y to ensure a fast convergence while avoiding the appearance of high unnatural cliffs.

3.4.1 Visual realism

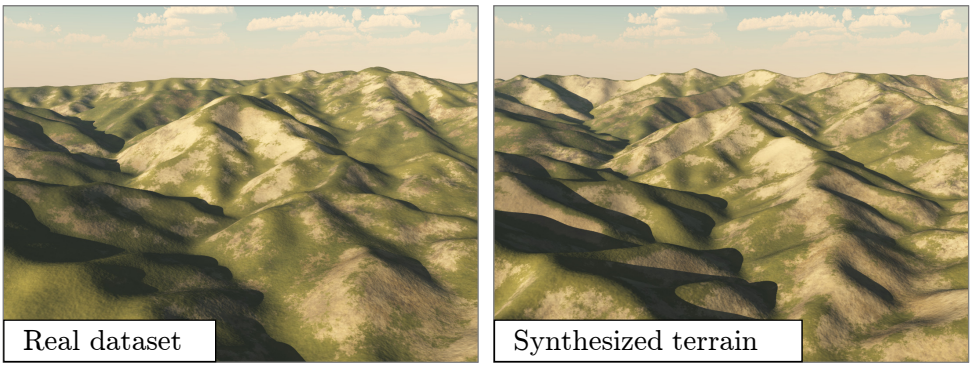


Figure 3.8: Comparison of a real terrain and a digital model produced by our fluvial erosion simulation.

We compared our stream-erosion simulation to real mountain data sets where fluvial erosion is the dominant factor: the San Gabriel mountains in California (data from http://www.usgs.gov/). Figure 3.8 shows a side by side comparison of real terrain and a result produced by our method. This confirms that our method can successfully generate coherent and plausible large scale dendritic patterns.

3.4.2 Rendering

We implemented three methods for rendering of our terrains. The first two approaches lend themselves for interactive modeling and provide a fast visual feedback, whereas the third method relies on a more computationally demanding conversion of the stream graph data-structure into a set of procedural primitives and targets visualization at a high level of detail.

The first interactive rendering method is achieved by generating a mesh from the stream graph by using Phong tessellation (Boubekeur et al. 2008). We flatten the shading of the edges traversed by a stream to emphasize the path of water, and we color the nodes depending on the drainage area to show the river network. We also visualize lakes by comparing the pass height with the height of all the points flowing into the bottom of the lake. An example is shown in Figure 3.9.

The second interactive rendering method consists in defining the surface of the terrain as an elevation function defined as the sum of Gaussian kernels centered at each node multiplied by the node height. The height is then normalized by the sum of the kernels at that point. This results in a smoother geometry than the Phong tessellation, but it is harder to emphasize the water network.

The third method is not interactive because it is based on a high level of detail terrain representation. This representation is obtained by converting the stream-graph model into

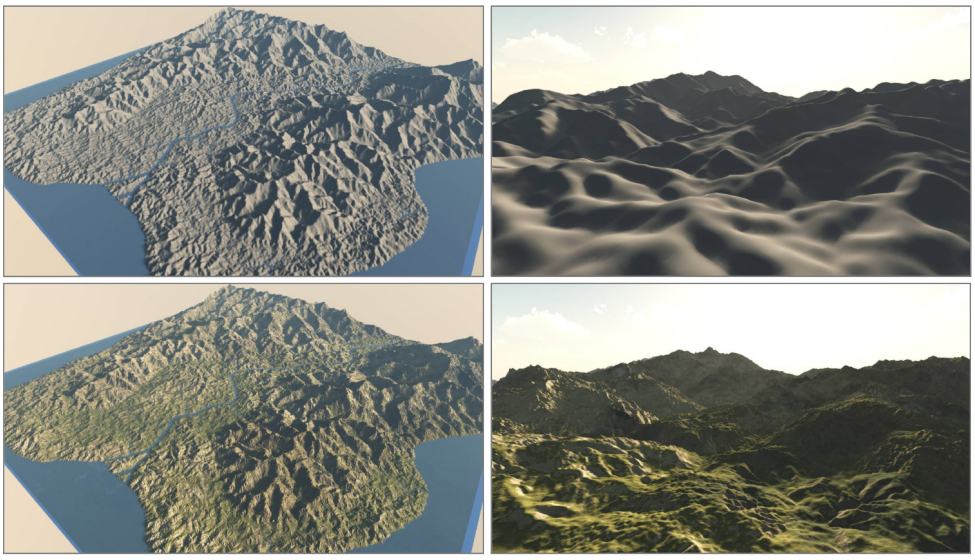


Figure 3.9: Comparison of two of the rendering methods used in our approach. The images show a far view (left) and a close-up of real-time tessellation (top) and procedural primitives (bottom).

a hierarchical primitive-based model, as described in (Génevaux et al. 2013; 2015). This model combines parameterized terrain primitives representing the different landform features (ridges, valleys, and rivers) into a hierarchical construction tree. This identification of landform features and the selection of the corresponding primitives is performed automatically by analyzing the coarse terrain elevation map produced by the stream power erosion process as described by Guérin, Digne, Peytavie, et al. 2016. Rivers are carved by using the elevation and drainage information embedded in the graph (Figure 3.9).

3.4.3 Performance

# samples	One time step (s)	Time to steady state(s)
10 000	0.031	6.4
40 000	0.177	35.4
90 000	0.401	78.0
160 000	1.273	252.0

Table 3.1: Simulation time as a function of the number of samples. Convergence is obtained after around 200 time steps.

Table 3.1 reports the performance of our method as a function of the number of sampling points. Although we did not fully optimize the implementation, our method provides interactive feedback at every time step which enables us to visualize a simulation at interactive rates and to tune parameters.

The number of iterations needed to obtain a fully-formed mountain chain is difficult to estimate because it depends on a number of input parameters (see the discussion below).

In our experiments, the mountains where fully shaped after 50 iterations and the geometry stops evolving after $100 - 300$ iterations, as shown in Figure 3.10. A solution to accelerate convergence could be to progressively refine the sampling grid. Note that this refinement needs to be uniform, in order to preserve the possible emergence of local stream and the details in the shape of rivers.

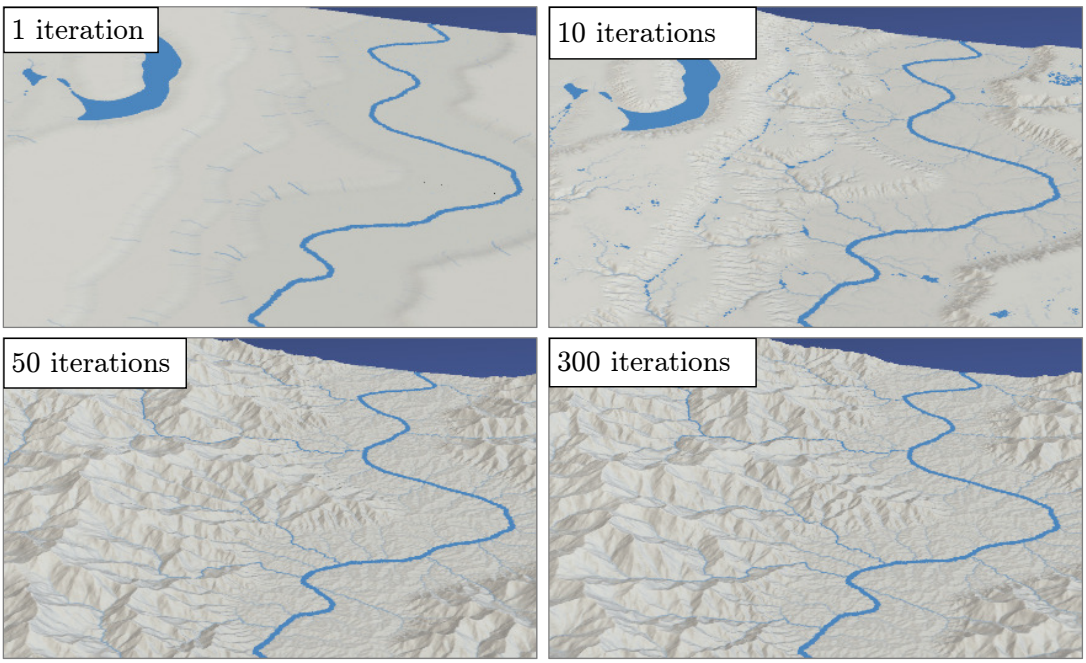


Figure 3.10: Different steps of our fluvial erosion algorithm. The mountains are formed at ~ 50 iterations, and steady state is reached after 100 to 300 steps.

3.4.4 Lake overflow

Failing to capture the flow in local minima leads to an important under-prediction of the eroded volume, as shown by Figure 3.11. A 1024×256 terrain is filled with a 500 m flat areas augmented with a small random perturbation, at the exception of the leftmost column, which is set at altitude 0 and exclusively defines the boundary. The erosion simulation is run 70 times with $1,000$ years time steps, with and without local minima correction. As shown by Figure 3.11, ignoring local minima results in a slower migration of the cliff. This is explained by a lower average drainage area, which starts at the cliff edge and does not cover all the remaining parts of the landscape, and by an inaccuracy in the implicit solution on large drainage: the erosion is limited in distance by a new local minimum at each step.

A simple experiment is performed to show the impact of the different methods for correcting receivers on an erosion time step. This example has been built on a regular grid with an 8-neighboring connectivity, but the results are similar on an irregular triangular network. The initial terrain is made of an inverted pyramid in a 100×100 regular grid, with a prescribed slope set to $45°$. A single node at the middle top is given altitude 0 (also the altitude of the base of the pyramid), and is set to be the boundary node (Figure 3.12 (top left)). A single

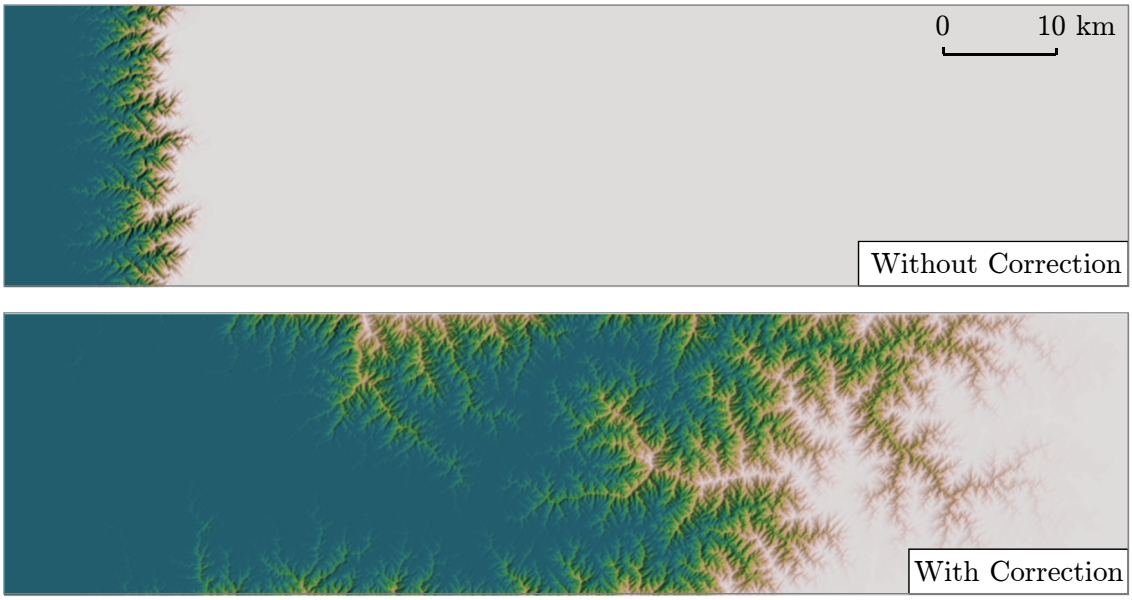


Figure 3.11: Simulation of erosion of 500 m high escarpment on a 1024×256 terrain. The nodes of the leftmost column are the only 0 altitude boundary nodes. We show the result of 70 time steps of $1,000$ years each, without (top) and with (bottom) connecting the lakes.

time step of $5,000$ years of erosion is performed with three different strategies for correcting the receivers in the flooded area:

Simple correction: the receiver of the local minimum is directly set to the higher node of the pass (here one of the neighbors of the outflow), and the receiver of this node is assigned to the outflow node. This simple strategy does not benefit from the implicitness of the stream power equation resolution: only the pass node is widely eroded due to the large drainage of the basin (Figure 3.12 (top right)).

Carving: the path obtained by following the receivers from the pass to the local minimum is inverted, as if a trench had been carved. The resulting erosion follows this path, actually carving a subset of this hypothetical trench (Figure 3.12 (bottom left)).

Filling: the receivers of the flooded area are all modified as if the water surface was replaced by a small slope. This results in a star like pattern centered at the outflow (Figure 3.12 (bottom right)). The number and disposition of branches of the star are due to the 8-neighboring lookup for the cells.

A particular attention should be drawn to the performance of the algorithm introduced to solve the flow routing problem (Section 3.2.3), because this part was the main bottleneck of previous implementations Braun et al. 2013 and Cordonnier et al. 2016. We compare our algorithm (Linear time version using (Mareš 2002) and Kruskal 's algorithm for computing a minimum spanning tree) with state of the art depression filling techniques (Barnes et al. 2014; Wei et al. 2018, Figure 3.13). First, we show how the algorithms perform on a fixed size $16,000,000$ nodes terrain, with an increasing number of local minima. The initial terrain

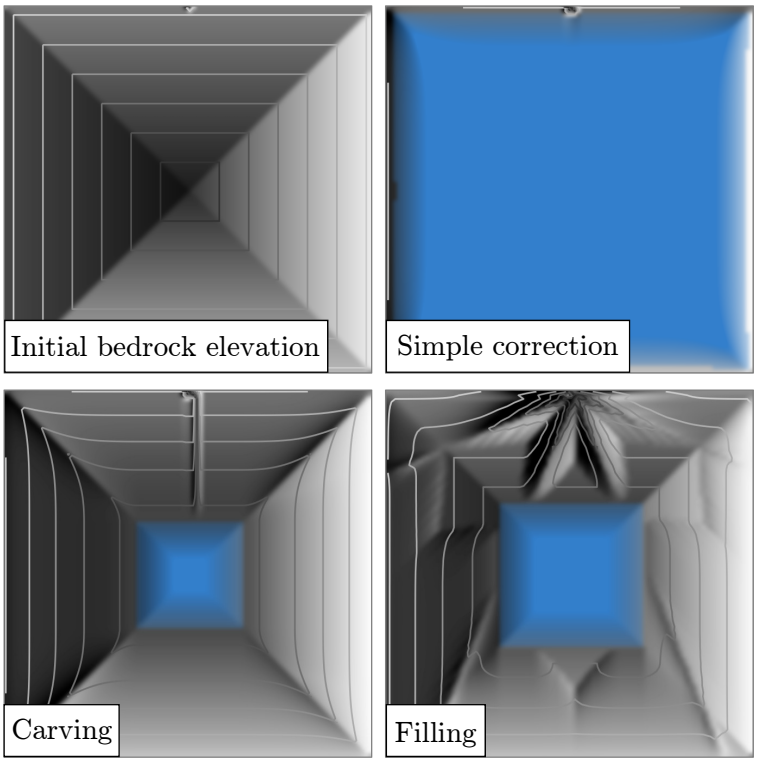
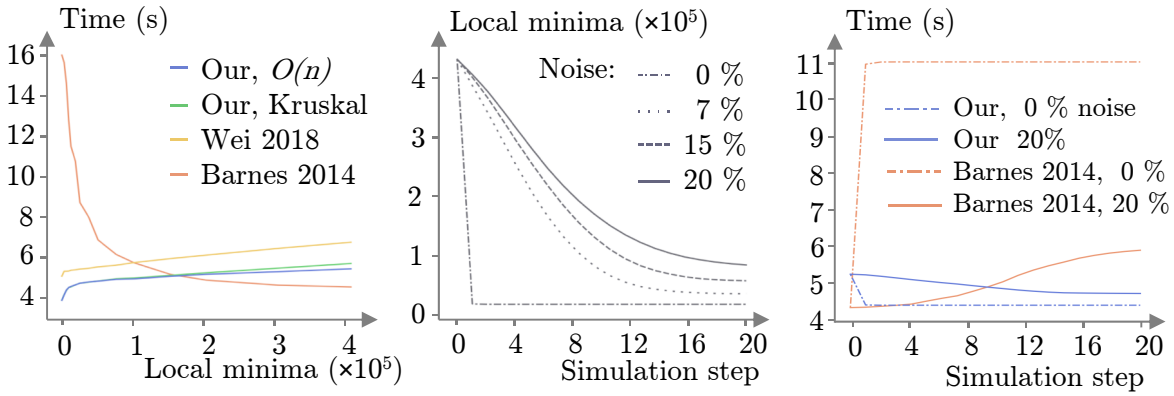


Figure 3.12: Demonstration of the effects of receivers correction on erosion (single time step of $5,000$ years of erosion). The initial terrain is a square hole with constant slopes on the sides (top left with contour plot), with a single boundary node at altitude 0 on the middle of the upper edge. The simplest correction connects the local minimum to the pass (top right). Depression carving (bottom left), and depression filling (bottom right). Water level after erosion is shown in blue. This example was generated on a 8-neighborhood regular grid, which explains the star-like pattern obtained after applying the depression filling strategy.

is at steady state without depressions, and we progressively add distinct artificial single-node holes, up to 2.5% of the total number of nodes. The behavior of our results is similar to the ones of Wei et al. 2018, but the algorithm from Barnes et al. 2014 shows a decrease of computation time when the number of local minima increases. This is consistent, because they use a priority queue only for nodes outside of a depression. When the number of nodes increases, the simplicity of this algorithm makes it very efficient. In order to choose which algorithm to use, we show the evolution of number of local minima on the 20 first steps of erosion, starting from a flat terrain. We study the reaction to different uplifts, from uniform to uniform plus 20% of Perlin noise. For all these uplift functions, the number of local minima decreases and stabilizes after around 16 steps. The performance per step is obtained by combining these two studies, and shows that for a large enough number of iterations, our algorithm may be preferred.



(a) One step performance **(b)** Local minima **(c)** Per step performance

Figure 3.13: We evaluate the performance our algorithm using either the $O(n \log n)$ *Kruskal* algorithm, or the $O(n)$ algorithm proposed by Mareš 2002 for computing the minimum spanning tree. These are compared with state of the art algorithms from Barnes et al. 2014 and Wei et al. 2018 for a single erosion time step on top of a $16,000,000$ terrain already pushed to steady state. The execution time is measured after adding distinct local minima artificially, from 0 to 2.5% of the total number of nodes (a). To relate these performances to an erosion simulation, we show the number of local minima (b) for the 20 first erosion steps where a uniform uplift of 5 $mm\ years^{-1}$ has been augmented with a pseudo-random noise which amplitude varies from 0 to 20% of the global uplift. Combining both gives the execution time per steps of the our $O(n)$ algorithm and (Barnes et al. 2014) with different noise proportions.

3.4.5 Stream power erosion

Figure 3.14 shows the result of the stream power equation erosion, combined with small scale details. on top of generating the large scale topography resulting from the combination of uplift and erosion, or method generates an hydrology network consistent with the terrain elevations. This network can be computed more generally on any input elevation.

The **erosion parameters** in the stream power erosion are not intuitive to set. Even in geology, the impact of these coefficients is not well-understood. The erosion coefficient k and the uplift u are both subject to a multiplication by $\mathrm{d}t$, so only their ratio is relevant. However, its value has a strong influence on the mountain height. We made a series of experiments in order to find a relationship between this ratio and the maximum mountain height. It turned out this relation is linear and the height in kilometers follows the rule $h_{max} = 2.244\, u/k$.

In our approach, we allow only the changes to the drainage area exponent m. Indeed, only the ratio between m and n has a meaning that we can deduce from the equations. Let us suppose we have reached an equilibrium state in a region where u and k are constant over the space. The stream power Equation (3.1) becomes:

$$\frac{\mathrm{d}h(\mathbf{p})}{\mathrm{d}t} = 0 = u(\mathbf{p}) - k\, A(\mathbf{p})^m\, s(\mathbf{p})^n.$$

From that, we have s and A proportional:

$$s \sim A^{-m/n}.$$

We note x the distance between $\mathbf{p}$ and the corresponding outflow, and d the distance between ep and a ridge. Then we assume that A is proportional to $(d-x)^2$. Recall that $s = dh/dx$, so we obtain:

$$\frac{dh}{dx} \sim (d-x)^{-2m/n}.$$

After integration, and imposing $h(0) = 0$, we have:

$$h \sim \begin{cases} d^{1-2m/n} - (d-x)^{1-2m/n} & \text{if } m/n < 1/2 \\ log(d) - log(d-x) & \text{if } m/n = 1/2 \\ \dfrac{1}{(d-x)^{2m/n-1}} - \dfrac{1}{(d)^{2m/n-1}} & \text{otherwise} \end{cases}$$

We use the result of these equations to choose the proper ratio m/n to shape the desired river profile.

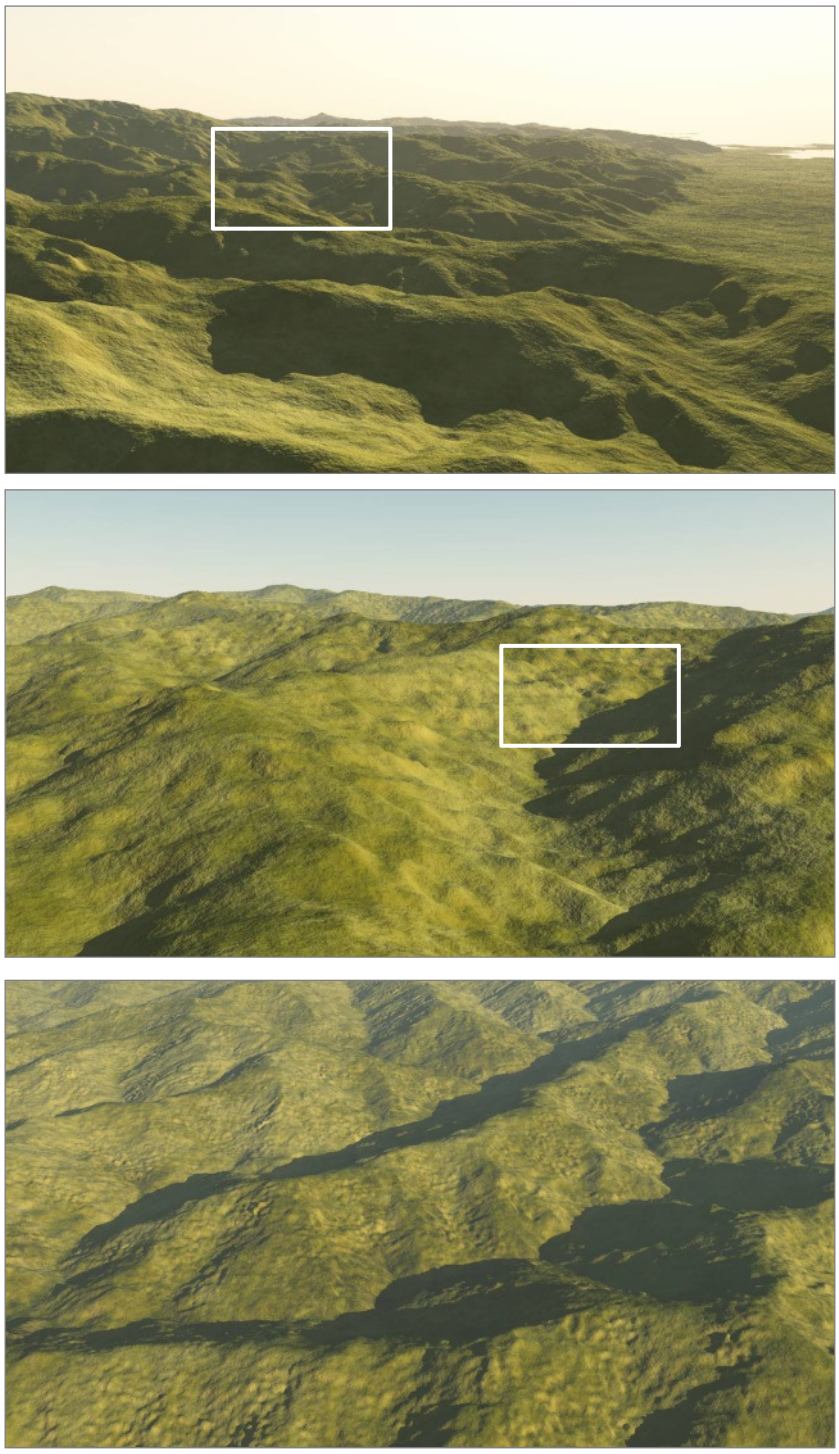


Figure 3.14: The graph representing the terrain was rendered by using a set of function based primitives corresponding to different landform features which are parameterized by the data embedded in the stream graph (Génevaux et al. 2015). This method provides varying level of detail as can be seen in this successive zoom (from top to bottom).

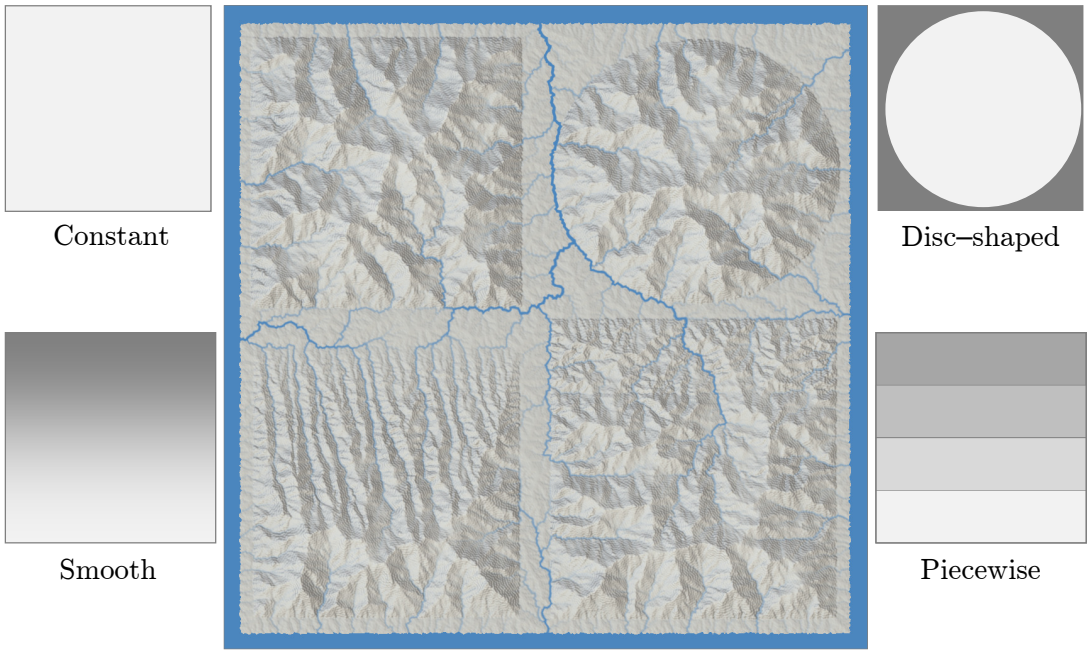


Figure 3.15: The uplift gradient has a strong influence on the resulting terrain.

The uplift has a strong influence on the resulting terrain and it is the main way the user affects the final shape. As shown in Figure 3.15, the main impact is in the gradient of the uplift values. The actual shape of the uplift does not have a strong influence, except on the boundaries. Having the same uplift shape, a slowly decreasing gradient leads to a very straight erosion in the gradient direction, whereas a set of steps of constant values gives more random valleys with sudden jumps on the gradient in the mountain heights. We can also observe that lower values of uplift lead to smaller valleys, and that the thermal erosion is important with regular slopes for high uplift values.

The **thermal erosion** has an important influence on the shape of the valleys. It affects their regularity as shown by Figure 3.16. If the maximum slope given for the mountain is 30°, which is the usual talus angle for thermal erosion (Musgrave et al. 1989), our erosion model results in a layout of very regular geometric valleys.

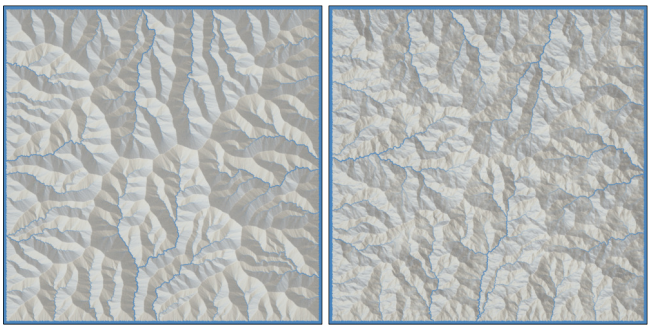


Figure 3.16: Uniform maximum talus angle gives regular pattern (left), whereas modulating the value by 3D Perlin noise gives more randomized results (right).

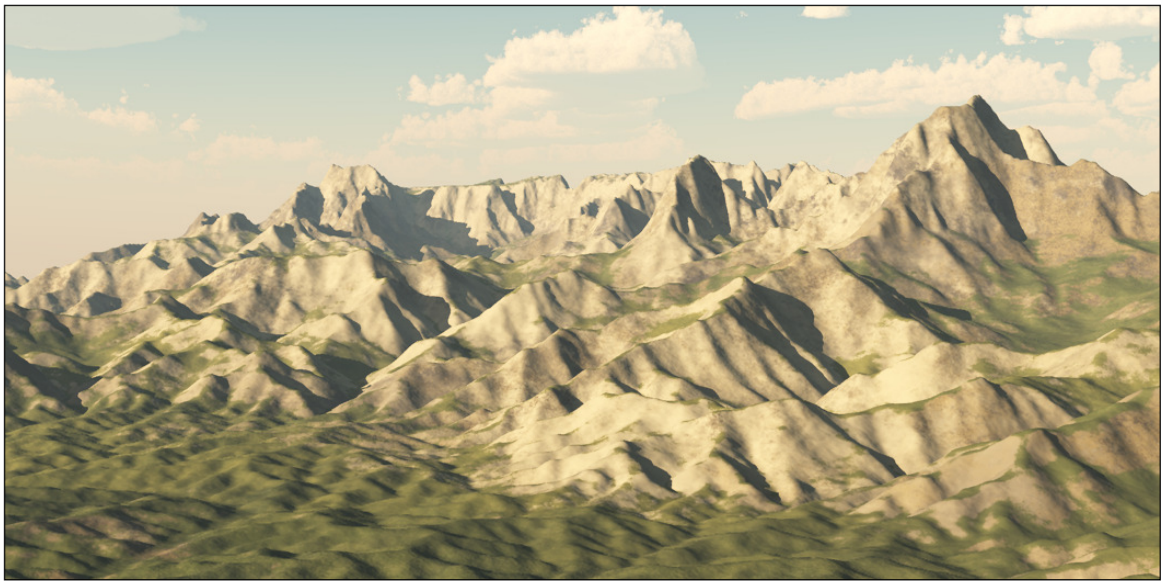


Figure 3.17: Choosing different maximum slope depending on the mountain height gives plausible cliffs effect.

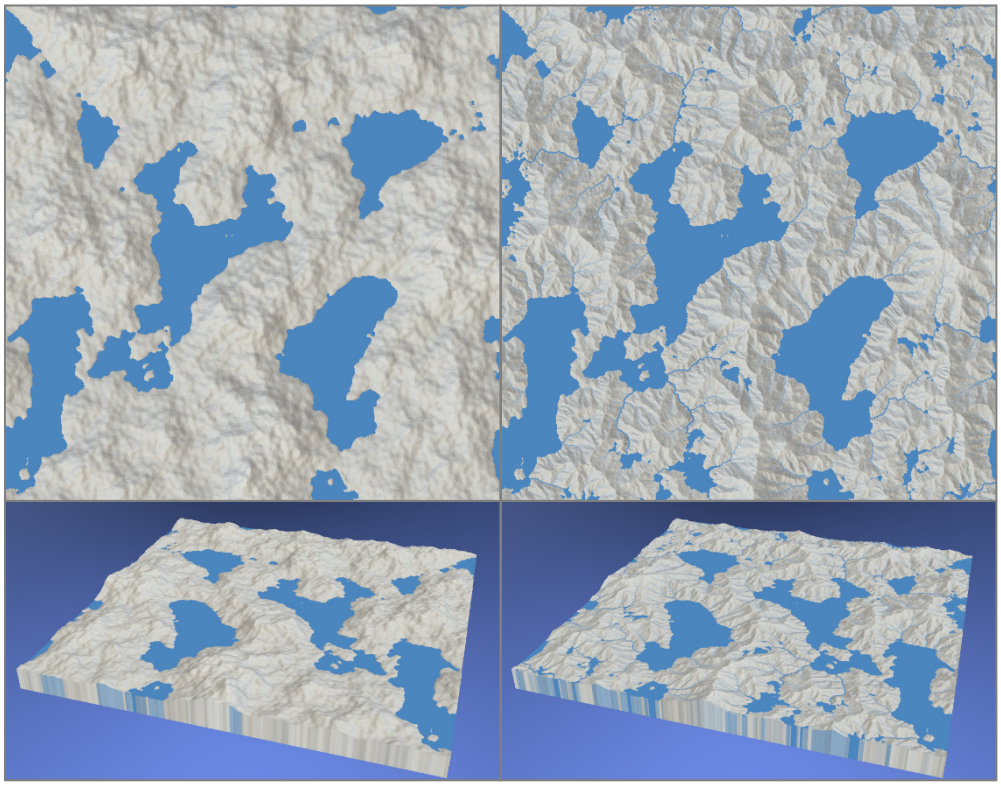


Figure 3.18: A fractal terrain (left) after the application of the stream power erosion (right). Note the creation of new rivers and lakes.

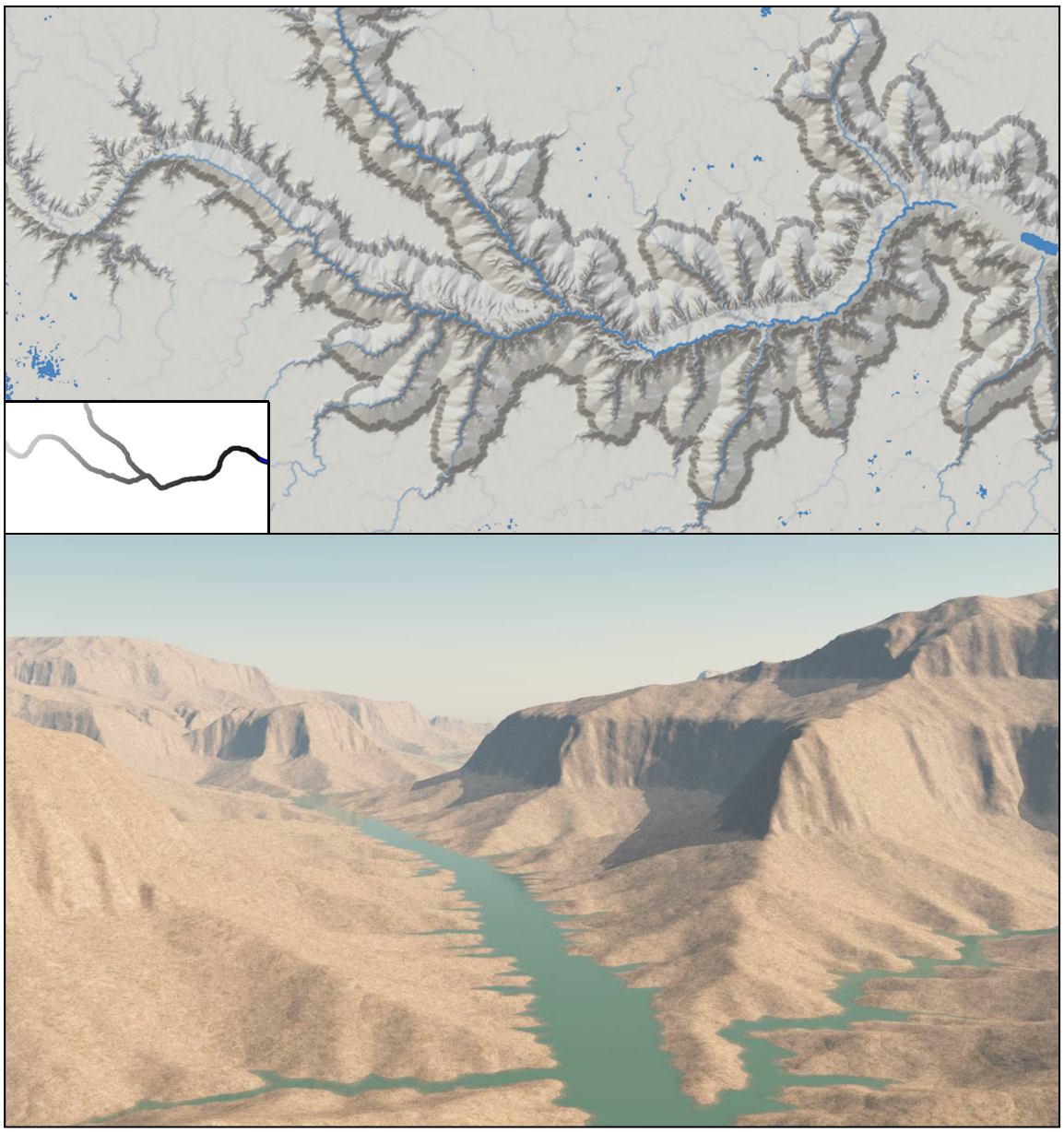


Figure 3.19: A flat terrain initialized with a carving river (top left) has been eroded with height dependent slopes (close-up at bottom).

We experimented by forcing the maximum slope to follow a 3D Perlin noise with a high persistence, to account for local different rocks strength. We choose the minimal and maximal slope angles to be 6° and 54° respectively. This results in a more random distribution of the erosion patterns in valleys as shown in Figure 3.16.

Furthermore, we can obtain interesting features by procedurally adjusting the thermal slope. Figure 3.17 shows a landscape with small thermal slopes below a given height, but higher slopes above it. This adds cliffs to the crests, which is typical for many mountains.

Stream power erosion without the uplift. The stream power erosion can be used without the uplift and it adds a global hydrological realism to an existing scene as shown on an example of a fractal terrain enhanced with erosion in Figure 3.18. As the erosion converges toward a flat terrain, it is necessary to use small time steps and to stop the simulation after only a few iterations.

Figure 3.19 shows that this method can model a large variety of landscape: starting from a simple height map formed with two strokes and a gradient, we obtain a plausible canyon. We chose a height dependent maximal slope for thermal erosion to obtain the succession of cliffs and slopes in the result.

3.5 Conclusion

We presented a method for terrain generation that takes into account large scale fluvial erosion. This is achieved by simultaneously considering uplift and erosion, a commonly observed behavior of mountains in nature.

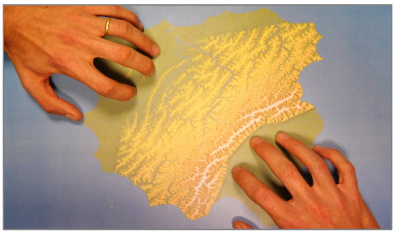
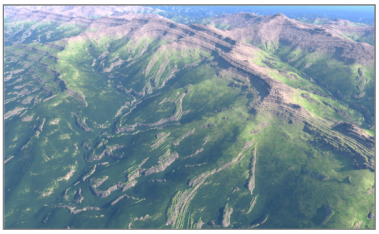
Contrary to previous erosion simulation methods, the user does not need to predefine an initial mountain on which erosion is applied, and which may affect the visual realism. In our approach the user paints a simple uplift map on a flat ground, enabling control of the shape of the main mountain ranges after a few iterations. Our simulation algorithm runs at interactive rates, and allows monitoring the results by tuning a single erosion parameter. Moreover, the erosion itself can be used without the tectonic uplift and it improves realism of existing terrain models. In addition to real-time visualization methods used during simulation, we can convert the vector data we compute to high-quality procedural terrain elements with detailed ridges and riverbeds.

Our algorithm has various limitations. Our validations are based only on visual observations and statements about the visual plausibility should be supported by some evaluation, for example by testing with human subjects. However, a difficulty of a fair comparison with real-world structures is that only main terrain structures should be compared, while users may base their visual comparison on details. This lack of evaluation was partially alleviated by the fact that this research has been conducted in collaboration with geologists, who provided at least some partial visual evaluation of the method. Another limitation is that the system behavior depends on the parameters that are not well-understood even in geology. While we attempted to provide meaning to those parameters and we document them meticulously in Section 3.4, a further insight into their values, dependencies, and effects could bring additional value to our approach. Some of those values could be, for example, measured in real terrains.

This erosion method will be reused in the next chapters, where we will see how to combine user gestures and state of the art knowledge on plate collision and crust folding to interactively build a more geologically accurate uplift map (Chapter 4). The fluvial erosion presented here will also be extended to take into account debris flow and hillslope erosion in Chapter 5, to be combined with glacial erosion.

Chapter 4

Interactive manipulation of tectonically driven uplift

Contents

4.1 Overview **60**
4.1.1 Plate tectonics in geology 60
4.1.2 Geologically-inspired interactive simulation 61
4.2 Earth crust as a viscous material **63**
4.2.1 Moving plates creation 63
4.2.2 Viscous compression 64
4.2.3 Uplift from thickness changes 65
4.3 Earth crust as layered sheets **65**
4.3.1 Folding of layered materials 66
4.3.2 Procedural fold generation 66
4.3.3 Uplift update from folds 69
4.4 Terrain surface generation **69**
4.4.1 Interactive terrain generation 69
4.4.2 Rock layers at the surface 70

4.5 Implementation, results and discussion **72**
4.5.1 Architecture 72
4.5.2 Qualitative and quantitative results 74
4.5.3 Validation and discussion 76
4.5.4 User study 80
4.6 Conclusion **80**

Earth surface in mountainous areas is mainly shaped by the competitive and simultaneous actions of the growth rate of mountains (uplift) and erosion (Chapter 3). The uplift is constrained by tectonic events acting on the earth crust through different phenomena, consistent among the various mountain ranges on Earth. Therefore, **accurately computing the uplift** is important when simulating mountain growth.

Geologists have developed models for mountain uplift; either focused on the mechanics of crustal deformation (Willett et al. 1993) or on computing the Earth's surface response to tectonic uplift (Braun et al. 2013). But their purpose was not to generate visually compelling 3D terrains at interactive rates. As we show in this chapter, bringing their observations to Computer Graphics can be used to improve the quality of generated landscapes by enabling the formation of large-scale landforms commonly observed in nature, while enabling direct control of the results. In particular, uplift systems consist in a variety of consistent faults and folds generated by tectonic events and dependent on subsurface geology, and are responsible for patterns that can be observed at various scales, from eroded cliffs to regularly spaced valleys.

In this chapter, we present a new volumetric model for the interactive generation of visually-compelling, large-scale terrains aimed at Computer Graphics applications. This is an extended version of the method published at TVCG 2018 (Cordonnier, Cani, et al. 2018). We address the long-standing challenge of providing the user with interactive control of large-scale terrains, such as shaping full mountain ranges, while automatically maintaining visual consistency with phenomena observed in nature, including the modeling of multi-scale folds. A key feature in our approach is the use of knowledge-based, phenomenological simulations instead of complex physical simulation, and we show that the loss in accuracy is well balanced by the gain in interactivity. We coin the term *geologically-inspired simulation* for these simulation methods that exploit first-order phenomenological knowledge from geology. We do not claim to reproduce the complexity of geological models (Braun et al. 2010), which incorporate formal mechanical calculations coupled with sophisticated erosion models but require heavy computations that prevent interactive control. Our geological approximations are justified by the interactivity of our sculpting system and by the targeted result: a geologically-plausible large scale eroded terrain surface that hides most of the higher order tectonic processes. Our framework could also be used for interactive illustration purposes in geology.

When tectonic plates collide, the Earth's crust is thickened in an incompressible manner, which causes the surface to be uplifted by isostasy: considering that tectonic plates float on the much less viscous and denser upper mantle like an iceberg floating on the ocean, any variation in crustal thickness must be accompanied by relative vertical motion of the plate's surface. Collision between plates often leads to the formation of faults or dipping shear zones that cut through the entire crust. Within the uplifting region, shortening is accommodated by smaller scale folding. The wavelength of the folds depends on the stratigraphy of the crust,

i.e., the thickness of layers previously deposited at the surface of the Earth and their relative competency. The surface uplift is balanced by *fluvial erosion* resulting in the formation of dendritic drainage systems and valleys. The rate of river incision is controlled in part by the competency of rocks, which means that the stratigraphy of the crust strongly influences the shape of river profiles and valley walls and may lead, for example, to the formation of local cliffs or ledges that stand out in the landscape.

Our multi-layered model of crustal material captures the combined effect of compression and folding. We consider that, at the scale of the crust, rocks behave in a viscous manner and deform by uniform shortening. This is similar to the thin-sheet approximation used in geodynamical models (Ellis 1996), based on the observation that, deep in the Earth's crust, rocks deform by ductile creep at very low strain rate. In the upper part of the crust faulting and folding take place. We use a procedural model to add multi-scale folds triggered by the compression field, based on the specific features of the chosen subsurface stratigraphy. The resulting deformation model is used to compute the uplift of the surface. This uplift is then used in a real-time erosion model, enabling interactive visualization of the resulting sculpting of the surface by erosional processes. The additional stratigraphic details of our model (smaller scale layers) enable us to generate multi-scale folds and produce the final, high quality terrain models used for final rendering.

Our method allows the hands-on interactive design of mountain ranges: the user can quickly draft and then progressively refine large scale terrains through gestural control, while ensuring that the generated mountain ranges are consistent with first-order geological processes. We choose sculpting control because it has long been one of the main expressive modeling metaphors (Galyean et al. 1991). In addition to physically-based virtual plasticine (Dewaele et al. 2004), sculpting systems have proven to be successful for editing as rigid as possible objects (Sorkine et al. 2007) or shapes with nested features (Stanculescu et al. 2013) as well as structured shapes, such as architectural models (Milliez et al. 2013). In a typical interactive session, the user sculpts and refines multi-layered earth-crust material as if it were plasticine, using displacement gestures applied using finger interaction on a multi-touch display. The first gestures define the tectonic plates and their relative motion. The resulting collision causes surface uplift and the growth of mountains, where the speed of growth is stored in an uplift map. The combination of the uplift and of an existing fluvial erosion simulation is visualized at interactive rates, enabling the user to refine his/her design. Enough data is generated to enable high quality rendering of the resulting terrain at different scales, showing the effect of a non-uniform input stratigraphy on valley flanks. These results could not be obtained with any example-based terrain synthesis methods because of the lack of stratigraphic information in scanned real terrain databases.

We validated our method by comparing our results to simulations from geology and to real terrain data. We show that various phenomena that take place at different scales are correctly captured, including the asymmetric large-scale shape of mountain ranges or the small scale, stratigraphically-controlled folds that appear along eroded cliffs. We also conducted a user study with artists and non-artists which confirmed the usability of our sculpting tool. Our contributions are:

- A new multi-layered model embedding stratigraphic information enabling for the interactive coupled simulations of crustal compression, folding of sedimentary layers and surface erosion.

- An expressive control tool for the sculpting of tectonic forces through user gestures on a tactile device.
- A validation of our results thought user study and comparisons with both simulation from geology and real data.

4.1 Overview

In this section we describe the geological background used as the basis for our approach and the choice of our algorithms and data structures.

4.1.1 Plate tectonics in geology

Continental collision is often the end product of the closure of an oceanic basin driven by subduction of the oceanic plate beneath one of the two continents. When continental plates collide, subduction of the lower part of one plate continues (left plate in Figure 4.1), while the upper layer (the continental crust) made of lighter, granitic rock resists subduction and experiences compression. This compression is accommodated by the formation of a main crustal-scale fault or shear zone, located above the stable, non-subducting plate, that accumulates the deformation by thrusting (Willett et al. 1993). In other situations, for example where the crust is hotter, the deformation is more diffuse and the crust deforms as a thin viscous layer (Ellis 1996). In all cases, this causes uniform uplift of the crustal block and we model this growth by approximating the crust as a viscous material (Section 4.2).

Simultaneously, this block deforms internally by smaller scale faulting and folding. Due to past tectonic events and/or the deposition of sedimentary rocks at the Earth's surface, the continental crust is commonly composed of a number of horizontal rock layers of different mechanical properties. On top of the basement, made of metamorphic and igneous rock, layers of sedimentary rock are deposited over millions of years, due to the erosion of older mountains or to the deposition of carbonate shells in ancient oceans. When compressed during plate collisions, the difference in mechanical properties between the sedimentary layers makes them fold rather than thicken. We compute the multi-scale folding from a thin sheet approximation of the crust (Section 4.3).

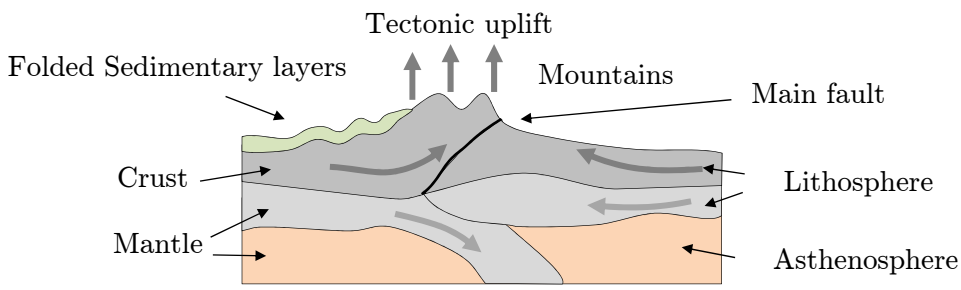


Figure 4.1: Colliding tectonic plates: plate experiencing subduction (left) with uplift and folding of crust material while the lithosphere plunges down, main fault (middle), and stable plate (right).

As a result, mountain ranges are typically asymmetrical, with uplift and folding leading to high topography and steep surface slopes on one side of the mountain where the main structure is located, and more subdued elevations and lower slopes on the other side (*i.e.,* the right of Figure 4.1). The fact that continental crust is made of a number of different rock layers causes different fold scales to appear (top left of Figure 4.1). The surface slopes resulting from these complex patterns of surface uplift lead to concomitant surface erosion controlled by the rate of fluvial incision (Braun et al. 2013) which, in turn, results from the shear stress imposed by debris transported by fast flowing rivers on the bedrock (Section 4.4).

4.1.2 Geologically-inspired interactive simulation

Our goal is to enable the sculpting of large scale terrains (hundreds of kilometers) at interactive rates, where the user's time corresponds to millions of years in real temporal domain. The corresponding simulated time is around 5 million years in our examples.

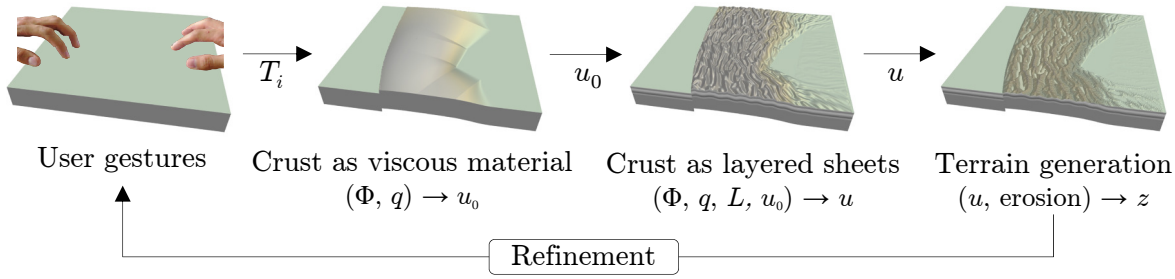


Figure 4.2: User gestures define tectonic plate movement that is used to simulate earth crust as if it was made of a viscous material. The compression and thickness change first yields a coarse uplift map u_0 and then a thin layered strata that produces a refined uplift map u. Fluvial erosion then produces the final terrain elevation z. The overall simulation is interactive and allows for real-time authoring by hand gestures.

User input gestures on a multi-touch input device define the shape of tectonic plates and put them in motion (see Figure 4.2). The user first places a reference hand on the table to define the first tectonic plate, serving as an anchor and intended to be stationary. He/she then moves the second hand in contact with the table to define and set the motion of the second plate (defined as a Voronoi region associated with the second hand) and can repeat this gesture to add new plates. Each time a plate is created, crust material is considered to move rigidly in the convex hull of finger positions, and finger motion is interpolated to define a single, rigid transformation T_i, interpreted as the rate at which the plate moves per time unit (Section 4.2). Assuming that, at this scale, *i.e.,* when geological temporal scales are remapped into a few seconds, the Earth's crust deforms like a soft, viscous material and inertial forces can be neglected, this motion is unaffected by the collision process and can therefore be assumed to remain constant over time–except when the user applies further edits. The output is a transport field denoted as Φ.

Figure 4.3: The initial stratigraphy: rock layers L_i have different mechanical properties, whereas sub-layers differ by erosion parameters only.

Subsurface stratigraphy (Figure 4.3) is defined prior to the simulation using a multi-layered, volumetric model of the Earth's crust: n layers of rock $L = \{L_i\}$, $i = 1, \ldots, n$ are defined with corresponding thicknesses $0.1 \leq \tau_i \leq 5$ km and viscosity μ_i (a measure of the competence of the rock). They represent variations in rock type and mechanical strength at a variety of scales. Thinner layers are used to represent sedimentary material located above thicker, basal rock layers that represent the layering of the crust, for example separating it into a quartz-rich upper crust and a feldspar-rich lower crust. In our implementation, layer thicknesses are spatially uniform, although extending our method to have them vary locally would be straightforward.

To increase visual realism, each layer can be subdivided into $1 - 10$ sub-layers of varying resistance against erosion (k, $\theta_{\max}$). This represents the fractal nature of natural stratigraphy where large competence contrast is likely across thick layers that are themselves composed of thinner layers of reduced competence contrast.

Geologically-inspired simulation consists of a simulation loop that uses three consecutive actions to compute the terrain deformation at each time step (Figure 4.2).

1. From the user-defined transport field Φ, we compute the elevation rate (or *uplift*) u_0 of the terrain caused by the compression between tectonic plates, assuming that the earth crust behaves as an incompressible viscous material in the considered geological time step (Section 4.2).

2. We then compute the uplift field u_i of each layer of rock according to the associated multi-scale folding processes. This is performed using an efficient procedural method based on results from the theory of folding of layered materials (Section 4.3).

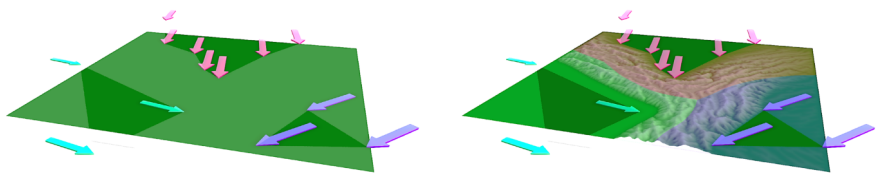
3. We apply fluvial erosion (Chapter 3, Section 4.4) to shape the terrain during mountain growth due to uplift. This results in a time dependent surface elevation $z(\mathbf{p}, t)$, in which we keep track of the original rock layer at each node position, for rendering purposes.

Time steps. Recall that we are sculpting *animated material*: the mountains grow and are eroded while they are sculpted. Therefore, there are two temporal loops: the *geological simulation* loop based on the current user input (which gives us a stationary compression field and thus stationary uplift and folding, then combined with erosion), and the *user interaction* loop (Figure 4.2). The compression field, the folds, and then the uplift need to be recomputed

only when a new interaction takes place (*e.g.,* when the motion of the plates is modified). Even when not recomputed, they are used to update the terrain elevation at each frame. The geological time simulation step (Section 4.2) is not the same as the time-step for any of these loops: it is a theoretical time step used for computing the stationary compression field from the current plates motions. Its value is not important since it expresses a stationary situation.

4.2 Earth crust as a viscous material

Shortening of the crust is the first phenomenon to be accounted for when tectonic plates collide. The crust at this scale can be assumed to behave as a viscous material, resulting in thickness changes and uplift. To capture this phenomenon, we use a 2D physical simulation that computes a *transport field* $\Phi(\mathbf{p})$ in the region of influence between plates. The transport field is a map $\Phi : R^2 \rightarrow R^2$ that describes where each point moves subject to the tectonic forces. It is coupled with a geometric volume preservation method to generate the uplift.



1. Regions A_i **2.** Influence regions

Figure 4.4: Gestural control: Three moving plates are created and refined by using successive hand motions and their rigid part are set to the convex hull of the interacting fingers (1). The region of influence of each plate is defined as the part of the plane that is closest to its rigid part (2).

4.2.1 Moving plates creation

Every plate is characterized by its supporting region A_i where the earth crust displacement is rigid. The associated transformation T_i (a rotation and a translation) represents the rigid displacement of the underlying crust material during a simulation step. While A_i does not move over time and T_i is constant, the continuous displacement of crust material causes surface uplift and mountain growth in colliding regions.

To define a new plate, the user moves his/her hand in contact with the touch table. We provide visual feedback by displaying an arrow from the initial position $\mathbf{c}_k^i$ of each finger to its final position, *i.e.,* when contact with the table ends. The support region A_i is defined as the convex hull of the initial finger positions $\mathbf{c}_k^i$. Without loss of generality, the vector $\Phi(\mathbf{c}_k^i) - \mathbf{c}_k^i$, corresponding to each arrow, is interpreted as the motion of the underlying earth crust during a given geological simulation step Δt.

Since the user gesture may not be fully rigid, the plate's rigid transformation T_i is computed from the user defined set of transport directions at control points $\Phi(\mathbf{c}_k^i)$ by using least squares minimization:

$$\text{argmin} \sum_k \|\Phi(\mathbf{c}_k^i) - T_i\, \mathbf{c}_k^i\|^2.$$

The term $\Phi(\mathbf{c}_k^i)$, and the corresponding arrows are then updated from T_i so that the arrows always depict the rigid motion of the crust material under the rigid part of the plate during a geological step Δt. The user can use extra gestures to add new arrows and edit T_i accordingly.

When crust material moves away from the regions A_i to collision regions, the assumption of rigid motion is no longer true. We call the *influence region* of each plate the portion of the crust closest to the associated A_i. We interpolate the displacements T_i to compute the transport field Φ in those regions of influence.

4.2.2 Viscous compression

The transport field Φ describes the local movement of earth crust material during Δt under the action of tectonic forces, and is computed as the interpolation of the transformations T_i given by the user. At the space and time scale of our simulation, the earth crust can be approximated as a thin plate of viscous material that shortens under the action of plate motion and collision (Figure 4.4 (b)) (Ellis 1996). At this scale rock deformation is isochoric and the viscous deformation can be assumed to be incompressible such that horizontal shortening yields thickness changes.

The transport field. To simulate the 2D compression, we adapt an existing model of viscous virtual plasticine (Dewaele et al. 2004). The amount of viscous crust material q is stored in a 2D regular grid, to be transported from cell to cell through Φ. The grid is defined by a step dx and an orthogonal frame $(\mathbf{e_x}, \mathbf{e_y})$. Each grid cell is represented by its center $\mathbf{x}_i$, and q_i represents the volume of crust rock inside the cell. The area of the grid cell is denoted by a_{cell}.

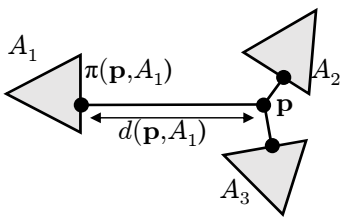


Figure 4.5: Transport at $\mathbf{p}$ is interpolated between support regions A_1, A_2 and A_3, depending on the projection $\pi(\mathbf{p}, A_i)$ of $\mathbf{p}$ onto T_i and distance $d(\mathbf{p}, A_i)$ between $\mathbf{p}$ and T_i.

In our implementation, computations are performed on a 1024×1024 resolution grid. Directly implementing a forward displacement approach yields unnatural patterns similar to Moiré'. Therefore, we solve this problem by computing the inverse of the displacement field Φ^{-1} for transporting material. For each grid point $\mathbf{p}$ inside the transformed rigid area $T_i(A_i)$ the inverse of the transport is set to the rigid transformation

$$\Phi^{-1}(\mathbf{p}) = T_i^{-1}\,\mathbf{p}. \tag{4.1}$$

Outside the region A_i, the field Φ^{-1} is computed by using an interpolation based on a distance weighting operator. Let $d(\mathbf{p}, A_i)$ denote the distance between $\mathbf{p}$ and region A_i and $\pi(\mathbf{p}, A_i)$ the projection of $\mathbf{p}$ onto the region A_i (Figure 4.5). We define:

$$\Phi^{-1}(\mathbf{p}) = \mathbf{p} + \sum_i \frac{T_i^{-1}\,\pi(\mathbf{p}, A_i) - \pi(\mathbf{p}, A_i)}{d(\mathbf{p}, A_i)^e} / \sum_i \frac{1}{d(\mathbf{p}, A_i)^e} \tag{4.2}$$

and we set the exponent $e = 0.9$ to get a smooth falloff of the influence of the distance.

Crust material transportation. We use Φ^{-1} to compute the amount of crust material q in grid cells after the geological step Δt, *i.e.,* when plate material is transported from the tip

to the end of the arrows (Figure 4.4).

Let q_0 be the initial amount of material in each grid cell. In order to obtain the amount of material $q(\mathbf{x}_i)$ in a cell c_i at the end of the plate's motion, we use the inverse transformation Φ^{-1} to compute the original positions of the four corners of c_i before motion, which defines a quadrangle of area a_i. Then we compute the value of q in the cell c_i after the simulation step: $q(\mathbf{x}_i) = q_0\, a_i/a_{cell}$. The initial amount of material is factored later to compute the uplift, so its value does not have any physical meaning. We reduce the equation to dimensionless quantities by setting $q_0 = 1$.

4.2.3 Uplift from thickness changes

Crustal deformation is isochoric, *i.e.,* crust material retains a constant volume over time. The local thickness of the crust is proportional to the volume of crust rock inside the associated cell and to the amount of material q in the 2D grid. Therefore, variations in the amount of material provide a map of thickness change rates: $\mathrm{d}q/\mathrm{d}t = (q - q_0)/\Delta t$.

Since our simulation method does not capture the main fault formation process and the resulting discontinuity in altitude (Section 4.1.1), we model them by a procedural approach. We define the fault as an offset of the boundary of the compressed region on the stable plate side that is computed as an iso-level of q. We then define the uplift due to compression as $u_0 = 0$ on the stable side of the fault and $u_0 = \alpha\, \mathrm{d}q/\mathrm{d}t$ elsewhere. The coefficient α is a scaling coefficient used to set the uplift to the appropriate geological time scale; in our system we set α so that the uplift is $5.0\, 10^{-4}$m /y on average. This approach captures the asymmetric nature of the earth crust compression. The resulting uplift map u_0 is further improved by incorporating folds, as described next.

Note that this procedure neglects two potentially important aspects of crustal shortening. First, we neglect isostasy by imposing that the compression-induced shortening results in surface uplift only. In most mountain belts, shortening also results in a vertical depression of the base of the crust (and of the plate). Second, we do not prescribe a dip to the main fault. In nature all compressive faults (or thrusts) dip at an angle of a few tens of degrees with respect to the surface. The consequence of our approximation is that we do not accurately compute the additional horizontal velocity field that arises from the deformation process. We note, however, that such an approximation is also commonly used in the geological community as the effects of horizontal advection of landforms have only been recently investigated (Castelltort et al. 2012).

4.3 Earth crust as layered sheets

The deformation and shape of mountain ranges are also controlled by folding of layers within the crust under compression. More precisely, the crust is composed of several layers of rocks of varying thickness and strength. Complex multi-scale folding occurs, where small folds appear on top of larger ones. Since simulating this complex folding process would be computationally demanding, we use a procedural model to capture this well understood behavior and then rectify the uplift map u_0, which only captures the viscous behavior of the crust.

4.3.1 Folding of layered materials

Folding is a mechanical instability that results from the growth of stress imbalance at the interface between layers with different competence. Researchers in both geology (Yamato et al. 2011) and Computer Graphics (Rémillard et al. 2013) have studied this phenomenon. They rely on the observation that the wavelength of folds λ of a layer k of thickness τ_k and viscosity μ embedded in a viscous material are correlated, which can be expressed by the following equation (Schmalholz et al. 2000):

$$\lambda_k \sim \left(\tau_k\, \tau_{k-1}\right)^{1/2} \left(\mu_k / \mu_{k-1}\right)^{1/6}. \tag{4.3}$$

Our work relies on this theoretical law as it provides a good approximation of the fold formation process. Similar expressions would be obtained assuming an elastic or viscoelastic rheology for the layers; the important point is that we make the wavelength of the folding instability proportional to the thickness of the layer(s) and their competence contrast (viscosity or elastic properties), both of which can be controlled as input parameters.

The user can define the wavelength of the folds that directly controls the folding effects. The fold thickness is computed automatically.

4.3.2 Procedural fold generation

Although folds of appropriate wavelengths could be set to emerge from layered sheet simulation (Rémillard et al. 2013) or could be generated by combining elastic thin sheet material (Narain et al. 2012) with smaller scale folds (Rohmer et al. 2010), we generate the folds procedurally on top of the viscous simulation (Section 4.2). This allows us to achieve interactive rates, which is key for seamless content creation with sculpting gestures. Moreover, directly reusing wavelengths from Equation (4.3) is sufficient to ensure visually plausible folds. Therefore, we introduce a procedural modeling method for multi-scale folds.

Our method forces the folds to follow a set of lines orthogonal to the compression direction. We constrain those lines to be separated by an average distance λ that is the theoretical wavelength. The lines are chosen based on the compression direction extracted from the transport field Φ. This choice corresponds to the places of maximum folding (called *top skeleton* below), between which the folds are interpolated. This provides visually plausible results consistent with the expected fold wavelength and height.

Extracting the compression field. We extract the folding direction and amplitude from the crust *compression field* C by first converting the inverse transport Φ^{-1} given by Equations (4.1) and (4.2) into a deformation tensor field $\mathbf{F}$:

$$\mathbf{F} = \left(\mathbf{I} - \nabla(\Phi^{-1} - \mathbf{I})\right)^{-1}.$$

The stretch tensor field $\mathbf{U}$ is computed by taking the square root of the Cauchy-Green deformation tensor (Rohmer et al. 2010). In our case, this can be written as $\mathbf{U} = \sqrt{\mathbf{F}^T\mathbf{F}}$. For every cell, the 2×2 matrix $\mathbf{U}_i$ can be diagonalized:

$$\mathbf{U}_i = \nu_1\, \mathbf{e}_1\, \mathbf{e}_1^T + \nu_2\, \mathbf{e}_2\, \mathbf{e}_2^T,$$

Vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are the unit eigenvectors in the plane and $\nu_1 \leq \nu_2$ with real positive eigenvalues. We compute a *compression field* C for the crust, defined in every cell i by: $C_i = (1-\nu_1)\,\mathbf{e}_1$ if $\nu_1 < 1$ and $C_i = 0$ otherwise. The vector $\mathbf{e}_1$ is the direction of compression and $1-\nu_1$ represents its strength. Geological events that would cause some local extension of crust are beyond the scope of this paper and are not considered.

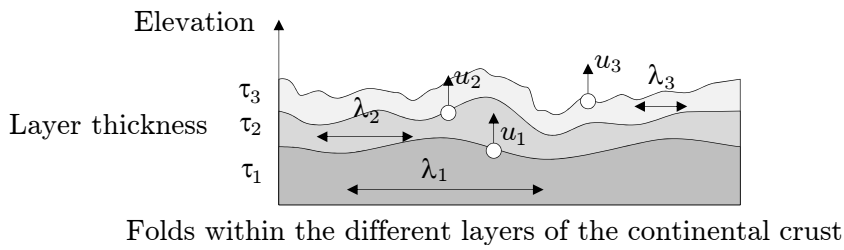


Figure 4.6: The different fold layers are stacked and their uplift contributions are summed to obtain the final uplift.

Iterative fold generation. We model the folds resulting from the layered structure by processing layers one after another (Figure 4.6) and summing the resulting elevation rates u_i to get the final uplift rate:

$$u = u_0 + \sum_{i=1}^{n} u_i. \tag{4.4}$$

This captures folding effects at different scales, since each u_i represents folding at the specific wavelength λ_i (Equation (4.3)) that results from the interaction between the pair of layers L_{i-1} and L_i. Folds are generated using sinusoidal interpolation (Biot 1961) between skeletal curves computed among the edges of a random 2D grid, and then used to compute u_i, as described below.

Fold skeletons. We compute crest and bottom lines for the folds called the *top* and the *bottom* skeletons. Similar to previous fold generation methods (Rohmer et al. 2010), these skeletons are set to be orthogonal, in average, to the compression direction. We also make sure that they cover the whole compression region. This is done as follows:

We first generate a planar mesh $\mathcal{M}$ independent from the terrain grid, and chosen such that the average edge length is equal to the desired wavelength λ_i from Equation (4.3). The mesh $\mathcal{M}$ is defined as a Delaunay triangulation of a Poisson disc sampling of the terrain, with λ_i as average edge length.

Then, some edges of $\mathcal{M}$ need to be selected as part of the *top skeleton* S_t (crest lines). We use the compression field C to compute S_t as follows. Let $[\mathbf{a}, \mathbf{b}]$ be an edge of $\mathcal{M}$ with center $\mathbf{m}$. For every edge surrounded by two faces F_0 and F_1, we define a dual edge as an edge linking the center of an edge of F_0 to the middle of an edge of F_1, both being distinct from $[\mathbf{a}, \mathbf{b}]$.

We define the *compression energy* e along an edge $[\mathbf{a}, \mathbf{b}]$ to measure its alignment with

the direction of compression:

$$e([\mathbf{a},\mathbf{b}]) = \left| \frac{(\mathbf{b}-\mathbf{a})}{\|\mathbf{b}-\mathbf{a}\|} \cdot \frac{C_i(\mathbf{m})}{||C_i(\mathbf{m})||} \right| .$$

An edge with low compression energy is more likely to be in the direction of the folds. Therefore, an edge E is added to S_t if an only if its folding energy is below one of its four dual edges. It can occur that the three edges of a face of $\mathcal{M}$ belong to S_t which would cause a singularity. We detect these cases and remove the edge of highest energy from S_t.

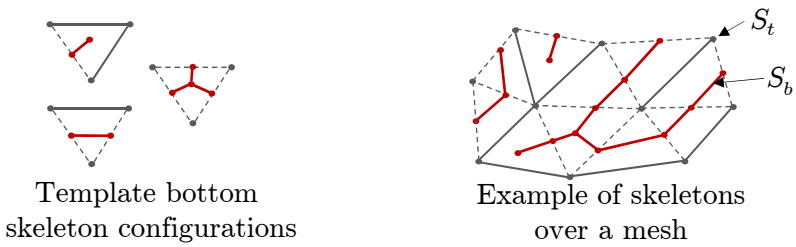


Figure 4.7: Folds computation starts by choosing edges for the top skeleton S_t (gray) over a mesh $\mathcal{M}$ (dotted gray) in the best fold direction. The bottom skeleton S_b (red) is deduced from S_t following the template (left)

The third step is to build the skeleton S_b representing the bottom parts of the folds. Figure 4.7 shows the template configurations that are used to generate S_b from S_t. For every triangle $F \in \mathcal{M}$ we count the number of edges n_F of F that belong to S_t and apply the following classification. If $n_F = 2$ we add an edge to S_b, linking the center of the edge of F that is not in S_t to the center of F. If $n_F = 1$ we add an edge to S_b that links the center of the two edges of F that are not in S_t. Otherwise we add three edges connecting the center of one edge of F to the center of F. Note that this construction ensures that the edges of S_b remain approximately parallel to the main folds direction, *i.e.,* to the average direction of neighboring S_t edges.

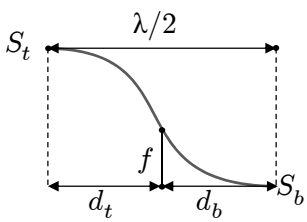


Figure 4.8: The local wavelength value is calculated from the distance from top to bottom skeletons, and used to set fold height.

Fold heights. In the last step, we build the fold surface between the bottom and the top skeletons. Although the average wavelength is near the theoretical value from Equation (4.3), it may vary among the folds and we need to know the precise wavelength at any point in order to compute its height. Let $d(\mathbf{p}, S_t)$ and $d(\mathbf{p}, S_b)$ denote the distance between point $\mathbf{p}$ and each skeleton. We define the local wavelength as:

$$\lambda(\mathbf{p}) = 2\ (d(\mathbf{p}, S_b) + d(\mathbf{p}, S_t)) ,$$

and the local height of the fold at time Δt (see Figure 4.8) as:

$$f_i(\mathbf{p}) = \frac{\sigma_i\, \Delta t, ||C_i(\mathbf{p})||}{2} \left(1 + \cos\left(\frac{2\,\pi\, d(\mathbf{p}, S_t)}{\lambda_i(\mathbf{p})}\right)\right) ,$$

where σ_i is the folding speed. We choose it as a fraction (between 0.01 and 0.5) of the uplift u_0. Note that we add one to the usual cosine curve used to shape the fold (Biot 1961) because we model the folding behavior of a rock layer that slides over some thicker layer below it, so $f_i(x)$ should be always kept positive.

Local fold refinement. This step aims at complementing our procedural fold model with a local, direct editing tool that allows for authoring the ridges and valleys in a mountain range. The user can change the direction of the folds by manually selecting edges and can also change the height of the mountain along a fold line. The advantage of this local refinement tool is that it offers direct control over the local ridges and valleys while ensuring to some extent the geological consistency of the results, since the computed fold skeletons serve as a geological guide for the user to maintain the coherency of the generated mountain.

4.3.3 Uplift update from folds

The uplift u_i resulting from the folding of the current fold pair of crust layers is defined as the folding speed Δf:

$$u_i = \Delta f = \frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t}.$$

This value is used in Equation (4.4) to compute the total uplift field u, and yields terrains that fold at multiple scales.

In practice, using several different folding scales is not necessary for the interactive visual feedback during a sculpting session, since only large folds are noticeable. The other fold layers are considered only during the offline rendering of the final terrain.

4.4 Terrain surface generation

The elevation of the terrain is generated from the combined uplift and stratigraphy of the crust by applying fluvial erosion and landsliding processes. In our system, we did not include a representation of glacial erosion (Chapter 5). Although the offline results would be strongly improved, the proposed method is not fast enough to enable interactive plate manipulation.

4.4.1 Interactive terrain generation

The terrain surface generation algorithm (Chapter 3, Section 3.3) combines the uplift (Equation 4.4) with a fluvial erosion law. Our method for solving this law relies on a triangle irregular network (TIN) that represents the topology of the terrain. The drainage area $\mathcal{A}(\mathbf{p})$ and slope $s(\mathbf{p})$ are derived for each point $\mathbf{p}$ in the triangulation in order to compute a rate of elevation change following the stream power law:

$$\frac{\partial\, z(\mathbf{p}, t)}{\partial t} = u(\mathbf{p}) - k\, \mathcal{A}(\mathbf{p})^m\, s(\mathbf{p})^n, \tag{4.5}$$

The coefficients m, and n are assumed to be constants and k varies with the rock type (see Section 4.4.2) (as it is commonly done in geological experiments, we used $n = 1$, $m = 0.5$, see

Table 4.1 for the different values used for k). The erosion equation captures the river carving effect. Moreover, the uplift u is uniformly rescaled so that its order of magnitude is close to $5.0\,10^{-4}\,m\,y^{-1}$, which is a geologically-consistent value (uplift rate in mountain belts rarely exceed a few millimeters per year) that produces visually plausible results.

In an attempt to improving the variability of the valley slopes, we extend the original model by coupling the fluvial erosion with a landslide erosion model that mimics the way cliffs are shaped by landslides. This method is derived from the original *thermal erosion* (Musgrave et al. 1989) used in Computer Graphics. We check the slope $s(\mathbf{p})$ after each step of fluvial erosion. If the slope $s(\mathbf{p})$ is larger than a corresponding talus slope θ, we update the height using:

$$\frac{\partial\, z(\mathbf{p}, t)}{\partial t} = l\,(s(\mathbf{p}) - \tan\theta) \quad \text{if } s(\mathbf{p}) > \tan\theta, \quad 0 \text{ overwise.}$$

The constant $l = 10^{-2} m\,y^{-1}$ has been set experimentally and describes the speed of the landslide effect. The talus slope is preset to the commonly accepted default value of $\theta = 30°$, but our model enables modifications for each layer of rock, to make it depend on material strength (Table 4.1).

Considering that the slope at $\mathbf{p}$ only depends on the height of its lowest neighbor $z(\mathbf{q})$ and on the horizontal distance between them $r = \|\mathbf{p} - \mathbf{q}\|$, we use:

$$s(\mathbf{p}) = \frac{z(\mathbf{p}) - z(\mathbf{q})}{r}.$$

We evaluate z using an implicit time scheme that is computed in linear time, without the need to invert a matrix: this is achieved by evaluating the heights in the right order, *i.e.,* $z(\mathbf{q}, t + \mathrm{d}t)$ before $z(\mathbf{p}, t + \mathrm{d}t)$.

4.4.2 Rock layers at the surface

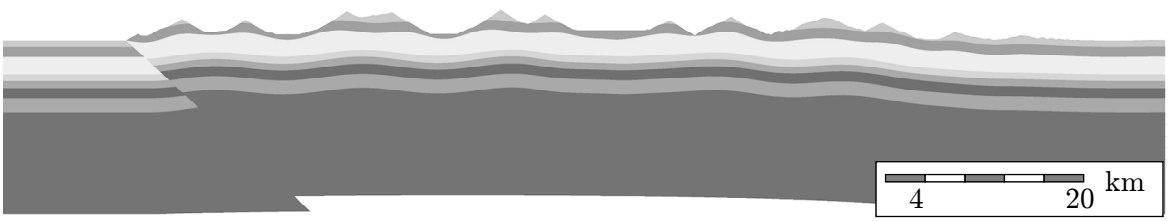


Figure 4.9: Schematic rendering of a geological cross-section of a synthesized mountain range.

The multi-layered nature of the crust controls not only the folding behavior, but also impacts the erosion process (see Figure 4.9). An important feature of our subsurface geologically-based simulation is that we can associate a specific rock layer with each visible point of the eroded terrain surface, and use this information to locally alter the erosional properties or to adapt the rendering.

The rock layer index associated to a point at the eroded surface is obtained from the stratigraphy of the original elevation h_0 of this point inside the crust. This elevation cannot be simply defined as the difference between the total height of uplifted material and the total

height of eroded material at that point, because uplift contribution of folds should not be considered when they are being eroded. Therefore, we compute the height of each fold layer at the current simulation step, and evaluate the original height of the surface from its position between the two nearest layers (Figure 4.10).

The height h_k of the k-th layer surface is computed by summing the heights of the layers below, starting with $h_0 = u_0\, t$, and adding the specific uplift due to folding of the current layer:

$$h_k(\mathbf{p}) = h_{k-1}(\mathbf{p}) + u_k(\mathbf{p})\, t,$$

where t is the current time in the simulation, since the beginning of mountain growth.

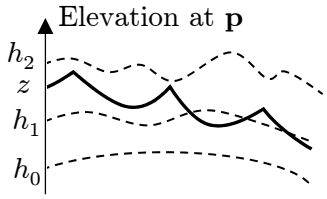


Figure 4.10: The nature of the rock at the surface, of altitude z, is deduced from its position wrt. fold layers of altitudes h_0, h_1 and h_2.

We first compute which crustal layer every point of the surface of the terrain $\mathbf{p}$ belongs to. This is achieved by computing the vertical displacement of each rock layer, from bottom to top, until the total thickness gets higher than the current surface altitude $z(\mathbf{p})$; thus the index k satisfies $h_{k-1} < z < h_k$. We then compute the elevation of the point between the two layers:

$$\gamma = \frac{h_k(\mathbf{p}) - z(\mathbf{p})}{h_k(\mathbf{p}) - h_{k-1}(\mathbf{p})}.$$

The height of the point before deformation is:

$$h_0(\mathbf{p}) = (1-\gamma)\, h_k(\mathbf{p}, t_0) + \gamma\, h_{k-1}(\mathbf{p}, t_0),$$

where $h_i(\mathbf{p}, t_0)$ denotes the altitude of the crust layer i before deformation at time t_0. We derive the layer index from the original height of the point. To imitate geological strata, we divide the layers into sub-layers of different strengths, and adjust the erosion parameters accordingly: we use a high erosion constant k and talus landslide slope ($\theta = 30°$) for weak rock layers, and a low erosion constant k with a high slope ($\theta = 50°$) for hard rock.

During the rendering step, a texture is added to strong layers by fetching the original height h_0 from a 1D look-up texture. This enables us to show folded rock layers on eroded cliffs.

4.5 Implementation, results and discussion

Δt	Total simulation time	$5\,10^6\,y$
n	Number of layers	$2-10$
τ_i	Layer thickness	$0.1-5\,km$
μ_i	Layer viscosity	$10^{10}\,Pa$
λ_i	Layer fold wavelength	$0.5-20\,km$
σ_i	Layer folding speed	$0.01\times\ u-0.5\times u$
u	Target average uplift	$5.0\,10^{-4} m\,y^{-1}$
k	Erosion constant	$2.5\,10^{-7}-8\,10^{-7} m^{-1}$
θ_{max}	Talus angle	$30^\circ-50^\circ$
l	Landslide speed	$10^{-2} m\,y^{-2}$

Table 4.1: Parameters of our system and ranges of values used in the examples.

We developed our algorithm in C++ and with OpenGL and GLSL APIs for real-time rendering. High quality images were directly rendered by streaming our synthesized terrains to *Terragen 3* which was also used to generate terrain surface details. All examples were generated on a desktop computer equipped with an Intel Xeon E5 CPU, clocked at 3.7GHz with 16GB of RAM. We used a Wacom CINTIQ 24HD multitouch table for the input.

4.5.1 Architecture

We separated the user-interaction and the tectonic simulation into two CPU threads. The first thread handles the user-interaction, the visual feedback and gesture analysis. The viscous material simulation, fold computation and tectonic simulation were performed in the second thread. The separation of tasks into independent threads allowed us to provide authoring in real time with an average performance over 30Hz.

The viscous crust compression is computed on a grid of resolution 1024×1024. The algorithm (Section 4.2) is straightforward to implement on GPU because each grid cell can be computed independently. We achieve high processing performance (10ms) with no visible grid or aliasing artifacts.

λ [m]	Skeleton edges	Skeletons [ms]	Height [ms]
$7,000$	955	0.9	2.1
$3,000$	$4,246$	3.7	1.4
$1,000$	$36,715$	53.0	2.8

Table 4.2: Statistics for the different folds: wavelength (in m), number of skeleton edges, and computation time for the skeleton and the elevation (in ms).

The folding simulation is divided into two parts: the construction of fold skeletons (Section 4.3) was implemented on the CPU, whereas the fold height computation using the inverse distance transform was developed on the GPU. Because the distances between the skeleton edges are small and regular, we developed a GPU algorithm that renders small rectangles around each line, no larger than the theoretical wavelength, and output the distance to the line at each pixel. The distance at the junction of two lines is defined as the minimum of the results. The performance of this step is detailed in Table 4.2.

The terrain surface is a mesh with $10,000$ vertices for a $100 \times 100\text{km}^2$ terrain. The generation process is fast enough to guarantee interactive visual feedback. One iteration takes 10 ms on average and a 5 million years simulation takes only 5 iterations. Thus, we can perform these iterations interactively before updating the terrain on screen. The parameters used in our setup are reported in Table 4.1.

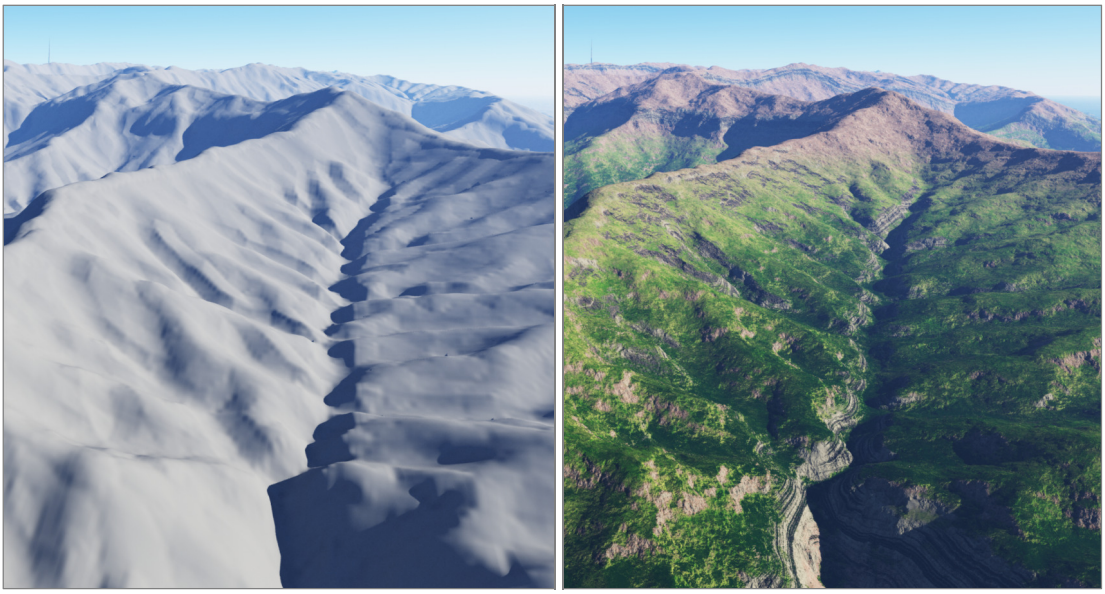


Figure 4.11: Real-time rendering (left) and the same terrain with details generated by Terragen (right).

Only a few post processing steps are needed to achieve high quality rendering. First, the elevation is recomputed off-line by combining the uplift maps resulting from our method with erosion simulation (Chapter 3) with a smaller time step and a larger resolution ($1,000,000$ vertices). A simulation step runs in 4s on average and we need ≈ 100 iterations to obtain a final terrain. Finally, we add noise weighted by a function of the slope and the elevation to obtain small surface displacements that are shaded during the final rendering step (Figure 4.11).

4.5.2 Qualitative and quantitative results

Sculpting sessions. Figure 4.12 demonstrates the interactive local and global control over the entire process which is one of the important contributions of our method. The user mainly controls his/her design by using hand-on interaction on a multitouch table. By placing and moving both hands on the table, the continental plates are set in motion. The arrows provide visual feedback about the plate shape and location and their rigid transformations are defined by using finger gestures. The use of the fold refinement tool is illustrated in Figure 4.12 (3).

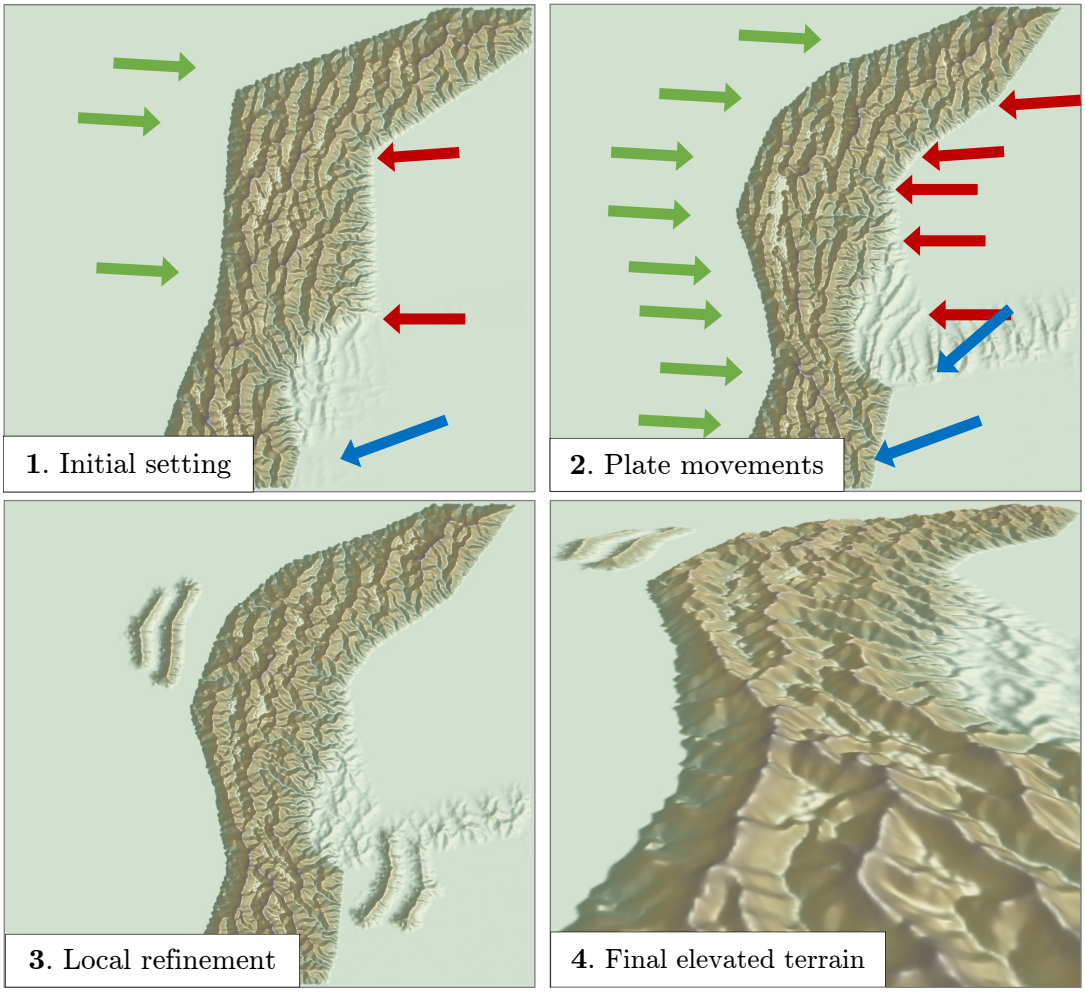


Figure 4.12: Successive steps of a sculpting session. The user starts by setting the movements of the plates (**1** and **2**), before adding local details by choosing folding directions and prescribing specific elevations (**3**) to get the final result (**4**).

Resulting terrains. Figures 4.18 and 4.13 show high quality renderings. Notice the overall asymmetric range, the alignment of the sub-ranges orthogonal to the compression direction, and at lower scales, the folded strata appearing along eroded cliffs. Figure 4.19 shows the effects induced by the layered erosion properties on the landforms of the terrain.

In addition to producing still versions of high quality terrains, our method can also be used to produce animations of the formation of mountain ranges in cross section and then as a full rendering.

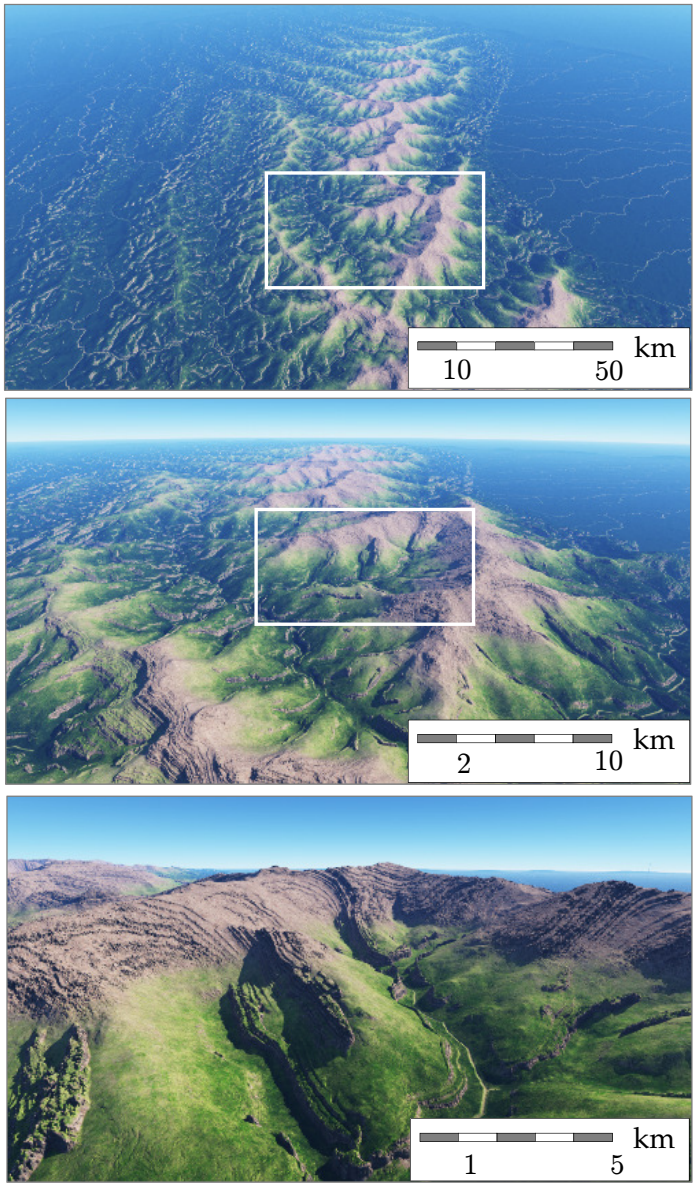


Figure 4.13: The user sculpts a mountain range. The results capture the whole range coherency, and massifs are aligned with the fold lines. Eventually, the folds leave marks after erosion.

4.5.3 Validation and discussion

This method was designed in collaboration with an expert in geology, Jean Braun, who helped us validating the modeling choices and the results. A detailed validation follows:

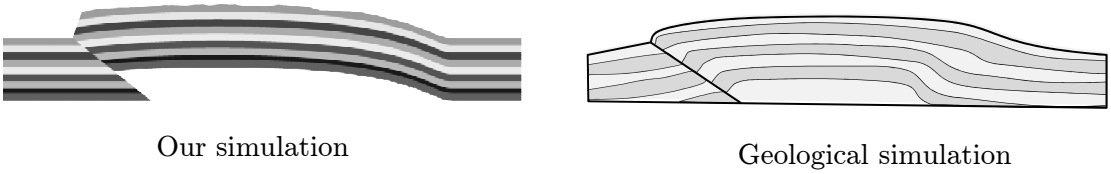


Figure 4.14: Comparison between a terrain profile generated by using the compression step only in our method (top) and a geological simulation result, also neglecting folds (bottom, courtesy of J. Braun (Braun et al. 1994)). Note the similar nearly uniform uplifts in the central parts.

Figure 4.14 shows a comparison between results from geology (Braun et al. 1994) and a mountain profile created by our algorithm. It validates the fact that our method adequately captures the vertical displacement due to crust compression. Figure 4.15 shows the folding of one layer in comparison to a geological simulation result (Fernandez et al. 2014).

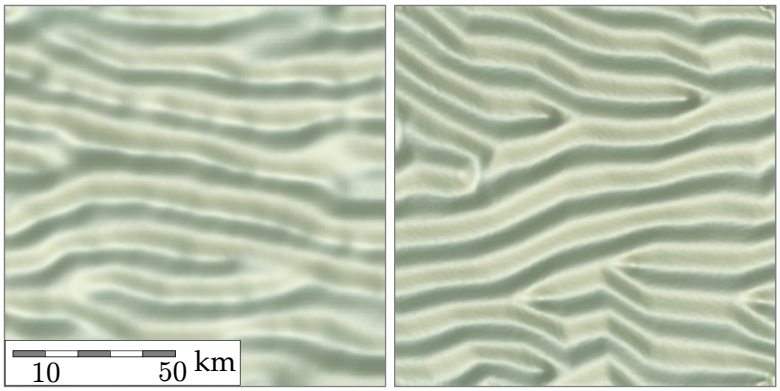


Figure 4.15: Folding validation: Comparison between a geological simulation result from (Fernandez et al. 2014) (left), and a top view of our results (right), both rendered using the same shading.

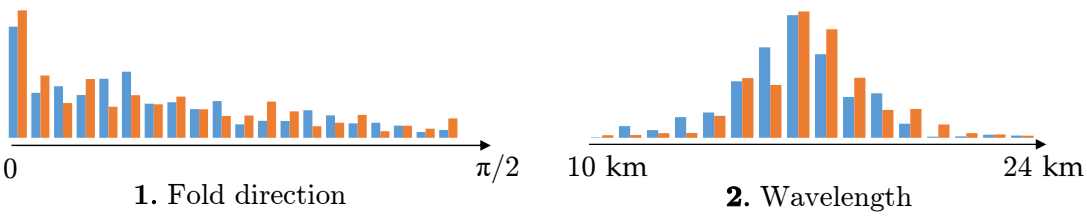


Figure 4.16: Folding validation: Comparison of the angles between fold normals and compression direction (left) and the histograms of wavelengths (right) between (Fernandez et al. 2014) (blue) and our results (red).

We also provide statistics computed on the graph of these folds. They include wavelength histograms and an histogram of angles between the fold direction and the orthogonal to the compression direction (Figure 4.16), and a comparison of the graph properties (Table 4.3). We choose to measure three values that are essential to the visual aspect of the folded regions, namely the number of end nodes, the number of triple joints, and the average fold length (between the nodes or triple joints).

	Our results	Geological simulation (Fernandez et al. 2014)
End nodes	21.2 %	25.6 %
Triple linkage	7.5 %	5.6 %
Average fold length	132 km	165 km

Table 4.3: Comparison of folds graph properties between our results and a geological simulation.

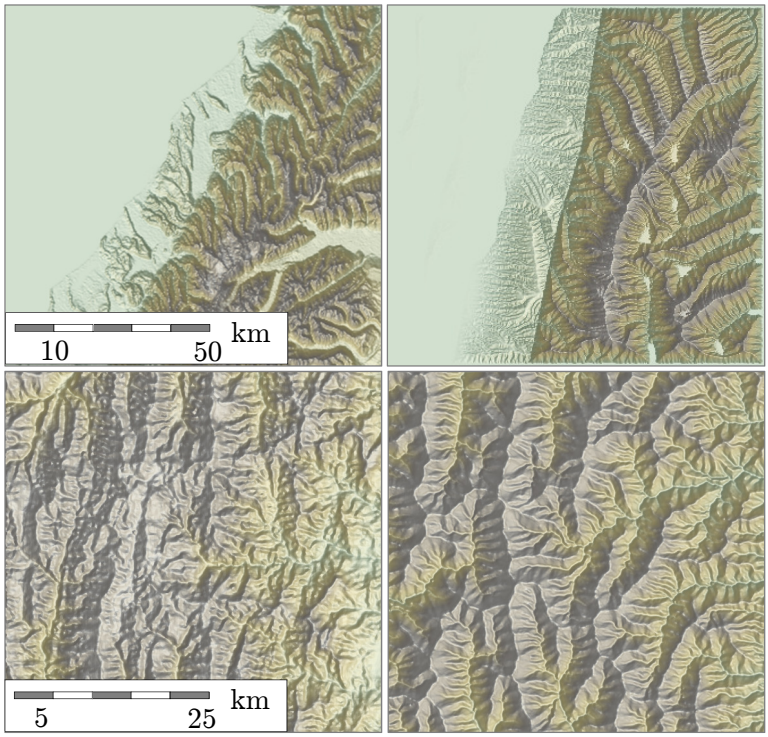


Figure 4.17: Comparisons between real mountains (left) and our results (right). Top: we compare the main fault with New Zealand's Alps. Bottom: we focus on eroded folds in the Andes.

Our results are compared side-by-side with real data from USGS in Figure 4.17. We selected two mountain ranges that form along plate boundaries, such as the main fault of New Zealand's Alps (Figure 4.17, top) and the Cerro Chorolque in Southern Bolivia (Figure 4.17, bottom), along the eastern side of the Andes. Both formed by thrusting and folding of layered rocks under compression. They differ mostly by their width, with the New Zealand Alps being much narrower than the Andes. Deformation in the New Zealand Alps is focused on a single

crustal-scale fault (the Alpine Fault) and built out of a single lithology, the Otago Schist, whereas in the Andes, the stratigraphy is more complex and controls the spacing between the crustal-scale faults and folds. The main difference between real data and our results is caused by glacial erosion (Chapter 5), which is not captured by our method, and results in broader, flat-bottom (or U-shaped) valleys.

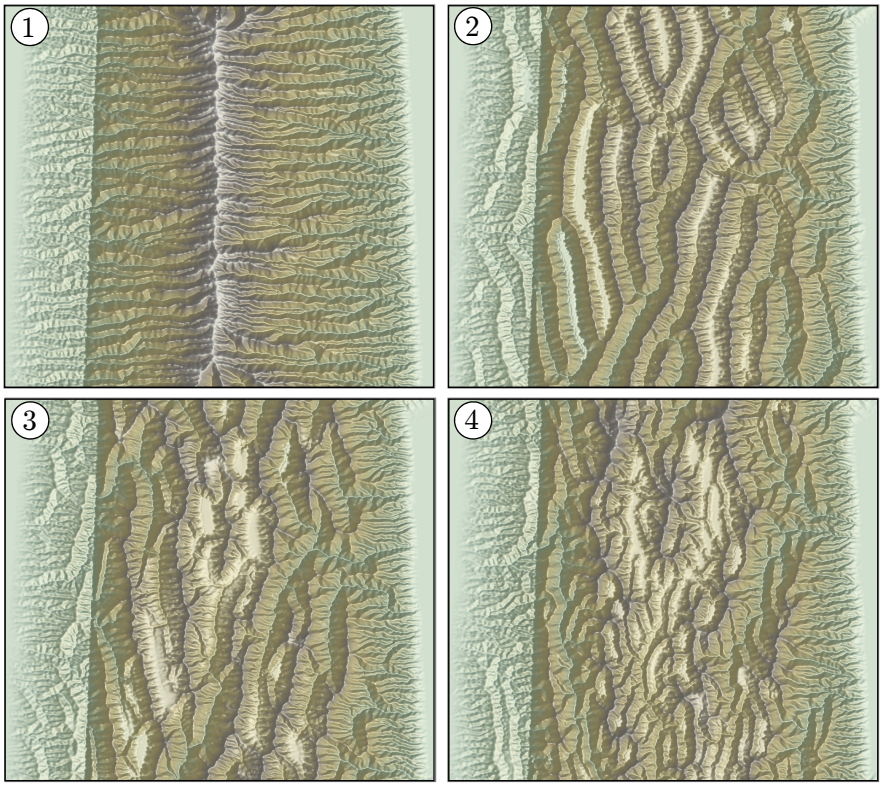


Figure 4.18: Various 100 km terrains obtained from the same overall uplift but with different folds layout: only viscous simulation (**1**), a single layer (**2**, wavelength $\lambda = 7$ km), two fold layers (**2**, $\lambda_1 = 10$ km, $\lambda_2 = 5$ km) and (**3**, $\lambda_1 = 15$ km, $\lambda_2 = 3$ km).

The importance of including folding and a stratigraphic control on erodibility is evidenced in our examples. Where such folding and stratigraphic control is not included (as shown in Figure 4.18.1), all valleys strike perpendicular to the axis of the mountain. Although such an organization of drainage can be observed locally, the morphology of most mountain ranges at the largest scale is strongly controlled by crustal-scale folding. In some mountain ranges, folding is mainly controlled by a single layer (Figure 4.18.2), such as in the Zagros Mountains in Iran which formed by folding of a sedimentary sequence above a weak salt layer. Various results can be obtained by changing the relative size of each fold layer: Figure 4.18.3 includes two layers of related fold wavelength (10 and 5 km), and can be compared to the real data from the Andes (Figure 4.17, bottom), while Figure 4.18.4 shows a mountain range formed from a larger difference in fold wavelength (15 and 3 km). At the smaller scale, most mountainous landforms are also strongly controlled by the progressive exhumation of rock layers of varying resistance to erosion (Figure 4.19). A dramatic example of this, that should be compared to our simulated landscape shown in Figures 4.20.2 and 4.19.3, is the so-called *velodrome* structure near the city of Digne in the French Alps (Figure 4.20.1) which formed by erosion

of a gently folded stack of sedimentary rocks of alternating resistance to erosion.

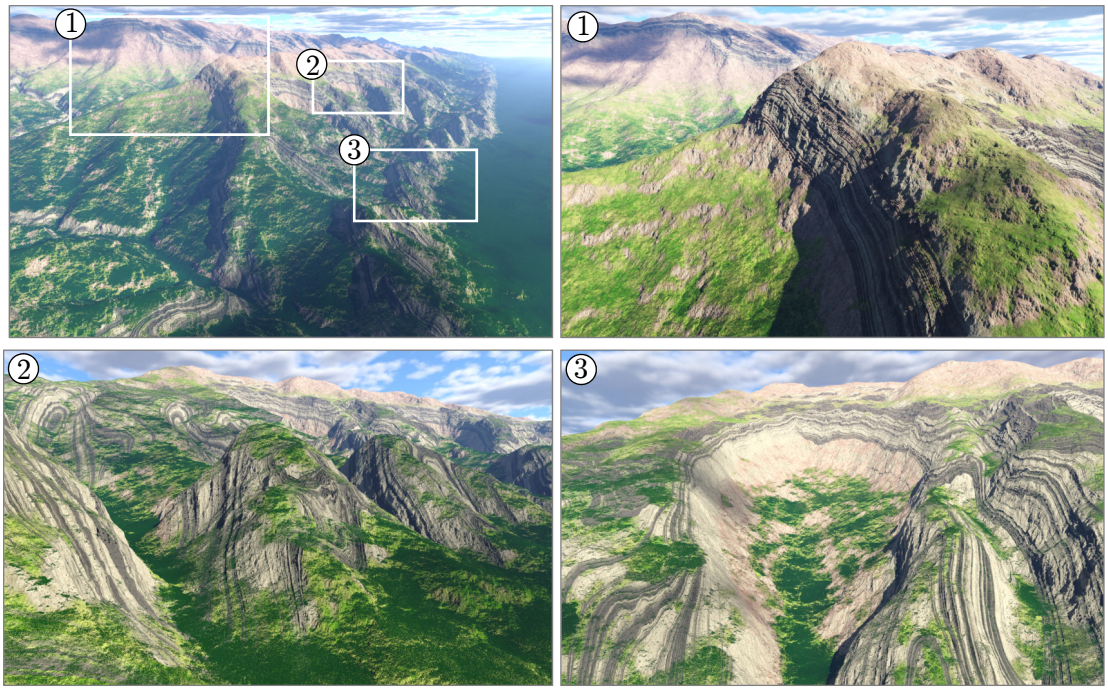


Figure 4.19: The difference of erosion properties among rock layers induces different effects: cliffs (**1**), individual small summits (**2**), or circus (**3**).

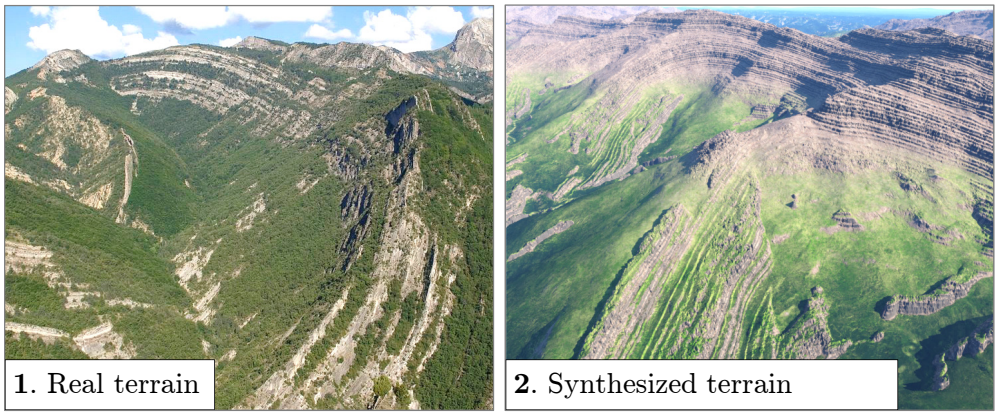


Figure 4.20: Comparison of a photograph showing real folds (1) and our synthesized terrain (2).

Limitations Our sculpting method has some limitations. First, the fold model we use is only valid at the start of the folding: when a real mountain fold grows too high, the relation with wavelength used in this paper is no longer true, and the fold does not have a sinusoid shape anymore (Schmalholz et al. 2000). A volumetric simulation would be necessary to compute more complex behaviors such as fold distortion (Kaus et al. 2006), as well as

secondary faults within the crust. While such a simulation would be interesting and would capture a wider range of geological situations in the generated folds, it would be also difficult to integrate in a fast and simple control tool.

Second, we do not allow for a plate setup that evolves over time, though it is the case in several real mountain ranges. This would be more geologically correct, but it would provide less intuitive manipulation.

Finally, our method does not lend itself for precise editing as it only provides the user with a high level of control over the final result. Although this limitation is alleviated by the possibility to edit fold skeletons, the user cannot author certain details, such as the number of dendritic patterns while keeping the other unchanged as shown in Figure 4.19.

4.5.4 User study

We conducted a user study to evaluate the usability of our mountain sculpting tool. It was tested on 11 subjects who had low to medium knowledge of geology. Among them, three were professional CG artists. The participants first performed an informal learning trial in order to understand the tool. After that, they were asked to reproduce real mountain ranges shown in two images: the Caucasus and the Alps. The users had to evaluate the interactivity and the effectiveness of the method, and the quality of their results. The measurements were rated on a 1-4 Likert's scale.

The quantitative part of the user study showed that the participants had a positive experience while working with our sculpting tool. The effectiveness of the tectonic gestures averaged 3.3, the local control tool received a mark of 2.8 and the overall intuitiveness was 3.0. The users rated their own achievement in reproducing the shape of real ranges with an average of 2.6.

The qualitative part of the user study included some positive comments from the subjects regarding the simplicity of the first contact, the enabled creativity, and the realism of the results. An important feature of the editing tool was the use of arrows to symbolize tectonic motion. The automatic update of arrows during interaction so that they always show the same rigid transformation, was pointed out by users: some found this counterintuitive, whereas most found it to be of great help. The interactivity and simplicity of our system have potential applications in education, in particular for teaching geology: in the field, the geometry of a fold may be partly obscured by the way erosion has revealed it to the geologist. Our system can show the fold as texture on a cliff wall after erosion as well as its true 3D subsurface geometry, making it a useful tool for training geologists in reconstructing the 3D representation of a folded layer from its exposed geometry.

4.6 Conclusion

Mountain uplift and erosion (Chapter 3) are among the dominant phenomena that shape terrain surface. We propose a new, *geologically-inspired* method to compute the uplift on sedimentary rock regions. We use two coupled sub-models, one for modeling the local thickening of the earth's crust due to compression of colliding plates, and the other for modeling multi-scale folding behavior. Interleaving these two sub-models results in a complex uplift

map simulating the growth rate of mountain ranges between tectonic plates. This enables us to generate visually compelling terrains at interactive rates.

The achieved interactivity allowed this method to be controlled through a sculpting metaphor on a tactile device. Thus, we propose the first approach enabling users to progressively sculpt large-scale mountain ranges by using hands-on interaction with displacement gestures, as if they are sculpting plasticine.

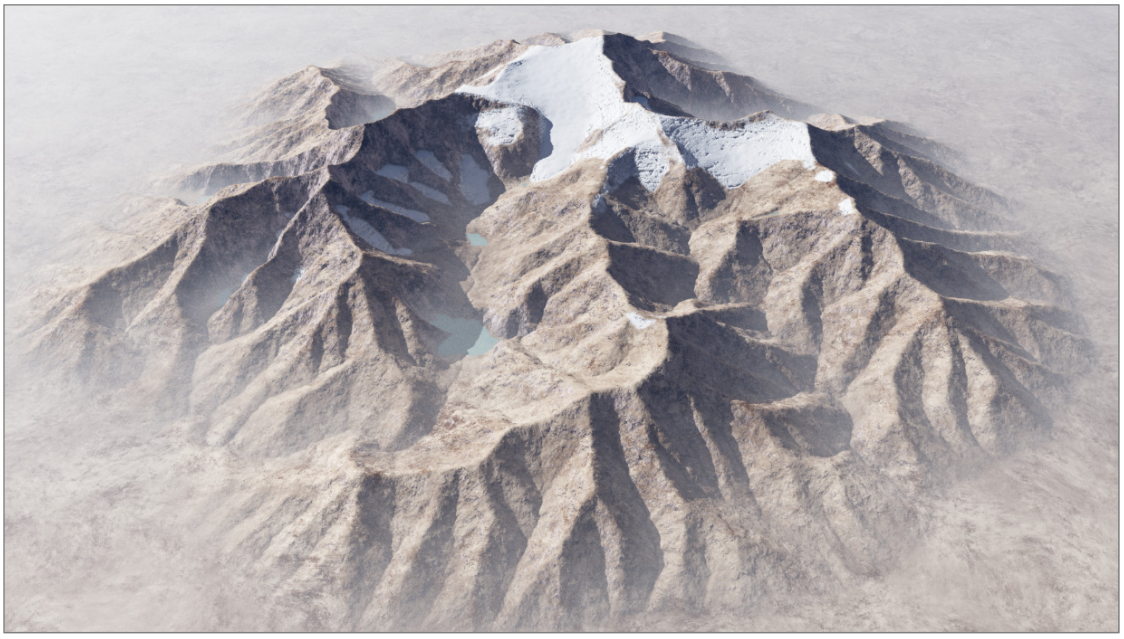
Independent of the interaction, the terrain the user manipulates evolves according to first-order geological laws. Consequently, our validations show that terrains modeled using our system capture important landform features of mountain ranges, such as the asymmetry in landform resulting from the presence of a crustal scale fault and multi-scale folding of crust layers appearing in eroded regions.

Of the limitations discussed in Section 4.5, the most important are the lack of secondaries faults and the handling of the history of plates motion. In particular, considering a lateral component to the uplift (Castelltort et al. 2012) and proposing a timeline (a control tool similar to the one used in Chapters 6 and 7) controlling plates activity would leverage the quality and diversity of our results. These are open possibilities for improving this method. A possible extension is the application of our sculpting system as an educational tool for teaching geology. Note that towards this direction, our model has already been selected to be demonstrated as an interactive experiment within the *MusX*, the museum of sciences at Ecole Polytechnique.

In the next chapter, we show how the quality of the terrain surface can be improved by simulating glacial erosion in addition to fluvial erosion effects.

Chapter 5

Glacial erosion



Contents

5.1 Overview **85**
- 5.1.1 Glacial erosion in Geology 85
- 5.1.2 Governing equations for glaciers 86
- 5.1.3 Efficient simulation of glacial erosion 87
- 5.1.4 Secondary erosion 88
- 5.1.5 Main algorithm 88

5.2 Ice flux propagation over the terrain **89**
- 5.2.1 Path graph computation 90
- 5.2.2 Ice flux propagation 90

5.3 Steady-state and erosion **91**
5.3.1 Computations at each iteration 92
5.3.2 Convergence 94
5.4 Debris flow, fluvial and hill slope erosion **95**
5.4.1 Debris flow and fluvial erosion 95
5.4.2 Hill-slope erosion 95
5.4.3 Interactions with glacial erosion 97
5.5 Results and discussion **97**
5.5.1 Validation experiments 98
5.5.2 Efficiency and speed 103
5.5.3 Limitations 105
5.6 Conclusion **106**

Glacial erosion carved large volumes of rock during the past glaciations. The effect of glaciers on a terrain is somehow dissimilar to the effect of other phenomena. Although the incidence of glaciers and ice sheets has decreased dramatically since the last major glaciation, some 18,000 years ago, their legacy remains. During previous ice ages, which have dominated the Earth's climate over the past three million years, the abrasion of fast flowing glaciers has shaped most of the terrains, sculpting valleys, fjords and mountain ranges such as the Cordillera and Alps. Mountain glaciers, in particular, create specific landforms, including U-shaped (or tunnel) valleys, cirques and arêtes, hanging valleys at the convergence of two glaciers (of which the Bridal Veil Valley in Yosemite is a well-known example), and glacial lakes, as well as smaller-scale moraines and drumlins that form near the margins of glaciers and ice sheets.

In previous chapters, we introduced geologically based methods for generating terrains by coupling fluvial erosion (Chapter 3) with tectonically driven uplift (Chapter 4). These techniques lead to V-shaped and dendritic valley patterns and cannot explain the formation of the glacial features–an issue shared with previous simulation methods in Computer Graphics (Chapter 2, Section 2.3.2). Therefore, we extend our approach by incorporating glacial erosion in the temporal simulation of large scale terrains.

From the key observation that the response time of glaciers is several orders of magnitude below the response time of glacial erosion, we build a new implicit simulation method for glacial erosion, under the assumption that the glacier flows at steady state.

In order to effectively simulate glaciers, it is important to understand their nature as permanent bodies of dense ice, resulting from a balance of snow accumulation at high altitudes, and melting and sublimation at lower altitudes. They flow very slowly under their own weight; the fastest glaciers on Earth move at a speed just above 10 km/yr. While moving, they abrade surfaces and transport debris from the underlying bedrock. As a result, the carving of U-shaped valleys occurs over an extended period, from tens of thousands to millions of years, and this makes glacial simulation challenging.

Although the flow of glaciers can be modeled using a shallow ice approximation (SIA) of the Navier-Stokes equations, time-steps used in existing computational methods are limited to a few days at most in order to achieve convergence. This makes simulation over the

required time-period impractical. To compensate, geologists use very coarse terrain grids for glaciation that are insufficiently detailed for the purposes of Computer Graphics.

We present a novel, geologically-inspired model of glacial erosion that allows for the generation of detailed heightfields (typically, extending over 50×50km, with discrete cells $50 - 100$m in extent). Our method generates medium to large-scale glacial landforms, such as U-shaped and hanging valleys, cliffs and glacial lakes.

Given the slow pace of ice flow, we are able to approximate the evolution of a glacier as a series of equilibrium states. This enables us to generate a new state and apply erosion every 1,000 years to account for a slowly-evolving climate. Using our method, we are able to capture the expansion and retreat of glaciers during typical glacial and interglacial periods, and their erosive impact on the underlying terrain. We also extend geomorphological models to incorporate erosion due to debris entrainment, which is essential for capturing the evolution of glaciated landforms over long periods, due to the steep slopes glaciers leave behind them. Our method generates characteristic landforms that are visually significant but impossible to obtain with existing simulations, such as hanging and U-shaped valleys, eroded cliffs, and glacial lakes.

In summary, our main technical contributions are:

- A novel, efficient solution for generation of the steady-states of glacier surfaces.
- A mechanism for enabling the simulation of a glacial erosion step within the same process.
- The incorporation of slope erosion on surfaces exposed by retreating ice.

5.1 Overview

We extend terrain generation to include the effects of glacial erosion. The challenge is twofold: achieving accuracy sufficient to capture the characteristic glacial shaping of terrains and coping with simulation over geological time-scales. Our method takes inspiration from geological models and adapts and extends them to fit the requirements of Computer Graphics. In particular we increase computational efficiency, and both spatial extent and sampling resolution, as well as incorporate complex phenomena such as erosion due to transportation of cliff debris.

The input to our algorithm is an initial digital elevation model (DEM) defined on a regular grid. Given that the time spanned by our simulations is sufficient for evidence of tectonic uplift, we also allow the user to submit an uplift map defining on a per-cell and per time-step basis a change in bedrock elevation that accounts for mountain growth.

Our algorithm simulates the combined action of ice flow and glacial erosion of the underlying bedrock, as well as smaller-scale erosive phenomena, such as debris transportation. The output is a layered model representing bedrock elevation and ice thickness at each simulation step.

5.1.1 Glacial erosion in Geology

In contrast to Computer Graphic research, geologists have explored and validated models for ice-flow and glacial erosion over many decades (Braun et al. 1999; Egholm et al. 2011; Harbor

et al. 1988; Headley et al. 2012; Mahaffy 1976; Sternai et al. 2013). Their motivation is a search for governing equations that match empirical observations, from measured ice flows to existing identified and classified landforms. Our choices for the ice flow equations and resolution methods are motivated by this literature, as explained next.

Glacier flow is usually modeled using a Shallow Ice Approximation (SIA) model. This is a 0-th order quasi-static variant of the Navier-Stokes equations (Mahaffy 1976), where viscosity is computed using Glen's law (Glen 1955; Nye 1957), stating a power law relationship between strain and stress (Headley et al. 2012).

Glacial erosion is considered proportional to the speed of the ice in contact with bedrock (Herman et al. 2015). Although ice movement is the sum of a deformation velocity $\mathbf{u_d}$ and a sliding velocity $\mathbf{u_s}$ (Knap et al. 1996), sliding has been shown to be dominant over internal glacier deformation in temperate, wet-based glaciers that form in high-relief mountainous areas. This is particularly true in regions where the ice thickness is low to moderate, as supported by field measurements (Savage et al. 1963). Therefore, we neglect the deformation velocity.

Although valid for large glaciers or ice sheets, standard SIA fails to account for constriction in narrow alpine valleys and thus cannot predict the formation of characteristic U-shaped valleys (Egholm et al. 2011). Braun et al. 1999 tackle this by adding a factor inversely proportional to the curvature of the valley in the sliding speed of the ice, which in turn dictates erosion of the underlying bedrock.

Solving the SIA and computing the associated glacial erosion is typically performed using an Eulerian approach with a linearized Alternating Direction Implicit (ADI) scheme (Mahaffy 1976). Unfortunately, the unconditional stability of ADI is not guaranteed for non-linear equations such as the SIA model, forcing the use of relatively small time steps of less than one year. This makes glacier simulation impractical on high resolution grids, given the time-spans involved (10^5 years or more). Therefore, geologists usually solve the equations for 2D profiles or coarse 32×32-sized grids, which do not provide accuracy sufficient for plausible terrains in Computer Graphics applications. Also, to the best of our knowledge, geologists have never combined glacial erosion with more local erosive phenomena, such as debris slippage from steep slopes left by retreating glaciers.

5.1.2 Governing equations for glaciers

Let us consider a glacier, on a bedrock of height z, with sliding but no deformation, of local ice thickness h, elevation $s = z + h$, and ice velocity $\mathbf{u_s}$ (see Figure 5.1).

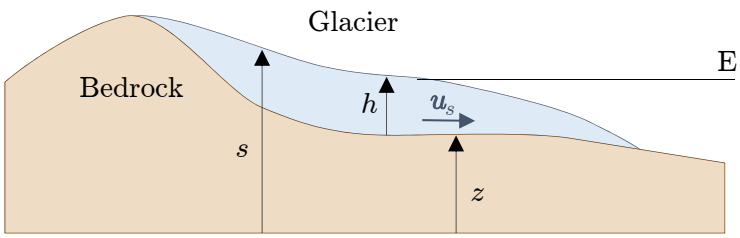


Figure 5.1: Notation for input values

The Shallow Ice Approximation (SIA) yields the following equations for a temporal change

in ice thickness (Headley et al. 2012):

$$\frac{\partial h}{\partial t} + \nabla \cdot h\, \mathbf{u_s} = M, \tag{5.1}$$

where M is the mass balance denoting the local difference between snow precipitation and melting at a point. This is computed as a proportion β of the difference between the local surface elevation and the equilibrium altitude E at which melting counterbalances precipitation:

$$M = \beta\,(s - E)\,. \tag{5.2}$$

Equation (5.1) only holds when $h > 0$; otherwise, h is set to 0 as a boundary condition. The sliding velocity $\mathbf{u_s}$ can be expressed as by Bindschadler 1983:

$$\mathbf{u_s} = -k_s\,\mu\,h^2 \|\nabla s\|^2 \nabla s, \tag{5.3}$$

where k_s is the sliding coefficient. The variable μ is used to mimic the constriction of valley walls on the ice flow responsible for generating the characteristic U-shapes (Braun et al. 1999), and is expressed as:

$$\mu = \left(1 + k_\mu \frac{\partial^2 z}{\partial_f^2 x}\right)^{-1}, \tag{5.4}$$

where k_μ is a parameter defining valley width and $\partial^2 z/\partial_f x$ is the second derivative of the bedrock elevation in a direction orthogonal to the ice flow.

The erosion of the underlying bedrock caused by a glacier can then be written as:

$$\frac{\partial z}{\partial t} = -k_e\,\|\mathbf{u_s}\|^l. \tag{5.5}$$

The erosion parameters l and k_e show little variability across rock types (Herman et al. 2015). Two possible setups are given: $l = 2.02$ and $k_e = 2.7\,10^{-7} m^{1-l}/year^{1-l}$, or $l = 1$ and $k_e = 10^{-4}$. We chose the second setting in our model, an exponent $l = 1$ drastically simplifying the equations.

5.1.3 Efficient simulation of glacial erosion

Although based on validated equations, our solution departs from the standard integration of the SIA equation used in geology. Rather, we approximate the ice surface as a succession of steady states, with a specific schedule of ice flow and erosion computation over the terrain grid. An important improvement is that our method allows very large time steps, well suited to the temporal scale of the modeled phenomena, and achieves linear computation time at each step with respect to the number of grid cells.

This relies on two key insights: first, that the fast pace of ice-flow (with respect to erosion time scale) enables a steady-state model of glacier evolution, and second, that the glacier formation and its erosive impact can be computed, efficiently and simultaneously, by applying a specific traversal order over grid cells.

More precisely, computing the flux of ice (namely, the quantity of ice exiting a cell) follows a specific task scheduling, which ensures that flux in a given cell at elevation s is

always computed before neighboring cells of lower elevation. Once the flux pass is complete, the new steady-state ice thickness and bedrock shifts due to erosion are propagated upwards in reverse. Note that this scheduling circumvents an $O(n \log(n))$ pre-sorting of n cells with respect to their current altitude at each iteration of the ice steady-state computation. In contrast, each iteration of our algorithm runs in linear time.

The combination of a steady-state model with a linear-time solution makes our method fast and stable, enabling the simulation of a full cycle of advancing and retreating glaciation over a $25 - 50$km terrain with a cell size of $50 - 100$m in less than 30 minutes.

5.1.4 Secondary erosion

Even though snow and ice cover much of a landscape during periods of glaciation there are still steep uncovered surfaces that are subject to other forms of erosion. Furthermore, exposure during interglaciation should not be neglected as it also contributes to the erosion history written in the terrain surface.

Fortunately, our Eulerian solution, where bedrock and ice are represented as distinct layers on a regular grid, allows us to switch between glacial and secondary erosion effects based on ice thickness. In particular, we incorporate debris flow (the steepening effect of rockslides scouring slopes) as well as standard fluvial erosion (water channels cut into the landscape) and hillslope erosion (the rounding out of crests and valleys due to sediment transport).

5.1.5 Main algorithm

In summary, our algorithm takes as input an initial terrain stored on a regular grid, and outputs the eroded terrain and the ice thickness. We assume that for each time step the ice surface is in a steady-state equilibrium. This state is iteratively computed for each time step (of duration dt) as follows:

Algorithm 5.1: At each time step:

```
if t = 0 then
    s_0(t = 0) ← z ;
else
    s_0(t) ← s(t − dt) ;
k ← 0 ;
repeat
    k ← k + 1 ;
    Build a computation order based on surface altitude s_{k−1}(t);
    Compute the ice flux q_k from highest to lowest altitude.;
    foreach cell c in upward computation order do
        if q_k(c) > 0 (there is ice) then
            Solve ice and erosion equations;
            s_k(t) ← z_k(t) + h_k(t);
    end
until changes in s(t) are below a defined threshold;
Add the effect of the other erosion phenomena.
```

Name	Value	Unit
Physical		
β	$1.0e^{-3}$	y^{-1}
k_s	1.7	$m^{-1}\,y^{-1}$
l	1	1
k_e	10^{-4}	1
Control		
E	$1e^3 - 4e^3$	m
k_μ	$1e^5$	m

Table 5.1: Glacial erosion parameters

User control is provided through a small set of intuitive parameters that have a straightforward relationship to the shape of the resulting eroded landscape, namely the equilibrium line altitude E, which defines the mean altitude of glaciers, the precipitation coefficient β, which controls glacial extent and the constriction parameter k_μ, which relates to the width of U-shaped valleys. The remaining parameters are all pre-set using validated values from geology literature (see Tables 5.1 and 5.2).

Name	Value	Unit
Fluvial erosion		
k_f	$5\,10^{-6}$	$m^{2m-1} year^{-1}$
m	0.4	1
n	1	1
Hill Slope		
k_h	$5\,10^{-3}$	$m^2\,year^{-1}$
Debris Flow		
k_{df}	10^{-4}	$m\,year^{-1}$
k_{da}	5	$m^{-2\,q}$
q	.2	1
q'	1.5	1

Table 5.2: Parameters used in the other erosion processes

5.2 Ice flux propagation over the terrain

Crucial to our technique is a proper ordering of cells preparatory to propagating ice-flux. This takes place for each time step and on every iteration of the glacial steady-state search.

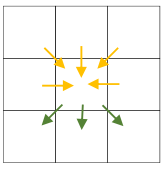
Equations (5.3) and (5.1) lead to following relationship between ice thickness and surface slope at steady-state equilibrium:

$$\nabla \cdot k_s\,\mu\,h^3 \|\nabla s\|^2 \nabla s + M = 0, \tag{5.6}$$

where the first term represents ice transport and M is the mass balance from Equation (5.2). We use Equation (5.6) to prepare the required order of evaluation for ice flux propagation.

5.2.1 Path graph computation

Let us consider a graph whose nodes are cells of the terrain, with edges that connect a cell to each of its eight neighbors. We first note that the glacier corresponds to all terrains cells with ice thickness $h > 0$. This always occurs where $M > 0$ or equivalently when $s > E$ (see Equation (5.2)). Therefore, there exists a path with positive ice thickness between each cell of the glacier and a cell where $s > E$ that also corresponds to a local maximum of the terrain. We can also show that at least one of these paths is of monotonically increasing surface altitude. Otherwise, there would be a local maximum on the surface with $M < 0$, indicating a flow out of that local maximum and thus $h = 0$. The set of all these paths forms a Directed Acyclic Graph, which we call the *Path Graph*, denoted as $\mathcal{G}$, and which is used to order computations on the ice surface.



To compute the Path Graph $\mathcal{G}$ for each cell c, we build a list of neighbors with strictly higher surface elevation (donors D_c) and, similarly, a list with lower elevation (recipients R_c). The computation order is obtained through a topological sort on $\mathcal{G}$, which ensures that all donors of a cell c are parsed before c. This is different from the stream graph introduced in Chapter 3 in that all lower neighbors are considered as receivers, and not only the one with the steepest slope.

Local minima in the ice elevation that would otherwise result in sink cells in the graph are removed by following the strategy introduced in Chapter 3, Section 3.2.3. The depression filling variant is chosen, where the ordering of the nodes in a depression is computed by assuming that the depression is filled.

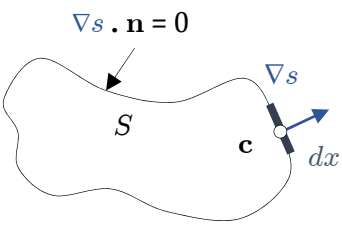
The next step consists in solving Equation (5.6) coupled with Equation (5.5) along this path graph.

5.2.2 Ice flux propagation

Considering a surface S on the terrain, delimited by a contour l, Gauss's theorem on Equation (5.6), gives:

$$\iint_S M \,\mathrm{d}S = -\iint_S \nabla\cdot k_s\,\mu\,h^3\|\nabla s\|^2\nabla s\,\mathrm{d}S = -\oint_l k_s\,\mu\,h^3\|\nabla s\|^2\nabla s\cdot\mathbf{n_l}\,\mathrm{d}l \tag{5.7}$$

where n_l is the local normal of the contour l.



We compute the ice flux by applying this equation to each discrete cell $\mathbf{c}$. This is done by constructing a contour l (see figure thereagainst) with a line segment of length dx (the grid cell size) orthogonal to the gradient of s, and with the remaining curve of the loop adjusting to the ice flow direction, such that $\nabla s\cdot\mathbf{n_l} = 0$.

With $q(\mathbf{c})$ denoting the flux leaving cell $\mathbf{c}$, we can rewrite Equation (5.7) as:

$$q(\mathbf{c}) = \iint_S M\,\mathrm{d}S = dx\,k_s\,\mu\,h^3(\mathbf{c})\|\nabla s(\mathbf{c})\|^3. \tag{5.8}$$

We solve this equation according to the previously defined computation order. This means that ∇s can only be computed from the recipients (lower neighbors) of $\mathbf{c}$. We choose to estimate $\|\nabla s\|$ as:

$$\|\nabla s(\mathbf{c})\| \simeq \frac{1}{|R(\mathbf{c})|} \sum_{r \in R(\mathbf{c})} \frac{s(\mathbf{c}) - s(\mathbf{r})}{d(\mathcal{C}c, \mathcal{C}r))} \tag{5.9}$$

where $|R(\mathbf{c})|$ is the number of recipients of $\mathbf{c}$ and $d(\mathcal{C}c, \mathcal{C}r)$ is the distance between the centroids of cells $\mathbf{c}$ and $\mathbf{r}$. The approximation is improved by including more neighbors: we thus choose recipients from the 8 direct neighbors of $\mathbf{c}$.

During iteration k, we propagate q_k based on the surface gradient $\nabla s_{k-1}(\mathbf{c})$ computed at iteration $k-1$. The flux is computed from high to low. Let us denote $q(\mathbf{c} \to \mathbf{r})$ as the flux of ice between a cell and one of its recipients. Mass conservation dictates that all of $q(\mathbf{c})$ should be distributed among the recipients of $\mathbf{c}$, so there exists an $\alpha(\mathbf{c} \to \mathbf{r}) > 0$ between all cell-recipient pairs, such that:

$$q(\mathbf{c} \to \mathbf{r}) = \alpha(\mathbf{c} \to \mathbf{r})\, q(\mathbf{c}) \quad \text{and} \quad \sum_{\mathbf{r} \in R(\mathbf{c})} \alpha(\mathbf{c} \to \mathbf{r}) = 1. \tag{5.10}$$

Experimentally, we found that the following value for α provides an effective distribution of flux.

$$\alpha(\mathbf{c} \to \mathbf{r}) = \frac{(s(\mathbf{c}) - s(\mathbf{r}))/d(\mathcal{C}c, \mathcal{C}r)}{\sum_{\mathbf{r}' \in R(\mathbf{c})} (s(\mathbf{c}) - s(\mathbf{r}'))/d(\mathcal{C}c, \mathcal{C}r)\|)}. \tag{5.11}$$

In summary, at iteration k of Algorithm 5.1 from Section 5.1.5, the building of computation order and calculation of ice flux expand as follows:

Algorithm 5.2: Flux Propagation

```
foreach cell c do q_k(c) ← S M ;
foreach cell c in descending computation order do
    foreach r ∈ R(c) do
        Compute α(c → r) from s_{k−1} (Equation (5.11));
        q_k(r) ← q_k(r) + α(c → r) min(0, q_k(c));
    end
end
```

Note that in the above, flux is clamped to 0 since the glacier is only defined where $q > 0$.

5.3 Steady-state and erosion

In the second half of each iteration (Algorithm 5.1), cells are processed upwards (in the inverse of flux order) in order to evaluate the steady state of ice h and bedrock z at a cell, from values at receiver cells. Unless specified otherwise, all computations below hold for cell $\mathbf{c}$, at time t and iteration k (from values computed at iteration $k-1$). Initialization considers two cases: 1) if $k = 0$ bedrock is initialized as $z = z(t - dt)$ everywhere; 2) otherwise, the

new flux information is used to reset ice and bedrock elevation outside the glacier: $h = 0$ and $z = z(t - dt)$ when $q \leq 0$.

5.3.1 Computations at each iteration

We compute glacial erosion by using Equation (5.8). Our fundamental approach is to solve the ice equation with a variable bedrock topography governed by Equation (5.5). Let y be the unknown depth of erosion. By combining Equations (5.5) and (5.8) we arrive at:

$$y = \frac{\partial z}{\partial t} = -k_e(\mu\, k_s)^{1/3} \left(\frac{q}{dx}\right)^{2/3} \|\nabla s\| \tag{5.12}$$

This is the primary erosion equation and the remainder of this section explains how we recast the two unknowns $\|\nabla s\|$ and μ in terms of y, so that the whole equation can be re-expressed purely as a function of y.

We begin with $\|\nabla s\|$, defined by Equation (5.9). We need to express $s = z + h$ as function of y. Computing the term-by-term quotient of Equation (5.12) with Equation (5.8) enables us to arrive at h as a function of y:

$$y\, dx/q = -k_e/h \Leftrightarrow h = -k_e\, q/(y\, dx). \tag{5.13}$$

Recall that $z = z(t - dt) + y\, dt$, and so this yields:

$$\|\nabla s(\mathbf{c})\| = \sigma_0\, y + \sigma_1 + \frac{\sigma_2}{y}. \tag{5.14}$$

The coefficients can be computed from values at the receiver cells and from bedrock elevation at time $t - dt$:

$$\begin{aligned}
\sigma_0 &= dt \sum_{r \in R(\mathbf{c})} \left(|R(\mathbf{c})|\, d(\mathcal{C}c, \mathcal{C}r)\right)^{-1} \\
\sigma_1 &= \frac{1}{|R(\mathbf{c})|} \sum_{r \in R(\mathbf{c})} \frac{z_{t-dt}(\mathbf{c}) - s(\mathbf{r})}{d(\mathcal{C}c, \mathcal{C}r)} \\
\sigma_2 &= -k_e \frac{q}{dx} \sum_{r \in R(\mathbf{c})} \left(|R(\mathbf{c})|\, d(\mathcal{C}c, \mathcal{C}r)\right)^{-1}.
\end{aligned}$$

The constriction factor μ (Equation (5.4)) is expressed as:

$$\mu^{-1} = 1 + k_\mu \frac{\partial^2 z}{\partial_f x} = \tau_0\, y + \tau_1 \tag{5.15}$$

where $\frac{\partial^2 z}{\partial^2_{\mathbf{f}} x}$ is the directional derivative of z in the direction $\mathbf{f}$ orthogonal to ∇s_{k-1}. We estimate ∇s as:

$$\nabla s \simeq \sum_{r \in R_{\mathbf{c}}} \alpha(\mathbf{c} \to \mathbf{r}) \frac{\mathbf{x}(\mathbf{r} - \mathbf{x}(\mathbf{c})}{d(\mathcal{C}c, \mathcal{C}r)}$$

where $\alpha(\mathbf{c} \to \mathbf{r})$ was computed in Equation (5.11), and $\mathbf{x}(\mathbf{c})$ is the position of cell c. The orthogonal direction is then obtained as $f = (-(\nabla s)_y, (\nabla s)_x)/\|\nabla s\|$.

The directional derivative is decomposed into the three second derivatives of z, which we express using finite difference convolution kernels:

$$\begin{aligned} D_{\frac{\partial^2}{\partial^2 x}} &= \frac{1}{dx^2}[-1, 2, 1] \\ D_{\frac{\partial^2 z}{\partial x \partial y}} &= \frac{1}{dx^2}[-1, 0, 1]^T[-1, 0, 1] \\ D_{\frac{\partial^2 z}{\partial^2 y}} &= \frac{1}{dx^2}[-1, 2, 1]^T \end{aligned}$$

Let K be the directional derivative kernel:

$$K = \mathbf{f}_x^2 D_{\frac{\partial^2 z}{\partial^2 x}} + 2\mathbf{f}_x\,\mathbf{f}_y D_{\frac{\partial^2 z}{\partial x \partial y}} + \mathbf{f}_y^2 D_{\frac{\partial^2 z}{\partial^2 y}}$$

We can now express the coefficients of the constriction factor:

$$\begin{aligned} \tau_0 &= -dt\, k_\mu\, K(1,1) \\ \tau_1 &= 1 - k_\mu\, K(1,1)\, z_{t-dt}(\mathbf{c}) + \sum_{\mathbf{n} \in N(\mathbf{c})} K(\mathbf{n} - \mathbf{c})\, z(\mathbf{n}) \end{aligned}$$

where N are the 8 direct neighbors to the cell $\mathbf{c}$, and $\mathbf{n} - \mathbf{c}$ is the 2 dimensional offset between cells $\mathbf{n}$ and $\mathbf{c}$ in the grid. When computing this directional derivative, the bedrock elevations coming from the receivers are already correct for this iteration, but others coming from the donors have not been updated yet. This is not an issue because these elevations are progressively improved at each iteration.

Finally, we combine Equations (5.14) and (5.15) to arrive at a new expression, where y is the only unknown:

$$(\tau_0 + y\,\tau_1)^{1/3}\, y + \sigma_0'\, y + \sigma_1' + \frac{\sigma_2'}{y} = 0, \tag{5.16}$$

with $\sigma_i' = k_e {k_s}^{1/3} \left(\frac{q}{dx}\right)^{2/3} \sigma_i$, for $i \in [|0, 2|]$.

Since there is no close-form solution for Equation (5.16), we solve it using a standard Newton-Raphson iterative scheme, as follows: We build a series of y_n that converge quickly to y, using:

$$y_n = y_{n-1} - F(y_n)/F''(y_n)$$

where $F(y) = 0$ is exactly given by Equation (5.16). The constriction only applies when the Laplacian is positive, *i.e.,* when $\tau_0 + y\,\tau_1 > 1$. Second, we try to find y, the variation of bedrock height, that is always negative. There are some constraints on the coefficients: $\sigma_0 > 0$, $\sigma_2 < 0$, $\tau_0 < 0$. The derivatives of F are:

$$F'(y) = \frac{\tau_1}{3}(\tau_0 + y\,\tau_1)^{-2/3} + \sigma_0' - \frac{\sigma_2'}{y^2}$$

$$F''(y) = -\frac{2\tau_1^2}{9}(\tau_0 + y\,\tau_1)^{-5/3} + \frac{2\sigma_2'}{y^3}$$

Therefore, $F'(y) > 0$ on the domain of admissible solutions and there exist at most one value of y where $F''(y)$ changes its sign. Moreover, $F''(y) > 0$ when $y \to 0$ and $F''(y) < 0$ when $y \to -\infty$.

We need to find an initial guess y_0 for the iterative Newton-Raphson process that makes y_n converges to the solution y. The initial guess y_0 should be between the final y and $\min(0, -\tau_0/\tau_1)$ (above which the equation is not defined). This is a condition to avoid that the series y_n becomes positive, and thus might not converge.

We design the following algorithm for finding an initial guess. Set $y_{-1} = (1 - \tau_0/\tau_1)$. If $y_{-1} < 0$ and $F(y_{-1}) < 0$, there is no possible solution with $\tau_0 + y\,\tau_1 > 1$. That means that the erosion can not be constrained by the valleys because of the too low initial curvature. This is solved by setting $k_\mu = 0$ and solving the second order equation:

$$(\sigma_0' + 1)\, y^2 + \sigma_1'\, y + \sigma_2' = 0$$

which has always a single negative root.

Else, if $F(y_{-1}) \geq 0$ and $y_{-1} < 0$, then $y_0 = y_{-1}$ verifies the good conditions for convergence.

The last case deserves more attention: $y_{-1} > 0$. If we find a bound B such that $(\tau_0 + y\,\tau_1)^{1/3} < B$, then an upper bound for y can be found by solving:

$$(\sigma_0' + B)\, y_+^2 + \sigma_1'\, y_+ + \sigma_2' = 0$$

And $y_0 = y_+$ is a possible initial solution. A lower bound of y can be found similarly by solving

$$\sigma_0'\, y_-^2 + \sigma_1'\, y_- + \sigma_2' = 0$$

and because $(\tau_0 + y\,\tau_1)^{1/3}$ is decreasing, the bound B can be found as $B = (\tau_0 + y_-\,\tau_1)^{1/3}$.

From the amount of erosion y, we finally update $h = -k_e\, q/y/dx$ and $z = z(t - dt) + y\, dt$.

5.3.2 Convergence

If applied as it stands, the iterative process may not converge. We need to apply a relaxation factor r directly after calculating y for a given cell, before progressing the next one. In our implementation, we progressively lower the relaxation factor from $r = 0.25$ to $r = 0.1$:

$$h(\mathbf{c}) = -r\,\frac{k_e\, q}{y\, dx} + (1 - r)\, h_{k-1}(\mathbf{c})$$

$$s(\mathbf{c}) = r\,\left(z(t - dt) + y\, dt + h(\mathbf{c})\right) + (1 - r)\, s_{k-1}(\mathbf{c})$$

$$z(\mathbf{c}) = s(\mathbf{c}) - h(\mathbf{c})$$

A special case occurs when a cell has negative or zero flux q, while some of its recipients do not. This can occur if successive iterations reach an invalid state, where a local maximum is found with negative mass balance. In this case, we set the ice surface to the average of its recipients. Given these adjustments, the process converged rapidly in all our experiments (see Section 5.5).

5.4 Debris flow, fluvial and hill slope erosion

Combining glacial erosion with other phenomena is essential to generating plausible terrains: Indeed span of mountain formation is several orders of magnitude longer than glaciation cycles, so most landscapes have been exposed to various climates, including interglacial periods where other types of erosion predominate. Even during ice ages, the lower glacier-free foothills of a mountain are shaped by other prevailing erosion processes. We have selected the following processes because of their impact on landscapes: debris flow, fluvial erosion and hill slope erosion.

5.4.1 Debris flow and fluvial erosion

Rock- and mud-slides on steep slopes gain a lot of inertia and have a huge erosive power that is responsible for gouging cliffs at small to medium scale. This phenomenon is crucial because retreating glaciers leave exposed cliff faces that are susceptible to debris flow. Our method improves on classical thermal erosion (Musgrave et al. 1989) that reduces bedrock slopes toward some equilibrium angle: we add an important term referenced in geology (Stock et al. 2003) that accounts for drainage:

$$\frac{\partial \mathcal{B}}{\partial t} = -k_{df}\ (1 + k_{da} A^q)\, \|\nabla \mathcal{B}\|^{q'}, \tag{5.17}$$

where the constants k_{df}, k_{da}, q', q are detailed in Table 5.2. The resolution method is very similar to the one explained in Chapter 3 : a graph ordering is constructed from a single recipient chosen among the neighboring nodes with the steepest slope. The drainage area is computed as the water flows from top to bottom, accumulating the cell area dx^2. The erosion is then computed implicitly from bottom to top, by approximating the gradient of the bedrock $\nabla \mathcal{B}$ using the slope toward the recipient $\mathcal{C}r$:

$$\|\nabla \mathcal{B}\| \simeq \frac{\mathcal{B}(\mathcal{C}c) - \mathcal{B}(\mathcal{C}r)}{d(\mathcal{C}r, \mathcal{C}c)}.$$

Fluvial erosion is solved similarly as explained in Chapter 3:

$$\frac{\partial \mathcal{B}}{\partial t} = -k_f\, A^m \|\nabla \mathcal{B}\|^n,$$

5.4.2 Hill-slope erosion

While *fluvial erosion* is valid for large drainage areas, *hill slope erosion* is dominant along ridges and accounts for more subtle erosion due to wind, rain, or gravel and sediment accumulating over the course of long-standing erosion of the terrain (Braun et al. 1997). It is modeled as a diffusion equation:

$$\frac{\partial \mathcal{B}}{\partial t} = k_h\, \Delta \mathcal{B}.$$

We solve this equation by using an Alternating Direction Implicit Scheme (Wachspress et al. 1960).

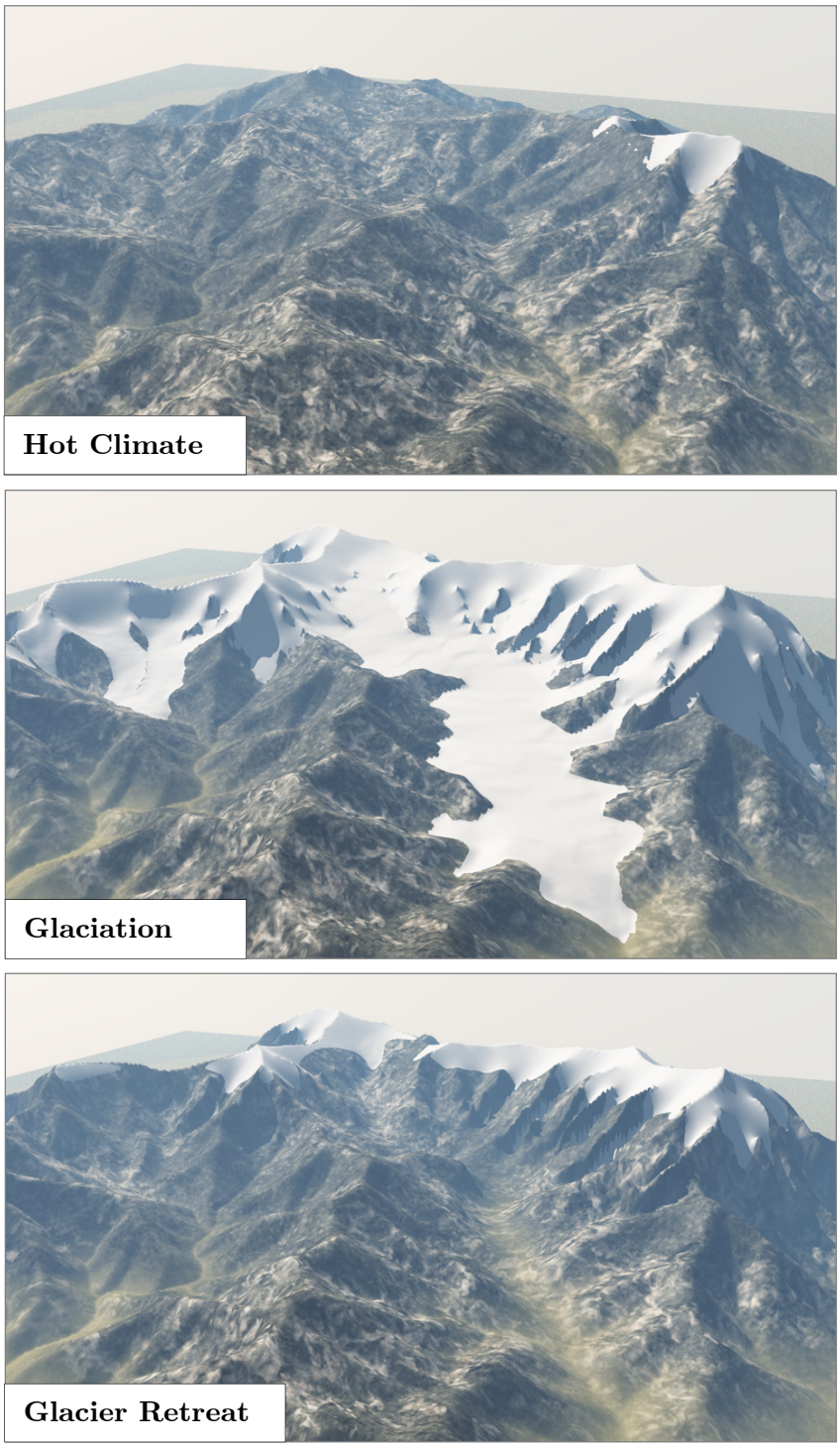


Figure 5.2: Timelapse of a landscape evolution. A mountain range is first computed for $3,000,000$ years with mainly fluvial erosion on a hot climate (top). Then, the environment undergoes a rapid cooling, and the ice erodes the bedrock for $100,000$ years (middle). The glacier finally retreats, leaving behind large U-shape valleys, hanging valleys and ridges.

5.4.3 Interactions with glacial erosion

Ice shields the terrain from other erosion processes so we do not apply them when the ice thickness is above a threshold ($3m$ in our experiments). This is, of course, an additional and important reason why we track ice thickness during simulation.

When used, the processes are applied in the following order: glacial erosion, then where ice is sufficiently thin, hill slope, followed by combined fluvial and debris flow erosion.

The erosion equations can be used on a static terrain for a short period of time. However, given the scale of our simulations, applying an uplift map that accounts for the simultaneous growth of mountain ranges under tectonic action is necessary to achieve plausible results.

5.5 Results and discussion

Alternation between warm interglacial periods and glaciation was simulated on a 50×50km terrain with 100m grid cells. As input, we used high persistence ($p = 0.6$) multi-fractal noise (Ebert et al. 2002) to simulate variations in uplift and generate mountainous landscapes. Warm climatic conditions were first simulated over a 3 million years period, with $dt = 10,000$ years. A high equilibrium altitude for glaciers ($E = 3,250$m) ensured that erosion came mainly from fluvial, hill-slope, and debris-flow processes, even though glaciers affected mountain tops. This was intended to form the terrain in preparation for glacial erosion (Figure 5.2, top).

To demonstrate glacial erosion, we applied fast cooling conditions, which lowered E to $2,650m$ over $30,000$ years. We reduced dt to $1,000$ years. The cooling continued for $70,000$ years, until E stabilized at $2,500\,m$ (Figure 5.2, middle).

Then, we set the climate to fast warming conditions (E rose to $3,500\,m$ in $20,000$ years), to allow other processes to progressively smooth glacier-dominated landforms (Figure 5.2, bottom).

The resulting terrain shows typical glacial patterns, namely U-shaped valleys, hanging valleys, and extensive cliffs left by retreating glaciers (Figure 5.4), that can be compared with landform features found in real mountain ranges (Figure 5.3).

Figure 5.3: Small side-valley glaciers converging on a large main glacier often leave hanging valleys. Our results (left) have a similar structure to the Bridalveil Fall in Yosemite National Park (right, ©Creative Commons).

We vary the erosion parameters to show a different landscape, illustrating the impact of

different glaciations strengths. For the bedrock erosion in Figure 5.5, the equilibrium line E was set to 110%, 70% and 30% of the initial bedrock elevation.

5.5.1 Validation experiments

We designed three experiments on a smaller grid size of 250×250 cells to show that our method can produce the principal patterns formed by glacial erosion.

U-Shaped valleys. Figure 5.6 shows the formation of a U-shaped valley. The bedrock was initialized with a V-shaped structure. We modified the mass balance function Equation (5.2) so that M had a Gaussian distribution over the ice source, and was negative elsewhere. We show the resulting glacier and the underlying eroded bedrock after 10 time steps of $10,000$ years each. As expected, the bedrock profile follows a parabolic curve, matching measurements in geology (Svensson 1959). Our results also show how the valley profiles are impacted by the constriction factor μ. This experiment shows not only the importance of the constriction factor that widens the bottom of valleys, but also its importance and effectiveness for the computation of erosion near the top of steady state glaciers. As the erosion is computed using implicit integration, ice and erosion only exist relatively to the next time step. Without constriction, the simulation always converges to a deep, infinitely thin trench. The constriction factor forces the glacier to slow down where bedrock exhibits high curvatures, which increases the amount of ice and enables the terrain to reach a stable equilibrium.

Hanging valleys We explored the formation of a hanging valleys (Figure 5.7). We added a second smaller V-shape valley to the terrain of the first experiment, using smaller localized absolute value function in the y direction. We added a smaller ice source at the top of it. We again show the resulting glacier, the eroded bedrock and the ice profile along the small valley–now a hanging valley. The cross-section view depicts the ice profile of a valley eroded with fluvial erosion only, for comparison. In contrast with fluvial erosion that smooths the bedrock surface in the valley's main direction, glaciers smooth the ice surface. When the volume of ice between two valleys is different, the smaller valley will only be eroded until the ice surface reaches the surface of the main glacier, leaving a strong discontinuity in the altitude of the bedrock.

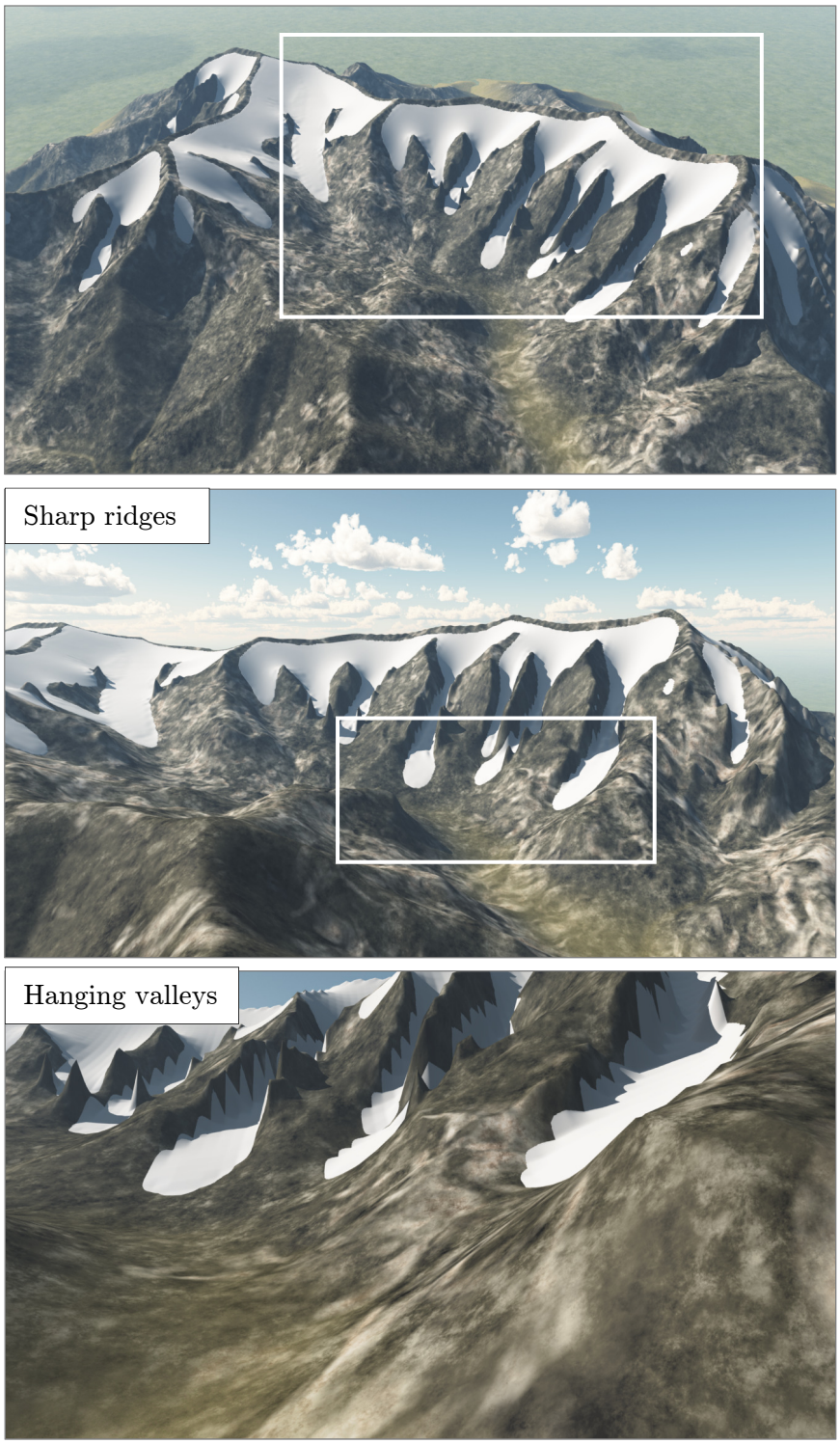


Figure 5.4: Retreating of the glacier left a typical glacial landscape (top), marked by sharp ridges and steep cliffs (middle), and hanging valleys (bottom). The main U-shape valley can be seen in all three images.

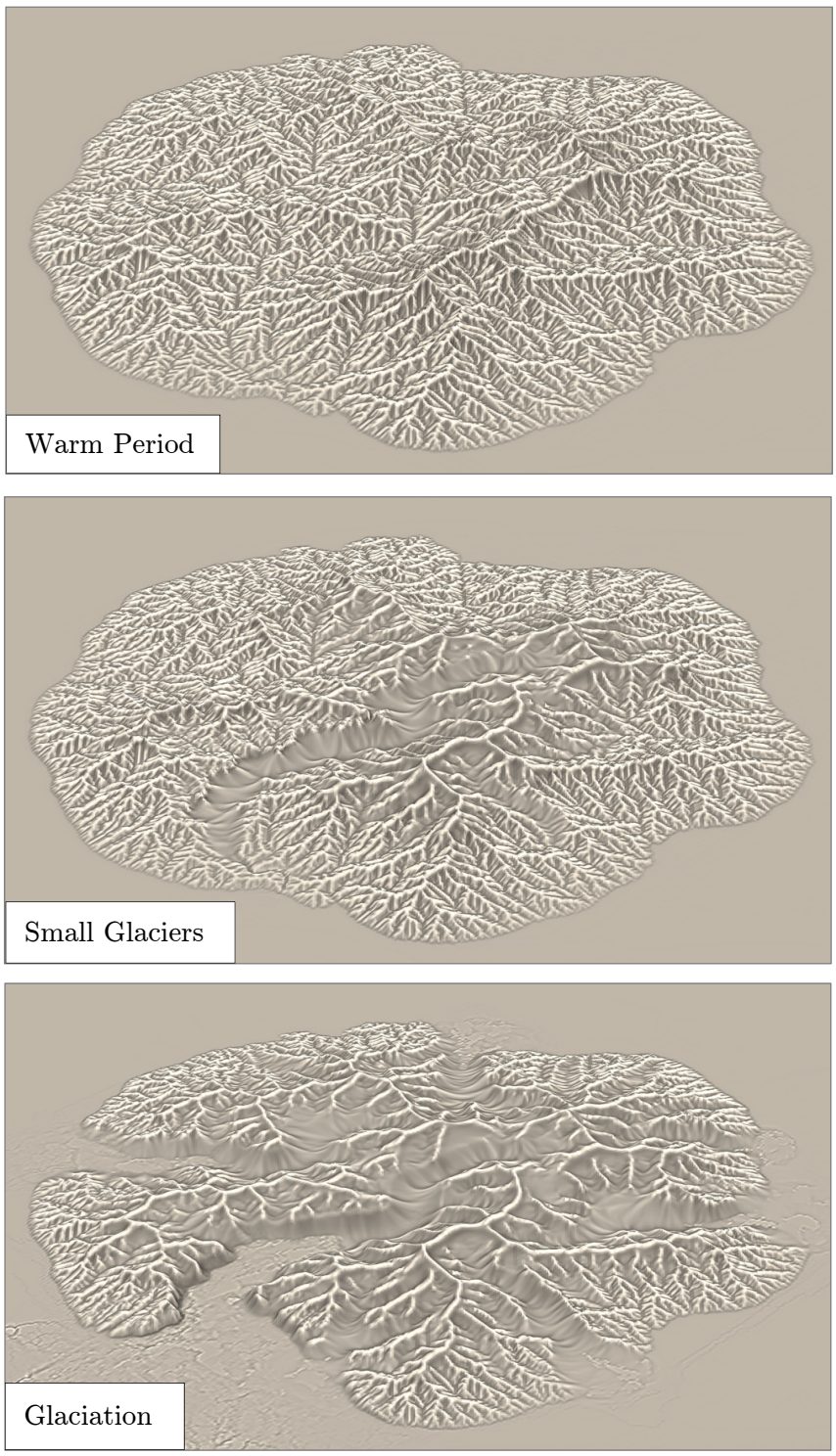


Figure 5.5: Non-photorealistic renderings enhancing ridges. Top: no glacier; middle: small glaciers; bottom: full glaciation. The ice has been removed to show the progressive erosion of the bedrock layer.

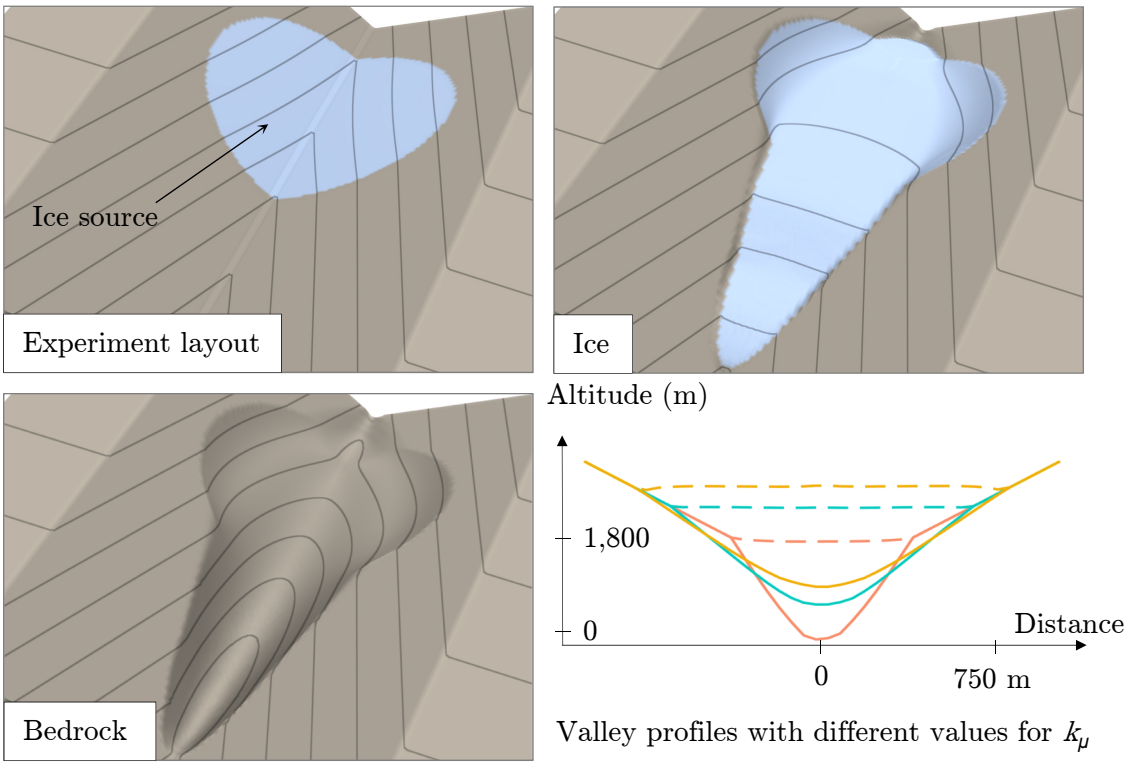


Figure 5.6: Experiment for U-shape valleys. The layout (top left) consists of a simple ice source on a slowly decreasing, regular V-shape valley. After 10 erosion time steps of $10,000$ years each, a glacier (top right) forms above an eroded U-shape valley in the bedrock (bottom left). A valley profile is captured (bottom right) for different values of k_μ: $10,000\,m$, $100,000\,m$ and $1,000,000\,m$. Solid lines show bedrock surface, dotted lines ice surface.

Glacial lakes. Our hypothesis for the formation of locally deeper areas due to glacial erosion was to relate them to abrupt changes of ice volume. To demonstrate this, we built a favorable landscape for lake formation. Three quarters of the landscape were subject to uplift with random variations, the remaining quarter being left flat as a rest area for the glacier. We artificially increased the altitude at the boundaries of the uplifted region during fluvial erosion simulation, except at a single spot, in order to force all water to leave the landscape at this specific spot. Running a fluvial erosion simulation during $1,000$ steps of $10,000$ years each led to a landscape with many convergent valleys (Figure 5.8, top left). The ice simulation was then performed on 95 steps of $10,000$ years in order to simulate the cumulated effect of several glaciations (Figure 5.8, top right), before progressively retreating the ice during 40 frames of $1,000$ years each. As a result, deep depressions dug by the glaciers left place to lakes (Figure 5.8, bottom). As expected, the convergence of valleys played an important part in the location of these major depressions.

Debris flows. Our last experiment shows the effect of debris flow on a terrain. We built a virtual initial peak on a 5×5km terrain (10m resolution) from Fractional Brownian Motion dampened with a Gaussian, and run the complete simulation with the following parameters

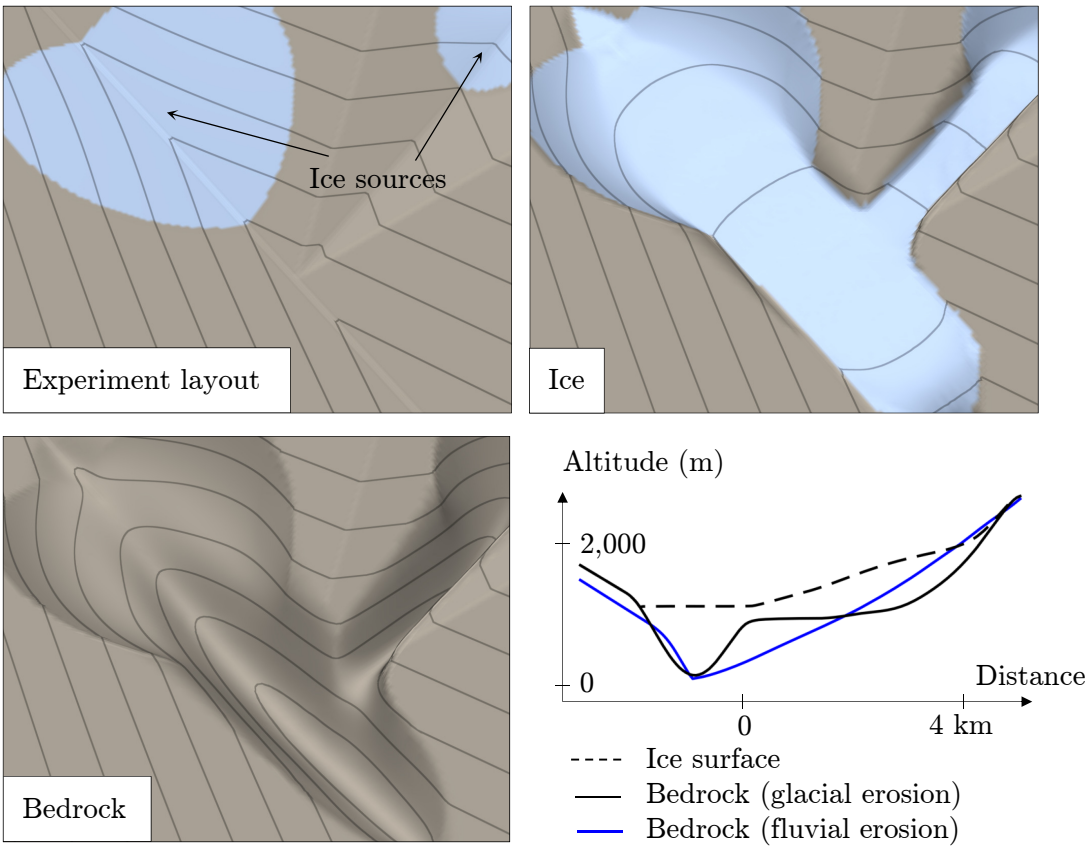


Figure 5.7: Experiment for hanging valleys. The initial conditions (top-left) are set with two V-shape valleys: a main one with wide lateral borders and tiny downward slope, and a smaller orthogonal side valley. Ice sources were set at the upper part of both valleys. Results, after 10 iterations of 10,000 years, show two convergent glaciers with a continuous ice surface at the junction (top right). The eroded bedrock (bottom left) shows a high discontinuity where the *hanging valley* forms. A longitudinal cut of the small valley was extracted (bottom right), showing the bedrock with a solid black line and the ice surface (dashed line). For comparison, we added the result of a fluvial erosion on the same initial layout, where the side valley connects directly to bottom of the main valley.

for erosion: (debris flow: $q = 0.2$, $q' = 2$; $k_{df} = 2.0\,10^{-4}\,m\,year^{-1}$; $k_{da} = 7.5\,m^{-0.4}$; fluvial erosion : $m = 0.7$; $n = 1.4$; $k_f = 3.5\,10^{-6} m^{-1.4}\,year^{-1}$; hill slope $3\,10^{-2} m^2\,year^{-1}$). The ice flow was constricted by an equilibrium line at 40% of the initial bedrock elevation. The simulation was performed on 30 time steps of 10,000 years each (Figure 5.9), top). Then the ice was removed and we performed two new simulations of 10 time steps of 10,000 years each, with and without debris flow erosion. Results show a strong difference in the eroded patterns in the main cliffs: while fluvial erosion digs deep trenches, separated by ridges smoothed by hill slope (Figure 5.9), middle), debris flow erosion results in strong, triangle-like patterns at the highest slopes, and is negligible elsewhere (Figure 5.9), bottom). This advocates for the fact that both erosion techniques should be used, especially in steep terrains such as the ones formed by glacial erosion.

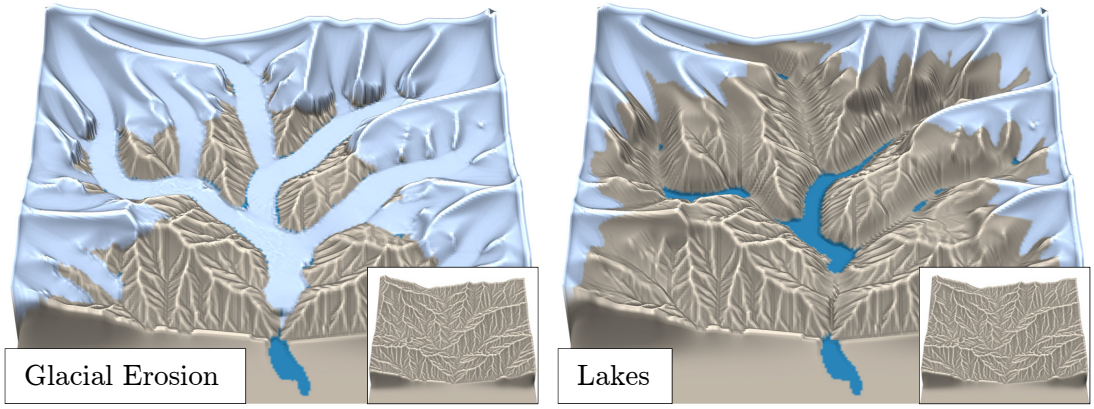


Figure 5.8: Lake formations. The initial conditions ensured many convergent valleys after 10 million years of fluvial erosion (insets). About 1 million years of glacial erosion was performed (left), deeply eroding depressions that turns into lakes when glaciers retreat (right).

5.5.2 Efficiency and speed

Designing a good threshold for convergence is hard, first because the equation being not defined at the boundaries of the ice, the iterations may lead to spurious osculations at the exterior of the glacier, second because the variations in glacier erosion may become negligible before the ice surface stabilizes. We found an upper bound on the number of iterations by analyzing the evolution of the difference of ice surface between two oscillations–which always stabilizes. In practice, the iteration number is set by hand, between 20 to 100, up to 200 in particularly complicated steps (ice added directly on top of a V-shaped landscapes, where the constriction varies drastically).

Timings were measured on a computer shipped with an Intel Core i7- 4710MQ CPU. The single threaded implementation was done in Python with Numba *just in time* compilation, which achieve near native performances. Table 5.3 shows the results for different grid sizes, along with the percentage of time needed for each different step of the algorithm. The total simulation time per time step sums 75 frames, which gives an idea of a scene with enough variety without being the most complex (for instance Figure 5.9). A typical simulation time for a time-lapse of 400 time steps on a 500^2 cells grid is 98 min.

Grid size	Iteration	Depr.	Order	Flux	Ice	Total
250^2	42 ms	21.3%	20.1%	5.9%	52.7%	3.19 s
500^2	190 ms	24.1%	21.8%	6.6%	47.5%	14.8 s
1000^2	838ms	24.7%	21.1%	7.3%	46.9%	62.9 s
2000^2	3.36s	26.5%	22.4%	6.9%	44.2%	251.8 s

Table 5.3: For different grid sizes, the time taken for one iteration, with percentage of used time between the different steps of the algorithm (depression filling, computation order, flux and ice computation).

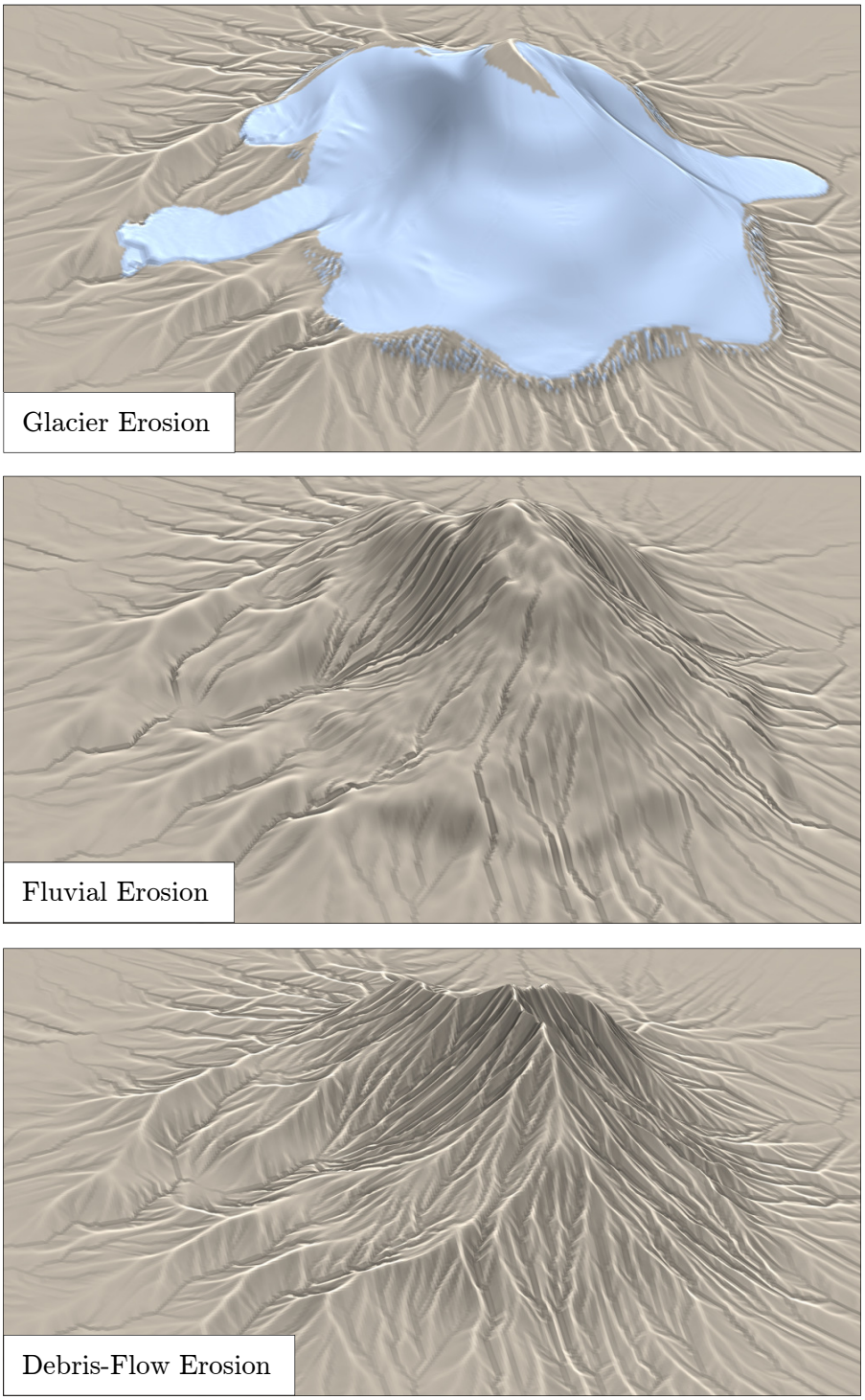


Figure 5.9: Secondary erosion effects. Steep slopes were carved on virtual peak by a glacier during $30,000$ years (top). Ice was suddenly removed, and the free slopes were eroded during another $10,000$ years by either only fluvial erosion and hill slope (middle), either all erosions methods, in particular debris-flow (bottom).

5.5.3 Limitations

A limitation of our steady-state approach is that the simulations cannot be used to compute a time-lapse animation showing the continuous evolution of the glacier. In theory, this could be achieved using smaller time steps, but only if convergence to the ice steady state was perfectly stable. Unfortunately, this is not the case in practice: with the values we used, in some rare cases, convergence to the steady state even showed noticeable oscillations of the ice limit. Fortunately, the first few iterations are sufficient to capture most glacial erosion, so such oscillations have no effect, in practice, on the eroded terrains we generate.

One important limitation remains, namely efficiency. Although we obtain, for the first time, a complete and tractable simulation of glacial erosion, our simulations take a few seconds per time-step. This makes them more difficult to control, since intuitive interactive brushes or handles cannot be provided to the user at runtime. Nevertheless, climate scenarios can be specified, as exemplified in our results.

5.6 Conclusion

In this chapter, we introduced the first method in Computer Graphics for the efficient simulation of glacial erosion. Our method is combined with secondary erosive phenomena, in particular fluvial, hillslope and cliff-based debris flow erosion. Our solution for glacial erosion allows for the simulation of the main characteristic glacial landforms, such as hanging and U-shaped valleys, cliffs and glacial lakes. It handles time spans covering several glacial and interglacial cycles.

The key idea is to link successive steady states of the glacier surface with large ice-erosion steps, using a specific scheduling of computations over the grid representing the terrain. Our method achieves the generation of large to medium-scale features in landscapes formed by glaciers.

Possible improvements include adding smaller-scale features, such as *moraines* (large flat areas formed by the transportation and deposition of rocks), *drumlins* (wave-like patterns formed by ice and water flowing over moraines), *eskers* (long and narrow winding ridges deposited by under-ice rivers during the retreat of glaciers), and glacial striations on steep slopes. To achieve these, our model would need to be combined with finer-scale physically-based or procedural models. For instance, the information on rock abrasion in our model could be used to parameterize the deposit of transported rocks into moraines. The sedimentation of debris flowing from cliffs into glacial lakes could also be modeled.

Although not tackling these glacial erosion features, lower-scale phenomena are considered in the next part. Chapter 6 introduces a novel framework for the interleaved simulation of erosive events and ecosystem simulation, and Chapter 7 proposes a method for the generation of snow cover. Both these applications could be linked with our glacier simulation: the mass balance could be deduced from the snow cover simulation, while transported rock debris created by glacial abrasion could be accurately instanced using the stochastic framework presented in Chapter 6.

Part II

Combining landscape simulation with medium scale phenomena

Chapter 6

Joint simulation of vegetation and erosion

Contents

6.1 Method overview **111**
6.1.1 Layered landscape model 111
6.1.2 Simulation 112
6.1.3 Control 114
6.2 Geomorphological events **115**
6.2.1 Rainfall and running water 115
6.2.2 Temperature 116
6.2.3 Lightning 117
6.2.4 Gravity 118
6.2.5 Fire 118
6.3 Ecosystem events **119**
6.4 Implementation **122**
6.5 Results and discussion **122**
6.6 Conclusion **132**

A major ingredient of a plausible virtual landscape lies in its coverage with meso-scale details, such as vegetation or rocks. This chapter presents a new simulation framework allowing for the interleaving of natural processes at various scales. In particular, we show how to combine erosional effects with ecosystem simulations to build a novel landscape authoring framework.

The first part of this thesis focused on the evolution of wide landscapes on large temporal extent. At the considered scale (time steps of several thousand of years and the grid resolution blurring details below 100 meters), landscape evolution was driven by large scale geological processes such as tectonic uplift and erosion. Smaller scale phenomena, such as the effects of vegetation or local rockslides were too punctual to be considered. Although unnecessary for the large scale simulation, the effects of these phenomena are of great importance when locally exploring a virtual landscape. It is always a challenging task for a designer to spread a large amount of plants and terrain details in a consistent way, and the human visual system is a very efficient discriminator of what is important or visually plausible.

Narrowing down the considered evolution scale to the time frame where smaller details are relevant does not completely hide the larger scale processes. On the contrary, landscapes exhibit huge spatial and temporal variance that is affected by multiple agents acting at different rates and scales. For instance pulses of hydraulic erosion, landslides, and lightnings can act very quickly and move or remove large amounts of plant and soil material within hours, while thermal erosion typically acts slowly and is expressed only over very long time spans. These different rates make it difficult to simulate the spatio-temporal evolution of landscapes, with classical simulation techniques. Geomorphological and ecosystemic phenomena also act differently depending on a plethora of factors such as location, slope, temperature, sun exposure, and soil type.

The key observation that drives this chapter is that the shape and temporal evolution of landscapes are significantly affected by the interaction between vegetation and erosion, a factor that has thus far not been considered in Computer Graphics. While vegetation may affect the speed of the geomorphological processes and reduce their effects such as landslides, many of them such as erosion, rock falls, and soil deposited by running water, also have a strong impact on the vegetation.

From this observation, we introduce a novel, unified framework for landscape modeling, that allows for interactive modeling of the mutual interaction between various geomorphological phenomena and their effects on terrain and vegetation. The input is an elevation map, which is modified, textured, and populated with consistent sets of details such as rocks, trees, or bushes by our simulation. The users retains full control over the terrain, vegetation, and their co-evolution during an interactive modeling session; they can tune the control parameters of a wide range of geomorphological phenomena, their interplay, and the weather. Users can also directly edit each layer state, such as the current plant cover, using brushes. As a result, the 3D layered model evolves and exhibits features that are difficult to achieve with other landscape modeling methods.

We claim the following technical contributions:

- An original framework for efficient, *stochastic simulation* of multiple phenomena and their interactions.

- A consistent set of models for the geomorphological and ecosystemic agents that simultaneously modify terrain and vegetation and influence their temporal evolution. In particular, we take into account the entire cycle of vegetation, from germination to death, and ultimate reduction to an organic mulch (humus), as well as providing a unified treatment of disturbance events, such as fire, lightning and landslides.
- Authoring tools to brush landscape materials and trigger natural phenomena, while navigating along the simulation time-line.

6.1 Method overview

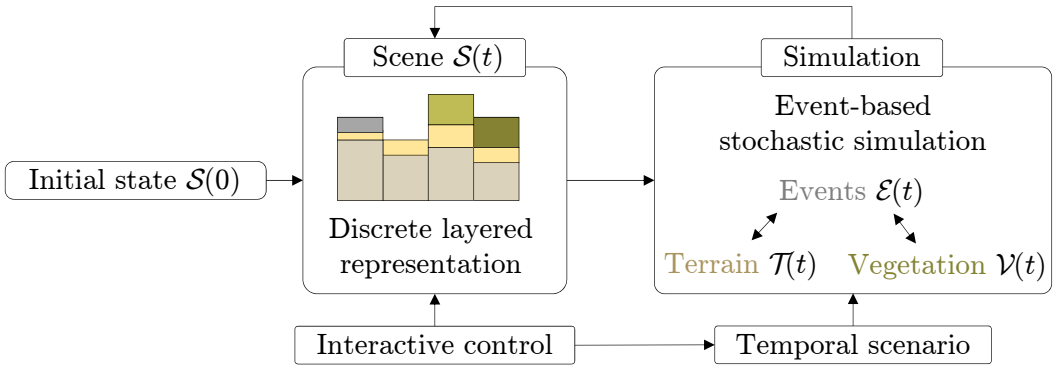


Figure 6.1: Framework: The layered scene $\mathcal{S}(t)$ is affected by the simulation, which includes an interplay of geomorphological and ecosystem events, vegetation and terrain.

Our method computes the temporal evolution of a terrain covered with vegetation under the combined action of various *geomorphological and ecological events*. Environmental effects such as rain, gravity, temperature, wind, fire, and lightning not only directly impact the evolution of the terrain and the development of vegetation, but also have an indirect impact, as vegetation and the upper terrain horizon (Grunwald 2016) mutually interact in a complex feedback loop. For example, vegetation can prevent rockslides but may equally be destroyed by falling rocks; vegetation absorbs water but flooding may also uproot plants.

Our framework (Figure 6.1) uses a discrete layered model that unifies the representation of different terrain materials and types of vegetation. Geomorphological and ecological events modify the data stored in the various layers during simulation. These layers collectively represent the state of the scene at each frame.

We use a discrete spatio-temporal simulation. At a given time step, the scene, denoted as $\mathcal{S}(t)$ is a set of 2D discrete layers composed of $n \times n$ cells. The simulation process computes the evolution of the scene $\mathcal{S}(t + \Delta t)$ from scene $\mathcal{S}(t)$ by stochastically applying a number of events to the cells of the terrain.

6.1.1 Layered landscape model

The layered model is an ordered sequence of layers from which a static landscape representation can be derived at any given time step. Alternatively, a series of simulation results can be

used to show the temporal evolution of the landscape over a chosen time span. Specifically, a discrete regular grid of size $n \times n$ cells is combined with a multi-layer ordered data-structure to represent different terrain materials and plant matter in every cell (Figure 6.2).

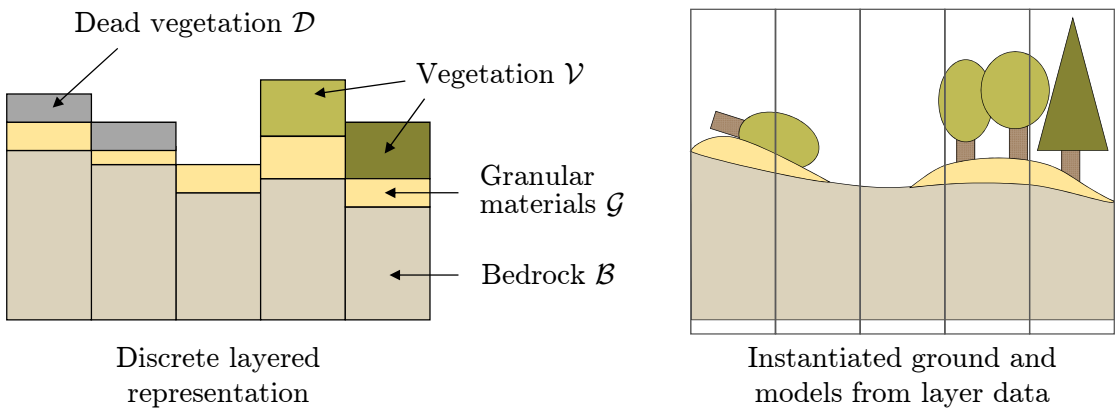


Figure 6.2: Representation of the ground and vegetation layers. Bedrock $\mathcal{B}$ and granular materials $\mathcal{G}$, consisting of rocks, sand, and humus, define the layers of terrain. Plants are represented using various vegetation layers $\mathcal{V}$, for grass, shrubs and trees. The layers store data used in simulations, such as canopy density and age for vegetation layers, and moisture for granular layers.

Terrain materials sit on top of a bedrock layer $\mathcal{B}$, which defines the base elevation. The broken rock, sand, and humus layers, denoted as $\mathcal{R}$, $\mathcal{K}$ and $\mathcal{H}$, represent the respective material thicknesses. In the remainder of the paper, we refer to these layers as *granular* material layers $\mathcal{G} = \{\mathcal{R}, \mathcal{K}, \mathcal{H}\}$. The ground elevation, denoted as $\mathcal{A}$, is defined as the sum of the bedrock and the granular material thicknesses: $\mathcal{A} = \mathcal{B} + \mathcal{G}$. The slope of the terrain between two points $\mathbf{p}$ and $\mathbf{q}$ will be defined $s(\mathbf{p}, \mathbf{q}) = (\mathcal{A}(\mathbf{p}) - \mathcal{A}(\mathbf{q}))/\|\mathbf{p} - \mathbf{q}\|$. In addition, cells are also characterized by values for soil moisture content $\mathcal{M}$, and the average daily duration of direct sun exposure $\mathcal{I}$, both of which are crucial to ecosystem simulation.

Vegetation layers store the density maps and other important parameters for every vegetation type. For trees and shrubs this includes the count, age, and height of plants in each grid cell, while for grass density alone is sufficient. We also model dead vegetation $\mathcal{D}$, which decays into humus and plays an important role in the ecosystem simulation. Total vegetation density $\mathcal{V}$ is given by the sum of the grass, shrub, and tree densities, weighted by their respective importance (Section 6.3).

6.1.2 Simulation

The novelty of our method lies in a stochastic simulation that supports robust and efficient integration of many phenomena acting on a landscape. In this section, we first describe the main challenges, and then explain our solution.

Challenges: An obvious approach to simulating multiple phenomena would be to jointly simulate the incremental actions of all agents (water flows, while thermal erosion fractures rock and plants grow), on each cell of the terrain, for each simulation frame.

Unfortunately, this strategy works against many of the data-structures and optimizations proposed in previous work, which presuppose a single agent acting in isolation. For instance, water-pile models accumulate water but often yield large water-level disparities between neighboring cells. Even with dampening, the disparity at intermediate stages can introduce significant instabilities when combined with chaotic events, such as rockfalls or plant growth.

This issue can be viewed from another perspective: a large, extensible simulation framework, with many parameters and unknown, and non-linear relationships, is most typically resolved using the forward Euler method. In this case, to achieve stability, the global time step must be chosen with respect to the most ill-conditioned equation in the simulation. This lockstep prevents efficient simulation of the other factors. Therefore, such an approach is limited to simple algorithms and data-structures or fine grids and small simulation steps.

Our solution: Instead of jointly simulating phenomena we successively generate a large number of individual events, which act separately on the landscape. Each event begins in a cell, typically follows a path through other cells (generally with some decay) and ends in bounded time. To generate the desired effect, both the sequence and specifics of individual events are stochastic. More precisely, in our formalism *events* obey the following rules:

- Propagate without backtracking to at most a single adjacent cell. An event may thus spawn a path as long as it does not branch, reverse course or form loops. It is important to note that this restriction only applies to propagation: an event may modify the layers of any number of neighboring cells along the way.
- Involve a bounded volume of material.
- Terminate in bounded simulation time δt, sufficiently short to be negligible with respect to the simulation time step $\delta t \ll \Delta t$.

The first rule limits execution time, the second promotes stability, and the third ensures convergence (by making it legal to perform such a decomposition).

This strategy is motivated by the seemingly stochastic nature of many events in real terrains, either because they are triggered at random (*e.g.,* rockslides) or their effects appear chaotic (*e.g.,* the spread of fire). Moreover, the path of such events is often determined by fine-scale features that are obfuscated by the discretization of a simulation grid. In practice, less artifacts are observed when sampling the next direction among a 8-neighborhood. Although a single event always propagates to a single neighboring cell, branching effects, such as rocks scattered during landslides, are achieved by the accumulation of many events in neighboring locations. The overall simulation can be described as follows:

- Choose a random position $\mathbf{p}_0$, with uniform probability, from among all terrain cells;
- Choose a random event, again with uniform probability, from all the available events;
- Activate the event at $\mathbf{p}_0$ and simulate it until it terminates;

- Store the effects (transported material, plant growth) in the relevant terrain layers.

This process is repeated a very large number of times N for each simulation step, which ensures a convergence to plausible results. In our implementation, N is a product of the number of grid cells and different event types ($N > 256 \times 256 \times 10 \simeq 650,000$ in all our examples), so that all terrain cells are likely starting points for at least a few events.

The time step of our simulation Δt is set to one year. This represents a balance between the relatively rapid development of vegetation and longer-term impact of erosion, while allowing efficient simulations that span several centuries. In practice, Δt can be reduced over the last few simulation months before exporting final geometry so as to account for seasonal variations in vegetation.

Initialization of the landscape model $\mathcal{S}(0)$ is controlled by the user, who can either provide data for the different layers or rely on procedural instantiation from a single input heightfield. In the latter case, the bedrock layer $\mathcal{B}(0)$ is initialized using elevation values from the heightfield, while sand $\mathcal{K}(0)$, rocks $\mathcal{R}(0)$ and vegetation $\mathcal{V}(0)$ are set to zero. The thickness of the humus layer $\mathcal{H}(0)$ is determined by the local slope of the bedrock: $\mathcal{H}(\mathbf{p}, 0) = g(\nabla\mathcal{B}(\mathbf{p}, 0))$ where ∇ is the gradient operator. The function $g : \mathbb{R}_+ \to \mathbb{R}_+$ is a smoothly decreasing function of the slope that maps $[0, +\infty]$ onto $[0, 1]$ with $g(\tan 30°) = 1/2$ so that there should be half as much humus on 30° slopes as on flat areas. In our implementation, we use a Gaussian $g(s) = e^{-s^2/\sigma^2}$ with $\sigma^2 = 3 \ln 2$. Moisture $\mathcal{M}$ in the granular material layers $\mathcal{G}$ is computed during rainfall simulation and need not be pre-calculated.

6.1.3 Control

The user is afforded pause-and-edit control over the simulation at several levels of abstraction, with changes stored in the timeline:

- *Environmental parameters* of the scene (rainfall patterns, frequency of lightning strikes, temperature, *etc.*) can be directly and interactively adjusted at any point in the simulation timeline.
- *Layered data* can be locally edited using interactive brushes, by adding or removing plants, or adjusting the thickness of materials, at any given time step. User edits may introduce inconsistencies, such as too much sand or too many trees on steep slopes, but these usually self-correct after a few simulation time steps.
- *Probability maps* allow the user to locally re-weight the chance of any chosen geomorphological events (such as enforcing a lightning strike in a given cell).
- *Environment scenarios* provide functionality to pre-define a script for any of the above interactions (environmental parameters, layer changes, and probability maps) as they evolve over time.

At each simulation step, the user is provided with a real-time visualization of the current state, enabling stop-and-restart editing of the simulation.

A common requirement of artists is the ability to lock portions of a landscape while leaving simulation unchecked elsewhere. In general, it is not possible to bound the propagation of events due to their randomness, but we can achieve a similar effect using a mask tool that

linearly interpolates between the content in frame n and frame $n-1$ as weighted by the mask, which allows content from a given frame to be carried forward unaltered.

6.2 Geomorphological events

In this section, we show how five significant and influential natural phenomena – rainfall, temperature, lightning, gravity and fire – have been incorporated into our unified framework.

6.2.1 Rainfall and running water

Rain, and the resulting running water, impact not only terrain shape (by hydraulic erosion) and soil composition (through material transport), but also vegetation growth. As a typical effect, water carves channels in the surface of the terrain and forms accretion cones (Figure 6.4). There are several factors to consider: the trajectory of runoff is heavily dependent on slope, drainage area and other features; the quantity of runoff is reduced by evapotranspiration off vegetation; and, finally plant roots bind soil and dampen hydraulic erosion.

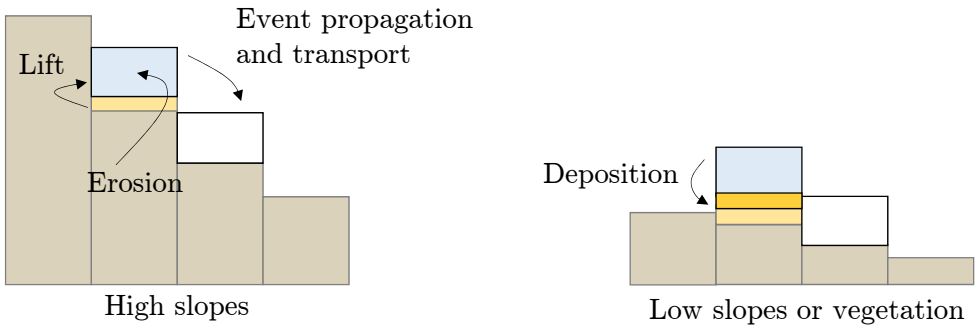


Figure 6.3: Runoff events use water as a vector to transport material. They perform erosion, lift, and deposition. Water is also partly absorbed and generates soil moisture.

We simulate all these effects through *runoff events*, which model the progression of surface water: its stochastic course across the terrain, its reduction due to soil absorption, and its ultimate impact in terms of erosion and material transport.

Runoff event simulation: A runoff event is parameterized by the volume of water w it transports. The event starts at a point $\mathbf{p}_0$ with $w = w_0$, follows a trajectory, and ends when $w = 0$ or a local terrain minimum is reached. Initially, w_0 is set to the total quantity of rain in a cell during a simulation time step Δt, and reduced by the proportion intercepted by plants and released to the atmosphere through evaporation, which depends on plant density.

The event is then simulated iteratively over the terrain grid as follows. Let $\mathbf{p}_k$ be the current position of the runoff. The subsequent position $\mathbf{p}_{k+1}$ is determined by a random choice from among the neighboring cells $\mathcal{N}(\mathbf{p}_k)$ having lower elevation than $\mathbf{p}_k$, with probability:

$$\rho(\mathbf{p}_{k+1}) = s(\mathbf{p}_k, \mathbf{p}_{k+1}) / \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p}_k)} s(\mathbf{p}_k, \mathbf{q}). \tag{6.1}$$

Once this random choice is made, the current slope between $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ is stored, or set to zero if $\mathbf{p}_k$ has no lower neighbor.

The runoff w_k is then reduced by soil absorption, which is inversely proportional to the slope. Absorbed water is added to the local moisture in cell $\mathbf{p}_k$, stored in layer $\mathcal{M}$. If this moisture exceeds the maximum holding capacity of the local soil layers (given preset parameters for bedrock, rock, sand, and humus), then the excess is returned to w_k because of soil saturation.

Interactions with the landscape: Inspired by the geology literature (Braun et al. 1997), we account for both hydraulic erosion, which grinds terrain material (bedrock and rocks) to a finer constituency (rocks and sand), and soil transportation, which involves water redistributing small quantities of rock, sand and humus.

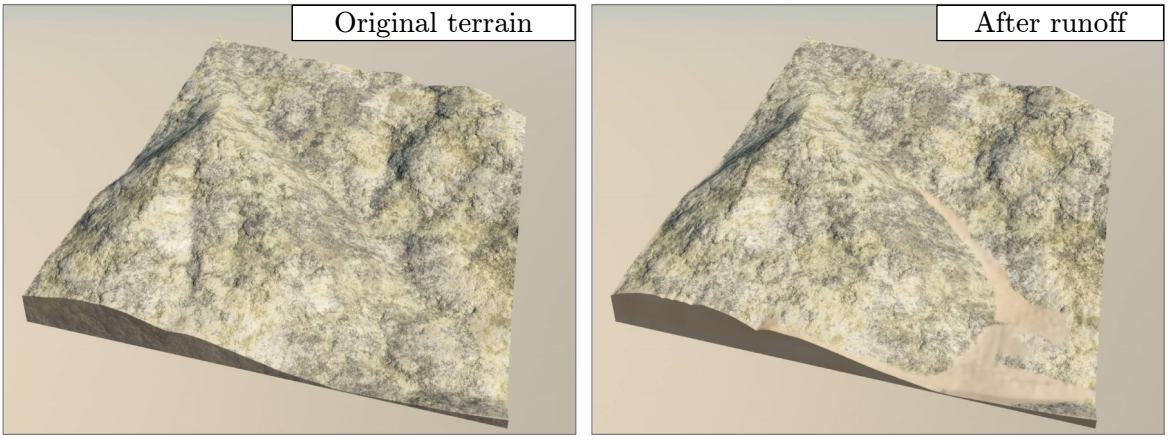


Figure 6.4: Results after several runoff events: channels form in the upper part, while accretion cones develop over those already present in the original terrain. One event has been initiated per cell and per year for 200 years.

We use a standard discrete model for erosion (Musgrave et al. 1989), where redistribution acts by lifting or depositing material, depending on a slope threshold (Figure 6.3). Lift only occurs until carrying capacity is reached, and deposition is inversely proportional to slope. Moreover, bedrock is partially shielded from erosion by intervening soil and vegetation layers. Finally, lift is impeded and deposition enhanced by the presence of vegetation, by virtually reducing the slope according to the amount of vegetation, for the purposes of redistribution.

Once the runoff sequence terminates we approximate the effects of plant transpiration and seepage into groundwater by reducing the moisture at the source $\mathbf{p}_0$ by a constant amount.

6.2.2 Temperature

Temperature variation plays an important role in triggering the fracture of bedrock – referred to as *thermal erosion* (Musgrave et al. 1989) – as well as in determining the niche suitability of vegetation. Thermal erosion occurs when water in rock cavities freezes, expands and breaks the bedrock into rocks that may then fall under gravity. Thermal erosion is most influential in regions with high temperature gradients, such as cliffs that are exposed to direct sunlight

and cold night-sky radiative transfer. Sand, humus and vegetation act to reduce the impact of temperature variations by shielding the bedrock.

In our framework, when a *thermal stress event* is launched at a random position $\mathbf{p}$, we first estimate the variation between daytime and nighttime temperatures denoted as ΔT using the elevation $\mathcal{A}(\mathbf{p})$ and illumination $\mathcal{I}(\mathbf{p})$. This ΔT value is then damped in proportion to the local density of vegetation $\mathcal{V}(\mathbf{p})$ and thickness of sand and humus $\mathcal{G}(\mathbf{p})$. The resulting value is used as the probability that a given quantity of local bedrock $\mathcal{B}$ will fracture into rocks $\mathcal{R}$:

$$f(\mathbf{p}) = k\Delta T\, s(\mathbf{p})/(1 + k_{\mathcal{G}}\mathcal{G}(\mathbf{p}) + k_{\mathcal{V}}\mathcal{V}(\mathbf{p})).$$

The coefficients k, $k_{\mathcal{G}}$ and $k_{\mathcal{V}}$ are constants and $s(\mathbf{p})$ is the steepest slope when considering the 8 direct neighbors. Since bedrock is often approximated as a granular material over large time scales (Densmore et al. 1998), we assume that the landslide effect is limited by a critical slope. The quantity of falling rock is computed from the difference between the local slope and this critical slope. By design, rocks are created in-place, and their fall is triggered and simulated according to gravity events (Section 6.2.4). The effect of temperature on local vegetation will be discussed further in Section 6.3.

6.2.3 Lightning

While it is well-known that lightning destroy trees, recent research in geomorphology (Knight et al. 2014) has demonstrated that bedrock struck by lightning is blasted into rocky material. A single strike may break down tons of bedrock and eject rocks up to several meters from the point of impact, resulting in a volume of up to $10,000m^3$ being moved per square kilometer per 100 years.

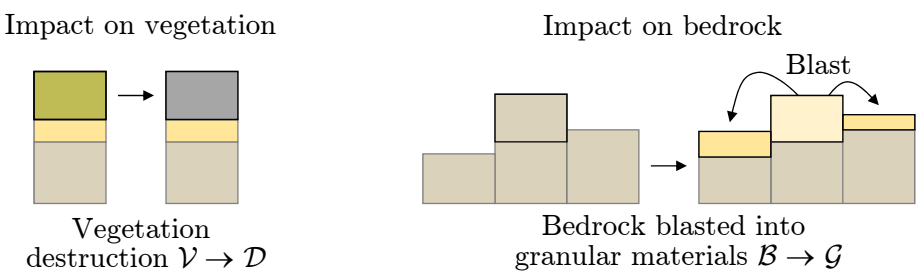


Figure 6.5: Effect of a lightning strike: vegetation is destroyed and part of the bedrock layer disintegrates into rock material, which is spread to neighboring cells.

We model this phenomenon through *lightning strike events*. At a random point of impact $\mathbf{p}_0$, the probability of damage is a function of elevation and the exposed character of the location, which we evaluate using local curvature of the terrain elevation $\mathcal{A}$:

$$l(\mathbf{p}) = k_{\mathcal{L}} \min\left(1, e^{k_{l_c}(\nabla\,\mathcal{A}(\mathbf{p}) - k_{l_s})}\right).$$

The coefficient $k_{\mathcal{L}}$ is the maximum probability that a lightning strike hits the cell at position $\mathbf{p}$, k_{l_s} is the minimum curvature for which this probability is achieved, and k_{l_c} is a scaling factor.

If the lightning strike occurs, we proceed as follows (Figure 6.5). Lightning destroys local vegetation, if present, and we reduce the number, and aggregate height and age of trees in the vegetation layer $\mathcal{V}$ and increase the density of dead vegetation $\mathcal{D}$ accordingly. Furthermore, in conducive conditions of high global temperature and low rainfall lightning has a chance of initiating a *fire event* at the same location (Section 6.2.5). Otherwise, a constant amount of bedrock material $\mathcal{B}$ is removed and spread as granular material $\mathcal{G}$ (rocks and sand) in the neighboring cells, while taking their relative elevation into account.

6.2.4 Gravity

We use separate events (*rock-slide*, *sand-slide* and *humus-slide*) to represent the collapse under gravity of different granular materials, since each has a specific friction angle. As with runoff, a slide event starts at a random position $\mathbf{p}_0$, propagates along a random slope-dependent trajectory (see Equation (6.1)), and terminates when the local slope falls below the friction angle. At each step, the amount of sliding material is computed as a random proportion of the material column that sits above the friction angle. We also add a contribution proportional to the curvature of the surface, to simulate a form of diffusion known as the hill-slope process on granular material (Braun et al. 1997).

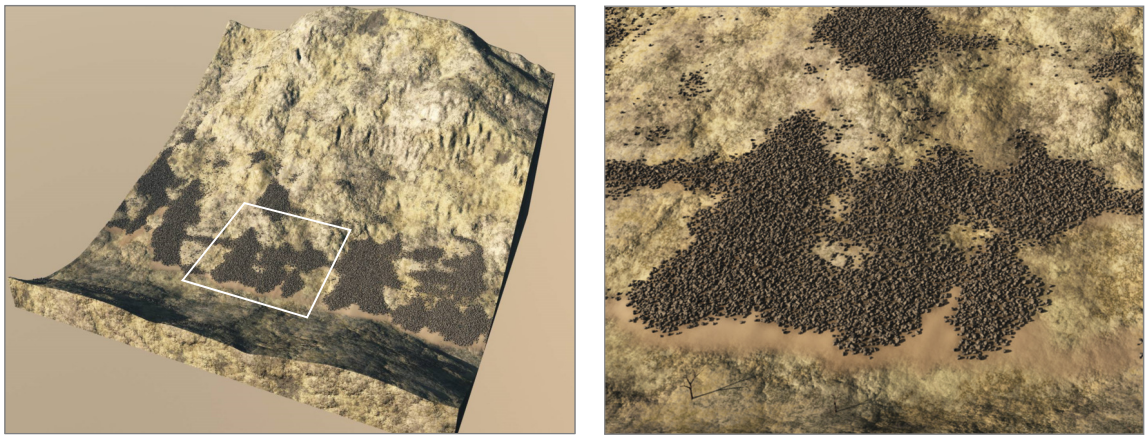


Figure 6.6: The effects of thermal stress and material slides. Bedrock fractures into rocks and sand, which slide under gravity. Here, rock is dark gray and sand is yellow. Note that the friction angle for sand is set lower than for rock.

Material slide effects are dampened by the presence of vegetation since the roots of trees form a stabilizing lattice on slopes. This is captured by a vegetation-dependent increase in the friction angle. Conversely, rockslides also damage vegetation, which we model by decreasing plants in the vegetation layers $\mathcal{V}$ and increasing the dead vegetation $\mathcal{D}$, accordingly: vegetation is destroyed in proportion to the volume of falling rocks.

6.2.5 Fire

Fire as a disturbance event is one of the primary causes of deforestation in hot and dry ecosystems. In our system, it also serves as a useful control tool for users.

Figure 6.7: The fire event start by seeding (left), where high slopes, dry areas under wind direction receive more fire intensity. Then, the vegetation is destroyed, and fire propagates to one of the seed (right).

We note that fires spread more strongly upwards and in the prevailing wind direction (Yassemi et al. 2008). Accordingly, a *fire event* at $\mathbf{p}_0$ starts with a seeding process igniting fires in neighboring cells, whose intensity is a function of local temperature, moisture, and vegetation density. The number, distance and direction of nearby cells damaged by the fire depends on this intensity and the prevailing wind (a constant user-defined 2D vector in our implementation). In particular, the fire spread direction follows a normal distribution centered on the wind direction, with variance related to the inverse of fire intensity, narrowing when the wind is strong. Next, vegetation is destroyed at the seeded locations in proportion to fire intensity and local slope. The cycle then begins again with one of the seeded locations chosen as the potential source of a new fire event. A fire dies out when there is no more vegetation to fuel it or if the fire intensity has dropped below the level required to ignite new trees.

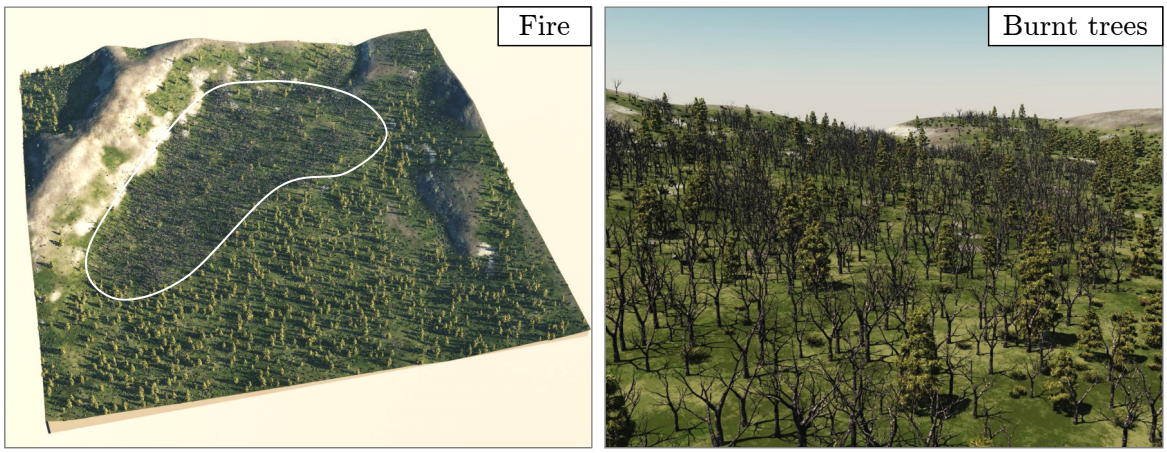


Figure 6.8: A fire set by the user and advected by a north wind, with subsequent regrowth over a few years.

6.3 Ecosystem events

In simulating vegetation, our point of departure from previous CG ecosystems is a consideration of the cyclic interaction between plants and soil. Plants rely on soil moisture but also impact it in various ways. For instance, due to evapotranspiration some rainfall is intercepted

by tree canopies and by litterfall (the detritus of fallen leaves and dead plants) and evaporates before reaching the ground, while water is also drawn up from the soil and transpires through leaves. Conversely, litterfall decays into humus, which changes the constituency of soil and improves its moisture holding capacity. We explicitly account for these effects, as well as the impact of vegetation on erosion (see Section 6.2.1).

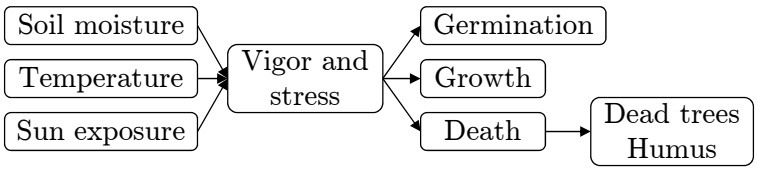


Figure 6.9: Ecological events use monthly soil moisture, temperature, and sunlight exposure to derive yearly stress and vigor values, which drive plant germination, growth, and death (contributing to the dead trees and humus layers).

Most ecosystem simulations in Computer Graphics (Benes et al. 2003; Bradbury et al. 2015; Ch'Ng 2013; Deussen et al. 1998; Lane et al. 2002) model individual plant specimens using a circular footprint to determine competition for resources. While this is viable for small-scale simulations (up to 1km^2) it becomes computationally costly at larger scales due to the correlation between terrain area and numbers of plants. Instead, we adopt a cell-based Eulerian approach, as favored by botanists and ecologists in their Dynamic Global Vegetation Models (DGVM) (Foley et al. 1996; Sato et al. 2007; Sitch et al. 2003). While plants are treated in aggregate it is still possible to incorporate competition for sunlight and soil moisture.

Another feature borrowed from the DGVM literature is our grouping of species into Plant Functional Types, representing plants with similar response to environmental conditions. A broad separation between tree, shrub and grass layers is sufficient in our case but the architecture supports finer categories, should differentiation between, for instance, evergreen and deciduous or needle-leaved and broad-leaved plants be required for a particular application.

As with other events, *ecosystem events* are generated at random in a given cell. Each of them accumulates growth, death and germination for a particular plant layer (Figure 6.9). This is based on the aggregation of a monthly suitability function that considers local temperature, soil moisture and sun exposure to arrive at a value for plant viability $V \in [-1, 1]$, for all types of plants (trees, shrubs, and grass, in our implementation). Negative values represent stresses to the ecosystem in response to extreme conditions, such as drought and frost, while positive values indicate proportional opportunities for growth and germination.

The response of plants is modeled using a piece-wise linear hat-like function $v(c)$ that captures the influence of a given climatic condition c (Gain et al. 2017):

$$v(c) = \begin{cases} -1 & \text{if } c < E_{\min} \text{or} c > E_{\max} \\ (c - E_{\min})/(I_{\min} - E_{\min}) & \text{if } E_{\min} \leq c < I_{\min} \\ 1 & \text{if } I_{\min} \leq c \leq I_{\max} \\ (c - I_{\max})/(E_{\max} - I_{\max}) & \text{if } I_{\max} < c \leq E_{\max} \end{cases},$$

where $E_{\min}, E_{\max}$ are the bioclimatic limits outside of which a particular plant type cannot survive (Sitch et al. 2003), $[I_{\min}, I_{\max}]$ is the ideal range for a plant, and c is a monthly

bioclimatic value for a given cell. We use a separate such function $v_i, i = \{$temperature, moisture, sunlight$\}$ for each combination of plant type and bioclimatic condition. Ultimately, $V = min_i(v_i)$, since viability is constrained by the weakest resource. For instance, a plant cannot flourish if it is over- or under-watered even if temperature and sunlight conditions are ideal, a principle known in Ecology as Leibig's law of the minimum (Cade et al. 1999).

The monthly bioclimatic inputs (c_i) are calculated as follows: soil moisture is derived by distributing a cell's yearly moisture value (layer $\mathcal{M}$) in proportion to monthly rainfall patterns; temperature is provided by the user as an average monthly value ($\bar{t}_i$), which is reduced according to cell altitude at a lapse rate of 6.5 °C per 1000 m: $\theta_i(\mathbf{p}) = \bar{\theta}_i - 0.0065\,\mathcal{G}(\mathbf{p})$; sun exposure $\mathcal{I}$ is calculated, based on latitude and compass direction, by intersecting ray's from the sun's position along its trajectory with the terrain. This captures terrain self-shadowing and provides average daily hours of direct sunlight per month for a cell.

Each plant layer in the cell, with the exception of grass, is encoded by the number of plants (n), sum of plant heights (h) and sum of plant ages (a). This allows an average plant specimen to be derived and hence an estimate of the density of plant coverage in a cell to be computed, as:

$$\mathcal{V} = n\pi(r \cdot h/n)^2/w^2,$$

where w is the width of a cell and r is the ratio between a plant's canopy radius and height.

In addition to its effect on the various geomorphological events in Section 6.2, the density of plants is useful to the ecosystem simulation itself in two respects. First, it provides a proxy for competitive pressure within plants of the same functional type. For instance, if $d > 1$ there is more than complete canopy coverage in a cell and self-thinning is mandated due to plants encroaching on each other. Second, it can be used to account for the shading of subordinate plants, such as shrubs shaded by trees, by reducing sun exposure in proportion to density.

Since the expected interval between vegetation events in a given cell is one year (based on Δt), we use the monthly viability to derive yearly values for vigor and stress and feed these into simple growth, germination and death processes. Vigor is the average viability during the growing season (when the average monthly temperature is at least 5 °C), while stress is the average of the four worst viability values, but is only considered if it is negative.

For seeding and germination, the event framework could support propagation to neighboring cells, in a similar fashion to fractured rock ejected by a lightning strike (Section 6.2.3). However, propagation processes are complicated by different forms of seed dispersal and variable delays in germination. For instance, in the case of obligate seeding forest fires are needed to spur germination. Instead, we make the simplifying assumption that an existing bank of seeds is present in the soil and model germination by adding a number of plants as seedlings to the cell in proportion to the vigor and available space $(1 - \mathcal{V})$, but only if there is no stress. For woody plant types we use an establishment rate of 0.24 saplings per m^2 (Prentice et al. 1993). For growth, we scale a constant yearly increase in plant height for each plant according to the vigor value. In the last process, plant death acts by removing average plant specimens from a cell into a litter pool according to self thinning, environmental stress and senescence rules. First, to avoid over-saturation plants are removed to enforce $d < 1$. Next, bioclimatic limits are accounted for by removing plants in proportion to the stress value. Finally, if the average age is above a threshold a small random proportion of plants are removed to represent death by old age.

It makes little sense to consider grass as individual plants, even in aggregate. So, instead we translate grass vigor directly into a density value and ignore issues of individual grass clump growth and death.

The final phase in an ecosystem event is to model the breakdown of dead plant matter $\mathcal{D}$ into humus $\mathcal{H}$. Higgins et al. 2007 provide a conversion from tree height to biomass, which when combined with an average green wood weight (m in kg / m^3) allows a derivation of bio-volume (b) from average plant height ($\bar{h}$), as: $b = 0.52\,\bar{h}^{\,2.55}/m$. Roughly 30% of litter fall is converted to humus in a given year (Sitch et al. 2003), while the remainder rots away and is released as carbon dioxide.

6.4 Implementation

Our system has been implemented in C++ and uses OpenGL for rapid previsualization and *Vue* for rendering photorealistic landscape images. All simulations were performed on a desktop computer equipped with an Intel Core i7 CPU clocked at 2.5 GHz. We did not use any graphics hardware acceleration.

We use elevation maps from real terrains downloaded from the (*U.S. Geological Survey*). The output can be either a single, static landscape, or several frames representing its evolution over time. In both cases we use our layered model to enhance the landscape with procedural detail, as follows: we provide a terrain heightfield, obtained by stacking the sand $\mathcal{K}$ and humus $\mathcal{H}$ layers on top of the bedrock elevation $\mathcal{B}$, and a surface texture, computed from the thickness of sand and humus and the density of grass $\mathcal{V}_g$. Plant geometry is instantiated from models stored in an atlas, in accordance with the vegetation layers $\mathcal{V}$, which define local plant data such as density, size, and age, for every cell. After scene geometry has been generated, it is exported to *Vue* for final rendering. The final rendering smooths the layers, and bedrock, sand, humus, and grass are rendered in order from the topmost layer to the bottommost. All geometric elements (rocks, shrubs, and trees, both living and dead) are instantiated.

To explore the possibility of acceleration, we have implemented a multi-threaded CPU version of our framework, where each thread handles a set of events. We solve race conditions, where two events converge on the same cell, using atomic instructions. Several cells can be locked iteratively during the propagation of an event, but, to avoid deadlock, the same thread cannot lock more than one cell at the same time. With this approach we achieve a speed-up factor of 4 on a 12-thread machine.

6.5 Results and discussion

The size of a cell in all our simulations is set to $10 \times 10\,\mathrm{m}^2$. This allows the capture of medium-scale erosion and ecosystem features, while allowing the simulation and authoring of landscapes up to $10 \times 10\,\mathrm{km}^2$ in extent at interactive rates.

Average timings for different types of events and for a complete scene are reported in Table 6.1. Performance is related to the scene resolution $n \times n$, the number of events and their overall complexity. Certain events have only a local extent, such as lightning strikes and ecosystem events. In contrast, events such as gravity (Section 6.2.4) and rain (Section 6.2.1), with hydraulic erosion, may propagate changes in the layered data-structure across many cells.

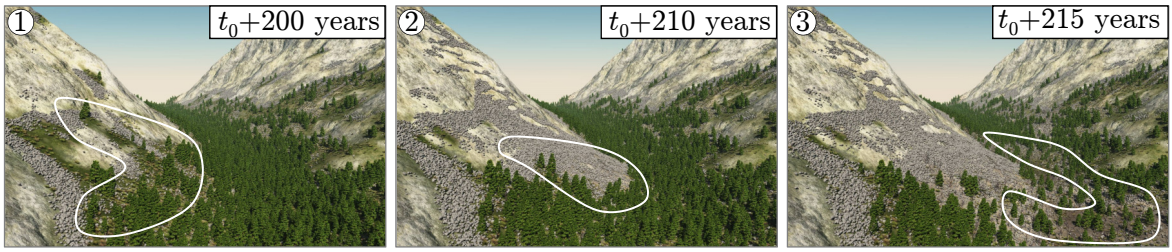


Figure 6.10: Our framework combines layered terrain and vegetation data and supports their interlinked simulation, which can be driven by users editing layers or triggering natural events. (1) The user first provides a bare-earth digital elevation map for time step t_0 and our framework simulates interleaved erosion and plant growth, up to $t_0 + 215$ years. (2) In the next time step at $t_0 + 210$ years, a landslide creates boulders that destroy vegetation. One year later, the designer triggers a fire in the valley, which spreads to consume part of the forest. (3) After four more years at $t_0 + 215$, the remaining trees have continued growing, new saplings have germinated and the humus layer is beginning to regenerate. The white loops indicate affected areas.

Event type	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
Rain	0.0695	0.43	3.92	36.5
Gravity	0.0067	0.027	0.08	0.43
Temperature	0.0016	0.0067	0.027	0.12
Lightning	0.0007	0.0027	0.01	0.05
Ecosystem	0.027	0.085	0.36	0.94
Total time	0.11	0.59	4.77	38

Table 6.1: Average performance (in s) for different events over the course of a simulation step Δt, and the total, as a function of terrain width n (in number of cells).

The worst case complexity is $O(n^4)$, where n is the terrain width, because the number of events per simulation step is proportional to n^2, and of these the most computationally demanding events trace a path with no cycles, spawning an upper bound of n^2 cells. In practice, we have found that execution is dominated by the runoff event, which exhibits average complexity of $O(n)$ for every cell and represents 75% of the computation for a $2.5 \times 2.5\,\mathrm{km}^2$ landscape. The overall average complexity varies from $O(n^2)$ for terrains with limited runoff to $O(n^3)$ for terrains with long channels.

User control: A variety of control mechanisms and achievable landscapes are showcased in Figures 6.10, 6.11, and 6.12. The first result in Figure 6.10 shows a landscape designed by triggering specific landslide and fire events, followed by periods of natural landscape evolution. Figure 6.11 demonstrates varied outcomes obtained simply by changing the initial 256×256 bedrock heightfield.

An example of interactive control during a 15 minute editing session is provided in Figure 6.12. Here, the user has modeled drought and sand-dune growth over a 128×128 subregion of an oceanic landscape.

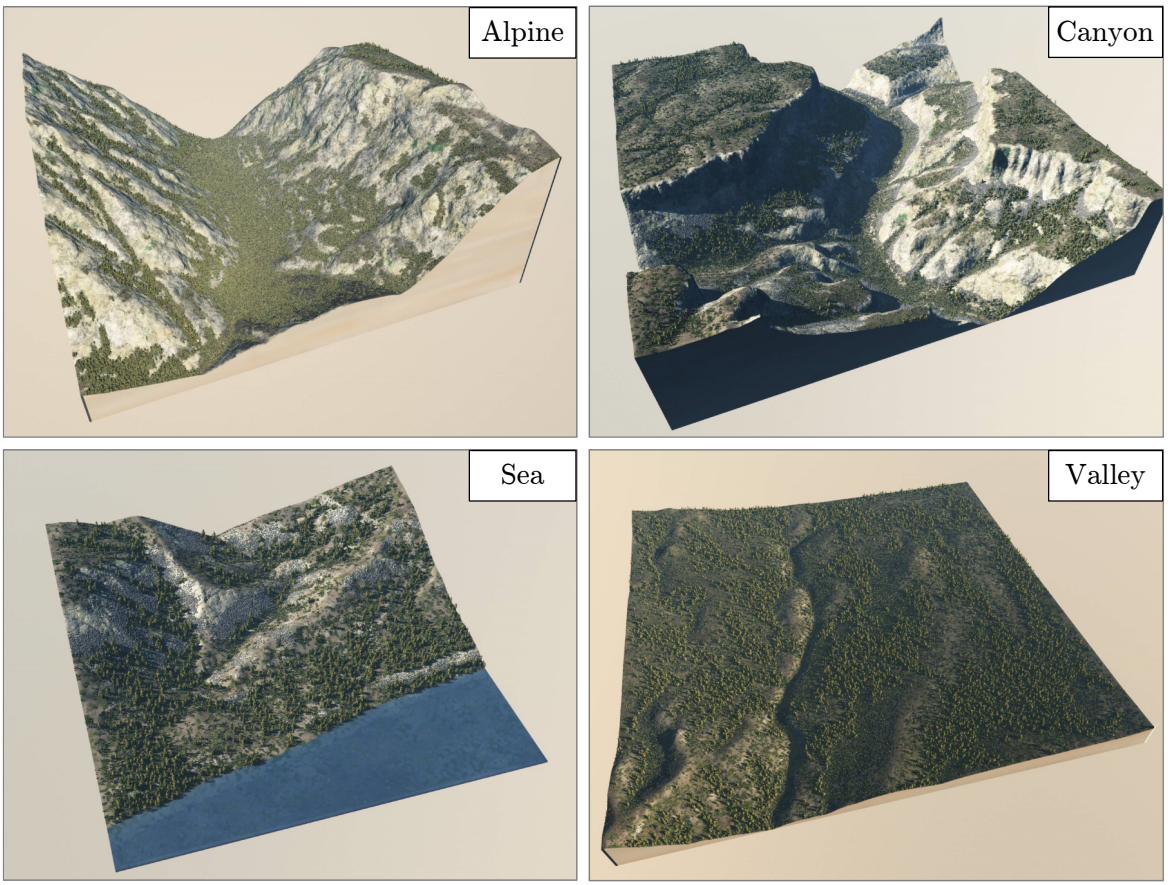


Figure 6.11: Results from different initial heightfield inputs produced using our simulation after 300 years of evolution. From left to right: an Alpine landscape from the U.S. Rockies and a portion of the Grand Canyon (top), a Mediterranean landscape and a forested valley (bottom).

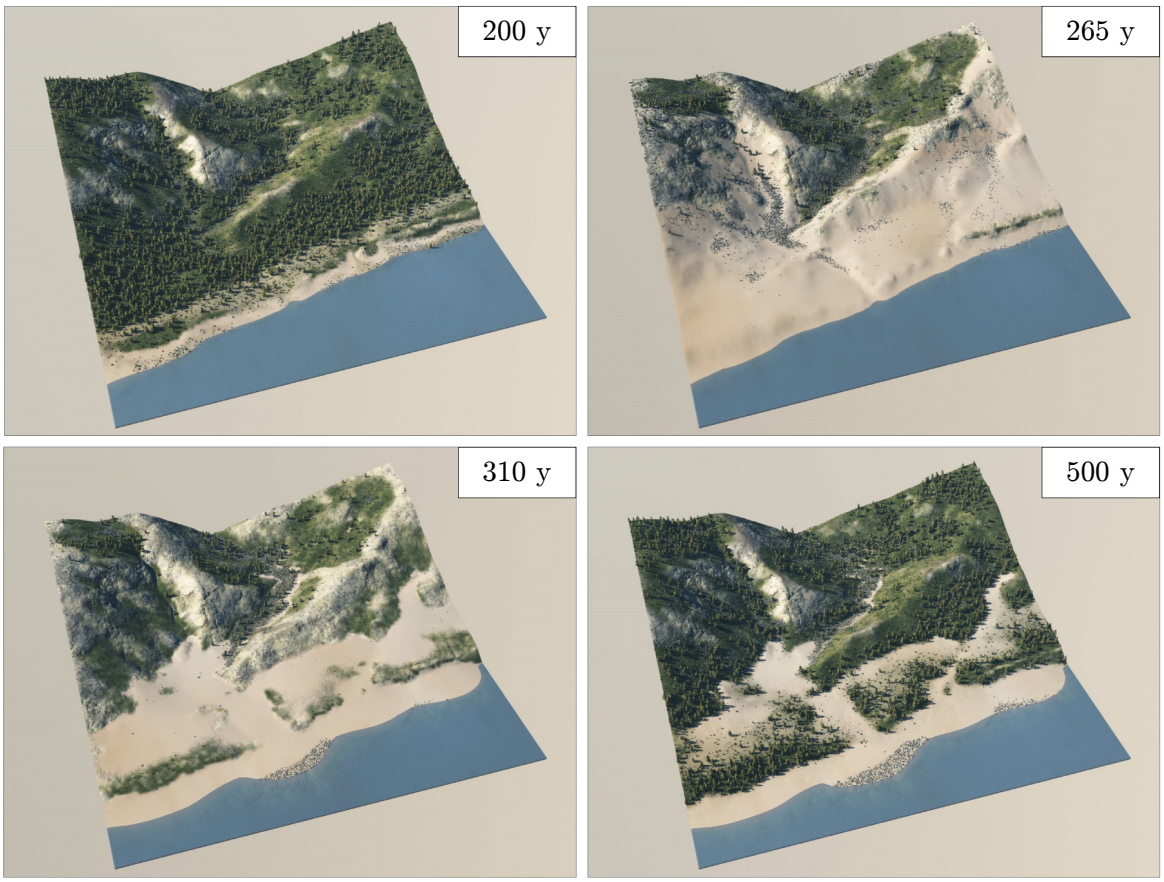


Figure 6.12: Evolution of the simulation in response to user edits. Initial state: the user paints sand along the beachfront and humus elsewhere. After vegetation growth at 200y the user layers sand across the lower half of the terrain, reduces precipitation, and destroys vegetation with fire. At 265y he increases rainfall. Finally, at 310y he adds humus to promote forest regrowth and waits until 500y.

Generic framework: The logical subdivision into independent tasks makes our framework flexible. It simplifies the main simulation loop (Section 6.1), making it both scalable and extensible: it is easy to add events to the simulation because each event is a function that takes as arguments a starting position, the terrain layers, and a few other simulation parameters, such as the main wind direction and strength. The relative importance of each event type can be easily adjusted to obtain a wide variety of landscapes (Figure 6.13).

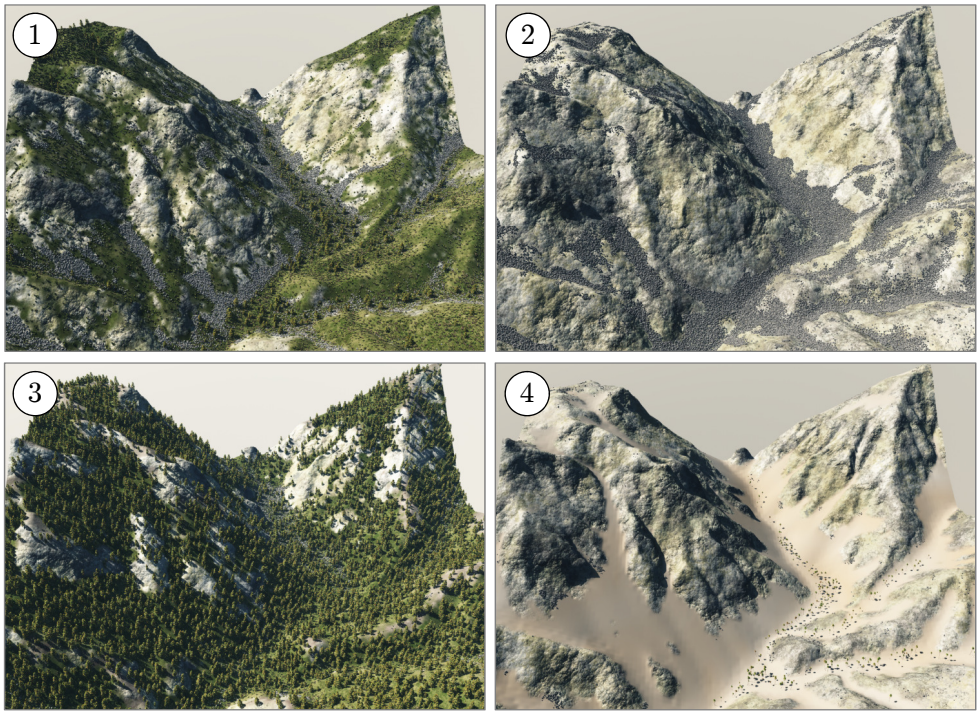


Figure 6.13: A terrain used to generate a variety of landscapes: (1) with default parameters, (2) a high altitude rocky region, (3) a dense forest on a fertile ground, and (4) a sand-filled desert with drought-adapted vegetation at the lower elevations.

Simulation quality: Our framework enhances terrain erosion by interlinking multiple geomorphological phenomena, including hydraulic erosion, material transport, shifts and slides due to gravity, and bedrock fracture from thermal erosion and lightning.

The results of erosion of a typical simulation (100–1,000 years) are subtle but distinctive; the final variation of height is on the order of a few meters. The visual consequences are mainly:

- Bedrock destruction evidenced by small channels and fractured cliff-faces (Figure 6.13.(2)).
- Correct buildup of materials (fallen rocks, sand, and humus) in both screes and accretion areas (Figure 6.14.(2)).
- That vegetation suppresses erosion and therefore softens slopes. This becomes visually salient when slopes are denuded by fire (Figure 6.15). However, the effects of vegetation on erosion are subtle in most scenes.

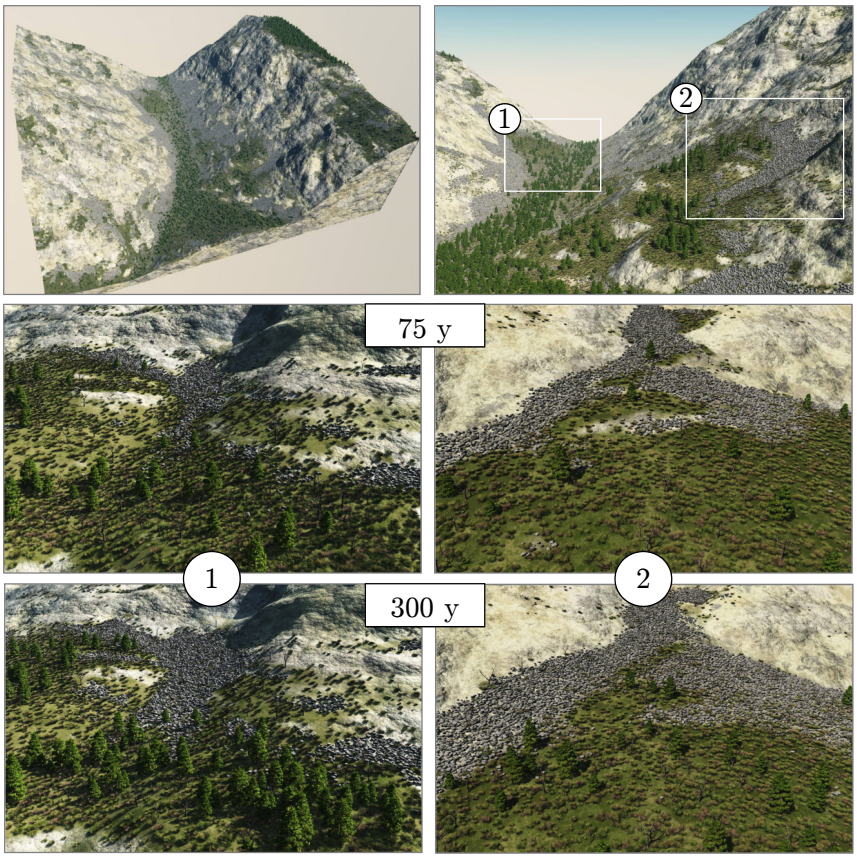


Figure 6.14: An example of combined erosion and ecosystem simulation. Images show in detail: (1) vegetation destroyed by falling rock and (2) plant growth on accretion zones, after 75 years and 150 years.

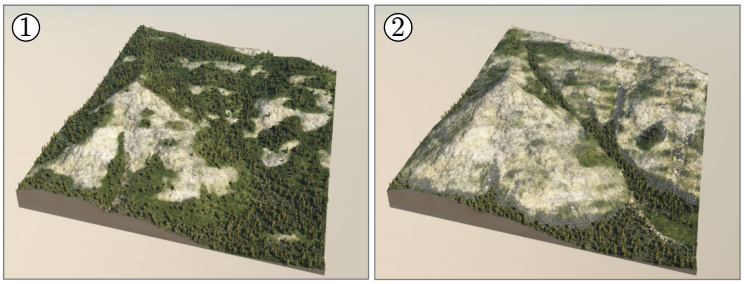


Figure 6.15: Impact of vegetation on soil erosion: vegetation protects the soil, damps erosion, and blocks rockfalls. (1) For 150 years the landscape has vegetation growth without erosion. (2) Conversely, over the same time span erosion affects the second terrain but without any vegetation. Finally, vegetation and erosion are combined for a further 150 years in both landscapes. The visual impact of erosion is highlighted in (2) by the sparser vegetation: the erosion removed soil, carved steep slopes and triggered rock falls that prevented vegetation growth.

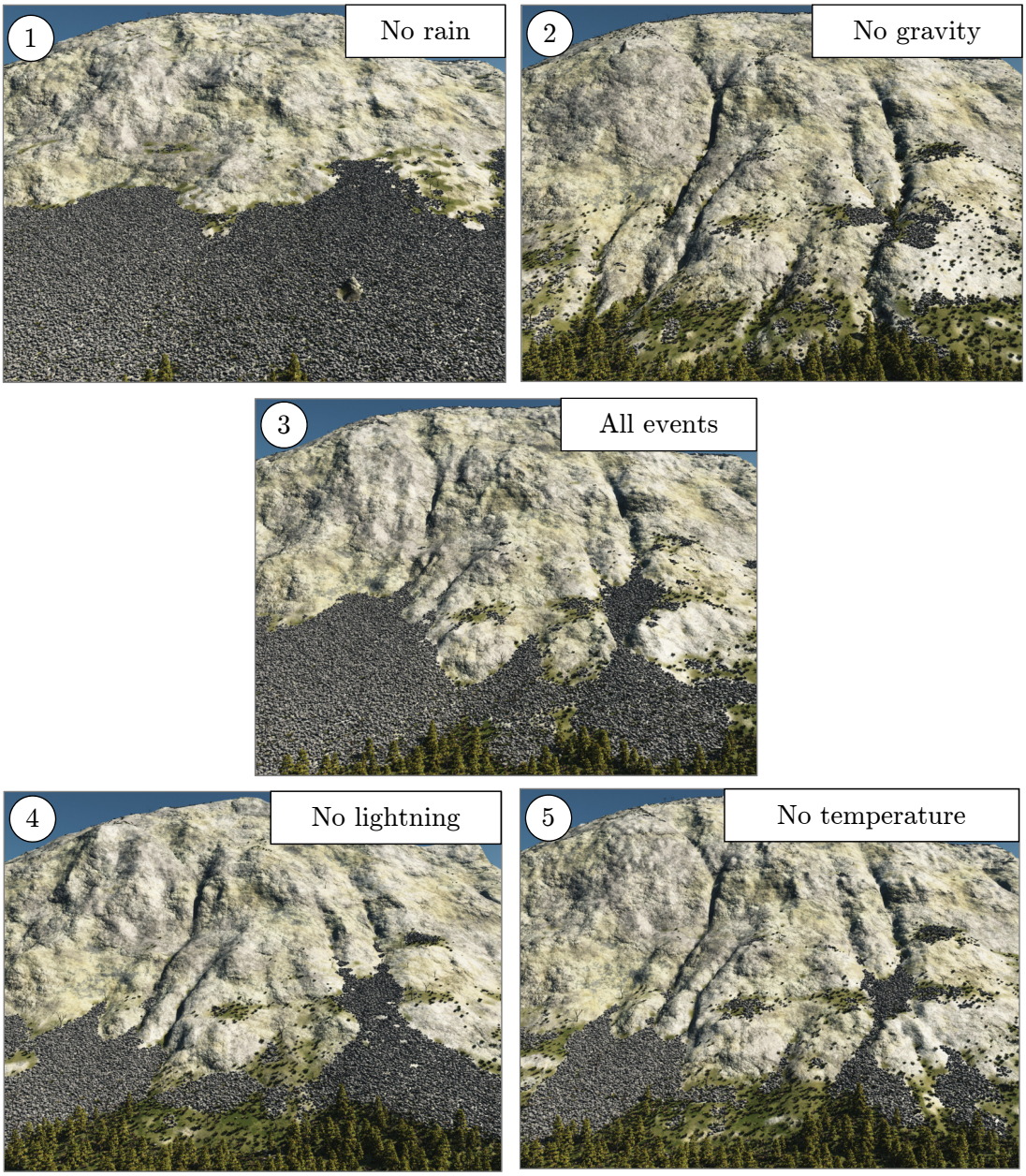


Figure 6.16: Disabling the events with respect to the complete simulation rendered in (3) shows their impact: (1) no rain, (2) no gravity, (4) no lightning, and (5) no temperature

Although each event type in isolation has little impact on a terrain, their significance is heightened by combined interaction. Compared to a combined simulation (Figure 6.16), an absence of erosion and sedimentation linked to rain prevents rock from being carved and soil from being transported into channels. Without gravity there are no rockslides to fill lower erosion pockets. A lack of lightning and temperature simulation prevents the erosion of exposed and steep slopes, respectively. The importance of fire in hot, dry landscapes is depicted in Figure 6.17.

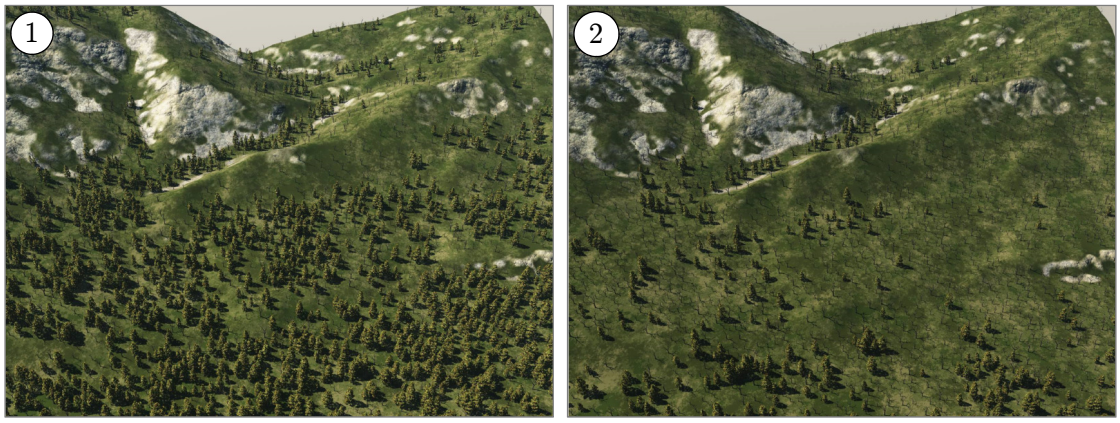


Figure 6.17: Effect of fires on a warm, dry landscape: no fire for 200 years (left), compared with regular fires every other year (right).

In terms of ecosystem simulation, our terrain erosion models capture visual outcomes not considered in previous work. First, our rainfall model accounts for moisture absorbed by different soil layers with differing moisture retention and these layers are distributed by erosion forces in a consistent fashion. For example, in Figure 6.18.(1) hilltops receive less moisture and thus less plant cover. Most importantly, our model considers humus generation, displacement, and exploitation by vegetation. For example, in Figure 6.18.(2) a pile of sand painted by the user to override humus retards vegetation growth. Second, erosion destroys plants through catastrophic events. For instance, rockfalls crush and bury trees, an effect that is characteristic of alpine forests.

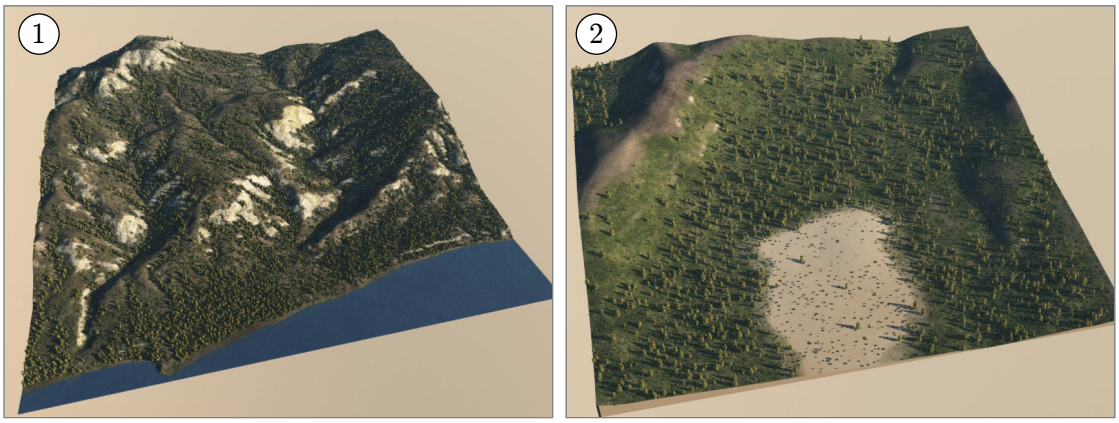


Figure 6.18: Effect of soil type on vegetation: (1) hilltops receive less moisture, (2) user-painted sand slows vegetation growth.

Real terrain data with different layers (humus, rock) is not readily available, making comparison difficult. This is exacerbated by temporal simulations that run anywhere from 100 to 1,000 years. Instead, we validate our results by comparing them with real phenomena illustrated by photographs. Figure 6.19(1) shows the destruction of vegetation by rock-slides, which we reproduce in Figure 6.14(1). Figure 6.19(2) has trees growing over accretion areas, and is mimicked in Figure 6.14(2). Finally, Figure 6.19(3) is a granular terrain, deeply eroded due to an absence of vegetation, which is simulated in Figure 6.20 by replacing bedrock with only granular materials.

Figure 6.19: Photographs of real-world landscapes show: (1) vegetation cleared by an avalanche of rocks, (2) vegetation growing on accretion cones, (3) terrain erosion accelerated by an absence of plants (© Creative Commons).

Limitations: One limitation of our framework is the computational cost of event simulation. Unfortunately, parallelizing the simulation is non-trivial, because events occur in sequence and may overlap spatially. In the interests of responsiveness, we limit users to interacting with coarser-scale landscapes ($n = 128$ to 256). One way to compensate for this is to store the events and user modifications and re-run them with an off-line upsampled simulation. Having many combining events increases both the number of simulation parameters and the complexity of their interdependence. These parameters tend to be hard to tune and are not artist-friendly.

Another limitation is the discrete nature of the simulator. While a discretized grid is more efficient, a continuous domain would be more appropriate for phenomena that act at different scales. In the same vein, we do not account for accurate fine-scale detail, so the method is less suited to close-up views. In future work, detailed geometry such as piles of rocks and boulders or fallen branches could be added using mass instancing (Guérin, Galin, et al. 2016).

Validation is a challenge common to all but the simplest simulation methods. While we included real images for comparison, it is difficult to quantify how closely the results match corresponding effects in nature. Our evaluation is only visual.

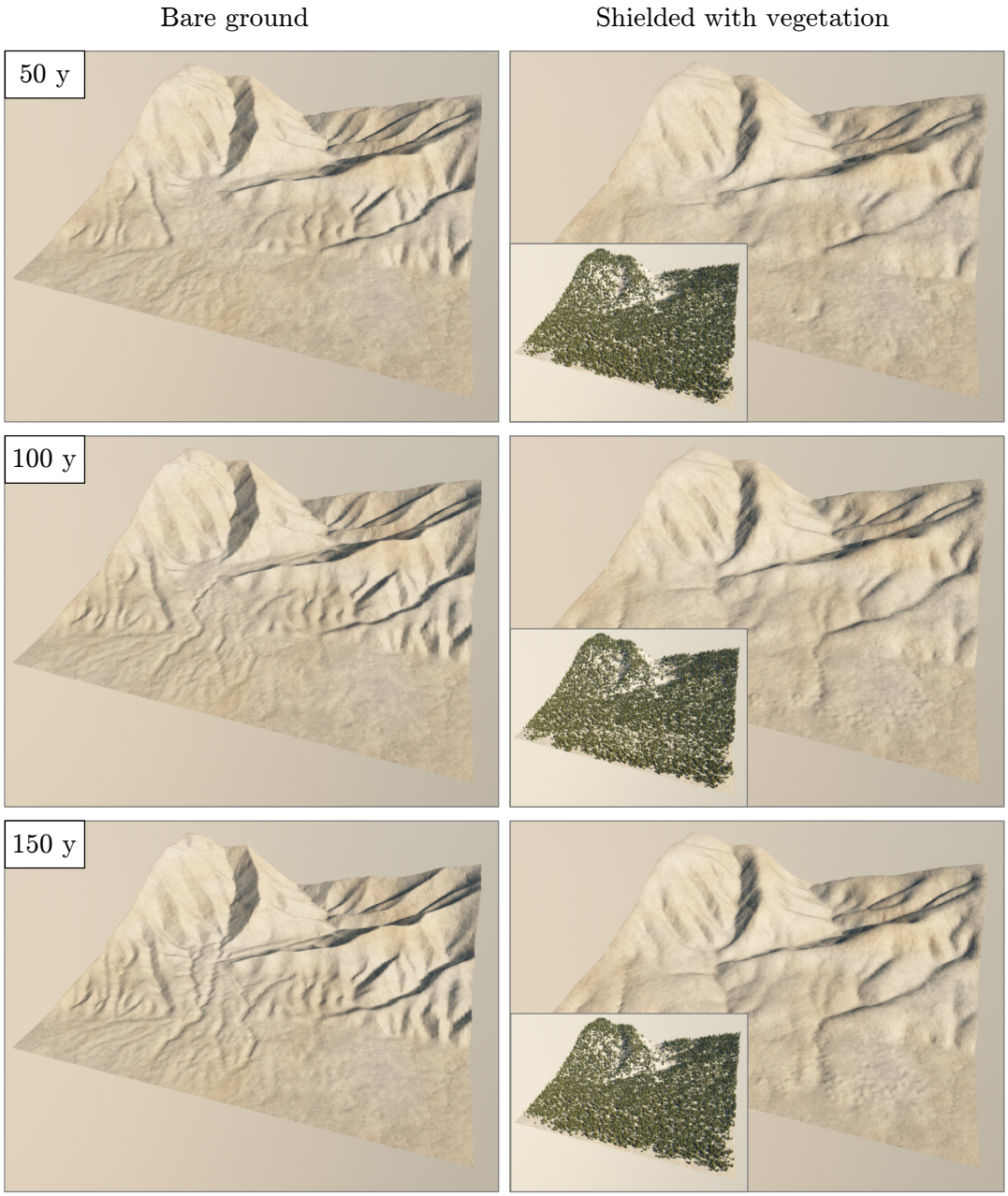


Figure 6.20: Shielding effects of vegetation: a time lapse of simulations on slopes with granular materials, (left) without vegetation to damp soil erosion, and (right) protected by vegetation.

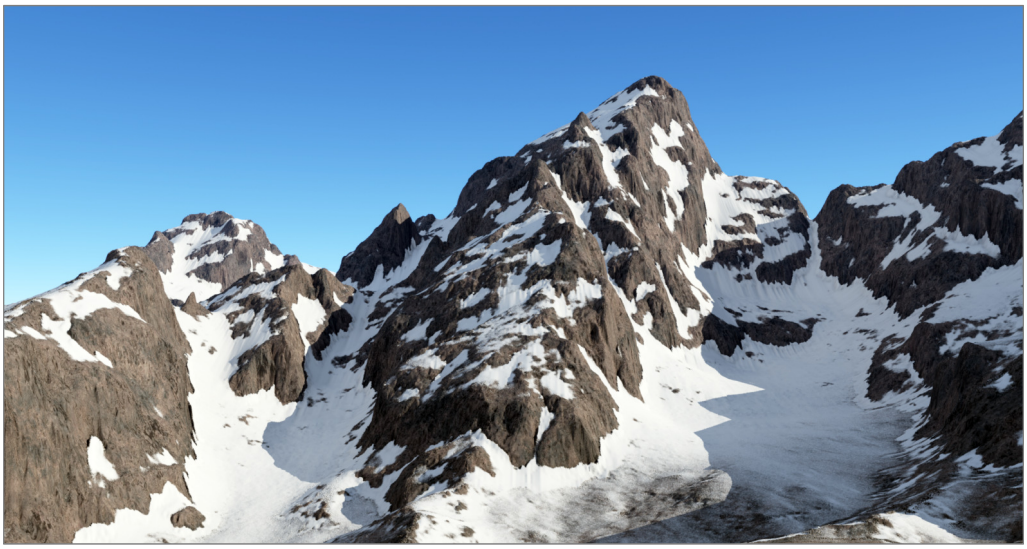
6.6 Conclusion

In this chapter, we presented a novel landscape editing framework, which enables the simulation of complex interactions between a variety of phenomena, ranging from vegetation lifecycles and terrain erosion to lightning and fire disturbance. This is achieved thanks to a layered landscape model, which stores terrain materials, vegetation densities, and other resources, and is evolved over time through a large number of stochastic events. We avoid a complex, joint simulation over all phenomena by launching these events in random order and restricting them to interact only through the landscape layers. This reduces computation time, simplifies the interactions between different phenomena, and enables interactive user control. Our editing tools enable the user to over-paint any of the landscape layers at runtime, throughout the simulation, thereby combining realism and control.

Due to the versatility of this approach, many additional events could be added to broaden the range of simulated phenomena to cover wind (affected by vegetation and topography), sand dunes, grazing herds, and different rock, soil and plant types. It would also be interesting to incorporate an extra layer for bodies of water, such as large rivers and lakes, updated using shallow water fluid simulation. This direction is explored in Chapter 7, not for general water flows but in the interesting case of a snow layer affected by sunlight, wind, skiers and avalanches. To tackle the challenge of interactive fluid simulation for snow cover, the simulation method is also extend to work on the GPU.

Chapter 7

Dynamic snow cover evolution



Contents

7.1 Overview **135**
7.1.1 Simulation method 136
7.1.2 Categories of events 138
7.2 Environmental conditions **138**
7.2.1 Temperature 138
7.2.2 Wind 140
7.3 Snow cover **142**
7.3.1 Snowfall 142
7.3.2 Snow state changes 143
7.3.3 Diffusion of powdery snow 144
7.3.4 Wind transport 144

7.4 Interactive phenomena **145**
7.4.1 Avalanches 146
7.4.2 Ski tracks 147
7.5 Implementation **149**
7.6 Results and discussion **152**
7.7 Conclusion **156**

Mountainous snow-covered landscapes are among the most visually-arresting vistas. In nature, snow coverage depends on altitude, but also a host of other phenomena, such as snow melting more on sun-facing slopes, snow shifted by the wind as channeled by topography, avalanches scouring some of the steepest slopes, and human activities, such as skiing, which leave visually-prominent imprints. Such snow-covered landscapes are heavily used in animated films and computer games, where their static portrayal provides a compelling backdrop, while dynamics elements (such as ski tracks and avalanches) serve a storytelling function, but a manual modeling process predominates. The challenge in instead generating these phenomena through simulation, lies is achieving both plausible results and the efficiency necessary for control.

Although the targeted result is very different in temporal scale and in visual aspect, the problem of generating snow-covered landscapes shows some similarity with the problem of temporal scale variability presented in Chapter 6. Snow generation encompasses a wide range of phenomena that take place at different scales both in space and time, from temperature variations (months) to avalanches (minutes). The stochastic framework presented earlier lends itself for the generation of snow, although it needs to be extended to handle the simultaneous interactive edition of both static and dynamic landscapes. This chapter describes our work presented at Eurographics 2018 (Cordonnier, Ecormier, et al. 2018).

We target medium scale scenarios (from 1×1 km to 10×10 km in extent, modeled using a $1\,024 \times 1\,024$ grid) – a resolution fine enough to capture snow-drift, avalanche and ski-track effects, but broad enough to encompass an expansive vista.

A key observation studied throughout this chapter is that avalanches and other phenomena such as snow-fall, snow-melt, snow-drift and Nordic skiing have significant visual impact on snow-covered landscapes. They should form part of the designer's tool-set, since they literally sculpt the landscape. While existing methods allow for the controllable placement of static snow cover, avalanches and snow-drift cannot be achieved as easily because they involve dynamic phenomena that *reallocate* snow mass. In contrast, these dynamic phenomena, optionally guided by the user, are at the heart of our modeling pipeline.

The input to our method is an elevation map overlaid with several qualitatively-different layers of snow (compacted, stable, unstable, and powdery), which are initialized by the user and set to evolve over time. For each time-step (*e.g.,* one day), the snow cover is updated as a consequence of various phenomena. First, we account for evolution in the local snow composition, such as a shift from stable to unstable snow with warmer temperatures. Second, a number of external events act on the snow layers. As examples: wind shifts powdery snow, avalanches may occur in steep areas with unstable snow, and skiing both compacts powdery snow and potentially triggers avalanches. These events sculpt the landscape, leading to the formation of characteristic features, such as overhangs on crest lines caused by snow-drift.

We extend the stochastic simulation framework presented in Chapter 7 with events specific to the phenomenological simulation of both the evolution of the snow cover and fast flowing avalanches. Our GPU acceleration enables interactive rates. We introduce a temporal zoom mechanism to visualize fast avalanches and dynamic ski tracks as details within the day long time-step. The user designs the landscape and specifies scenarios for its temporal evolution by using brushes: any point on the simulation time-line can be selected and tools applied to adjust the snow layers, either globally or locally. The probability of avalanches and skiing areas can also be prescribed. The user can thus rapidly explore design alternatives within the parameter space of events.

Our method is based on the following technical contributions:

- A novel interactive framework for modeling snow-covered countryside built on the GPU.
- An efficient method for visually simulating avalanches that combines viscous fluid and granular material behavior in a balance dictated by external conditions. This is seamlessly integrated with the stochastic simulation framework.
- A simple, yet accurate method for efficiently generating ski-tracks, which accounts for human impact on the uppermost snow layer and provides compelling visual detail.

Although we will detail the fourth contribution for the sake of completeness, it should be noted that this contribution about the impact of ski tracks is from the second author, Pierre Ecormier.

7.1 Overview

Our method enables users to interactively design static and dynamic medium-scale snow-covered landscapes (typically, $1-10\,\mathrm{km}$ on a side), at a ground-plane and temporal sampling resolution that is sufficient (typically, $1-10\,\mathrm{m}$ per cell and one day per time step). In particular, we want to capture characteristic snow-field effects, such as snow cornices on the leeward side of mountain crests, buildup at the base of slopes scoured by avalanches, and ski-tracks that weave downhill. Lastly, to enhance the user experience and improve control, we want to visualize fast dynamic events such as avalanches or new ski tracks generated at the speed a user expects from real life, even if the remainder of the simulation runs at a faster pace.

Our method generates snow cover for a scene, represented by a heightfield grid with additional layers for compacted, stable, unstable and powdery snow, by drawing from a set of concurrent phenomena, including precipitation (snowing in our case), snow melt and wind transport (Figure 7.1), which interact, mutually but indirectly, through the shared snow layers.

These phenomena, in turn, are strongly influenced by time-linked environmental conditions, like temperature, prevailing wind, and sunlight intensity. Avalanches and skiing represent a class of complex limited-duration phenomena that require special treatment, particularly if they are to be incorporated as dynamic, real-time effects in games and virtual environments. There are intricate evolving interactions between the layered scene, the evolution of snow cover over time, and the environment. One causal chain might involve wind

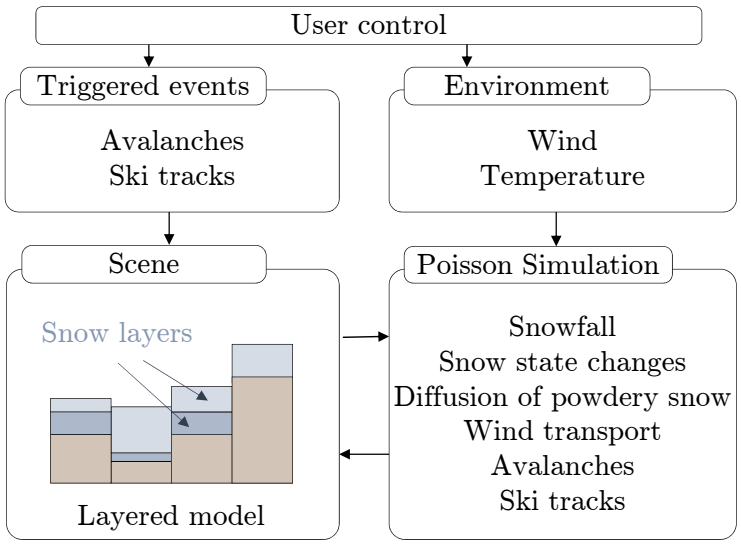


Figure 7.1: System overview: The user can define the environmental conditions that influence the snow simulation, or directly and interactively trigger events such as avalanches or new ski tracks. These are used in our unified stochastic Poisson simulation. The output is a static or dynamic snow-covered landscape.

shifting powdery and unstable snow across a crest onto a sheer slope, setting up ideal conditions for an avalanche to be triggered by an oblivious skier. To achieve such intricate simulation at interactive rates, the key idea is to use Poisson processes for interweaving incremental steps of the various events to be simulated, often at quite different temporal scales (presented in Section 7.1.1 and Figure 7.2).

The user can direct the simulation in several ways: by defining a temporal scenario for changes in environmental conditions, by directly painting height changes into the snow layers, by interactively exploring the phenomena parameters, or by triggering avalanches and skiing. In the interests of interactive performance, all components of the framework are implemented on a GPU.

7.1.1 Simulation method

The scene model consists of a regular grid. Each cell contains a stack of height values representing a set of ordered layers: bedrock (the initial static heightfield input), compacted snow (an icelike layer subjected to pressure from above), stable snow (a cohesive layer bound strongly to the terrain), unstable snow (a weak layer susceptible to slippage when perturbed), and powdery snow (an aerated layer with little cohesion subject to constant small spills).

Environmental conditions: An interactive session starts with a pre-computation step, where the environmental conditions are used to compute initial snow cover (Section 7.2). They include temperature (computed from altitude and illumination) and wind (computed from altitude and wind speed at sea level). These phenomena are only considered when the simulation is launched, or if the user decides to change environmental conditions. Results are

stored as static layers over the terrain using a scalar field for temperature and a vector field for wind speed, and used, together with the snow cover layers, to compute the local effect of the different runtime events, all of them being applied over the whole terrain.

Poisson stochastic simulation: At runtime, the phenomena of interest require different time scales: for instance, the typical time-scale of melting snow is far longer than required for a running avalanche (Figure 7.2).

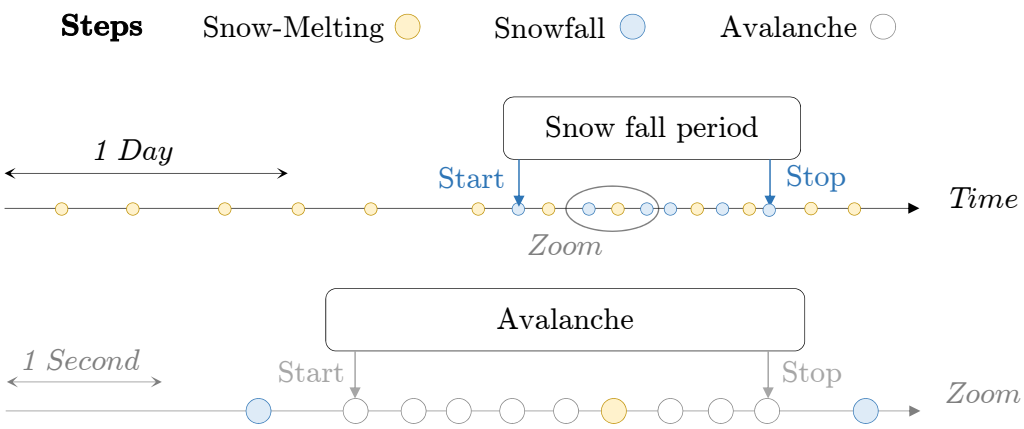


Figure 7.2: Temporal development of snowfall, snow melting, and an avalanche. Each event is handled by a Poisson process, some, such as the snowfall and avalanches, are preceded and followed by start and stop events. Avalanche steps have a high frequency and an entire avalanche may take place between two lower frequency events (such as snowfall steps).

To handle this, the stochastic simulation of Chapter 6 is extended as follows: a Poisson process is associated with each phenomenon, which is thus expressed through a series of individual stochastic events, applied on the whole terrain at a given mean-frequency f_e. We use a random variable t_e to represent the time at which the next event of type e is to be triggered. The variable t_e follows an exponential distribution defined by the probability density function:

$$\mathcal{P}_e(x) = f_e e^{-f_e x} \text{ if} x \geq 0 \qquad \mathcal{P}_e(x) = 0 \text{ otherwise.}$$

Since the set of times t_e for the different event types are independent stochastic variables, the probability of the next event being e is:

$$P\big(\, e \mid t_e = \min\left(t_1, \cdots, t_n\right)\big) = f_e / \sum_{i=1}^{n} f_i \tag{7.1}$$

The simulation algorithm thus runs as follows: At each simulation step, the system randomly selects which event is next according to the distribution of probabilities in Equation (7.1), and triggers it. The global simulation time is then increased the mean time-step associated with the current phenomenon.

Such event-based time-steps seamlessly support *temporal zooming* when high-frequency events are triggered. For example avalanches and ski tracks, are assigned a much higher fre-

quency than other events, enabling them to be perceived as dynamic phenomena (Figure 7.2).

7.1.2 Categories of events

We split the phenomena into a series of stochastic, infinitesimal *events*, each applied to the whole terrain (with, however, a different local effect depending on environmental conditions and snow layers). In addition to simulation step events, some phenomena such as snowfall require a start event followed by given a number of steps. We use two categories of events:

Snow cover evolution encompasses phenomena that take place over a longer period through a slow series of stochastic events (Section 7.3). For instance, *Snow-Melting* and *Wind-Transport* (snow transported elsewhere by the wind) are two phenomena that are always active and that take place through a series of *Snow-Melting-Step* events and *Wind-Transport-Step* events, respectively, each occurring twice a day on average. Another phenomenon is *Snowfall*, triggered by a *Snowfall-Start* event occurring once per week on average, followed by a series of *Snowfall-Step* events that are triggered twice a day over a period of a few days. Each *Step* event updates both snow distribution and snow stability.

Interactive phenomena include avalanches and ski tracks in our framework. Both take place over a short period (Section 7.4). In addition to being stochastically called at runtime (*e.g.,* once a week on average for avalanches on a one square kilometer terrain, and once an hour for ski tracks), events such as avalanches and ski tracks can be directly triggered by the user. This is achieved by increasing the frequency of the associated *Avalanche-Start* (respectively *Ski-Track-Start*) event, after creating favorable conditions at the user selected location, until the requested event is launched. The same method is used to enforce catastrophic events, such as ski-tracks triggering immediate avalanches when they cross unstable snow layers. While an avalanche (respectively a Ski-Track) phenomenon is active, *Avalanche-Step* (respectively *Ski-Track-Step*) events take place every $0.1s$ on average, which enables the user to see the dynamic phenomenon taking place at its natural (real-time) speed.

7.2 Environmental conditions

Snow coverage changes depending on the locally-varying environmental conditions, among which the most important are temperature (as affected by sunlight) and wind. These conditions depend primarily on terrain topography with the snow layers having negligible impact. Because of this, we can precompute these conditions and update them only when the user changes the terrain or an input parameter (such as wind direction or sea-level temperature).

7.2.1 Temperature

In our simulations, temperature is a key environmental input that depends primarily on altitude $\mathcal{A}$ and sunlight exposure $\mathcal{I}$. Strictly speaking, total altitude should be the combination of terrain altitude $\mathcal{B}$ and snow thickness $\mathcal{D}$. However, since the snow contribution is small ($\mathcal{D} \ll \mathcal{B}$) we use an approximation $\mathcal{A} = \mathcal{B}$ that allows precomputation. Temperature $\mathcal{T}$ is

calculated as:

$$\mathcal{T} = \mathcal{T}_0 + k_t \mathcal{A} + k_i \mathcal{I}.$$

$\mathcal{T}_0$ is the temperature at sea level, $k_t = -0.01°C\, m^{-1}$ is the per meter decrease in temperature with altitude, and $k_i = 10°C$ is the temperature increase due to 24-hours of direct sunlight.

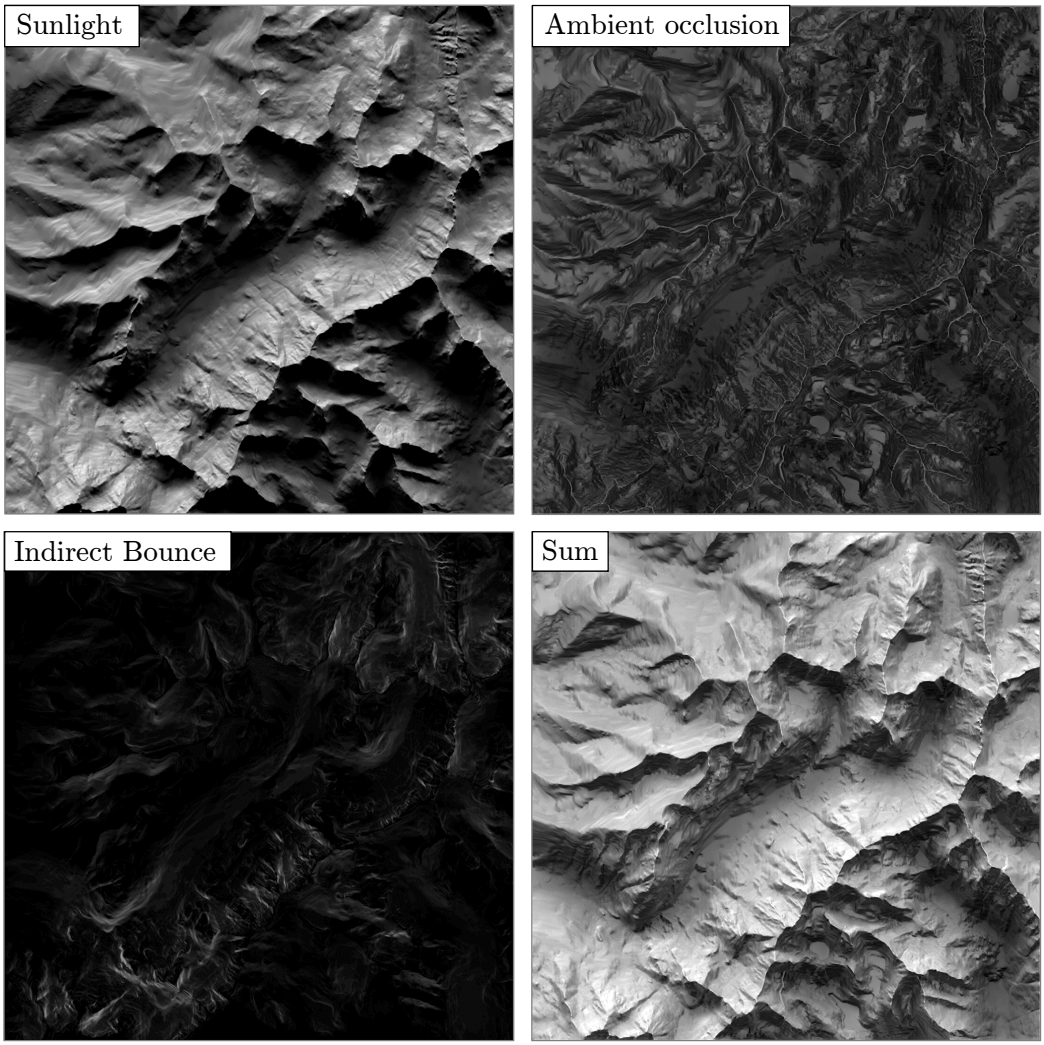


Figure 7.3: Sun exposure is computed by summing the direct sunlight, ambient occlusion, and one bounce of the indirect sunlight.

The calculation of sun exposure is more involved, since we factor in direct illumination potentially blocked by topography, ambient occlusion resulting from scattering due to clouds (Foldes et al. 2007; Maréchal et al. 2010), and a single ray bounce (Ritschel et al. 2009) accounting for the high reflectivity of snow. Total exposure (Figure 7.3) is then a sum of direct $\mathcal{I}_{sun}$, ambient $\mathcal{I}_{sky}$, and indirect sunlight $\mathcal{I}_{ind}$: $\mathcal{I} = \mathcal{I}_{sun} + \mathcal{I}_{sky} + \mathcal{I}_{ind}$.

Direct sunlight for a point $\mathbf{p}$ on the terrain is measured as the proportion of sun exposure during a day and is given by:

$$\mathcal{I}_{sun} = k_{sun} \sum_h s_h \, \mathbf{n} \cdot \mathbf{i}_h,$$

where k_{sun} is sun intensity, s_h indicates topographic shadowing of an incident ray $\mathbf{i}_h$ from the sun's position at hour h (see Chapter 6, Section 6.3 or (Gain et al. 2017)), and $\mathbf{n}$ is the terrain normal at the point $\mathbf{p}$.

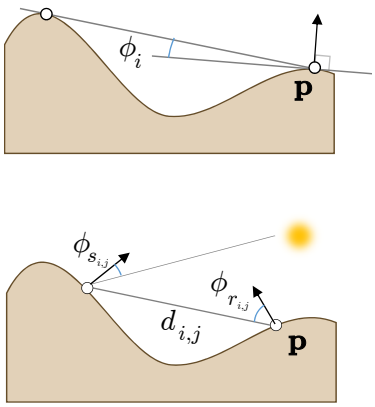


Figure 7.4: Terms used in the calculation of indirect sun exposure.

For ambient exposure, we make the assumption that ambient light $\mathcal{I}_{sky}$ is rotation invariant. We sample directions $\theta_i \in [0, 2\pi]$ around a terrain point $\mathbf{p}$ and obtain the maximal angle ϕ_i between the tangent to the terrain and the line between $\mathbf{p}$ and the highest visible point on the terrain in this direction (Figure 7.4, left). Then for an infinitesimal angle $\mathrm{d}\theta$ around θ_i, the ambient exposure is:

$$\mathrm{d}\mathcal{I}_{sky,i} = k_{sky} \int_{x=\phi_i}^{\pi/2} \cos(x)\sin(x)\,\mathrm{d}\theta\,\mathrm{d}x = k_{sky}\frac{\cos^2(\phi_i)}{2}\mathrm{d}\theta.$$

Total ambient sunlight at a point is the integral of the term above, which we discretize as a sum:

$$\mathcal{I}_{sky} = \sum_{i\in[1,N]} \frac{2\pi}{N}\frac{\mathrm{d}\mathcal{I}_{sky,i}}{\mathrm{d}\theta} = k_{sky}\frac{\pi}{N}\sum_{i\in[1,N]} \cos^2(\phi_i).$$

We evaluate indirect sunlight at a point $\mathbf{p}$ (the receiver) using a polar coordinate sampling of the nearby direct sunlight over random directions θ_i and distances $\{r_0, \ldots, r_j\} \in [0, r_{max}]$. For each such sample $\mathbf{p}_{i,j}$ (the sender) we define an angle $\phi_{r_{i,j}}$ between the receiver's normal and the sampling direction $(\mathbf{p}_{i,j} - \mathbf{p})$. Conversely, $\phi_{s_{i,j}}$ is the angle between sender's normal and the inverse sampling direction (Figure 7.4, right). Indirect sun exposure is calculated as:

$$\mathcal{I}_{ind}(\mathbf{p}) = k_{ind}\sum_i\sum_j A_{i,j}\,\mathcal{I}_{sun}(\mathbf{p}_{i,j})\,\frac{\cos(\phi_{s_{i,j}})\,\cos(\phi_{r_{i,j}})}{d_{i,j}^2},$$

where $A_{i,j} = 2\pi(r_{j+1/2}^2 - r_{j-1/2}^2)/N$ is the sampling area, for N angular samples, $r_{j+1/2} = (r_j + r_{j+1})/2$, and $d_{i,j}$ is the distance between $\mathbf{p}$ and $\mathbf{p}_{i,j}$.

7.2.2 Wind

In a snow-cover simulation, wind is both critical for its role in snow transport, and complex since it depends on topography. As a first approximation, we treat wind as a 2D velocity field that sits atop the terrain, and is initialized according to a globally uniform dominant 2D wind vector $\mathcal{W}_d$. This field is then enhanced by taking into account the effects of altitude, local slope direction, and wind shadows (Figure 7.6). Here again, we set $\mathcal{A} = \mathcal{B}$, allowing for precomputation.

First, we account for *Venturi* effects that accelerate wind at high altitudes, by scaling wind velocity with height:

$$\mathcal{W}_{venturi} = (1 + k_{venturi}\,\mathcal{A})\mathcal{W}_d.$$

While this linearization of the Venturi effect is not physically accurate it has the advantages of being efficient and easily controllable. Moreover, in our experiments the resulting snow

coverage was not visually distinguishable from results obtained with more accurate simulation.

Second, we redirect wind according to terrain slope. Let $\mathbf{n}_{xy}$ be the horizontal component of the normal vector to the surface. We define $\mathbf{n}_{xy}^{\perp}$ as the 2D vector obtained after 90° rotation of $\mathbf{n}_{xy}$ in the direction of $\mathcal{W}_{venturi}$ ($\mathcal{W}_{venturi} \cdot \mathbf{n}_{xy}^{\perp} \geq 0$). When the terrain is almost flat, $\|\mathbf{n}_{xy}\|$ is small and the wind direction does not change. Otherwise, the wind tends to align with $\mathbf{n}_{xy}^{\perp}$ as captured by the equation:

$$\mathcal{W} = \mathcal{W}_{venturi}\ (1 - \|\mathbf{n}_{xy}\|) + k_{terrain}\|\mathcal{W}_{venturi}\|\mathbf{n}_{xy}^{\perp}. \tag{7.2}$$

Finally, wind shadows form leeward of crests and ridge lines and this leads to the characteristic build-up of snow cornices. To achieve this, we assume that wind shadows are binary (wind at full strength or no wind at all), and we create a *wind-effect surface* representing the lowest altitude above which the wind blows at full strength.

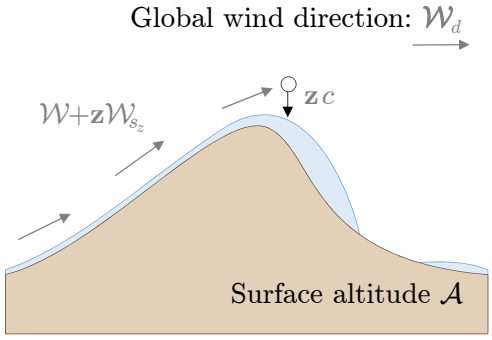


Figure 7.5: While the wind-effect surface is in contact with the ground ($\mathcal{W}_z = \mathcal{A}$), its vertical speed $\mathcal{W}_{s_z}$ is forced to follow the terrain gradient. Otherwise, $\mathcal{W}_{s_z}$ decreases by a constant value c, which makes the wind-effect surface follow a parabolic shape.

The horizontal wind velocity for this layer is set to $\mathcal{W}$ (Equation 7.2). The altitude of the wind-effect surface $\mathcal{W}_z$ is related to the vertical wind speed $\mathcal{W}_{s_z}$ as follows:

$$\begin{aligned} \mathcal{W} \cdot \nabla \mathcal{W}_{s_z} &= -c && \text{if } \mathcal{W}_z > \mathcal{A} \\ \mathcal{W}_{s_z} &= \mathcal{W} \cdot \nabla \mathcal{A} && \text{otherwise,} \end{aligned}$$

where c is the constant by which the vertical wind speed decreases when the wind-effect surface is strictly above ground level (Figure 7.5).

The value of c is set experimentally to $0.7\,m\,s^{-2}$, and controls the extent of the wind-shadow cap. The smaller this value, the bigger the cap. The layer's altitude is then derived from:

$$\mathcal{W} \cdot \nabla \mathcal{W}_z = \mathcal{W}_{s_z}.$$

The equation is undefined for $\mathcal{W} = \mathbf{0}$, in which case we set $\mathcal{W}_{s_z} = \mathcal{A}$. Leeward of ridge lines, the gradient of the elevation and the wind act in opposite directions creating a negative vertical speed that forces the wind-effect layer to slowly decrease, generating a parabolic wind-shadow cap. Windward, the wind is pushed toward the surface, where $\mathcal{W}_{s_z}$ is directed by the 2D gradient of $\mathcal{A}$.

Both altitude $\mathcal{W}_z$ and vertical speed $\mathcal{W}_{s_z}$ are computed using a Gauss-Seidel scheme. They are initially set to $\mathcal{W}_z = \mathcal{A}$ and $\mathcal{W}_{s_z} = \mathcal{W} \cdot \nabla \mathcal{A}$. Iteration proceeds in a black-and-white checkerboard fashion. On odd iterations, all white-assigned grid cells are updated with respect to black-assigned cells, as follows: first, $\mathcal{W}_{s_z}$ is calculated, independently from $\mathcal{W}_z$; then $\mathcal{W}_z$ is updated based on the new value of $\mathcal{W}_{s_z}$; and finally, $\mathcal{W}_{s_z}$ is corrected if $\mathcal{W}_z \leq \mathcal{A}$. On even iterations the black cells are re-evaluated.

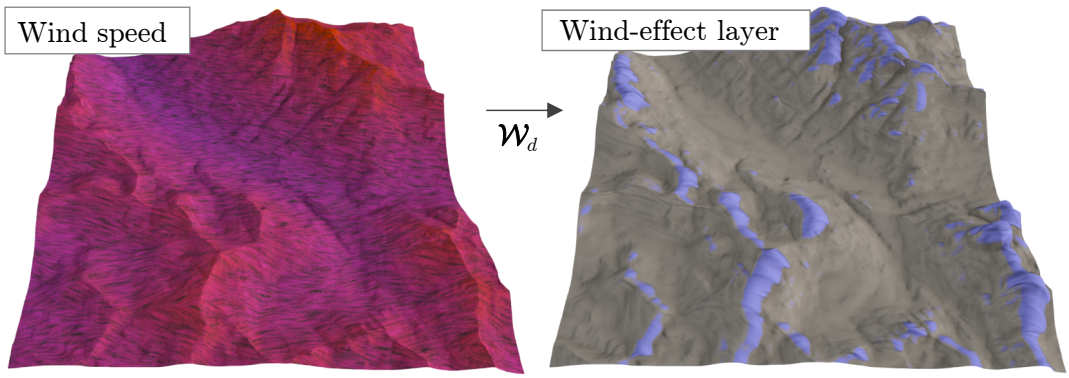


Figure 7.6: Wind speed increases with altitude – shown as color ranging from blue to red (left) with dark lines indicating the local wind directions; the wind-effect surface hugs the ground except leeward of ridges, where caps (in blue) are formed (right).

7.3 Snow cover

Snow is rarely homogeneous in either structure or composition. To express this we partition snow cover ($\mathcal{D}$) into four layers: compacted ($\mathcal{C}$), stable ($\mathcal{S}$), unstable ($\mathcal{U}$), and powdery ($\mathcal{P}$) snow. The distinction between the latter two is necessary because powdery snow often trickles downwards immediately after deposition, while unstable snow shifts only with an avalanche event. The unstable layer serves as a proxy for the interleaving of brittle and strong layers in real snow that are likely to trigger avalanches. Key to of our framework is tracking the evolution of these snow layers over time, under the action of precipitation, melting, diffusion, and wind shifts (see Figure 7.7).

7.3.1 Snowfall

Snowfall is a blanketing event that influences snow cover by adding to the uppermost layers. We use a mean triggering interval of one week and a mean duration of three days as defaults. Apart from these, the user can also adjust snowfall strength k_{snow}.

The Foehn effect allows a straightforward relationship between snowfall and altitude:

$$\mathcal{D}(t + \delta t_{\mathrm{p}}) = \mathcal{D}(t) + \delta t_{\mathrm{p}}\, k_{snow}\, \max(0, \mathcal{A} - \mathcal{A}_{0\,precip}),$$

where $\mathcal{A}_{0\,precip}$ is the altitude at which rainfall transitions to snowfall, and $\delta t_{\mathrm{p}} = 1\,\mathrm{day}$ is the precipitation time step. Snowfall is initially divided into powdery and normal snow, with the latter assigned to $\mathcal{S}$ and $\mathcal{U}$ layers after landing. First, some of the precipitated snow is converted into powdery snow (*i.e.,* snow that cannot bond to bare slopes or the existing snow layer). The proportion of snow that becomes powdery is $x_{\mathcal{P}} = k_{powdery}\,(\|\nabla\mathcal{A}\| - s_c)$ (clamped between 0 and 1), where $k_{powdery}$ is a user defined constant that scales the slope influence, $\|\nabla\mathcal{A}\|$ is the norm of the gradient of snow surface altitude, and s_c is a critical slope beyond which the snow cannot fall. Based on the assumption that this critical slope depends on the inertial temperature of the underlying ground, the critical slope increases when the

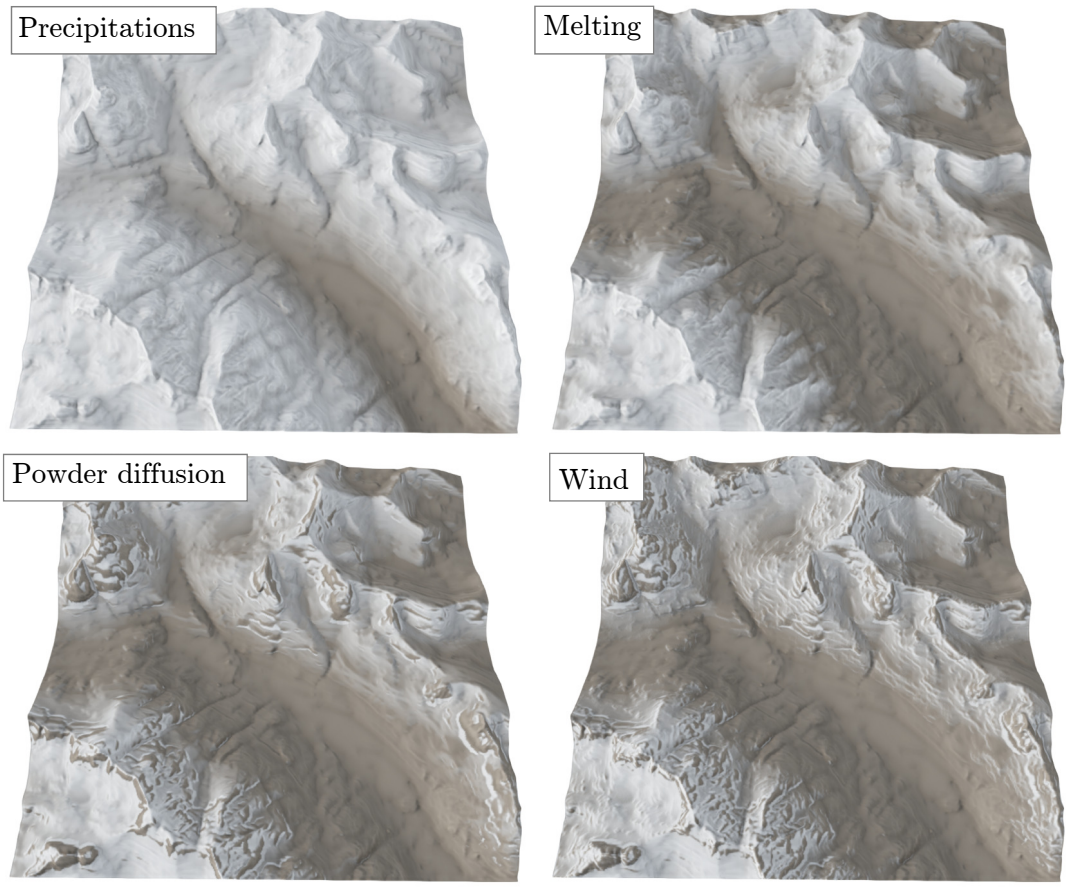


Figure 7.7: Effects of different phenomena on snow cover.

temperature decreases:

$$s_c = s_{c_0} + k_{s_c\,powdery} \max(0, \mathcal{T}_{0\,powdery} - \mathcal{T}),$$

where $k_{s_c\,powdery}$ scales the temperature's influence, and $\mathcal{T}_{0\,powdery}$ is the highest temperature influencing the critical angle. Then, the remaining proportion of snow $(1-x_{\mathcal{P}})$ is split between unstable and stable snow, where the proportion of unstable snow is: $x_{\mathcal{U}} = k_{unstable}\,(s - s_{\mathcal{U}})$, clamped between zero and one.

7.3.2 Snow state changes

Temperature (Maréchal et al. 2010) and slope (Lehning et al. 2008) are prime determinants of change in snow stability and hence shifts between layer categories. We propose a simplified model that takes both into account. Temperature induces melting of the snow cover according to:

$$\mathcal{D}(t + \delta t_{\mathrm{m}}) = \mathcal{D}(t) - \delta t_{\mathrm{m}}\, k_{\mathrm{melt}} \max(0, \mathcal{T} - \mathcal{T}_{0\,melt}),$$

where $k_{\mathrm{melt}} = 0.01 m\, day^{-1}\,{}^{\circ}C^{-1}$ is a global constant melting rate, $\mathcal{T}_{0\,melt} = 0^{\circ}C$ is the minimum temperature at which melting occurs, and $\delta t_{\mathrm{m}} = 0.5$ day is the state change time

step. Melting occurs layer by layer, progressing from top (unstable) to bottom (compact). The resulting water is not taken into account in our model.

Transitions in snow stability depend on temperature. At warmer temperatures ($\mathcal{T}_{warm}$) snow becomes very unstable. At cool temperatures around freezing ($\mathcal{T}_{cool}$) oscillations between partial melting and refreezing act to cement snow stability. At cold temperatures ($\mathcal{T}_{freeze}$) snow stabilizes only under the pressure of its own weight. By experimenting with our model, we have found that this dynamic is captured by linearly interpolating stability changes depending on temperature: we use parameter values of $-k_w$, k_c, and 0 for warm, cool and cold temperatures, respectively, where $-k_w$ and k_c are user defined constants that relate to the speed of state change. Slope is also taken into account: instability increases slower on gentle slopes, and conversely stability increases slower on steeper slopes.

7.3.3 Diffusion of powdery snow

Because of its consistency, powdery snow undergoes an almost constant diffusion process of localized shifts and spills. We apply this diffusion on the powdery layer ($\mathcal{P}$) at a high update frequency. It is computed for each cell $\mathbf{p}$ using the slope of each of the 4-cell neighbors $\mathbf{p}_n$:

$$s_d = (\mathcal{A}(\mathbf{p}) - \mathcal{A}(\mathbf{p}_n))/\|\mathbf{p} - \mathbf{p}_n\|.$$

The proportion of powdery snow shifted in direction d is a function of slope s_d in that direction, a constant rest slope value (s_{d_0}), and a shift rate parameter (k_m):

$$m_d(\mathbf{p}) = \begin{cases} \delta t_\mathrm{d}\, k_m \max(0, s_d - s_{d_0}) & \text{if } s_d \geq 0 \\ \delta t_\mathrm{d}\, k_m \min(0, s_d + s_{d_0}) & \text{otherwise,} \end{cases}$$

where δt_d is the time-step of the diffusion events. We define snow incoming i or outgoing o from $\mathbf{p}$ as the sum of all negative or positive m_d, respectively. Each outgoing positive m_d is scaled in proportion to the powder layer $\mathcal{P}$ by $(i + \mathcal{P})/o$ if $o > i + \mathcal{P}$ to prevent negative quantities of powdery snow. The powdery layer is then updated by shifting snow from or to neighboring cells with respect to m_d.

7.3.4 Wind transport

Wind acts on powdery snow to scour and advect it based on the curvature of the snow surface (Figure 7.8). To achieve this, we compute the wind resistance of the snow surface, taking into account wind velocity $\mathcal{W}$ and altitude $\mathcal{A}$:

$$c_\mathcal{W} = |\mathcal{W}_x| \frac{\partial^2 \mathcal{A}}{\partial x^2} + |\mathcal{W}_y| \frac{\partial^2 \mathcal{A}}{\partial y^2}.$$

In practice, we zero $c_\mathcal{W}$ if it is positive, or if the wind-effect surface is distinct from the snow surface (as in the blue wind-shadow regions of Figure 7.6 right). Snow is then eroded from a cell in proportion to the concavity: the amount eroded is $\max(\mathcal{D}, -k_{erosion}\, c_\mathcal{W})$ and it is deposited to two of its neighbors in the direction of $\mathcal{W}$ (refer to Section 7.5 for implementation details).

Wind also weakens the stability of snow cover by tamping slopes, thereby forming a thin

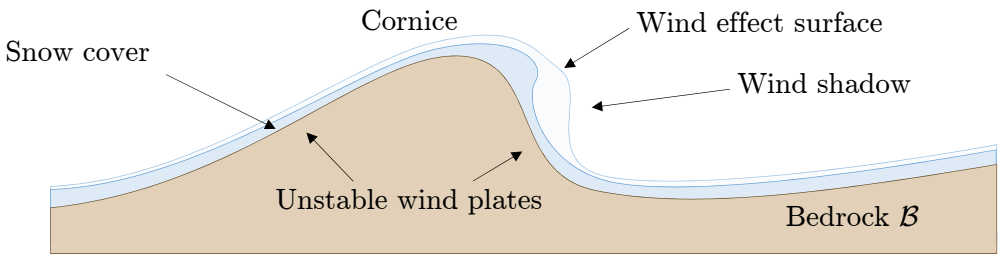


Figure 7.8: The impact of wind on snow cover includes increased instability and formation of cornices.

brittle shell over snow that would otherwise spill downslope. To account for this effect, we increase instability in proportion to positive changes in vertical wind speed, as illustrated in Figure 7.9.

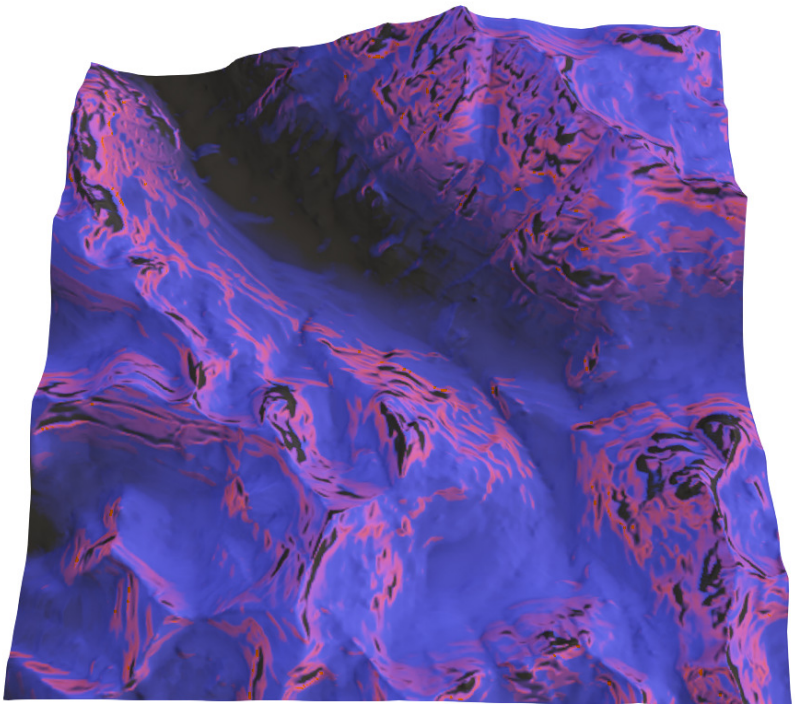


Figure 7.9: Stability map: when present, unstable and powdery snow are shown in purple on top of compacted and stable snow in blue. Regions with no snow are in black.

7.4 Interactive phenomena

The phenomena described in this section are either triggered automatically (with a low per-cell probability so that their occurrence is rare), or manually by the user selecting a seed point. The latter is most useful if real and simulation time are synchronized so that users can best judge the dynamics of the phenomenon.

7.4.1 Avalanches

Avalanche behavior involves many correlated conditions, including terrain shape and initial environmental conditions, and a variety of constituents, including entrained air and different forms of snow. Here, we focus on two principle avalanche types: (1) *Dry snow avalanches* composed of powdery snow mixed with cohesive ice-blocks. Such avalanches are best treated as a flow of granular material; (2) *Wet snow avalanches* with heavy, part-melted snow that behaves as a viscous fluid. These types represent two poles of a continuum: our unified treatment allows a mix of behaviors parameterized by temperature.

We should note that snow type (either wet or dry) is only one axis in the standard classification of snow avalanches (McClung et al. 2006). Other axes of the zone of origin include the type of start zone (point or slab) and the level of the sliding layer (surface or full). Within this classification scheme we support surface sliding point zone avalanches for both wet and dry snow types.

Fluid simulation. Both avalanche types can be encompassed by a fluid simulation and for this we choose to implement a hydrostatic pipe-model from (O'Brien et al. 1995; Šťava et al. 2008). This 2D Eulerian method discretizes the simulated domain into 2D cells corresponding to columns of snow that are connected by virtual pipes. A pipe's pressure depends on the fluid content, and the simulation stabilizes the flow of fluid induced by pressure differences. For a given cell $\mathcal{C}$ in the terrain grid, we use Moore's neighborhood, which considers all eight neighboring cells. The difference in pressure ΔP_i between two neighboring cells $\mathcal{C}$ and $\mathcal{C}_i$ is:

$$\Delta P_i = \rho\, g\, (\mathcal{A}(\mathcal{C}) - \mathcal{A}(\mathcal{C}_i)),$$

where $\mathcal{A} = \mathcal{B} + \mathcal{D}$ is the altitude of the uppermost surface, ρ is the density of the fluid, and $g = 9.81$ is the gravitational constant. The acceleration a_i of the snow between neighboring cells is:

$$a_i = \Delta P_i / (\rho \mathrm{d}x),$$

where dx is the cell width. The flow in the pipe evolves as:

$$\phi_{\mathcal{C}\to\mathcal{C}_i}(t+\delta t) = \phi_{\mathcal{C}\to\mathcal{C}_i}(t) + \delta t\, c\, a_i,$$

with c being the cross section of the pipe (set as constant as $c = dx^2$). Finally the height of the unstable snow layer $\mathcal{U}$ is adjusted according to:

$$\mathcal{U}(t+\delta t) = \mathcal{U}(t) - \delta t\, dx^{-2} \sum_i \phi_{\mathcal{C}\to\mathcal{C}_i}(t). \tag{7.3}$$

To enforce snow removal by a positive amount, we use a re-scaled version of $\phi_{\mathbf{p}\to\mathbf{p}_i}(t+\delta t)$ in Equation (7.3), with the scaling factor: $\min(V_{out}, dx^2\mathcal{U}(t))/V_{out}$, where V_{out} is the total outflow and:

$$V_{out} = \delta t \sum_i \max(0, \phi_{\mathcal{C}\to\mathcal{C}_i}(t+\delta t)).$$

Granular material flows are commonly modeled by adding a yield criterion to the fluid, *i.e.,* a friction force opposing the flow, of norm:

$$|\mathbf{f}| = g \tan(\theta_c),$$

and where θ_c is the rest angle of the snow. If the flow is null, or applying this force would reverse flow direction, then flow is set to zero to model static friction. Such a yield criterion can introduce gridding artifacts in an Eulerian simulation. However, in our case the avalanche footprint is small relative to the terrain and no visual artifacts are apparent.

Viscosity in the avalanche is obtained by adding a viscous force, opposed to the flow direction $\phi_{\mathbf{p}\to\mathbf{p}_i}$:

$$\mathbf{v} = -k_{\mathbf{v}}\,\mathcal{U}\,\phi_{\mathbf{p}\to\mathbf{p}_i}.$$

We augment the method of Št'ava et al. 2008 by adding these forces in the acceleration term:

$$a_i = \Delta P_i/(\rho \mathrm{d}x) + \alpha\mathbf{f} + (1-\alpha)\mathbf{v}$$

where $\alpha \in [0,1]$ is proportional to snow temperature.

Application to our framework: In our simulations, only the unstable layer $\mathcal{U}$ and the powdery snow $\mathcal{P}$ on top (if not yet diffused) are subject to avalanche redistribution. It is assumed that the stable and compacted layers are too strongly fused (making our avalanches of surface layer avalanche type).

In our framework, we use a *snow moving* boolean layer to decide if unstable snow is locally still or in motion. The snow is initially still. A *rupture point* can then be triggered either automatically or by the user. If the unstable layer $\mathcal{U}$ is non empty, then the avalanche rapidly propagates to neighboring areas, with reach proportional to the thickness of $\mathcal{U}$. All unstable snow in the rupture zone is immediately set into motion, as well as all unstable snow in the downslope reach of the avalanche.

7.4.2 Ski tracks

Human action on a terrain is another non negligible factor in modeling realistic mountainous landscapes. In particular, Nordic skiing not only compresses snow layers and leaves tracks, which is important for a more authentic feel to the scene, but can also trigger avalanches when performed on unstable layers of snow.

The size of our simulation grid (10 m per cell) only allows a consideration of the general direction of the skiers. While this is sufficient when modeling the impact of Nordic skiing on snow state during the simulation, we need more precise paths to achieve realistic rendering. We thus opted to integrate a procedural method to generate plausible refined tracks.

Global path search. As with avalanches, skiers that are not user-triggered are automatically generated by a *Ski-Track-Start* event called on the terrain, where each cell has a low probability of spawning a skier. This probability is influenced by the length and viability of a ski route. Therefore, we pre-compute a map registering the distance from each cell to

its farthest down-slope end point (local minima or edge of the terrain). We use a simple cellular automaton that finds minima on the terrain and propagates the distance to them in subsequent steps to efficiently compute this map, directly on the GPU. This distance map is used both as a weighted mask when automatically spawning new skiers, and as a guide to discourage skiers from reaching dead-ends, as discussed next.

Large scale paths are computed to approximate general direction and movement of the skiers, without the detail of the curves used to regulate their speed. For that, we take into account the slope of the mountain by defining an ideal slope angle s_t that skiers will be comfortable with and try to follow, and make sure that there is enough snow for them to ski on. In this work, skiers are modeled as independent agents, defined by their position and orientation, responsible for deciding their next move using a local search, based on the amount of snow present in neighboring cells, the corresponding relative slope s_n and a pre-computed weight w_d related to the distance to a terminus. In practice, skiers have a small lookahead window of a few cells, and will steer in the best candidate direction defined by the center of a nearby cell. The probability for a skier aiming towards a given cell n is:

$$P(n) = \mathbf{1}_{snow>threshold}\, f(|s_n - s_t|)\, w_d, \tag{7.4}$$

where $\mathbf{1}$ is the indicator function, s_n is the slope between the current cell and cell n, and f is a function that assigns a weight, which can be changed to tune the behavior of skiers with regard to slope. To avoid sharp changes in direction, a smooth transition to the new steering direction is computed and applied to the orientation of the skier. The steering direction is re-evaluated at each animation step.

Refined tracks are created for rendering purposes. They are approximated based on the observation that the precise movement of skiers is analogous to sine waves of varying amplitude and frequency, with sharp turns used to slow down and straight paths to gain speed. With this in mind, we model movement on each straight segment of a global ski trajectory using $\mathbf{p}(t) = a\sin(2\pi f t)$, where a is the amplitude and f is the frequency, dynamically updated with the terrain's varying slope.

Indeed, a skier moving straight down a mountain will go faster than one with a trajectory following the isoline. To account for this, we compute the effective slope s_e of the skier's trajectory as:

$$s_e = \arcsin \frac{\sin(s_n)}{2f\,l} \qquad l = \sqrt{\frac{1}{4f^2} + 4a^2},$$

where l is the distance between sine curve extrema. This provides the local frequency value required for skiers to reach their comfortable target slope s_t:

$$f = \frac{\sqrt{\sin^2(s_n) - \sin^2(s_t)}}{4a\sin(s_t)}$$

Continuity with the previous refined position and orientation of the skier is ensured by choosing an appropriate starting phase value along the sine curve. At each animation step, the resulting movement detail is mapped on the fly onto the lower resolution trajectory computed using Equation (7.4). This is done using a local update to a ski tracks texture layer covering

the whole terrain. An alternative is to export this texture as a displacement map for off-line rendering.

Interaction between ski tracks and snow is two-way. Once a skier enters a cell it transforms a fixed amount of snow from unstable to stable, or from stable to compacted if no more unstable snow remains. If there is unstable snow still remaining then the probability of an avalanche is increased. Conversely, the impact of snow on rendered ski tracks is taken into consideration: as snow is deposited along the path or shifted by the wind, the tracks fade dynamically depending on the quantities involved.

7.5 Implementation

Our system is implemented in C++, using OpenGL and GLSL for rendering and CUDA for simulation. All examples were measured on a laptop equipped with an NVidia Quadro K2100M GPU. Off-line rendering was done with Terragen software.

Wind transport and diffusion of powdery snow can move snow from the current cell to one of its neighbors. This can cause race condition if the simulation executes on all cells concurrently. To avoid this situation, we run the kernel in multiple passes, each time affecting only cells that would not cause parallel write conflicts. We perform the simulation on cells according to a cross-shaped tiling, which requires five kernel launches (one per arm and one at the cross center). This is a very tight arrangement of kernels that allows for conflict free usage of direct neighboring cells. In our experiments, this implementation gave slightly better performances than using atomic instructions, and was also both easier to implement and more readable.

5	1	2	3	4	5
2	3	4	5	1	2
4	5	1	2	3	4
1	2	3	4	5	1
3	4	5	1	2	3
5	1	2	3	4	5

Avalanche simulation on the GPU is inspired by the work of Šťava et al. 2008, with the distinction that we store four floating point values for signed flow instead of the eight outflows: our GPU implementation allows writing to neighboring cells, as long as two cells are not written simultaneously. When computing the flow for the given cell $\mathbf{p}$, the resulting flows are written at $\mathbf{p}$ only if they are positive, and in the neighboring cells if they are negative. In this way, all the flows are updated and race conditions are avoided while maintaining high occupancy.

Ski tracks are also computed on the GPU. Skiers are spawned in parallel in each cell with a small probability, and then moved still in parallel on a cell-by-cell basis using the same tiling pattern described for wind transport. For large-scale paths, we only store the number, position an orientation of skiers present in each cell, and every thread dispatches its skiers to neighboring cells. For fine-scale tracks, we also store the frequency and phase of the sine curve used to refine each straight segment, since this is necessary to render continuous curves. An alpha-blending coefficient, updated when snow moves into a cell, is used to progressively fade-out ski tracks.

Performance is mainly impacted by temperature and wind calculations. Therefore, they are performed once in entirety as a pre-processing step and require approximatively 7.8 and 10.5 seconds, respectively, to compute. The simulation and the vizualization are done on the GPU. We achieve interactive simulation rates of 30Hz, on a 1024×1024 grid (Table 7.1). Wind transport and powdery snow diffusion are relatively slow because they require five kernel launches each. In contrast, precipitation and snow-state events are less demanding since they only require a single kernel launch.

Event	Time	n	Tot. time	%
Wind transport	60	2	120	10
Powdery snow diffusion	64	2	128	11
Avalanche	38	10	380	32
Snowing	12	1	12	1
Snow state change	16	2	32	3
Skier movement	50	10	500	43
Total	240	37	1 172	100

Table 7.1: Average time for one frame (in ms), number of occurrences of events n, total time (in ms), and percentage of the total for each stage on a $1\,024 \times 1\,024$ grid. In practice, the snow cover evolution runs for about 100 frames (days). An accumulated 1 000 shorter frames (representing fractions of a second) are used to visualize avalanches and ski tracks. Subsets of these frames are packed along the duration of such shorter-term events, usually between two longer-term frames.

Because we want to display a regular temporal evolution of one frame per day (except during temporal zoom), the number of occurrences of each event per day impacts performance. In our implementation, we perform on average 200 wind transport, powdery snow diffusion and melting events, 100 precipitation events, and 300 avalanche events over 100 days. We achieve an interactive time step of half a second, and the user has to wait about 60 seconds for the final landscape. When the user manually launches avalanches or ski tracks, the temporal zoom suppresses occurrence of all the other events, so that the interactive phenomenon is displayed in real time.

Constant	Description	Value
T_0	Sea level temperature	-
k_t	Temp. offset with altitude	$-0.01°C\,m^{-1}$
k_i	Temp. offset per daylight hour	$0.41°C\,h^{-1}$
k_{sun}	Amount of direct illumination	$0.9\,h$
k_{sky}	Amount of ambient illumination	$0.1\,h$
k_{ind}	Amount of indirect illumination	$0.9\,h$
$k_{venturi}$	Wind speed offset with altitude	$10^{-3}\,m^{-1}$
$k_{terrain}$	Topography effect on wind	0.5
$\|\mathcal{W}\|$	Wind strength at sea level	$10\,m\,s^{-1}$
c	Decrease of wind effect vs hight	$0.7\,m\,s^{-2}$
k_{snow}	Snow fall strength	$10^{-4}\,day^{-1}$
$\mathcal{A}_{0\,precip}$	Rain-snow limit altitude	-
$k_{powdery}$	Powdery snow from precipitation	5
s_{c_0}	Powdery snow min critical slope	0.5
$k_{s_c\,powdery}$	Temp. influence on s_{c_0}	$5 \times 10^{-2}\,°C^{-1}$
$\mathcal{T}_{0\,powdery}$	Highest temp. that affects s_{c_0}	$-10°C$
$k_{unstable}$	Unstable snow from precipitation	1
$s_{\mathcal{U}}$	Critical slope of unstable snow	0.3
k_{melt}	Melting rate	$0.01m\,day^{-1}\,°C^{-1}$
$\mathcal{T}_{0\,melt}$	Melting temp.	$0°C$
T_{warm}	Temp. of the unstable snow	$5°C$
T_{cool}	Optimal temp. for stable snow	$-5°C$
T_{freeze}	Min temp. affecting stability	$-20°C$
k_w	Instability induced by warmth	$10^4\,m\,day^{-1}$
k_c	Stability induced by cold	$10^4\,m\,day^{-1}$
s_{d_0}	Rest slope for snow diffusion	0.5
k_m	Diffusion rate	$0.5day^{-1}$
$k_{erosion}$	Wind erosion rate of snow	$0.1m\,s^{-1}\,day^{-1}$
ρ	Snow density	$0.5g\,cm^{-3}$
g	Gravitational acceleration	$9.8m\,s^{-2}$
θ_c	Rest angle of avalanche snow	$30°$
$k_{viscous}$	Snow viscosity	$5 \times 10^{-4}s^{-2}$
s_t	Target slope for skiers	$30°$
a	Tracks sine wave target amplitude	$5\,m$
f	Tracks sine wave target frequency	$0.01\,m^{-1}$

Table 7.2: Simulation parameters with symbol, meaning, and default value (including physical units, "Temp." stands for temperature). Sea level temperature T_0 and rain-snow limit altitude $\mathcal{A}_{0\,precip}$ are highly variable depending on the input terrain and are set by the user.

7.6 Results and discussion

We have sourced mountainous DEM terrains from the U.S. Geological Survey, specifically from on or near: Steens Mountain, Oregon; Glacier Park, Montana; the Teton Range, Wyoming; and Mount Elbert, Colorado. Unless otherwise indicated, the terrain and simulations are sampled at 10 m cell resolution and have an extent of 10×10 km. In addition, we compare against real snow depth data provided by the NASA Airborne Snow Observatory for Tuolumne Basin, California and Conojes Basin, Colorado at 10 m sampling, and Bassies in the Pyrenees, France (Marti et al. 2016) at 2 m sampling.

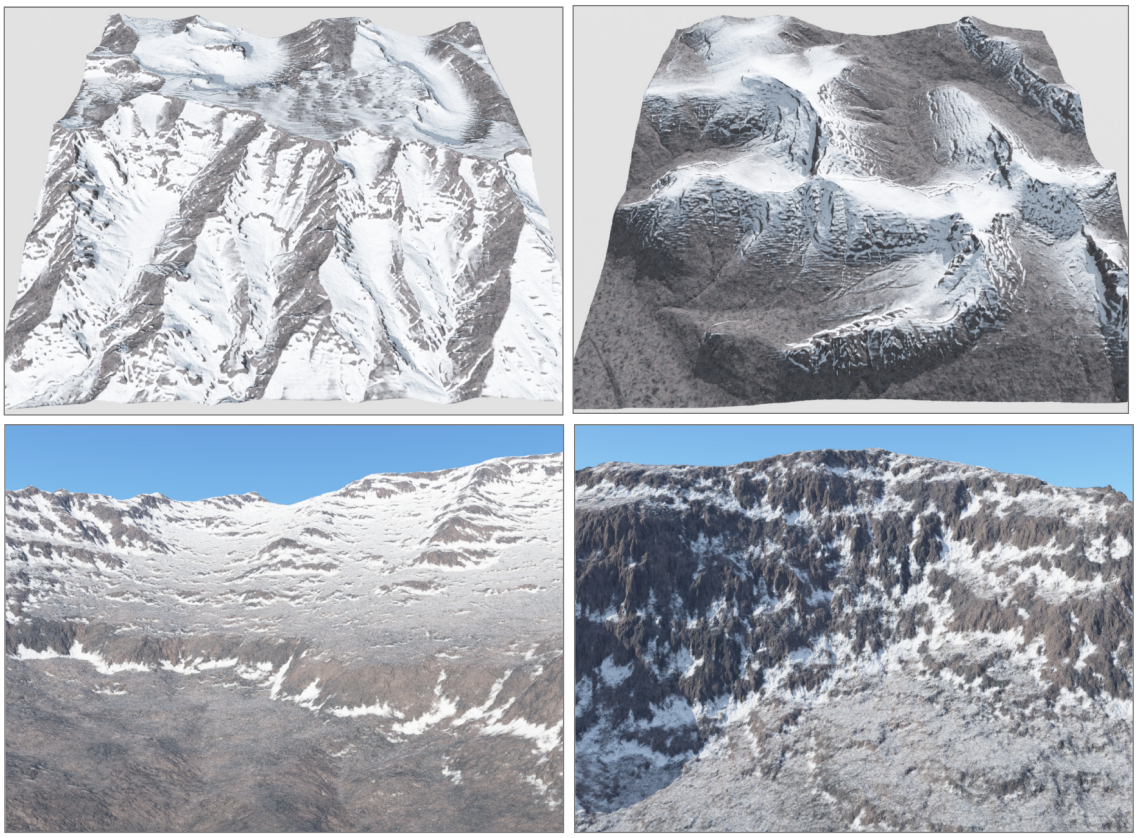


Figure 7.10: Various results, highlighting the effects of sun exposure, diffusion and slope.

Here we showcase a variety of outcomes that are visually consistent with underlying mountainous terrain. Figure 7.10 shows both far and near landscape renderings of (from left to right): a high-altitude region near Steens Mountain, with sun-exposed slopes; a sparser covering of an area from Glacier Park; a higher-resolution simulation of Steens Mountain, highlighting wind impact; and another subsampling showing snow collected below cliffs. The latter two simulations are at 2 m resolution over 2×2 km. Collectively, these results demonstrate snow deposition and melting due to snowfalls and state changes, and how diffusion and avalanches clear steep slopes.

Figure 7.11 shows summer, autumn, late winter, and spring melt stages for a landscape subjected to an evolving user-specified weather scenario. In autumn, snow settles at high altitudes on gentler sun-shadowed slopes. In the heart of winter, even steep slopes are covered

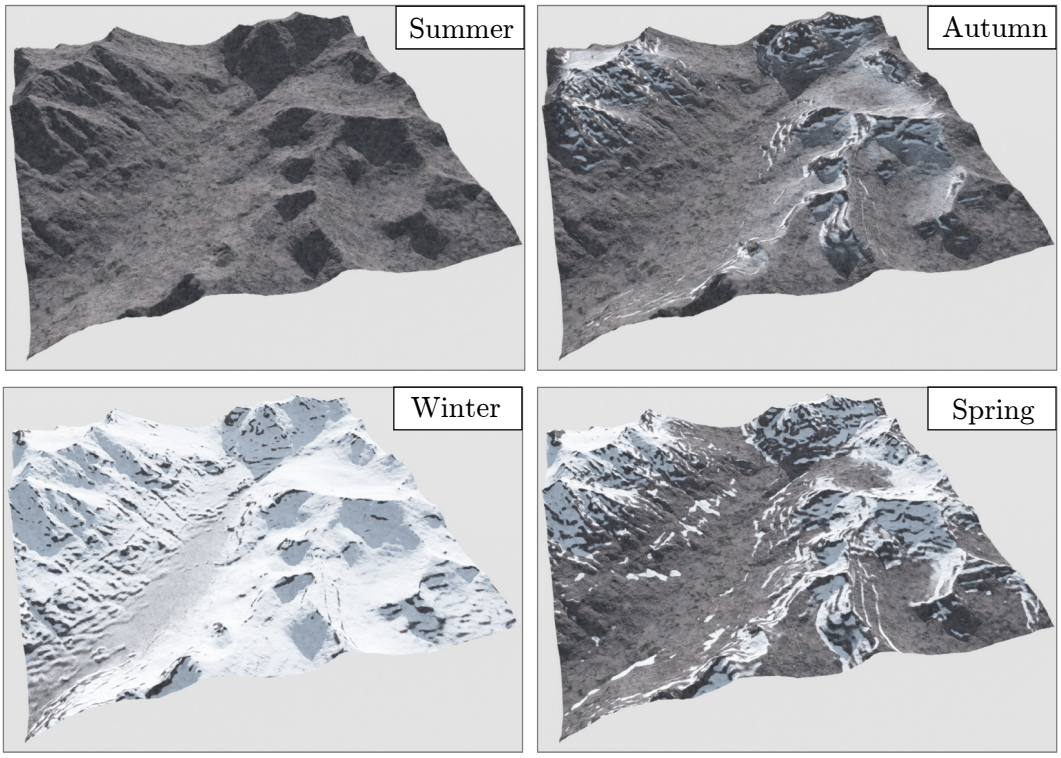


Figure 7.11: Dynamic landscape: bare rock in summer, the first autumn snows, snow in late winter, and melting in spring.

and cornices develop. In spring, lower, flatter areas melt first, with snow retained at high altitudes and for larger deposits.

The snow distribution resulting from an avalanche plays a role in the improved consistency of the landscape after a user modification. An example is given by Figure 7.12 where a user adds some snow (Figure 7.12, middle) on top of an automatically generated snow cover (Figure 7.12, top). An avalanche is triggered, cleans the slope from exceeding snow, and deposits the fallen snow in the bottommost gentle slopes (Figure 7.12, bottom).

Figure 7.13 shows different snow specific effects obtained from our simulation, such as ski tracks, avalanche outcome or cornices.

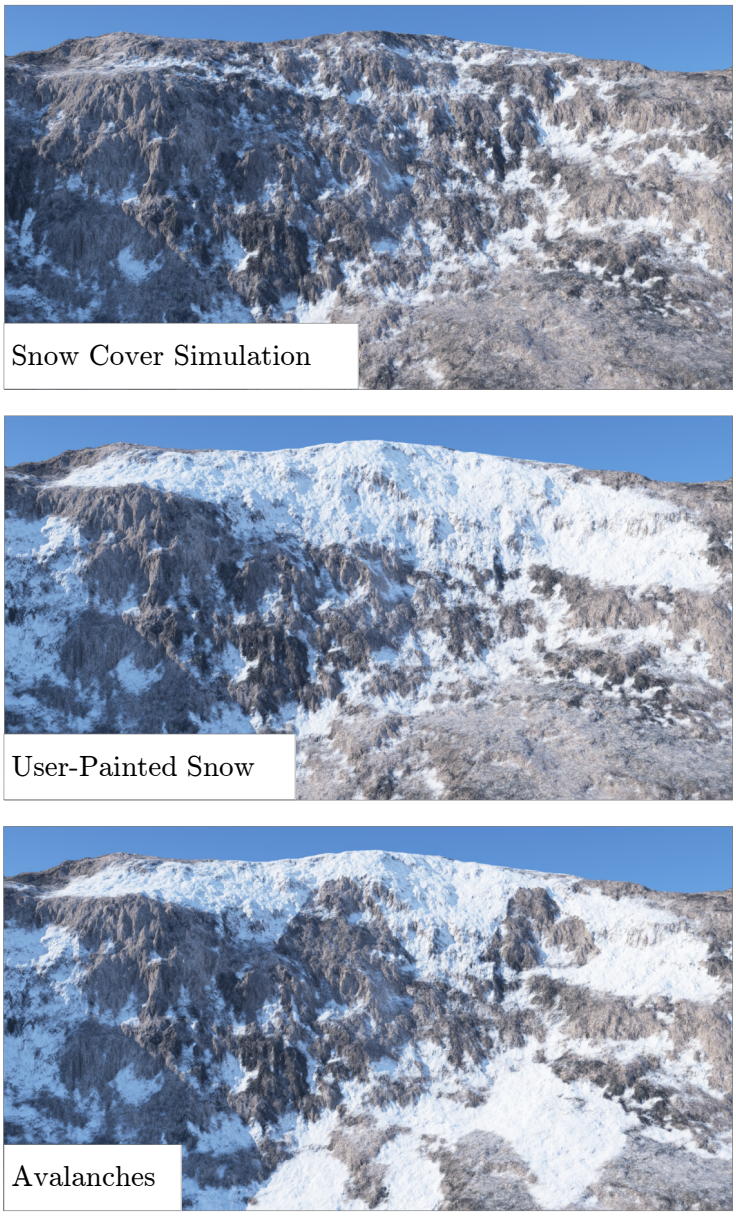


Figure 7.12: Snow-covered landscapes with triggered avalanches. The input scene (top) is enhanced with user-painted snow (middle) and avalanches are triggered, leading to a complete scene with snow beneath the mountains (bottom).

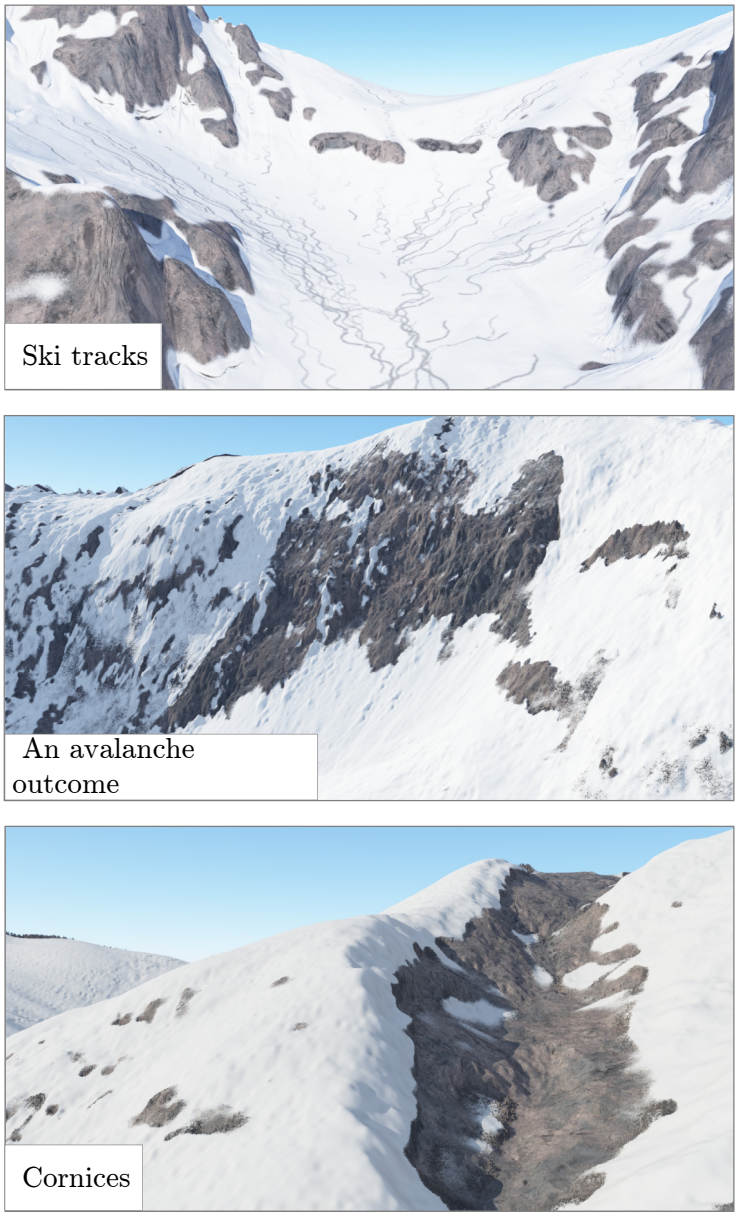


Figure 7.13: Specific effects simulated by our method.

Validation - perceptual user study. For validation purposes, the ideal would have been to compare our simulated snow coverage results against real snow coverage data on the same terrain topography. Unfortunately, to achieve such a match and set our simulation parameters would require significant additional information on terrain type, water bodies, moisture levels, ice coverage, prevailing winds, and human activity. Instead, we validated physical plausibility through a perceptual user study. We chose to compare results against both real snow data and a previous method, using a two-alternative forced choice (2-AFC) user experiment. For comparison to prior work, we selected the occlusion method of Foldes et al. 2007 since this method is intended for landscapes of a similar scale and requires comparable computation. Moreover, since we had no physical ski-track data or previous ski track models to compare against, we excluded ski-tracks from the user study.

Participants in the study were presented with nine pairs of images, and asked to select the most *natural* in each case. All images where rendered from synthetic or real data using the same render settings, with image pairs drawn from the set of {real data, our results, ambient occlusion} such that each two-element subset appeared three times. Care was also taken to use the same mountain range an equal number of times, with the data sources as indicated previously. The overall order of presentation, as well as the order within pairs, were randomized (some examples are show in Figure 7.14).

We had 57 participants between the ages of 21 and 78. Some had significant Computer Graphics exposure, or were accustomed to high altitude winter scenery. Our experiment shows that overall a particular generation method was selected on average (with standard deviation σ), as follows: real data 68.2% ($\sigma = 17\%$), our results 47.8% ($\sigma = 26\%$), and previous ambient occlusion 33.9% ($\sigma = 22\%$) of the time. This indicates that, although we improve markedly over previous work (in particular participants favored our results over previous work 67% of the time on average), our simulation is not yet indistinguishable from physical reality. This is likely due to certain secondary effects that we do not consider, such as trees, soil constituency, running water, and surface ice formation.

With a more extensive user study it would be possible to evaluate the visual impact of the individual components and parameters of our model. Although time consuming, such a study would be a useful extension for future work.

7.7 Conclusion

This chapter was dedicated to a new modeling method for dynamic, snow-covered landscapes. The stochastic simulation of Chapter 6 was extended with a Poisson-based scheduling of the diverse phenomena involved. By applying it to snow related natural processes, we capture complex physical interactions, such as snow melted by both direct and indirect sunlight, powdery snow shifted by the wind, and avalanches initiated in steep areas of unstable snow. Combined, all these events sculpt the landscape over time, leading to the formation of visually consistent features, such as overhangs of snow under ridges, piles of snow at the base of slopes cleared by avalanches, or ski-tracks consistently snaking down-slope. Our method also integrates a GPU implementation, that allows for interactive modeling that supports real-time phenomena. We validated our method with a user study and showed a variety of practical examples.

Further work would be needed to incorporate many secondary effects. In particular, trees,

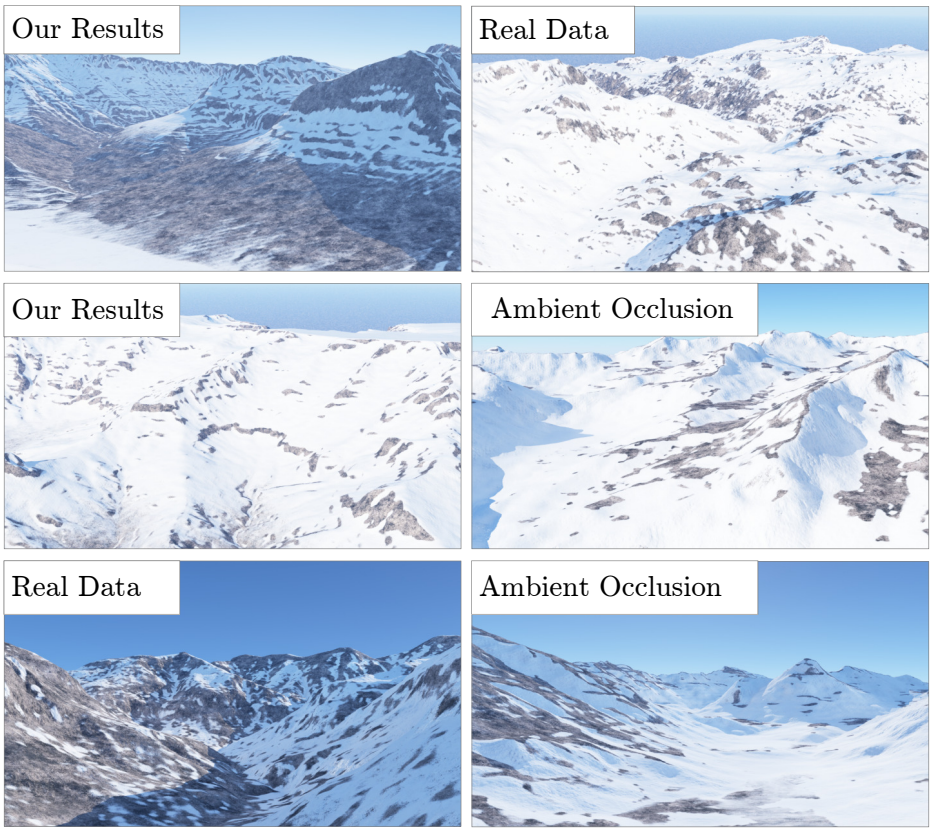


Figure 7.14: Validation by forced-choice comparison between real data, our results and an existing ambient occlusion-based techniques. Each row represents a typical pairing with randomized order as presented in our user study.

rocks, soft soil, grass, ice formation and running water are important for snow formation. The difference in the ranges of time steps between the simulations presented in this chapter and in Chapter 6 prevents all the events described in both chapters to be simulated together, not because of an incompatibility of time step - the simulation method being designed to handle this, but because the high frequency of snow related events would request either a very long simulation, either a simulation with no other events. A solution for this would be to provide multiple versions for an event showing its averaged effect at different time scales. The event version would be chosen with respect to a characteristic simulation time, which decreases when approaching present day.

Finally, our use of different landscape scales, with cell widths from 2 to 10 meters, demonstrates that the method is robust to scale changes between simulation scenarios. However, an interesting research direction would be to incorporate a pyramidal approach capable of handling both large scales and fine details within the same terrain.

Chapter 8

Conclusion

We introduced novel contributions to the generation of large-scale time-evolving landscapes. In the following, we will summarize these contributions and give an outline for future research.

8.1 Summary

In this thesis, we focused on the problem of simulating landscapes at two scales: the temporal and spatial extent of full mountain ranges, and a medium scale where details such as vegetation or seasonal snow cover are relevant. This targeted medium scale encompass landforms formed by phenomena at very different scales in terms of time and space: from large spatial scale (mountains) to medium scale landforms (valleys, bluffs, riverbanks, cliffs) and from large time scale (million of years for tectonics) to medium and small scale (a few seconds for avalanche simulation).

Large scale mountain formation We studied the formation of terrains at very large spatial and temporal scales. At these scales, we observed the formation of whole mountain ranges above tens of kilometers in extent, during millions of years. We first proposed a method to couple fluvial erosion modeled through the stream power law and tectonically driven uplift (Chapter 3). By ordering the nodes of the terrain by increasing altitude, we were able to use a linear time implicit scheme to compute the combined action of uplift and erosion. The efficiency and stability of this method enables fast computation using large time steps, allowing simple user control on the uplift through masks. Then, we improved the characterization of the uplift thanks to a geologically inspired simulation (Chapter 4), where the earth crust was successively considered as a viscous material and as a block of layered sedimentary sheets. The first representation was used to compute the overall shape of the mountain range and to define the internal compressive tectonic forces. From this, the second representation was considered to generate the folding of the crust responsible for the distribution of mountain and valleys. This method was used in conjunction with a sculpting tool enabling the user to interactively shape tectonic forces. The generated uplift was finally coupled with the previously introduced fluvial erosion method, instantaneously generating the mountain surface. My last contribution was to introduce novel erosion tools. In particular, we considered glacial erosion, which is a prominent factor of present day landscape features in

many mountain ranges. This erosion method benefits from a novel technique for solving non-linear steady state ice flow on steep mountains, which is coupled with a fully implicit scheme for erosion. This enables for the efficient modeling of glacier-carved valleys with interesting glacial features.

We solved the main challenges of large scale landscape generation identified in Chapter 1 with the following contributions. First, we specifically considered large scale landscapes, something that has been neglected for a long time in Computer Graphics except in some recent work (Génevaux et al. 2015). Second, we took inspiration from earth sciences in our models, improving both the visual plausibility and the physical consistency of the results. Lastly, we tried to alleviate one of the common drawback of simulations techniques by designing computationally efficients methods (implicit schemes allowing for large time steps but without inverting matrices), which allowed us to study new methods for user control.

Combining medium scale phenomena We narrowed down the scale of our study, to consider landscapes where various phenomena combine and mutually influence each other, resulting in complete landscapes with vegetation and terrain details. We designed a new stochastic approach to handle the interleaving of these phenomena, which typically exhibit a large variability of temporal scales (Chapter 6). This method was applied to the joint simulation of erosion and vegetation effects, considering many competitive phenomena such as hydrolic erosion, rock falls, material diffusion, vegetation life, fires and lightnings. Thanks to a timeline, the user can brush different material amount as well as locally trigger some phenomena, and the system automatically removes inconsistencies. We extended this method to handle dynamic events within a longer landscape evolution, enabling to interleave avalanches and ski tracks with the simulation of snow cover throughout the year. Both the efficiency and controllability of the method where improved with a GPU implementation.

After generating the terrain, this part plays an important role in the plausibility of the landscape itself. By combining phenomena at different scales, we bridged the gap between medium and large scales. We also captured the interactions between phenomena resulting in the complex organization of natural patterns, extending the plausibility of the generated landscapes. Lastly, we enabled user manipulation of the local material amounts and of the launched phenomena in a timeline, allowing for a control of the whole landscape evolution, while ensuring that inconsistencies possibly introduced by the user are smoothed out by the subsequent simulation.

8.2 Future work

Despite our contributions, modeling terrains and realistic landscapes still remains a challenging area of research with many directions worth investigating.

Erosion processes. We developed algorithms for the efficient simulation of fluvial and glacial erosion on large scale terrains. We also introduced hill-slope process and extended thermal erosion with debris flow from researches in Geology. Although we believe that these erosion techniques may be sufficient to model the main landforms in large-scale mountain ranges, additional studies are needed in their parametrization. For example, directional hill-slope may account for a dominant wind direction, and other formulations of the drainage area

need to be found for erosion happening on specific climatic conditions. For instance, isolated precipitations on desert can result in instant flood, with massive erosional strength because of the lack of vegetation. Rock type also matters, where our equation was mostly valid for granite-like rock. On the other hand, limestone like rocks are deeply dissolved by water, progressively forming large underground network of cave and tunnels. The impact on the surface is weak, except after the resurgence where deep canyons are carved. Furthermore, our implementation of fluvial erosion does not handle the transport and deposition of sediments. Modeling and simulating the sediments moving downstream and their deposition is of crucial importance to generate plains and valleys. Surprisingly, those landforms have seldom been modeled, maybe because they have less dramatic shapes than large mountain ranges. Still, it impacts the plausibility of the most accessible parts of the mountains, and are thus worth considering. Moreover, it is an explanation for the meandering of rivers (Kurowski 2012) that needs to be investigated. Lastly, some other erosional processes have smaller impact on mountain ranges but are worth studying, such as coastal erosion.

Uplift. Uplift is responsible for the main mountainous landforms. Faults have impact at many scale: they influence the distribution of valleys and mountains, they enforce some parts of river paths, they bring closer rocks of different types leading into discontinuities within the erosion patterns and vegetation types, and they have a fundamental role in the visual aspect of landslides. Furthermore, faults are not only characterized by a discontinuity in the uplift, but also by the horizontal components of the plate motion. This horizontally extended uplift could be added to our framework to generate more diverse large scale erosion patterns. Other related geological features can be considered, such as escarpments or volcanoes, to extend the range of reachable landscapes.

Multi-scale phenomena. One of our contributionss is the new simulation method presented in Chapter 6 to handle the simultaneous simulation of phenomena at various temporal scales. A large amount of future work can be built on top of it. A first possibility is to design new phenomena, such as wind forming dunes, or water flow, that would results in rivers and lakes. Another one is to extend the range of the possible scales by giving multiple versions for a given phenomenon, by integrating its effects at different temporal scales. A landscape could be formed by the subsequent versions, decreasing the temporal scale when approaching present day. The advantage would be a fast creation of the overall landscape altogether with a very precise modeling of the recent small scale features.

Control. Although we tried to take user control into account when designing our modeling and simulation tools, further work would be beneficial to improve user control. Multi grid techniques, in particular a coarse to fine version of the simulations would dramatically lower the time needed to wait for a result, resulting in easier control. Specific controls can be added, for example sketch to guide rivers of brushes to feed glaciers. Overall, inverse control tools would greatly extend the expressibility allowed by our models: users provide a rough sketch of their intents, and the system guesses the initial conditions for the simulation such that the result is as close as possible to the designed output. The result would then be both consistent, because obtained from a simulation with reasonable starting conditions, and close to the user's intent. Machine learning would be a good candidate to solve this difficult

problem thanks to its generative capacities, as shown by a promising work from Guérin et al. 2017.

Validation. One important open problem is to provide an evaluation mechanism for further comparing the results of landscape modeling techniques with real data. We proposed several options, from forced choice experiments to the recognition of typical landforms features. Although interesting for particular cases, these solutions do not generalize to a global approach that would allow for the classification of landscape modeling techniques. The main challenge in providing such a validation is that research in landscape modeling usually focuses on a limited number of features and it is hard to segment the targeted phenomena from real data to compare them with synthetic results.

Knowledge from Natural Sciences was of tremendous help to build generation tools for large scale time-evolving landscapes. Our approach was to approximate natural laws, so that they were both accurate at the considered scale and fast enough to compute, to leverage controllability. But the user experience is still weakened by the use of simulation methods, and could be improved by the use of machine learning techniques, especially their generative variants, both for generating the landscapes and for improving the control tools. Some specific challenges arise with the use of machine learning for landscapes, in particular when physical consistency is considered. Machine learning results are generally very convincing visually, but it is much harder to enforce physical knowledge such as proper flow routing. We envision two possibilities to solve this problem: either use learned parameters as an input of a simulation, or use a simulation step within the learning process to enforce physically plausible results. This problem is extended by our focus on time evolving landscapes, where the result should be consistent through time. Finally, although learning can be performed on the large amount of available data on surface elevation, such data severely lacks when subsurface is considered. An option is to use simulation techniques to generate training data. A first direction could be to simulate the distribution of faults within Earth crust and their effects on the neighboring rock layers. Results would be used to train a machine learning system on the correlation between faults and the above surface layer, enabling to predict faults from surface information.

Bibliography

Alsweis, Monssef and Oliver Deussen (2006). "Wang-tiles for the simulation and visualization of plant competition". In: *Computer Graphics International: Advances in Computer Graphics*. Springer, pp. 1–11 (page 24).

Alsweis, Monssef and Olivier Deussen (2005). "Modeling and visualization of symmetric and asymmetric plant competition". In: *Eurographics Workshop on Natural Phenomena*. Ed. by Pierre Poulin and Eric Galin. The Eurographics Association, pp. 83–88 (page 23).

Andújar, Carlos, Antoni Chica, MA Vico, S Moya, and Pere Brunet (2014). "Inexpensive reconstruction and rendering of realistic roadside landscapes". In: *Computer Graphics Forum* 33.6, pp. 101–117 (page 24).

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 (page 29).

Ariyan, Maryam and David Mould (2015). "Terrain synthesis using curve networks". In: *Proceedings of Graphics Interface*, pp. 9–16 (page 14).

Barnes, Richard (2016). "Parallel Priority-Flood depression filling for trillion cell digital elevation models on desktops or clusters". In: *Computers and Geosciences* 96, pp. 56–68 (page 39).

Barnes, Richard, Clarence Lehman, and David Mulla (2014). "Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models". In: *Computers and Geosciences* 62, pp. 117–127 (pages 39, 42, 48–50).

Beardall, Matthew, Mckay Farley, Darius Ouderkirk, Jeremy Smith, Michael Jones, and Parris K Egbert (2007). "Goblins by spheroidal weathering". In: *Eurographics Workshop on Natural Phenomena*, pp. 7–14 (pages 17, 18).

Beaumont, C, P Fullsack, and J Hamilton (1992). "Erosional control of active compressional orogens". In: *Thrust Tectonics*. New York: Chapman and Hall, pp. 1–18 (page 34).

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*, 10:1–10:9 (page 14).

Belhadj, Farès (2007). "Terrain modeling: a constrained fractal model". In: *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. ACM, pp. 197–204 (page 13).

Belhadj, Farès and Pierre Audibert (2005). "Modeling landscapes with ridges and rivers: bottom up approach". In: *Proceedings International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. ACM, pp. 447–450 (page 14).

Benes, Bedrich (2007). "Real-time erosion using shallow water simulation". In: *VRIPHYS*, pp. 43–50 (page 19).

Benes, Bedrich, Nathan Andrysco, and Ondřej Šťava (2009). "Interactive modeling of virtual ecosystems". In: *Proceedings of Eurographics Workshop on Natural Phenomena*. Munich, Germany, pp. 9–16 (page 21).

Beneš, Bedřich and Xabier Arriaga (2005). "Table mountains by virtual erosion". In: *Proceedings of the Eurographics Workshop on Natural Phenomena*, pp. 33–39 (page 19).

Benes, Bedrich and Enrique David Espinosa (May 2003). "Modeling virtual ecosystems with the proactive guidance of agents". In: *Proceedings of CASA*, pp. 126–131 (pages 23, 120).

Benes, Bedrich and Rafael Forsbach (2001). "Layered data representation for visual simulation of terrain erosion". In: *Proceedings of the Spring Conference on Computer Graphics*. Vol. 25(4). IEEE Computer Society, pp. 80–86 (pages 11, 19).

Beneš, Bedřich and Rafael Forsbach (2001). "Parallel implementation of terrain erosion applied to the surface of mars". In: *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. Camps Bay, South Africa: ACM, pp. 53–57 (page 19).

Beneš, Bedřich and Rafael Forsbach (2002). "Visual simulation of hydraulic erosion". In: *Journal of the World Society for Computer Graphics* 10.1-3, pp. 79–86 (pages 5, 18).

Benes, Bedrich, Václav Těšínskỳ, Jan Hornyš, and Sanjiv K Bhatia (2006). "Hydraulic erosion". In: *Computer Animation and Virtual Worlds* 17.2, pp. 99–108 (pages 5, 18).

Bernhardt, Adrien, André Maximo, Luiz Velho, Houssam Hnaidi, and Marie-Paule Cani (2011). "Real-time terrain modeling using CPU-GPU coupled computation". In: *Proceedings of the Conference on Graphics, Patterns and Images*. Maceió, Brazil: IEEE, pp. 64–71 (page 14).

Bindschadler, Robert (1983). "The importance of pressurized subglacial water in separation and sliding at the glacier bed". In: *Journal of Glaciology* 29.101, pp. 3–19 (page 87).

Biot, Maurice Anthony (1961). "Theory of folding of stratified viscoelastic media and its implications in tectonics and orogenesis". In: *Geological Society of America Bulletin* 72.11, pp. 1595–1620 (pages 67, 69).

Bloomenthal, Jules (1985). "Modeling the mighty maple". In: *Proceedings of SIGGRAPH* 19.3, pp. 305–311 (page 21).

Bornhofen, Stefan and Claude Lattaud (2009). "Competition and evolution in virtual plant communities: a new modeling approach". In: *Natural Computing* 8.2, pp. 349–385 (page 22).

Boubekeur, Tamy and Marc Alexa (Dec. 2008). "Phong tessellation". In: *ACM Transactions on Graphics* 27.5, 141:1–141:5 (page 45).

Bradbury, Gwyneth A, Kartic Subr, Charalampos Koniaris, Kenny Mitchell, and Tim Weyrich (Nov. 2015). "Guided ecological simulation for artistic editing of plant distributions in nat-

ural scenes". In: *Journal of Computer Graphics Techniques* 4.4, pp. 28–53 (pages 5, 23, 25, 120).

Braun, Jean and Malcolm Sambridge (1994). "Dynamical Lagrangian Remeshing (DLR): a new algorithm for solving large strain deformation problems and its application to fault-propagation folding". In: *Earth and Planetary Science Letters* 124.1, pp. 211–220 (page 76).

Braun, Jean and Malcolm Sambridge (1997). "Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization". In: *Basin Research* 9.1, pp. 27–52 (pages 95, 116, 118).

Braun, Jean and Sean Willett (2013). "A very efficient O (n), implicit and parallel method to solve the stream power equation governing fluvial incision and landscape evolution". In: *Geomorphology* 180, pp. 170–179 (pages 34, 39, 48, 58, 61).

Braun, Jean and Philippe Yamato (2010). "Structural evolution of a three-dimensional, finite-width crustal wedge". In: *Tectonophysics* 484, pp. 181–192 (page 58).

Braun, Jean, Dan Zwartz, and Jonathan H. Tomkin (1999). "A new surface-processes model combining glacial and fluvial erosion". In: *Annals of Glaciology* 28, pp. 282–290 (pages 85–87).

Bruneton, Éric and Fabrice Neyret (2008). "Real-time rendering and editing of vector-based terrains". In: *Computer Graphics Forum* 27.2, pp. 311–320 (page 14).

Cade, Brian, James Terrell, and Richard Schroeder (1999). "Estimating effects of limiting factors with regression quantiles". In: *Ecology* 80.1, pp. 311–323 (page 121).

Carpentier, Giliam de and Rafael Bidarra (2009). "Interactive GPU-based procedural height-field brushes". In: *Proceedings of the International Conference on Foundations of Digital Games*. ACM, pp. 55–62 (page 13).

Castelltort, Sébastien, Liran Goren, Sean D Willett, Jean-Daniel Champagnac, Frédéric Herman, and Jean Braun (2012). "River drainage patterns in the New Zealand Alps primarily controlled by plate tectonic strain". In: *Nature Geoscience* 5, pp. 744–748 (pages 65, 81).

Ch'Ng, Eugene (July 2011). "Realistic placement of plants for virtual environments". In: *IEEE Computer Graphics and Applications* 31.4, pp. 66–77 (page 23).

Ch'Ng, Eugene (May 2013). "Model resolution in complex systems simulation: Agent preferences, behavior, dynamics and n-tiered networks". In: *Simulation* 89.5, pp. 635–639 (page 120).

Chiba, Norishige, Kazunobu Muraoka, and Kunihiko 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, pp. 185–194 (page 19).

Cordonnier, Guillaume, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Eric Guérin (2016). "Large scale terrain generation from tectonic uplift and fluvial erosion". In: *Computer Graphics Forum* 35.2, pp. 165–175 (pages 7, 34, 35, 41, 42, 48).

Cordonnier, Guillaume, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin (2018). "Sculpting mountains: Interactive terrain modeling based on subsurface geology". In: *IEEE Transactions on Visualization and Computer Graphics* 24.5, pp. 1756–1769 (pages 7, 58).

Cordonnier, Guillaume, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani (2018). "Interactive generation of time-evolving, snow-covered landscapes with avalanches". In: *Computer Graphics Forum* 37.2, pp. 497–509 (pages 7, 134).

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, pp. 134:1–134:12 (page 7).

Croissant, Thomas and Jean Braun (2014). "Constraining the stream power law: a novel approach combining a landscape evolution model and an inversion method". In: *EGU General Assembly Conference Abstracts*. Vol. 16, p. 10657 (page 36).

Dagenais, François, Jonathan Gagnon, and Eric Paquette (2016). "An efficient layered simulation workflow for snow imprints". In: *The Visual Computer* 32.6-8, pp. 881–890 (page 26).

De Carli, Daniel Michelon, Cesar Tadeu Pozzer, Victor Schetinger, and Fernando Bevilacqua (2014). "Procedural generation of 3D canyons". In: *Proceedings of the Conference on Graphics, Patterns and Images*. Rio de Janeiro, Brazil: IEEE, pp. 103–110 (page 15).

Densmore, Alexander L, Michael A Ellis, and Robert S Anderson (1998). "Landsliding and the evolution of normal-fault-bounded mountains". In: *Journal of geophysical research: solid earth* 103.B7, pp. 15203–15219 (page 117).

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 (page 14).

Deussen, Oliver, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz (1998). "Realistic modeling and rendering of plant ecosystems". In: *Proceedings of SIGGRAPH*. ACM, pp. 275–286 (pages 23, 24, 120).

Deussen, Oliver and Bernd Lintermann (2006). *Digital design of nature: computer generated plants and organics*. Springer Science & Business Media (page 21).

Dewaele, Guillaume and Marie-Paule Cani (2004). "Interactive global and local deformations for virtual clay". In: *Graphical Models* 66.6, pp. 352–369 (pages 59, 64).

Doran, Jonathon and Ian Parberry (2010). "Controlled procedural terrain generation using software agents". In: *Transactions on Computational Intelligence and AI in Games* 2.2, pp. 111–119 (page 16).

Dorsey, Julie, Alen Edelman, Henrik Wann Jensen, and Hans Kohling Pedersen (1999). "Modeling and rendering of weathered stone". In: *Proceedings of SIGGRAPH*. Vol. 25(4). ACM, pp. 225–234 (page 18).

Ebert, David S., F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley (2002). *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc. (pages 4, 13, 97).

Egholm, David L, Mads F Knudsen, Chris D Clark, and Jerome E Lesemann (2011). "Modeling the flow of glaciers in steep terrains: The integrated second-order shallow ice approximation (iSOSIA)". In: *Journal of Geophysical Research: Earth Surface* 116.F2 (pages 85, 86).

Ellis, Susan (1996). "Forces driving continental collision: reconciling indentation and mantle subduction tectonics". In: *Geology* 24.4, pp. 699–702 (pages 59, 60, 64).

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 (pages 4, 15, 28).

Emilien, Arnaud, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes (July 2015). "WorldBrush: Interactive example-based synthesis of procedural virtual worlds". In: *ACM Transactions on Graphics* 34.4, 106:1–106:11 (pages 5, 24, 28).

Étienne, Jocelyn, Pierre Saramito, and Emil J Hopfinger (2004). "Numerical simulations of dense clouds on steep slopes: application to powder-snow avalanches". In: *Annals of Glaciology* 38.1, pp. 379–383 (page 27).

Fearing, Paul (2000). "Computer modelling of fallen snow". In: *Proceedings of SIGGRAPH*, pp. 37–46 (page 25).

Fernandez, Naiara and Boris Kaus (2014). "Fold interaction and wavelength selection in 3D models of multilayer detachment folding". In: *Tectonophysics* 632, pp. 199–217 (pages 76, 77).

Festenberg, Niels V. and Stefan Gumhold (2009). "A geometric algorithm for snow distribution in virtual scenes". In: *Proceedings of the Eurographics Conference on Natural Phenomena*, pp. 17–25 (page 27).

Festenberg, Niels and Stefan Gumhold (2011). "Diffusion-based snow cover generation". In: *Computer Graphics Forum* 30.6, pp. 1837–1849 (page 27).

Foldes, David and Bedrich Benes (2007). "Occlusion-based snow accumulation simulation". In: *VRIPHYS*, pp. 35–41 (pages 27, 139, 156).

Foley, Jonathan, Colin Prentice, Navin Ramankutty, Samuel Levis, David Pollard, Steven Sitch, and Alex Haxeltine (1996). "An integrated biosphere model of land surface processes, terrestrial carbon balance, and vegetation dynamics". In: *Global Biogeochemical Cycles* 10.4, pp. 603–628 (pages 22, 120).

Fournier, Alain, Don Fussell, and Loren Carpenter (1982). "Computer rendering of stochastic models". In: *Communications of the ACM* 25.6, pp. 371–384 (page 13).

Frade, Miguel Monteiro de Sousa, Francisco Fernàndez de Vega, and Carlos Cotta (2010). "Evolution of artificial terrains for video games based on accessibility". In: *Applications of evolutionary computation* (page 16).

Gain, James, Harry Long, Guillaume Cordonnier, and Marie-Paule Cani (2017). "EcoBrush: Interactive control of visually consistent large-scale ecosystems". In: *Computer Graphics Forum* 36.2, pp. 63–73 (pages 7, 23, 24, 120, 140).

Gain, James, Bruce Merry, and Patrick Marais (2015). "Parallel, realistic and controllable terrain synthesis". In: *Computer Graphics Forum* 34.2, pp. 105–116 (pages 5, 15, 16).

Galyean, Tinsley A. and John F. Hughes (July 1991). "Sculpting: An interactive volumetric modeling technique". In: *Computer Graphics (Proceedings of SIGGRAPH 91)*. Vol. 25, pp. 267–274 (page 59).

Gamito, Manuel N. and Kenton Forest Musgrave (2001). "Procedural landscapes with overhangs". In: *Proceedings of the Portuguese Computer Graphics Meeting*. Lisbon, Portugal (page 12).

Gaume, J, T Gast, J Teran, A van Herwijnen, and C Jiang (2018). "Dynamic anticrack propagation in snow". In: *Nature communications* 9.1, p. 3047 (pages 27, 28).

Génevaux, Jean-David, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedrich Benes (2013). "Terrain generation using procedural models based on hydrology". In: *ACM Transactions on Graphics* 32.4, 143:1–143:13 (pages 4, 14, 46).

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 (pages 4, 12, 35, 38, 46, 52, 160).

Glen, John W (1955). "The creep of polycrystalline ice". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 228.1175, pp. 519–538 (page 86).

Grosbellet, François, Adrien Peytavie, Éric Guérin, Éric Galin, Stéphane Mérillou, and Bedrich Benes (Feb. 2016). "Environmental objects for authoring procedural scenes". In: *Computer Graphics Forum* 35.1, pp. 296–308 (page 29).

Grunwald, Sabine (2016). *Environmental soil-landscape modeling: Geographic information technologies and pedometrics*. CRC Press (page 111).

Guérin, Éric, Julie Digne, Éric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez (Nov. 2017). "Interactive example-based terrain authoring with conditional generative adversarial networks". In: *ACM Transactions on Graphics* 36.6, 228:1–228:13 (pages 5, 16, 162).

Guérin, Eric, Julie Digne, Adrien Peytavie, and Eric Galin (2016). "Sparse representation of terrains for procedural modeling". In: *Computer Graphics Forum* 35.2, pp. 177–187 (pages 12, 16, 35, 46).

Guérin, Eric, Eric Galin, François Grosbellet, Adrien Peytavie, and Jean-David Geneveaux (2016). "Efficient modeling of entangled details for natural scenes". en. In: *Computer Graphics Forum* 35.7, pp. 257–267 (pages 29, 130).

Hädrich, Torsten, Bedrich Benes, Oliver Deussen, and Sören Pirk (2017). "Interactive modeling and authoring of climbing plants". In: *Computer Graphics Forum* 36.2, pp. 49–61 (page 21).

Harbor, Jonathan M, Bernard Hallet, and Charles F Raymond (1988). "A numerical model of landform development by glacial erosion". In: *Nature* 333.6171, p. 347 (page 85).

Headley, Rachel M., Gerard Roe, and Bernard Hallet (2012). "Glacier longitudinal profiles in regions of active uplift". In: *Earth and Planetary Science Letters* 317-318, pp. 354–362 (pages 86, 87).

Herman, Frédéric, Olivier Beyssac, Mattia Brughelli, Stuart N Lane, Sébastien Leprince, Thierry Adatte, Jiao YY Lin, Jean-Philippe Avouac, and Simon C Cox (2015). "Erosion by an Alpine glacier". In: *Science* 350.6257, pp. 193–195 (pages 86, 87).

Higgins, Steven I, William J Bond, Edmund C February, Andries Bronn, Douglas IW Euston-Brown, Beukes Enslin, Navashni Govender, Louise Rademan, Sean O'Regan, Andre LF Potgieter, et al. (2007). "Effects of four decades of fire manipulation on woody vegetation structure in savanna". In: *Ecology* 88.5, pp. 1119–1125 (page 122).

Hinks, Tommy and Ken Museth (2009). "Wind-driven snow buildup using a level set approach". In: *Eurographics Ireland Workshop Series*, pp. 19–26 (pages 25, 26).

Hnaidi, Houssam, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin (2010). "Feature based terrain generation using diffusion equation". In: *Computer Graphics Forum* 29.7, pp. 2179–2186 (pages 4, 14).

Howard, Alan D (1994). "A detachment-limited model of drainage basin evolution". In: *Water resources research* 30.7, pp. 2261–2285 (page 36).

Hudák, Matej and Roman Ďurikovič (2011). "Terrain models for mass movement erosion". In: *Theory and Practice of Computer Graphics* (page 19).

Hurtut, Thomas, Pierre-Edouard Landes, Joëlle Thollot, Yann Gousseau, Remy Drouillhet, and Jean-François Coeurjolly (2009). "Appearance-guided synthesis of element arrangements by example". In: *Proceedings of the International Symposium on Non-photorealistic Animation and Rendering*. ACM, pp. 51–60 (page 25).

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 (pages 5, 20).

Jákó, Balázs and Balázs Tóth (2011). "Fast hydraulic and thermal erosion on the GPU". In: *Proceedings of the Central European Seminar on Computer Graphics*. Viničné, Slovakia (page 19).

Jenson, Susan and Julia Domingue (1988). "Extracting topographic structure from digital elevation data for geographic information system analysis". In: *Photogrammetric Engineering and Remote Sensing* 54, pp. 1593–1600 (page 39).

Kamal, Raiyan and Yusuf Sarwar Uddin (2007). "Parametrically controlled terrain generation". In: *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. Perth, Australia: ACM, pp. 17–23 (page 13).

Kaus, Boris and Stefan Schmalholz (2006). "3D finite amplitude folding: implications for stress evolution during crustal and lithospheric deformation". In: *Geophysical Research Letters* 33.14 (page 79).

Kelley, Alex, Michael Malin, and Gregory Nielson (1988). "Terrain simulation using a model of stream erosion". In: *ACM Transactions on Graphics*, pp. 263–268 (page 14).

Knap, Wouter, Johannes Oerlemans, and Martin Cabée (1996). "Climate sensitivity of the ice cap of king George Island, South Shetland Islands, Antarctica". In: *Annals of Glaciology* 23, pp. 154–159 (page 86).

Knight, Jasper and Stefan Grab (2014). "Lightning as a geomorphic agent on mountain summits: Evidence from southern Africa". In: *Geomorphology* 204, pp. 61–70 (page 117).

Krištof, Peter, Bedrich Benes, Jaroslav Křivánek, and Ondřej Šťava (Mar. 2009). "Hydraulic erosion using smoothed particle hydrodynamics". In: *Computer Graphics Forum* 28.2, pp. 219–228 (pages 5, 19).

Kurowski, Michał (2012). "Procedural generation of meandering rivers inspired by erosion". In: *Journal of the World Society for Computer Graphics*, pp. 79–86 (pages 5, 19, 161).

Lagae, Ares, Lefebvre Syvalin, Robert L. Cook, Tony DeRose, Georges Drettakis, David S. Ebert, J. P. Lewis, Ken Perlin, and Zwicker M (2010). "A survey of procedural noise functions". In: *Computer Graphics Forum* 29.8, pp. 2579–2600 (page 13).

Lague, Dimitri (2014a). "The stream power river incision model: evidence, theory and beyond". In: *Earth Surface Processes and Landforms* 39.1, pp. 38–61 (page 17).

Lague, Dimitri (2014b). "The stream power river incision model: evidence, theory and beyond". In: *Earth Surface Processes and Landforms* 39.1, pp. 38–61 (page 36).

Laine, Samuli and Tero Karras (2011). "Efficient sparse voxel octrees". In: *IEEE Transactions on Visualization and Computer Graphics* 17.8, pp. 1048–1059 (page 12).

Lane, Brendan and Przemyslaw Prusinkiewicz (May 2002). "Generating spatial distributions for multilevel models of plant communities". In: *Proceedings of Graphics Interface*. Calgary, Alberta, Canada, pp. 69–80 (pages 5, 23, 24, 120).

Law, Richard, Janine Illian, David Burslem, Georg Gratzer, C. Gunatilleke, and I. Gunatilleke (2009). "Ecological information from spatial patterns of plants: insights from point process theory". In: *Journal of Ecology* 97.4, pp. 616–628 (page 22).

Lawick, Joost van Pabst van and Hans Jense (1995). "Dynamic terrain generation based on multifractal techniques". In: *High Performance Computing for Computer Graphics and Visualisation*, pp. 186–203 (page 13).

Lehning, Michael, H Löwe, M Ryser, and N Raderschall (2008). "Inhomogeneous precipitation distribution and snow transport in steep terrain". In: *Water Resources Research* 44.7 (page 143).

Lewis, John (1987). "Generalized stochastic subdivision". In: *ACM Transactions on Graphics* 6.3, pp. 167–190 (page 13).

Lindenmayer, Aristid (1968). "Mathematical models for cellular interaction in development". In: *Journal of Theoretical Biology* Parts I and II.18, pp. 280–315 (page 21).

Lindsay, John (2016). "Efficient hybrid breaching-filling sink removal methods for flow path enforcement in digital elevation models". In: *Hydrological Processes* 30.6, pp. 846–857 (pages 39, 42).

Lorensen, William and Harvey Cline (1987). "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM SIGGRAPH computer graphics*. Vol. 21. 4. ACM, pp. 163–169 (page 11).

Mahaffy, MW (1976). "A three-dimensional numerical model of ice sheets: Tests on the Barnes Ice Cap, Northwest Territories". In: *Journal of Geophysical Research* 81.6, pp. 1059–1066 (page 86).

Mandelbrot, Benoit and Roberto Pignoni (1983). *The Fractal Geometry of Nature*. San Francisco: W.H. Freeman and Company (page 13).

Mandelbrot, Benoit and John Van Ness (1968). "Fractional Brownian motions, fractional noises and applications". In: *SIAM review* 10.4, pp. 422–437 (page 13).

Maréchal, Nicolas, Eric Guérin, Eric Galin, Stéphane Mérillou, and Nicolas Mérillou (2010). "Heat transfer simulation for modeling realistic winter sceneries". In: *Computer Graphics Forum* 29.2, pp. 449–458 (pages 26, 139, 143).

Mareš, Martin (2002). *Two linear time algorithms for MST on minor closed graph classes*. ETHZ, Institute for Mathematical Research (pages 41, 48, 50).

Marti, Renaud, Simon Gascoin, Etienne Berthier, M. de Pinel, T. Houet, and D. Laffly (2016). "Mapping snow depth in open alpine terrain from stereo satellite imagery". In: *The Cryosphere* 10.4, pp. 1361–1380 (page 152).

Masselot, Alexandre and Bastien Chopard (1995). "Cellular automata modeling of snow transport by wind". In: *International Workshop on Applied Parallel Computing*, pp. 429–435 (page 26).

McClay, Ken (1992). *Thrust Tectonics*. Springer, Dordrecht (page 17).

McClung, David and Peter Schaerer (2006). *The Avalanche Handbook*. The Mountaineers (page 146).

Měch, Radomír and Przemyslaw Prusinkiewicz (1996). "Visual models of plants interacting with their environment". In: *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, pp. 397–410 (page 21).

Mei, Xing, Philippe Decaudin, and Bao-Gang Hu (2007). "Fast hydraulic erosion simulation and visualization on GPU". In: *Pacific Graphics*. IEEE Computer Society, pp. 47–56 (page 19).

Mérillou, Stéphane and Djamchid Ghazanfarpour (2008). "A survey of aging and weathering phenomena in computer graphics". In: *Computers & Graphics* 32.2, pp. 159–174 (page 17).

Michel, Elie, Arnaud Emilien, and Marie-Paule Cani (2015). "Generation of folded terrains from simple vector maps". In: *Eurographics 2015 short paper proceedings*. Zurich, Switzerland, p. 4 (page 20).

Miller, Gavin SP (1986). "The definition and rendering of terrain maps". In: *ACM SIGGRAPH Computer Graphics*. Vol. 20. 4. ACM, pp. 39–48 (page 13).

Milliez, Antoine, Michael Wand, Marie-Paule Cani, and Hans-Peter Seidel (2013). "Mutable elastic models for sculpting structured shapes". In: *Computer Graphics Forum* 32.2pt1, pp. 21–30 (page 59).

Moeslund, Claus, Thomas anvisud Madsen, Michael Aagaard, and Dennis Lerche (2005). "Modeling falling and accumulating snow". In: *Vision, Video and Graphics* (page 26).

Montgomery, David R (2002). "Valley formation by fluvial and glacial erosion". In: *Geology* 30.11, pp. 1047–1050 (page 17).

Moriya, Tomoaki and Tokiichiro Takahashi (2010). "A real time computer model for wind-driven fallen snow". In: *ACM SIGGRAPH ASIA 2010 Sketches*, 26:1–26:2 (page 27).

Muraoka, Kazunobu and Norishige Chiba (2000). "Visual simulation of snowfall, snow cover and snowmelt". In: *Proceedings of the Parallel and Distributed Systems: Workshops* (page 26).

Musgrave, Kenton, Craig Kolb, and Robert Mace (1989). "The synthesis and rendering of eroded fractal terrains". In: *ACM SIGGRAPH Computer Graphics* 23.3, pp. 41–50 (pages 5, 18, 19, 44, 53, 70, 95, 116).

Nagashima, Kenji (1998). "Computer generation of eroded valley and mountain terrains". In: *The Visual Computer* 13.9-10, pp. 456–464 (page 18).

Narain, Rahul, Armin Samii, and James O'Brien (2012). "Adaptive anisotropic remeshing for cloth simulation". In: *ACM Transactions on Graphics* 31.6, p. 152 (page 66).

Natali, Mattia, EM Lidal, J Parulek, I Viola, and D Patel (2013). "Modeling terrains and subsurface geology". In: *Proceedings of Eurographics State of the Art Reports*, pp. 155–173 (page 10).

Neidhold, Benjamin, Markus Wacker, and Oliver Deussen (2005). "Interactive physically based fluid and erosion simulation". In: *Eurographics Workshop on Natural Phenomena*, pp. 25–33 (page 19).

Nishita, Tomoyuki, Hiroshi Iwasaki, Yoshinori Dobashi, and Eihachiro Nakamae (1997). "A modeling and rendering method for qnow by using metaballs". In: *Computer Graphics Forum* 16.3, pp. C357–C364 (page 25).

Nye, John Frederick (1957). "The distribution of stress and velocity in glaciers and ice-sheets". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 239.1216, pp. 113–133 (page 86).

O'Brien, James F and Jessica K Hodgins (1995). "Dynamic simulation of splashing fluids". In: *Computer Animation*. IEEE, pp. 198–205 (page 146).

Passos, Vladimir Alves dos and Takeo Igarashi (2013). "LandSketch: A first person point-of-view example-based terrain modeling approach". In: *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. Anaheim, USA: ACL, pp. 61–68 (page 15).

Peucker, Thomas, Robert Fowler, James Little, and David Mark (1978). "The triangulated irregular network". In: *Proceedings of Digital Terrain Models Symposium*. St. Louis, USA: American Society of Photogrammetry (page 11).

Peytavie, Adrien, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou (2009). "Arches: A framework for modeling complex terrains". In: *Computer Graphics Forum* 28.2, pp. 457–467 (page 12).

Premože, Simon, William Thompson, and Peter Shirley (1999). "Geospecific rendering of Alpine terrain". In: *Eurographics Conference on Rendering*. Granada, Spain, pp. 107–118 (page 27).

Prentice, Colin, Martin Sykes, and Wolfgang Cramer (1993). "A simulation model for the transient effects of climate change on forest landscapes". In: *Ecological modelling* 65.1, pp. 51–70 (pages 22, 121).

Prusinkiewicz, Przemyslaw and Mark Hammel (1993). "A fractal model of mountains with rivers". In: *Proceedings of Graphics Interface*. Vol. 30(4), pp. 174–180 (page 14).

Prusinkiewicz, Przemyslaw and Aristid Lindenmayer (2012). *The algorithmic beauty of plants*. Springer Science & Business Media (page 21).

Prusinkiewicz, Przemyslaw, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane (2001). "The use of positional information in the modeling of plants". In: *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*. ACM, pp. 289–300 (page 21).

Pytel, Alex and Stephen Mann (2013). "Self-organized approach to modeling hydraulic erosion features". In: *Computers & Graphics* 37.4, pp. 280–292 (page 5).

Rastello, Marie and EJ Hopfinger (2004). "Sediment-entraining suspension clouds: a model of powder-snow avalanches". In: *Journal of fluid mechanics* 509, pp. 181–206 (page 27).

Rémillard, Olivier and Paul Kry (2013). "Embedded thin shells for wrinkle simulation". In: *ACM Transactions on Graphics* 32.4 (page 66).

Reynolds, Daniel Tobias, Stephen D Laycock, and A.M. Day (2015). "Real-time accumulation of occlusion-based snow". In: *The Visual Computer* 31.5, pp. 689–700 (page 27).

Ritschel, Tobias, Thorsten Grosch, and Hans-Peter Seidel (2009). "Approximating dynamic global illumination in image space". In: *Proceedings of the Interactive 3D graphics and games*. ACM, pp. 75–82 (page 139).

Rohmer, Damien, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer (2010). "Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles". In: *ACM Transactions on Graphics*. Vol. 29. ACM, p. 157 (pages 66, 67).

Roudier, Pascale, Bernard Peroche, and Michel Perrin (1993). "Landscapes synthesis achieved through erosion and deposition process simulation". In: *Computer Graphics Forum* 12.3, pp. 375–383 (page 20).

Rusnell, Brennan, David Mould, and Mark G. Eramian (2009). "Feature-rich distance-based terrain synthesis". In: *The Visual Computer* 25.5-7, pp. 573–579 (page 14).

Sai-Keung, Wong and Fu I-Ting (2015). "Hybrid-based snow simulation and snow rendering with shell textures". In: *Computer Animation and Virtual Worlds* 26.3-4, pp. 413–421 (page 26).

Saltvik, Ingar, Anne Elster, and Henrik Nagel (2007). "Parallel methods for real-time visualization of snow". In: *Proceedings of the International Workshop on Applied Parallel Computing*, pp. 218–227 (page 26).

Sato, Hisashi, Akihiko Itoh, and Takashi Kohyama (2007). "SEIB–DGVM: A new Dynamic Global Vegetation Model using a spatially explicit individual-based approach". In: *Ecological Modelling* 200.3–4, pp. 279–307 (pages 22, 120).

Savage, JC and WSB Paterson (1963). "Borehole measurements in the Athabasca Glacier". In: *Journal of Geophysical Research* 68.15, pp. 4521–4536 (page 86).

Schmalholz, Stefan M. and Yuri Yu Podladchikov (2000). "Finite amplitude folding: Transition from exponential to layer length controlled growth". In: *Earth and Planetary Science Letters* 179, pp. 363–377 (pages 66, 79).

Schneider, Jens, Tobias Boldte, and Rüdiger Westermann (2006). "Real-time editing, synthesis, and rendering of infinite landscapes on GPUs". In: *Proceedings of Vision, Modeling, and Visualization*. Aachen, Germany: IOS Press, p. 145 (page 13).

Shifley, Stephen R, Hong S He, Heike Lischke, Wen J Wang, Wenchi Jin, Eric J Gustafson, Jonathan R Thompson, Frank R Thompson, William D Dijak, and Jian Yang (2017). "The past and future of modeling forest dynamics: from growth and yield curves to forest landscape models". In: *Landscape Ecology* 32.7, pp. 1307–1325 (page 22).

Sitch, Stephen, Chris Huntingford, N. Gedney, P. E. Levy, M. Lomas, S. L. Piao, R. Betts, P. Ciais, P. Cox, P. Friedlingstein, C. D. Jones, I. C. Prentice, and F. I. Woodward (2008). "Evaluation of the terrestrial carbon cycle, future plant geography and climate-carbon cycle feedbacks using five Dynamic Global Vegetation Models (DGVMs)". In: *Global Change Biology* 14.9, pp. 2015–2039 (page 22).

Sitch, Stephen, Benjamin Smith, I Colin Prentice, Almut Arneth, A Bondeau, W Cramer, JO Kaplan, Samuel Levis, W Lucht, M Thonicke Sykes, et al. (2003). "Evaluation of ecosystem dynamics, plant geography and terrestrial carbon cycling in the LPJ dynamic global vegetation model". In: *Global Change Biology* 9.2, pp. 161–185 (pages 120, 122).

Smelik, Ruben M., Tim Tutenel, Rafael Bidarra, and Bedrich Benes (2014). "A survey on procedural modelling for virtual worlds". In: *Computer Graphics Forum* 33.6, pp. 31–50 (page 10).

Soler, Cyril, François X. Sillion, Frédéric Blaise, and Philippe De Reffye (2001). *A physiological Plant Growth Simulation Engine Based on Accurate Radiant Energy Transfer*. Research Report RR-4116. INRIA (page 21).

Sorkine, Olga and Marc Alexa (2007). "As-rigid-as-possible surface modeling". In: *Proceedings of the Symposium on Geometry processing*, pp. 109–116 (page 59).

Šťava, Ondřej, Bedrich Benes, Matthew Brisbin, and Jaroslav Křivánek (2008). "Interactive terrain modeling using hydraulic erosion". In: *Proceedings of the Symposium on Computer Animation*. Eurographics Association, pp. 201–210 (pages 19, 146, 147, 149).

Stanculescu, Lucian, Raphaëlle Chaine, Marie-Paule Cani, and Karan Singh (2013). "Sculpting multi-dimensional nested structures". In: *Computer and Graphics* 37.6, pp. 753–763 (page 59).

Sternai, Pietro, Frédéric Herman, Pierre G Valla, and Jean-Daniel Champagnac (2013). "Spatial and temporal variations of glacial erosion in the Rhône valley (Swiss Alps): Insights from numerical modeling". In: *Earth and Planetary Science Letters* 368, pp. 119–131 (page 86).

Stock, J and William E Dietrich (2003). "Valley incision by debris flows: Evidence of a topographic signature". In: *Water Resources Research* 39.4 (page 95).

Stomakhin, Alexey, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle (2013). "A material point method for snow simulation". In: *ACM Transactions on Graphics* 32.4, 102:1–102:10 (pages 26, 28).

Svensson, Harald (1959). "Is the cross-section of a glacial valley a parabola?" In: *Journal of Glaciology* 3.25, pp. 362–363 (page 98).

Talgorn, François-Xavier and Farès Belhadj (2018). "Real-time sketch-based terrain generation". In: *Proceedings of Computer Graphics International 2018*. New York, NY, USA: ACM, pp. 13–18 (page 13).

Tasse, Flora Ponjou, Arnaud Emilien, Marie-Paule Cani, Stefanie Hahmann, and Neil Dodgson (2014). "Feature-based terrain editing from complex sketches". In: *Computers & Graphics* 45, pp. 101–115 (page 15).

Tasse, Flora Ponjou, James E. Gain, and Patrick Marais (2012). "Enhanced texture-based terrain synthesis on graphics hardware". In: *Computer Graphics Forum* 31.6, pp. 1959–1972 (page 16).

Tokoi, Kohe (2006). "A shadow buffer technique for simulating snow-covered shapes". In: *International Conference on Computer Graphics, Imaging and Visualisation*, pp. 310–316 (page 27).

Tsuda, Yusuke, Yonghao Yue, Yoshinori Dobashi, and Tomoyuki Nishita (2010). "Visual simulation of mixed-motion avalanches with interactions between snow layers". In: *The Visual Computer* 26.6, pp. 883–891 (page 27).

Tucker, Gregory E and Gregory R Hancock (2010). "Modelling landscape evolution". In: *Earth Surface Processes and Landforms* 35.1, pp. 28–50 (page 17).

Tychonievich, Luther A. and Mike D. Jones (2010). "Delaunay deformable mesh for the weathering and erosion of 3D terrain". In: *The Visual Computer* 26.12, pp. 1485–1495 (page 18).

Vanek, Juraj, Bedrich Benes, Adam Herout, and Ondrej Stava (2011). "Large-scale physics-based terrain editing using adaptive tiles on the GPU". In: *IEEE Computer Graphics and Applications* 31.6, pp. 35–44 (page 19).

Voss, Richard F. (1991). "Random fractal forgeries". In: *Fundamental Algorithms for Computer Graphics*. Vol. 17. Springer, pp. 805–835 (page 13).

Wachspress, Eugene L and GJ Habetler (1960). "An alternating-direction-implicit iteration technique". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2, pp. 403–423 (page 95).

Wang, Changbo, Zhangye Wang, Tian Xia, and Qunsheng Peng (May 2006). "Real-time snowing simulation". In: *The Visual Computer* 22.5, pp. 315–323 (page 26).

Wei, Hongqiang, Guiyun Zhou, and Suhua Fu (2018). "Efficient Priority-Flood depression filling in raster digital elevation models". In: *International Journal of Digital Earth* 0.0, pp. 1–13 (pages 39, 48–50).

Whipple, Kelin X and Gregory E Tucker (1999). "Dynamics of the stream-power river incision model: Implications for height limits of mountain ranges, landscape response timescales, and research needs". In: *Journal of Geophysical Research: Solid Earth (1978–2012)* 104.B8, pp. 17661–17674 (pages 34, 36).

Willett, Sean, Christopher Beaumont, and Philippe Fullsack (1993). "Mechanical model for the tectonics of doubly vergent compressional orogens". In: *Geology* 21.4, pp. 371–374 (pages 35, 58, 60).

Wither, Jamie, Frédéric Boudon, Marie-Paule Cani, and Christophe Godin (Apr. 2009). "Structure from silhouettes: a new paradigm for fast sketch-based design of trees". In: *Computer Graphics Forum* 28.2, pp. 541–550 (page 21).

Wojtan, Christopher, Mark Carlson, Peter J. Mucha, and Greg Turk (2007). "Animating corrosion and erosion". In: *Eurographics Workshop on Natural Phenomena*, pp. 15–22 (page 18).

Wullschleger, Stan D, Howard E Epstein, Elgene O Box, Eugénie S Euskirchen, Santonu Goswami, Colleen M Iversen, Jens Kattge, Richard J Norby, Peter M van Bodegom, and Xiaofeng Xu (2014). "Plant functional types in Earth system models: past experiences and future directions for application of dynamic vegetation models in high-latitude ecosystems". In: *Annals of botany* 114.1, pp. 1–16 (page 22).

Wyvill, Geoff, Craig McPheeters, and Brian Wyvill (1986). "Soft objects". In: *Advanced Computer Graphics*. Springer, pp. 113–128 (page 11).

Yamato, Philippe, Boris JP Kaus, Frédéric Mouthereau, and Sébastien Castelltort (2011). "Dynamic constraints on the crustal-scale rheology of the Zagros fold belt, Iran". In: *Geology* 39.9, pp. 815–818 (page 66).

Yassemi, Shahram, Suzana Dragićević, and Margaret Schmidt (2008). "Design and implementation of an integrated GIS-based cellular automata model to characterize forest fire behaviour". In: *ecological modelling* 210.1, pp. 71–84 (page 119).

Zhou, Guiyun, Xiaoli Liu, Suhua Fu, and Zhongxuan Sun (2017). "Parallel identification and filling of depressions in raster digital elevation models". In: *International Journal of Geographical Information Science* 31.6, pp. 1061–1078 (page 39).

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 (page 15).