THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En informatique

École doctorale I2S

Unité de recherche LIRMM

Génération procédurale d'environnements sous-marins

Présentée par Marc HARTLEY
le [XX mois année]

Sous la direction de Christophe FIORIO, Noura FARAJ et Karen GODARY-DEJEAN

Devant le jury composé de

[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]
[Prénom NOM, Titre, Affiliation]	[Statut jury]

UNIVERSITÉ DE MONTPELLIER

Contents

Remerciements

- ...

Abstract

- ...

Résumé long

- ...

CHAPTER 1

Introduction

Contents

Back to summary

- ...

1.1 Procedural Generation

- ...

1.1.1 Definition

- ...

Official Definition

- ...

My Definition

- ...

1.1.2 History

- ...

1.1.3 Models Represented

- ...

Noise

- ...

Cellular Automata

- ...

Neural Networks

- ...

Physical Phenomena Modeling

- ...

1.1.4 User Interaction

- ...

Realism-Speed-Control Balance

- Main issue in terrain generation
- Explanation of realism
** Generated landscapes are close to what is found in reality
** Generation incorporates natural processes to be realistic
** Requires extensive physical simulations, expert knowledge
** Useful in applications such as natural disaster simulations
- Explanation of speed
** Fastest possible generation, aiming for real-time generation
** J. Gain classifies: real-time (< 30ms), interactive (< 3s), near-interactive (< 5min), and long-term.
** Useful in "infinity-scroll" video games, for example
- Explanation of control
** Aims to meet user demands
** Major issue being "impossible" user demands
** Useful in most procedural generation applications: speeding up artists' work, for example
- ...

Regeneration

- Manual actions
- Issues with regeneration
** What to regenerate?
** How to regenerate?
** Problems with user interactions?
- Action storage file in JSON (?)
- ...

1.2 Terrain Representation

- ...

1.2.1 2.5D Terrains

- ...

Height Maps

- ...

Height Functions

- ...

1.2.2 3D Terrains

- Need for 3D concepts
** Geological information
** Volumetric data
- ...

Main Issues

- Memory
- Visualization
- Modifications
- Conversion between representations
** Information loss
*** Error propagation on geometry (approximations on normals, Z resolution, surface, etc.)
*** Loss of subsurface information
- ...

Types, Definitions, Advantages, Disadvantages

- Voxel grids
- Material stacks
- Meshes
- Implicit surfaces
- ...

1.2.3 Other Models

- Concept of semantics
- ...

1.2.4 Underwater Landscapes

- 3D Data
** Coral landscapes filled with voids
** Many cavities (caves, grottos, karst networks)
- Interdisciplinary Data
** Rather common with terrain generation
** => Geological validation with experts
** For underwater,
*** Fewer experts,
*** More uncertainties
*** Based more on observations
*** Few data (coral landscapes < 0.1** Mix of geology, biology, hydrology, and physics (especially fluid dynamics)
- Need for multi-scale
** Not limited to underwater
** Integrate large elements (mountains) with small elements (vegetation)
** LOD
- ...

Fluid Simulations

- Very important in procedural terrain generation
- Allows justifying the geophysics of a simulation/generation
- Quite fast solutions in 2D (PIC, FLIP, Stable Fluids, SPH, etc.)
- But becomes much heavier and memory-intensive in 3D
- ...

1.3 Coral Reefs (Biological Aspects)

- Historical discovery of coral reefs
- Islands, barriers, atolls
- Atoll theories
- Importance in biodiversity
- Threats, protection, importance of understanding them
- ...

1.4 Geometry and Data Structures

- Presentation of used structures
- ...

1.4.1 Geometry

- ...

Points

- Defined in 3D space as $(x, y, z)^T$.
- When projected in 2D, $z = 0$ is implicit.
- Represented in the manuscript as: $\mathbf{p} \in \mathbb{R}^3$
- ...

Curves

- Parametric function $C : [0, 1] \rightarrow \mathbb{R}^3$.
- Unless otherwise specified, use of Centripetal Catmull–Rom spline [CITE CATMULL 1974]:
** Let $\mathbf{p}_i$ denote a point. For a curve segment C defined by points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and knot sequence t_0, t_1, t_2, t_3, the centripetal Catmull-Rom spline can be produced by:

$$C(t) = \frac{t_2 - t}{t_2 - t_1} B_1 + \frac{t - t_1}{t_2 - t_1} B_2 \tag{1.1}$$

where

$$B_1(t) = \frac{t_2 - t}{t_2 - t_0} A_1(t) + \frac{t - t_0}{t_2 - t_0} A_2(t) \tag{1.2}$$

$$B_2(t) = \frac{t_3 - t}{t_3 - t_1} A_2(t) + \frac{t - t_1}{t_3 - t_1} A_3(t) \tag{1.3}$$

$$A_1(t) = \frac{t_1 - t}{t_1 - t_0} \mathbf{p}_0 + \frac{t - t_0}{t_1 - t_0} \mathbf{p}_1 \tag{1.4}$$

$$A_2(t) = \frac{t_2 - t}{t_2 - t_1} \mathbf{p}_1 + \frac{t - t_1}{t_2 - t_1} \mathbf{p}_2 \tag{1.5}$$

$$A_3(t) = \frac{t_3 - t}{t_3 - t_2} \mathbf{p}_2 + \frac{t - t_2}{t_3 - t_2} \mathbf{p}_3 \tag{1.6}$$

and

$$t_{i+1} = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}^{\alpha} + t_i \tag{1.7}$$

where α ranges from 0 to 1 for knot parameterization, and $i = 0, 1, 2, 3$ with $t_0 = 0$. For centripetal Catmull-Rom spline, the value of α is 0.5. When $\alpha = 0$, the resulting curve is the standard uniform Catmull-Rom spline; when $\alpha = 1$, the result is a chordal Catmull-Rom spline.
** We will keep $\alpha = 0.5$ for all the work in this manuscript, as it felt like a good compromise between smoothness and control on the curve. The value of α has not been studied deeply.
- Advantages: Centripetal Catmull-Rom spline has several desirable mathematical properties compared to the original and other types of Catmull-Rom formulations. First, it will not form loops or self-intersections within a curve segment. Second, cusps will never occur within a curve segment. Third, it follows the control points more tightly. [COPY PASTE WIKIPEDIA]
- Additionally, we do not use "handles" (invisible control points) like for Bézier curves. At the cost of a little user control, I feel the use is simplified.
- The calculation of first and second derivatives (tangent and normal) is quick.
- ...

1.4.2 Data Structures

- ...

3D Grids

- In this manuscript, all 3D grids are defined as signed float32.
- Not optimal, especially for representing binary voxels (uses 32x more memory and computation than necessary), but flexible...
- Voxel grids stored as lists of sub-grids ("local modifications") to navigate undo-redo. Cell evaluation by summing sub-grids.
- ...

1.5 Prototype Creation

- C++23 and Qt5.12
- OpenGL 4.6 and GLSL

- Marching Cubes on geometry shader
** Bad idea, but justify why
- Renderings:
** With the prototype:
*** Marching Cubes on geometry shader
*** Triplanar texture
*** Real-time results
*** Textures based on materials
** With Unreal Engine 5:
*** Static meshes
*** Added procedural vegetation with plugin [PLUGIN NAME] and ocean with [PLUGIN NAME]
** With Blender 4.1:
*** Static meshes
*** Easier script usage
- ...

1.6 Contributions and Plan

- Chronological order of terrain generation
- Proposes an abstract representation halfway between computing and terrain expertise
** Offering a generalization of the desired landscape type (underwater, but also terrestrial)
- Proposes new types of landscapes (karsts and coral islands)
** Maintaining the notion of sparseness (implicit volumes)
- Proposes a particle-based erosion simulation method
** Terrain representation agnostic
** Lightweight, fast, easy to implement
- Aims to keep maximum control for the user
** In the generation process, but also to correct details upstream

1.6.1 Semantics

- Work oriented towards underwater generation
- Collaboration with a marine biologist
-

1.6.2 Modeling

- Generation of some landscape elements still new (karsts referring to Axel Paris, but coral islands new)
- Karst networks represented in a highly user-friendly manner
** Viewing karsts as a directed acyclic graph => close to tree structure
** Enhanced method for fractal generation with cycles (generation in multiple iterations)
- Coral islands using an interpretation of Darwin's theory
** Based on observations
** Based on travel journals

1.6.3 Amplification

- Increasing realism by adding details
- Particle-based erosion method
** Generalization for flexibility
** Speed, parallelization
- Towards a continuous erosion method.

Part I

Semantic Representation

Abstract

Simulating underwater landscape growth is complex due to the interplay of biological, environmental, physical, geological, and human factors. Our method addresses this complexity by introducing a sparse representation of the terrain elements using environmental objects, defined as parametric models based on point, curve, or region skeletons, which interact locally to spawn, grow, and die. This approach removes the need for complex interconnections, enabling a parallelizable and scalable generation process. By using fitting functions to incorporate biological rules and focusing on geological plausibility, our method enables interactive underwater landscape simulations. The main contributions of this work are the introduction of sparse terrain element representation, the use of fitting functions to simulate biological processes without complex physics, and the development of an interactive simulation framework based on geological events.

CHAPTER 2

Generation de terrain sémantique

Contents

Back to summary

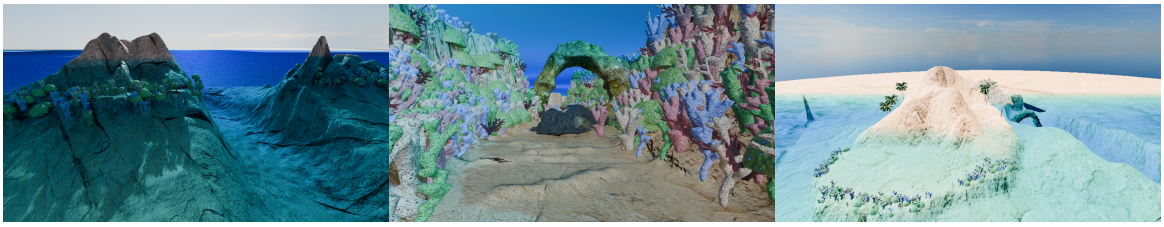


Figure 2.1: Our method can produce different scenes including coral islands and canyons at multi-scale using environmental objects to represent terrain features.

2.1 Introduction

Automated terrain generation is a key component of natural scene digital modeling for animated movies and video games. Many landscapes have been studied and are synthesised with more and more realism. Different processes can be used and combined to achieve these scenes: fractal terrains (Musgrave et al., 1989; Prusinkiewicz and Hammel, 1993), erosion simulation (Cordonnier et al., 2023; Mei et al., 2007), manual modeling (de Carpentier and Bidarra, 2009; E. Guérin et al., 2022), geological simulation (Cordonnier et al., 2017a; Cortial et al., 2019), ... The high quality of synthesis for such environment is due to the possibility to observe these environments from many point of views: long-distance gazing, hiking on mountains, remote sensing, aerial imaging, ... Thanks to the quality of the digital modeling, the entertainment industry display often breathtaking land scenes.

Underwater scenes are rarely created in these media for multiple reasons: these environments are not completely understood and mastered as much as land environments because they are difficult to access, we lack the capacity to see them at a larger scale (unlike mountains for example) and the underlying process that forms these landscapes are much more complex to simulate.

These limitations cause animated movies and video games studios to avoid as much as possible underwater environments.

However, these environments are important for the study of biology, geology, and, by extension, robotics. Due to the complexity, limitations and danger of underwater human operations, underwater robots are more and more used for marine environment monitoring(Dunbabin et al., 2020; Maslin et al., 2021; Palmer et al., 2021; Williams et al., 2016). Yet the validation process of such underwater robot is expensive, with heavy logistic, and it is often impossible to find the appropriated environment to test the system. Thus, underwater robotics requires simulation capacities, to be able to test the robot's algorithms in very specific environment and conditions. But for now, roboticians are lacking the capacity to test on realistic virtual scenes, and so only test them on synthetic scenarios that do not correlate with real world terrains. For that, they need to manipulate more precisely the characteristics of the underwater environment, at different scales.

The difficulties to visualize and study the underwater environments on a large scale at the same time as a small scale is an obstacle to the procedural generation and simulation of scenes that are coherent in these two different scales. A solution to this problem could be a bottom-up approach, simulating at the smallest scale the behaviour of all the elements of the environment. Computing such a simulation in order to generate an entire ecosystem is an near-impossible task due to time and memory complexity.

The method we propose provides a way to procedurally generate environments on multiple scales without introducing such complexity by using a sparse representation of the environment using environmental objects. The environmental objects only have access to local values of the environments to spawn, grow and die. The use of local interaction with environment values removes the need for complex interconnections between all elements of the terrain, providing a parallelisable generation process at large to small scales. Defining environmental objects of the terrain as parametric models based on point, curve or region skeletons provides a lightweight representation of the terrain that the user can interact with. The interaction with the simulation process is still a difficult task (Smelik et al., 2014). Our method does not aim for a visually realistic generation, but for a plausible terrain depending on geological and biological constraints, guided by the user. Using state of the art modeling of the geometry of the environmental objects of the terrain could achieve realistic results. We illustrate the method through the generation of coral islands and reefs.

Our main contribution are the introduction of a sparse representation of the terrain elements as environmental objects, the use of fitting functions to incorporate biological rules in the simulation process avoiding physic simulation and an interactive simulation based on geological events for underwater landscapes.

2.2 Related works

Procedural terrain generation has been heavily studied for the last 40 years (Galin et al., 2019). Researches in this topic try to find new solutions to compromise between realism, user control and efficiency (Gain et al., 2009). Using fractal noise parametrized to resemble real landscape has been an important first step (Musgrave et al., 1989) as it's a fast and light solution to generate procedurally the appearance of mountains. The lack of user control pushed newer works toward the use of controlled noise by including real DEM in the process through learning (Brosz et al., 2007; Kapp et al., 2020), while the rise of deep learning technologies gave higher control to the user through sketches (É. Guérin et al., 2017; Talgorn and Belhadj, 2018).
By including expert knowledge of tectonic process and subsurface geology, some algorithms tend to get more realistic (Cortial et al., 2019; Michel et al., 2015; Patel et al., 2021).
While these algorithms are able to generate large-scale landscapes, the finer details of the terrain is often computed by the use of erosion simulation (Cordonnier et al., 2023; Paris et al., 2019b; Schott et al., 2023). This process can be expensive in time but results in more plausible surfaces.

All the algorithms aim to reproduce plausible relief in terrestrial landscapes, mostly limited to alpine landscapes, but a lack of research can be found in almost all other biomes. Underwater landscapes generation, for example, has been almost completely absent from literature for many reasons: the difficulty of accessing the area, the lack of visibility under water and the complex physics of underwater geology and biology make the algorithms adapted for this environment scarce.
The majority of the ocean floor can be represented as a fractal terrain (Mareschal, 1989). While stochastic noise can be sufficient to model the ocean floor, this process won't cover areas with the biggest biomass, near shallower waters such as near coasts and islands.
Due to the impossibility to observe the large-scale and the small-scale of underwater environments, some works related to geology model large structures like the profile shape of the coral reef (BOSSCHER and SCHLAGER, 1992), simulate its surface growth (Li, 2021), or use procedural algorithms for single polyp (Abela et al., 2015). We however don't have a mix of the different scales, and neither methods take into account the environment such as the topography or the interaction of different terrain elements. This is mainly due to the fact that the evolution time for each scale varies from a span of weeks to thousands of years.

In an ecosystem, any element of the system has an impact on their surrounding. Simulating each physical properties such as shading, heat, humidity may require enormous computation power. By considering these properties as scalar fields surrounding the whole scene, and that elements of the terrain affect locally the scalar fields, we can simplify the computation of the physical properties of the environment (Grosbellet et al., 2016; É. Guérin et al., 2016). This process provides a scalable system from which scene details are rendered in a plausible way. In a similar way, other works represent the wind flow as a composition of local vector fields (Wejchert and Haumann, 1991), avoiding complex fluid simulation while providing user control in a lightweight model. We extend these works by incorporating a time-evolution system such that the scene can be dynamic.

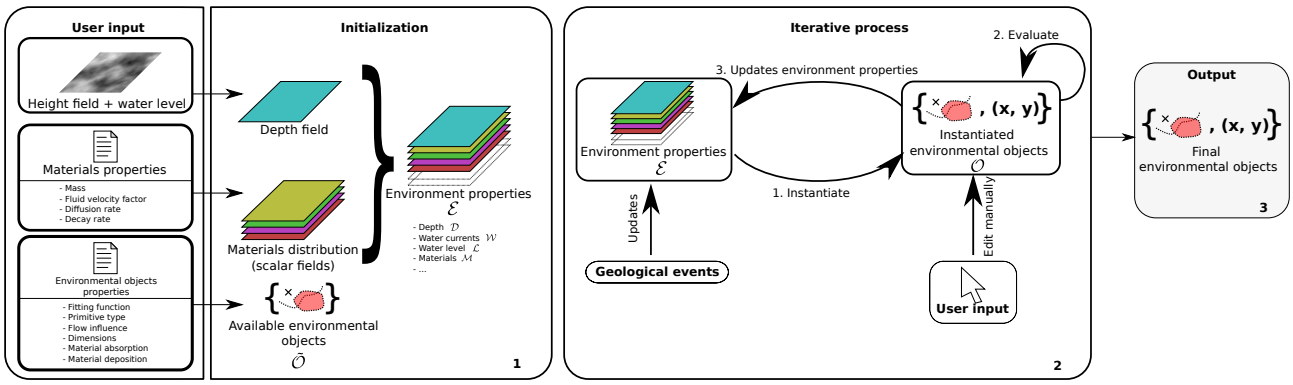


Figure 2.2: Overview of the pipeline of the method. The user provides as input an initial height field and sets the water level, as well as a definition of the materials properties and environmental objects properties that will be used in the iterative process. These inputs are initialized as an initial set of environmental objects and scalar fields that represents the environment values. In the iterative loop, new environmental objects are instantiated using the current state of the environment at their optimal position. The existing environmental objects in the terrain reevaluate their fitting function to grow or die and update the environment values locally. At each iteration, geological events can update the environment values, while the user can interact directly with the environmental objects. The result of the whole process is a set of environmental objects which is a sparse representation of the elements of the scene.

2.3 Method

The overall pipeline of the method is based on simple incremental generation like most rule-based systems. In this type of system, the final state is defined either by reaching equilibrium, or by verifying specific conditions, such as a maximum number of iterations. We define our pipeline in three phases (Figure 2.2): the initialization phase that describe the generation and simulation rules, the iterative phase generating populating the terrain with our environmental objects and finally the output.

2.3.1 Pipeline overview

The generation of the terrain is initialized using an initial height field $\mathcal{H}$ and a water level $\mathcal{L}$. The height field provides variation on the depth $\mathcal{D}$, which can influence the generation process of the scene. We set $\mathcal{D} = \mathcal{H} - \mathcal{L}$.
The list of available environmental objects $\tilde{\mathcal{O}}$, representing the different elements that can be present in the scene, are provided with their properties: type, size, generation rules, growing conditions and effects on the environment values (Section 2.3.2).
Finally, different materials can be defined with their properties such as diffusion speed, mass, damping factor and influence from the water currents. Materials distributions are represented as a scalar field $\mathcal{M} : \mathbb{R}^2 \to \mathbb{R}$ and water currents as a vector field $\mathcal{W} : \mathbb{R}^2 \to \mathbb{R}^2$ that can be evaluated by the environmental objects of the scene to simulate their growth and spawn at the most probable position. The environmental properties $\mathcal{E} = (\mathcal{D}, \mathcal{W}, \mathcal{L}, \mathcal{M})$ is composed of depth, water currents, water level and materials distribution information at any point of the terrain (Section 2.3.3).
The definition of environmental objects' properties and environment properties is done with field experts, providing the pertinent parameters required to model the evolution of the

terrain elements using expert knowledge (Section 2.4). Additional properties can easily be added to the environment properties $\mathcal{E}$ in order to fit to the experts needs, such as atmospheric pressure, humidity, temperature, ...
The generation phase can optionally be started with an initial set of environmental objects present in the scene.

Once the initialization phase is done, the generation begins. The generation process is incremental and its main loop is composed of two different steps: the instantiation of new environmental objects then the update of the environment. At each iteration, new environmental objects can be created at their most fitting locations if possible. The generation rules provided in the initialization phase are used to find the optimal position from stochastic sampling (Section 2.3.2). All environmental objects are evaluating their state analytically using the fitting function provided as input (Section 2.3.2). Once the instantiation step is done, the environment's values are updated by each environmental object by deposing and absorbing some of the available materials (Section 2.3.3) while modifying the water currents (Section 2.3.3) around them. Finally, water currents and height field's gradient displace materials of the terrain at each iteration. During the generation process, the user can alter directly the distribution and shapes of the environmental objects (Section 2.3.4) and perturb the generation process by planning geological events that have impacts on the environment values (Section 2.3.4).

The output of our system is a set of environmental objects disposed in the plane. We do not provide the 3D representation of the environmental objects, letting the user define the rendering method. The figures used in the paper use a mix of implicit surfaces and triangular meshes.

2.3.2 Environmental objects

Environmental objects are rule-based objects following rules depending on their local environment for evaluation their state in their life cycle. We can see them as a life form in the way that they are created and eroded with time. During their lifetime, they influence their local environment by depositing and absorbing materials around them and influencing the water currents. The environment objects are described spatially as a single point, a parametric curve or a region.

We consider that all environmental objects follow a life cycle of spawning, growing and dying. While many environmental objects of a terrain is not a living being, we assume that evolution of relief, for example, starts at one point in time, grow as the geological factors force it to and is eroded until a point where this environmental object can not be distinguished from the rest of the environment.

Environmental objects are spawn stochastically in the terrain at the optimal fitting position. This position is determined from a generation rule given by the user for each of the environmental objects, which is dependant on the environment state. Once the environmental object is present in the scene, it will continuously evaluate its fitting function to determine its state in the life cycle. If the evaluation results as less than zero, the environmental object dies and it is removed from the list of environmental objects present in the scene. While the environmental object remains, it will continue influencing its environment, by absorbing and depositing material around it and by influencing the water currents.

Generation rules

Generation rules provides, for each environmental object, a fitting function Γ_{obj} defining the most probable location for an environmental object to spawn. Fitting functions' parameters contains, for every point $\mathbf{p}$, the environmental values $\mathcal{E}_p$ (the amount of each material available $\mathcal{M}(\mathbf{p})$ and the velocity and direction of the water currents $\mathcal{W}(\mathbf{p})$) and information about surrounding environmental objects $\mathcal{O}$ (signed distance from the closest punctual environmental object or curve defining curve- and region-based environmental objects, curvature of the curve, and start and end points of the curve-based environmental objects).

The seed point of a spawning environmental object is defined by a stochastic sampling of the plane. We propose different optimization means to find the optimal fitting position, depending on the environmental object shape.

The spawning position of a punctual environmental object is found at the local maxima of the fitting function from a seed point. The optimisation process simply follows the field's gradient $\nabla\Gamma_{obj}$ until the local maxima is reached.

A region is defined as an isocontour of the field for which the target area A is found. From the seed point, we follow the isolevel of the fitting function $\nabla\Gamma^{\perp}_{obj}$ until a loop is created to define the initial condition of the shape. Using the Active Contours algorithm, we can optimize the region's energy defined as $E = E_{\text{internal}} + E_{\text{shape}}$ with

$$E_{\text{internal}} = \frac{1}{2}\left(\alpha(t)\left\|\frac{dv}{dt}(t)\right\|^2 + \beta(t)\left\|\frac{d^2v}{dt^2}(t)\right\|^2\right). \tag{2.1}$$

The internal energy E_{internal} force the shape compact while the shape energy E_{shape} force the shape into a specific target. For the environmental objects generated in the following examples, we used a constraint on a target area.

$$E_{\text{shape}} = (A - a)^2 \tag{2.2}$$

with a the current area of the shape and A the target area, provided by the user for each shape, with some randomness.

A curve have different generation rules. It can either follow the gradient of the fitting function $\nabla\Gamma_{obj}$, follow the isocontour $\nabla\Gamma^{\perp}_{obj}$, or follow the heat points. While the first two possibilities are trivial, the later can also be optimized using the Active Contours algorithm by optimizing the energy $E = E_{\text{internal}} + E_{\text{shape}}$ with Equation (2.1). We applied a length constraint on the curves :

$$E_{\text{shape}} = (L - l)^2$$

with l the curve's length and L the target length. This algorithm is sensible to the initial shape of the curve, so we start with a straight line following the isolevel at the seed point.

Environment object's evaluation

Environment objects are evaluated at every iteration in order to determine the current state of the life cycle of the environmental object. For punctual environmental objects, this evaluation is applied at its position $\Gamma_{obj} = f(\mathcal{E}_p)$. Curve environmental objects are evaluated along the parametric curve C such that $\Gamma_{obj} = \int_C f(\mathcal{E}_{C(t)})\,dt$. In practice, we compute the evaluation as the average of all control points of the curve $\frac{1}{n}\sum_i^n f(\mathcal{E}_{C_i})$. Region environmental objects are evaluated inside their region Ω as such $\Gamma_{obj} = \int_\Omega f(\mathcal{E}_p)\,dp$. In practice, we compute the

average of random points in the region $\frac{1}{n}\sum_i^n f(\mathcal{E}_{p_i})$. When deformations on the environmental object's shape is applied, we apply cage deformation using Green's coordinates in order to keep consistent evaluation points during the whole life cycle of an environmental object.

2.3.3 Environment values

All environmental objects of an ecosystem has an impact on all the other environmental objects, which would result in an exponentially growing computation effort as the number of environmental objects of the terrain increase. We avoid this problem by considering the environment values as a proxy to allow any environmental object to interact with any other one. Each of the environmental object have a local impact on the environment values without knowledge of neighboring environmental objects. This modification of the environment values can be due to an absorption and deposition of some material $\mathcal{M}$ or an influence on the water currents.

Environment materials

The environment is composed of a scalar field for each of the possible material that can be found in the terrain. The scalar fields represents the availability of the material at any point, but not a height field. Each material is defined with a mass m, a fluid velocity factor ν, a diffusion rate D and finally a decay rate k.

Each environmental object in the terrain is a source and a sink of materials. It is the main mean of communication between environmental objects as it allows them to interact with their surrounding environment. We define the amount of deposed material with $D_{\mathcal{M}}$ and $A_{\mathcal{M}}$ the amount of material deposed and absorbed by the environmental object and $\gamma(t) \in [0, 1]$ a factor related with the current state of the environmental object, which state that more material will be displaced when the environmental object is fully formed than when it was just spawn:

$$\int_0^t \gamma(t)\,(D_{\mathcal{M}} - A_{\mathcal{M}})\,dt$$

The deposition and absorption around an environmental object is defined using the Gaussian kernel distance computation from the skeleton.

The scalar field for the material $\mathcal{M}$ is displaced by using a warp operator ω, taking into account the water flow $\mathcal{W}$ and the terrain slope $\nabla\mathcal{H}$. We unified the warp with m the mass of the material and ν a influence factor of the fluid on the material:

$$\omega(\mathbf{p}, t) = m\nabla\mathcal{H}(\mathbf{p}, t) + \nu\mathcal{W}(\mathbf{p}, t)$$

The materials are also dispersed at a diffusion rate D, for which we can use the advection-diffusion-reaction equation to evaluate the distribution after a time t

$$\frac{\partial\mathcal{M}}{\partial t}\omega\nabla\mathcal{M} = D\nabla^2\mathcal{M} - k\mathcal{M} \tag{2.3}$$

We solve (2.3) numerically using Euler integration

$$\begin{aligned}\mathcal{M}(\mathbf{p}, t + dt) = \mathcal{M}(\mathbf{p}, t) + dt(D\nabla^2\mathcal{M}(\mathbf{p}, t) - k\mathcal{M}(\mathbf{p}, t) \\ - \omega(\mathbf{p}, t)\nabla\mathcal{M}(\mathbf{p}, t))\end{aligned} \tag{2.4}$$

The introduction of the decay rate k in the equation allows for the reach of a steady-state, where we can consider the simulation stable. As the user updates the state of the simulation manually, we observe the reach of this steady state before continuing the iterative steps.

Water currents

We define our water currents as a vector field defined as

$$\mathcal{W}(\mathbf{p}) = \mathcal{W}_{\text{user}}(\mathbf{p}) + \mathcal{W}_{\text{simulation}}(\mathbf{p}) + \mathcal{W}_{\text{objects}}(\mathbf{p})$$

With $\mathcal{W}_{\text{user}}$ a user-defined vector field, $\mathcal{W}_{\text{simulation}}$ an analytical solution inspired by a wind flow simulation (Paris et al., 2019a), and $\mathcal{W}_{\text{objects}}$ the water flow alteration computed from the environmental objects. The component $\mathcal{W}_{\text{simulation}}$ is terrain-induced. Given an input flow direction a, we modify the vector field by warping it with the terrain gradient smoothed at multiple scales :

$$\mathcal{W}_{\text{simulation}}(\mathbf{p}) = \sum_{i=0}^{i=n} c_i \omega_i \cdot v$$

with $v = (a(1 + k_w \mathcal{D}(\mathbf{p}))$ and $\mathcal{D}(\mathbf{p})$ the depth at point $\mathbf{p}$ and k_w a scaling factor, used to simulate the Venturi effects. $\omega_i \cdot v$ is the warping operator at scale i with a coefficient c_i defined as

$$\omega_i \cdot v = (1 - \alpha)v + \alpha k_i \nabla \tilde{h_i}^{\perp}(\mathbf{p}) \qquad \alpha = \left\| \nabla \tilde{h_i}(\mathbf{p}) \right\|$$

with k_i a deviation coefficient, α the slope of the smoothed terrain and $\nabla \tilde{h_i}^{\perp}(\mathbf{p})$ the orthogonal vector of the smoothed terrain. As proposed by the authors, we used two scaling levels $n = 2$ with gaussian kernels of radii 200m and 50m with weights 0.8 and 0.2 and deviation coefficients k_0 and k_1 of 30 and 5.

$\mathcal{W}_{\text{objects}}$ is a deformation field defined as the accumulation of flow primitives (Wejchert and Haumann, 1991). Kelvinlets are applied on each environmental objects to deflect the water flow. We use the scale and grab formulations of the regularized Kelvinlets brushes (Goes and James, 2017), denoted as $s_\varepsilon(r)$ and $g_\varepsilon(r)$ respectively to simulate obstruction and diversion, are defined as

$$s_\varepsilon(r) = (2b - a)\left(\frac{1}{r_\varepsilon^3} + \frac{1}{2r_\varepsilon^5}\right)(sr)$$
$$g_\varepsilon(r) = \left[\frac{a - b)}{r_\varepsilon} I + \frac{b}{r_\varepsilon^3} r r^t + \frac{a\varepsilon^2}{2r_\varepsilon^3}\mathbf{I}\right]\mathbf{F}$$

with $a = \frac{1}{4\pi\mu}$ and $b = \frac{a}{4(1-v)}$ provided μ a shear modulus and v a Poisson ratio provided for each Kelvinlet, $r = \mathbf{p} - \mathbf{q}$ for $\mathbf{p}$ the evaluation position and $\mathbf{q}$ the center point of the Kelvinlet, $r_\varepsilon = \sqrt{\|r\|^2 + \varepsilon^2}$ the regularized distance, ε a radial scale for the deformation field, s a scaling factor and $\mathbf{F}$ the force vector of the grab operation. Deformations defined on curves use $\mathbf{q} = C(\mathbf{p})$ with $C(\mathbf{p})$ the closest point on the curve from the point $\mathbf{p}$ and $f = C'(\mathbf{p})$. We can then define $u_o(\mathbf{p}) = s_\varepsilon(\mathbf{q} - \mathbf{p}) + g_\varepsilon(\mathbf{p} - \mathbf{q})$.
Finally, we can retrieve the velocity field from the objects:

$$\mathcal{W}_{\text{objects}}(\mathbf{p}) = \sum_{o \in \mathcal{O}} \lambda_o u_o(\mathbf{p})$$

2.3.4 User interaction

The user can guide the generation process. The use of simple shapes as environmental objects facilitate the edition of the simulation, as we can interactively add, remove or modify

Figure 2.3: Starting from a coral colony developed around a canyon (*left*), the user edits the shape of the canyon, resulting in a different configuration of the scene, killing the corals that ends too deep in the water (*center*) and the development and growth of new corals at the previous location of the canyon (*right*).

environmental objects, or focus the generation process in a restricted area. Interaction with the environment values is also provided as geological events, that the user can invoke during the simulation. While the direct interactions on the environmental objects are instantaneous, as a the geological events are active on a given duration.

Direct interactions on the environmental objects

The interactive nature of our simulation enables the user to modify the state of the terrain by manipulating directly the environmental objects of the scene. We assume the modifications applied between two iterations of the simulation.

Translating an environmental object is trivial, we simply requires to evaluate the state of the environmental objects at a translated position. The deformation of environmental objects can be applied on curve and region environmental objects by updating the control points of the skeleton and recomputing the resulting implicit surfaces. The evaluation positions used for region environmental objects are displaced by applying a cage deformation of the 2D shape using the Green coordinates of points in the shape. After the alteration of the region, evaluation points should be keeping a similar distribution than before, avoiding unexpected results during the interaction. By modifying an environmental object, the environment values may change, which can result in the destruction of the now incompatible environment objects in the scene (Figure 2.3).

As long as a non-zero fitting function is defined in the terrain, new environmental objects can be forced by the user at any point of the simulation.

Control over the region of the terrain that should be updated can be given by adjusting all fitting functions through a scalar field $\lambda : \mathbb{R}^2 \to \mathbb{R}$ such that the fitting function $\Gamma_{obj}(\mathbf{p})$ of any new environmental object is evaluated as $\Gamma^*_{obj}(\mathbf{p}) = \lambda\mathbf{p}\Gamma_{obj}(\mathbf{p})$. This is especially useful in the planning of robotic simulations as we can first generate the overall shape of our terrain and secondly focus the generation process around the areas that may be visited by the robot, avoiding useless simulations and computer power. Figure 2.7 shows an example of colonization of the coral polyps that we limited manually into an annulus.

Our water current simulation is modeled as a simple vector field. As such, the user is able to interact with it at any moment of the simulation, allowing for the death of sensible environmental objects while it will guide the simulation into a new landscape. By modifying the water currents, the user also modifies the transport rate of materials at this position. The modification of currents is given as a stroke, a parametric curve C for which we evaluate $\Delta\mathcal{W}_{\text{user}}(\mathbf{p})$ just as for curved environment objects (Section 2.3.3).

Figure 2.4: Lowering the water level by a few meters caused most of the coral objects to satisfy $\Gamma_{obj} \leq 0$, causing their death. Since the water level (blue) decrease slowly, new coral objects spawn progressively at a lower altitude.

Geological events

A configuration file can define in advance the different events that should be triggered during the simulation. This can be useful to generate landscapes that are close to some existing locations. Multiple geological events can be triggered either as sudden or continuous environmental changes. These changes play a huge role in the morphology of landscapes. We define events with a starting point and an ending point, such that at any time of the simulation we can compute the progress of the event as $t_e \in [0, 1]$.

Water level changes are important events that shape the underwater landscapes. As previously submerged environmental objects get elevated above water level, flora and fauna terrain elements dry and die. Deprived from the living part of the elements, everything is more affected by terrestrial erosion. By updating the value of the depth $\mathcal{D}$ evaluated in the fitting functions, any environmental object that is sensible to the depth will be impacted automatically, that may be causing death (Figure 2.4). The modification of the water level is defined as

$$\mathcal{D}(\mathbf{p}) = \mathcal{D}_0(\mathbf{p}) + \sum_{e \in \text{events}} \Delta\mathcal{D}_e t_e$$

with $\Delta\mathcal{D}_e$ the amount of water rising or lowering during an event. We assumed a linear evolution of the water level during an event. This allows to evaluate the depth at any point in space and in time.

Subsidence and uplift are the main geological events that create or destroy islands in the long term. These events are simulated as a simple factor on the height field of the generated terrain (Figure 2.5). Subsidence is not always uniform in the terrain. As such, the user can provide a position $\mathbf{q}$ at which the subsidence is the strongest, the amount of subsidence applied $\Delta\mathcal{H}_e$ and a standard deviation σ for which we can then compute at any point in space and time of the simulation the height of the terrain

$$\mathcal{H}(\mathbf{p}) = \mathcal{H}_0(\mathbf{p}) \cdot \sum_{e \in \text{events}} \frac{G(\|\mathbf{p} - \mathbf{q}\|)}{G(0)} \Delta\mathcal{H}_e t_e$$

with $G(x)$ the Gaussian function

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Figure 2.5: Simulating subsidence on a part of the terrain (brown area) cause the depth value to change locally, resulting in the death of coral objects that find themselves too deep to survive. Here two subsidence events are triggered in parallel.

Storms are factors of the geomorphology of coral reefs (Oron et al., 2023; Vila-Concejo and Kench, 2016) and coasts (Cowart et al., 2010; Domínguez et al., 2005). Due to the extreme wind and wave velocities coasts are highly eroded in a short time period and the more fragile corals near the water surface are broken, possibly causing breaches in the reefs and spreading polyps in the currents direction. While there are many factors at play to understand the apparition of storms and the hydrodynamics affecting it, we simplified the model of storms to the user as a single epicenter $\mathbf{q}$ with a wind velocity v_{wind} and a standard deviation σ representing the spread around the epicenter (Figure 2.6). The computation of water currents are then computed as

$$\mathcal{W}_{\text{user}}(\mathbf{p}) = \mathcal{W}^{*}_{\text{user}}(\mathbf{p}) + \sum_{e \in \text{events} | t_e \in [0,1]} v_{\text{wind}} \frac{G(\|\mathbf{p} - \mathbf{q}\|}{G(0)}$$

In this case, we did not include the linear factor t_e as storms are usually conserving a constant force for the time of the few weeks or months of their occurrence.

with ΔT_e the change of heat during an event, T_0 the temperature at the water surface, and c a very small factor.

The framework can easily be extended as the geological event system stays similar for all events. Including higher level simulations in the event system can be added, such as the simulation of tectonic activity, the use of fluid dynamics for tsunami events, the integration of human activity, ...

2.4 Expert knowledge integration

The definition of the fitting functions of the environmental objects are inspired by the biological and geological factors that rule the evolution of underwater landscapes. The main factors are depth, light, water currents and biodiversity. External events have direct and indirect repercussions on the biodiversity of underwater environments. Coral islands are complex bio systems in which fauna, flora and geology are mixed together.

2.4.1 Environmental objects description

We have represented with environmental objects some geologic elements, animal elements and flora elements. The low island is most often raised in a circular shape as the process

Figure 2.6: The result of a storm localized on one side of the island (red area) modifies the result of the evaluation of environmental objects around its epicenter for a short period of time. Most of the coral objects died from the event, except few environmental objects less sensible to water currents strength.

mainly appear around a hot spot under the ground. The evolution of an island into a coral island requires that the environmental conditions are sufficient for coral development: corals will grow slightly below the water surface as waves will break its growth and at a shallow depth (around 3m to 30m deep) in order for light to reach it. As coral grow and die, the skeleton is transformed into porous limestone, providing shelter to surrounding animals and reducing the impact of water erosion on the island. Corals drop polyps that are transported by the water flow and when they stick to a hard surface, as a rock or the reef itself, the coral may grow and colonize the area. As subsidence cause the island to lower, the living part of the coral reef keep growing toward light, which lead to a reef that is constantly close to the water level without reaching it due to wave erosion. The survival of reefs depends on the equilibrium between coral growth and and erosion. Eroded parts of the reef falling in the sheltered part of the reef accumulates, ending up by forming a lagoon. An island formed by a hot spot will inevitably subside in time, until it is completely flatten. As the coral reefs keep growing, only the lagoon remain, resulting in an atoll.
In this work we we integrate the biological and geological knowledge in the fitting functions of the environmental objects we want to generate. We represent the islands as regions that can be appearing with a uniform distribution. From the formulation of the region description (2.1), we mostly create circular islands. The coral elements, environmental objects described as a single point, have a fitting function that take into account the depth of the ground, the amount of sand, fresh water and polyps in their environment, as well as the strength of water currents. Each coral species have different living conditions, but we reduced our work to soft coral which are sensible to water strength and stony corals that are more resistant to erosion. Reefs are formed as coral's skeleton are transformed into calcareous stone, describing then as an environmental object representing multiple others.

2.4.2 Simplifications

The environmental factors simulated are greatly simplified as the real processes are in a very small time scale, that computer simulation are not able to simulate in interactive time. The use of environmental objects aim to represent a plausible results, while avoiding modeling the smaller scale events. Examples of simplifications are the geometry and material of each environmental object, which have an influence on the water currents through friction, the water currents represented as stationary flows, while the water flow dynamics are a complex system that may change completely at two different times of the day, the animal influence on the reefs that they transform by the ingestion and deposition of sediments, ...

2.5 Results

Our method provides a way to generate scenes at different scales. We demonstrate this capacity with the generation of a large scene of an island (Figure 2.1) after what we focused the generation process in a canyon (Figure 2.8), then a small-scale visualization of coral colonies (Figure 2.7). In the examples, we rendered the environmental objects as a implicit tree or as individual meshes. The island, lagoons, reefs, canyons and sand ripples as implicit surfaces

A canyon scene can be generated using our method. The water flow is affected by the curve of the canyon such that the currents are oriented in the direction of the curve's tangent.In this example, we force the position of arches to be inside the canyon. The arches deposits a material "rock deposit", which is the main element of the fitting function of the Rock object. The "rock deposit" is slightly affected by water currents, but its mass make it highly affected by gravity. As such, rocks will spawn underneath arches. In reality, an arch is often created as part of a large coral boulder that sees the calcareous bottom part detached by the water currents, often resulting in an arch surrounded by big rocks and smaller rocks from the erosion of the first rocks. As such, we define an environmental object "Arch" with a fitting function $\Gamma_{arch}(\mathbf{p}) = 5 - d(canyon - \mathbf{p}) * \|\mathcal{W}(\mathbf{p})\|$, an environmental object "Rock" using $\Gamma_{rock}(\mathbf{p}) = \mathcal{M}_{rock\_deposit}(\mathbf{p})$ and Pebble using $\Gamma_{pebble}(\mathbf{p}) = \mathcal{M}_{smaller\_rock\_deposit}(\mathbf{p})$. Finally, sand ripples are simply described as curves appearing where there is a lot of sand available: $\Gamma_{ripple}(\mathbf{p}) = \mathcal{M}_{sand}(\mathbf{p})$. Following these simple rules, Figure 2.8 shows the emergence of details in the scene.

In this example we defined three different types of corals, coralA, coralB and coralC, to illustrate the possibility to model behaviours from the choice of fitting functions. Each of the coral types deposits a material "coral polyp" and "coral polyp A" ("coral polyp B" and "coral polyp C" respectively). By considering a fitting function that minimize the ratio $\frac{\text{coral polyp}}{\text{coral polyp A}}$, we can see an emergent behavior of the three types of coral fighting for the space colonization. Figure 2.7 shows the result of this simulation at three different interations. At the border between two colonies, none of the colonies make progression due to the amount of coral polyp specific from the other colony.

2.6 Discussion

The proposed method aims to generate plausible landscapes using simplified versions of the evolution of an ecosystem and of the 3D representation. The biological realism of the result is highly correlated to the amount of simplification and assumptions, while the visual

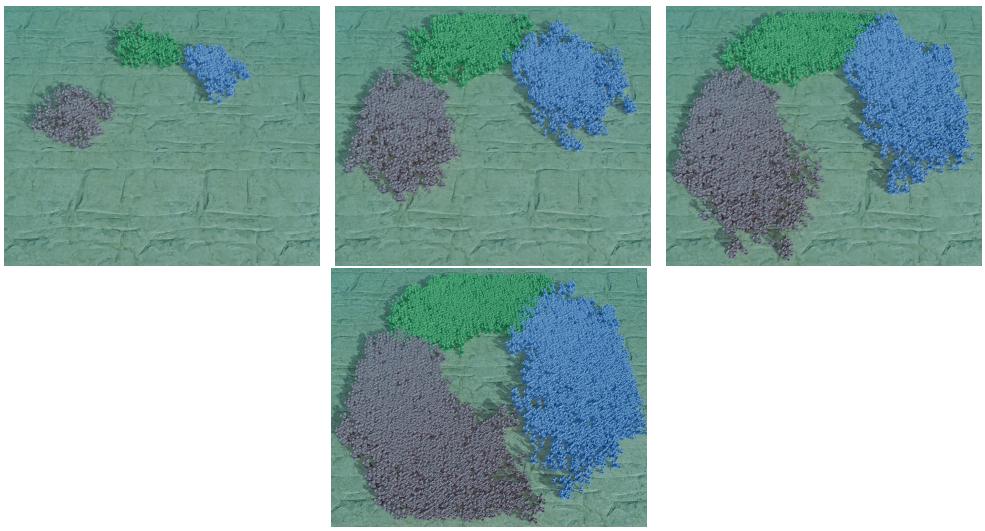


Figure 2.7: Three colonies of coral (red, blue, green) restricted to an annulus the middle section of the terrain fighting for the space.

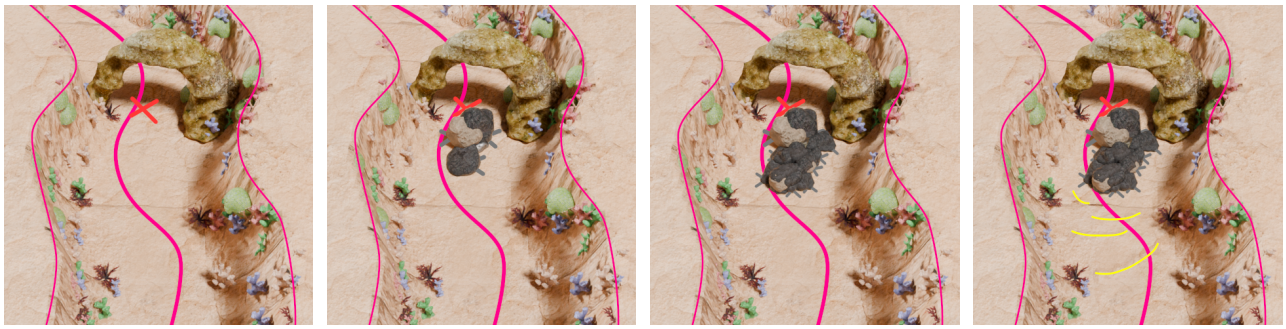


Figure 2.8: Evolution of a canyon scene at different iterations of the simulation. The apparition of an arch causes the spawning of rocks, pebbles, and finally some deposition of sand at the bottom of the canyon, spawning ripples.

realism is completely dependent to the geometric functions used for the 3D modeling of the environmental objects. While proposing a flexible method that propose a generic approach for terrain generation, a close collaboration with fields experts and with graphists is needed to achieve optimal results.

Most simulation algorithm's quality depends on the size of the time step used, but with the introduction of a decay rate in the materials properties, we limit the influence of time steps by considering that steady-state are reachable. The material deposition and absorption on punctual environmental objects can be seen as a Dirac function δ centered at their position resulting in the advantage that material displacement function can use the definition of the diffusion equation instead of the advection-diffusion-reaction equation. This equation allowing us to evaluate the state of the material $\mathcal{M}$ without intermediate steps, but this is not applicable with curve- and region-based environmental objects.

2.7 Conclusion

We have proposed a method to generate terrains procedurally using sparse representations. This representation, the environmental objects, enables to introduce expert knowledge by the mean of the fitting functions that rule the environmental objects life cycle, but also to integrate the user in the loop during the generation process. We reduced the terrain resolution limitations by defining the environment objects as parametric elements. Thanks to the sparse representation based on single points, curves and regions, we allow for direct manipulation of the environmental objects of the scene by the user which, thanks to the environment steady state consideration, also enables to include these interactions in the automatic simulation process.

Integrating environmental properties in the fitting function of environmental objects allows the user to guide the generation through geological events. Our method enables each environmental object of the scene to influence the environment locally, reducing the need of computations while also retrieving environment values locally, which result in a parallelizable life-like simulation process. The genericity of the environment properties definitions should be sufficient for plausible generation of other landscape types as long as expert knowledge can be translated to environmental object's formalism.

We limited our work to the use of 2D scalar fields as they are more easily differentiable, interpretable and lighter than volumetric representations. However, future works include using 3D representations of the terrain and the environment to generate 3D terrains, including cavities, sub-terrestrial areas and the interior of coral structures.

Figure 2.9: A simple coral island is generated using an island, a lagoon, reefs coral polyps, beaches, trees and algae environmental objects. Trees appear on beaches and algae grow in the lagoon's sand.

Part II

Modelisation

Abstract

- Methods completely different
** Physical phenomena are different
** => Simulation/generation methods must be specific for one landscape
- Methods are developed at different instants of the thesis work,
** We will see different procedural generation domains:
*** Analytical solutions (coral islands #1)
*** Deep Neural Networks (coral islands #2)
*** 3D user interaction (karst networks)
- Deep Learning tends to replace procedural methods in 2D domain,
** But still too complex for 3D models
*** (lack of data, lack of research interest for now)
** Still requires a lot of data, which, is (while not easy) possible using aerial images, but is way too sparse with underwater landscapes or underground biomes.
- We want to control the area in which an element is modeled in the terrain
** Because that's how we define them in the previous chapter.
** Thus we cannot use simple random noise methods => no bounds
*** Only solution is to use falloff maps, but meh...
- As such, we want to keep the skeleton of our elements using primitives (points, curves, regions)
** => Much easier to add constraints / manipulate primitives in a "procedural generation" way.
- Warning: usage of Deep Learning is at the limit of "procedural generation" and is not considered as part of it by the whole community.
(Complete with the Reddit poll).
- ...

CHAPTER 3

Volumetric Terrain Modeling [MAYBE TO BE MOVED TO INTRO]

Contents

Back to summary

- Volumetric modeling is important for representing 3D structures
- Allows for the representation of cavities, arches, overlays, etc.
- The concept of materials allows for including much more information for the following parts: amplification and rendering
** Amplification (e.g., erosion) needs to know the type of soil at the surface and subsurface to be realistic
** Rendering needs to know the material at the surface to correctly display textures
- ...

3.1 Implicit Terrains with Materials

- ...

3.1.1 Material Density

- ...

Material Granularity

- ...

Soil Triangle

- ...

3.1.2 Scalar Functions

- ...

3.1.3 Blending Functions

- ...

3.1.4 Placement Functions

- ...

3.1.5 Material Usage

- ...

Defining the Final Material

- ...

Post-processing: Material Transformation

- ...

3.2 Graphical Representation of Environmental Objects

- Implicit surfaces
- Meshes
- ...

CHAPTER 4

Automatic Generation of Coral Islands

Contents

Back to summary

4.1 Introduction

- Definition of coral islands
** Different types of coral islands

*** Here, volcanic islands
- Presentation of corals
- Difference from regular landscapes
** Concept of corals
*** Long-term evolution (island) and short-term evolution (corals)
**** Geological process of island subsidence
- ...

4.1.1 Darwinian Theory

- Several theories
- Difficulty in studying environments
** Use of observations
- Theory refuted by (Droxler and Jorry, 2021)
** But it's too early to judge
** Practical for our case.
- ...

Multiple Theories

- Clarify other theories:
** "Growth on Submarine Mountains Theory" (John Murray): Reefs start on submarine mountains and guyots, then gradually rise to the surface.
** "Sea Level Change Theory" (Reginald Daly) ["The Coral Reef Problem"] (Daly, 1915): [TO BE EXPLORED, I DIDN'T UNDERSTAND IT]
** "Erosion and Sedimentation Theory" (Maurice Ewing and William Donn): [TO BE EXPLORED]
** "Platform Reef Theory" (William Morris Davis): [TO BE EXPLORED]
- The theories are not necessarily contradictory; they can be complementary in explaining cases that others do not.
- ...

Darwin's Voyage

- ...

4.1.2 Overview

- Proposed tool
** Sketching the island
** Sketching the profile
** Wind simulation
- Automation of examples
** Data augmentation
- ...

4.2 Related Works

- Perlin + falloff
** But lacks control
- Uplift [SCHOTT UPLIFT] (Cordonnier et al., 2016; 2017a)
** But we propose a method that limits the required interaction
- Sketching [PEYTAVIE SKETCHING, EMILIEN FIRST PERSON] (Gain et al., 2009)
** Added radial constraint to simplify interaction
- Geological modeling (Patel et al., 2021)
** For us, it's hard to know what's happening underground (?)
- Unlike the literature, integration of the underwater part
** Integration of observation data into our process
*** -> Typical shape and profile of an island
** Difficult to integrate physical simulations due to uncertainty
- Layer-based generation could improve the realism of examples by considering layer/environment interactions.
- ...

4.3 Example generation

- We use Darwin's theory in our case because generation is relatively simple.
- 2 steps:
** Generation of the island
** Generation of the reef
- Numerous assumptions:
** Island has a relatively round shape
** Coral grows to a constant height around the island
** All "features" are radial
** Deformations of islands caused by winds and waves
** Islands are independent of each other
** The profile is relatively identical all around the island
- ...

Input

- Sketching of the island from above + profile view
- Wind strength
- Water level
- Subsidence rate
- ...

"Simulation"

- Subsidence calculated by simple scaling
** Note: No geological consistency
*** Here, using a zero level to scale in Z
*** No consideration of different soil materials
- Reef growth

** Does not consider any weather events
** Considers the "keep up" strategy (as opposed to "give up" and "catch up") [large simplification]
- Wind deformation
** using directly $f(\mathbf{p}) = f(\omega(\mathbf{p}))$.
- ...

Output

- Height map of the island's surface
- Island zones and coral zones
- Possibility to recalculate ground height and coral height
- ...

4.3.1 Closed form of coral growth

- Proposes a closed-form solution for surface calculation
- Calculation of ground height:
** Calculation of height by revolution around the origin point
** Deformation of the revolution profile using the top-down sketch
** Deformation of the height field by the wind map.
- Calculation of growth:
** On the initial heightmap,
*** Any surface $z_{min} < h(p) < z_{max}$ becomes coral
*** Calculation of the "low" contour $h(p) = z_{min}$ and "high" contour $h(p) = z_{max}$
*** ...
- Deformation of the map using the wind map:
** ...
- ...

4.3.2 Labeling of the map

- Using top-down sketch:
** Features "Mountain", "Island borders", "Beach" are radial (θ, r), so we can label each point of the map as the "next" feature
- Using coral simulation information:
** Provides the labels "Lagoon" and "Reef {begin, peak, end}". - The height map is directly associated to the feature map
- This is perfect to feed a cGAN.
- ...

4.3.3 Automation

- Take advantage of the radial nature of the features
- Some features can be optional (mountains)
- Deformation of the feature lines
** Influence on the radius: $\tilde{r}(\theta) = r(\theta) + \eta(\theta)$ with η continuous noise function 2π-periodic.
- ...

4.4 cGAN

- Conditional GAN: A type of Generative Adversarial Network (GAN) where the generation process is conditioned on additional information.
- ...

4.4.1 Definition of cGAN

- Two Networks: Consists of two neural networks, a Generator and a Discriminator, which compete against each other.
** Generator: Takes both random noise and additional information (like class labels or data) to produce synthetic data.
** Discriminator: Evaluates whether a given data instance is real (from the actual dataset) or fake (produced by the Generator), while also considering the additional information.
** Additional Information: This can be labels, data from other modalities, or any other contextual information that guides the generation process.
- Training Process: The Generator tries to create realistic data to fool the Discriminator, while the Discriminator tries to correctly classify data as real or fake based on both the data and additional information.
- Objective: The goal is to improve the Generator's ability to produce realistic data that matches the given conditions and to enhance the Discriminator's ability to distinguish between real and fake data.
- Applications: Used in various domains including image-to-image translation, text-to-image synthesis, and other tasks where generating data based on specific conditions is required.
- ...

4.4.2 Why cGAN?

- Flexibility of input
- Moving beyond the "radial" input condition
- Output even for "inconsistent" data (e.g., ocean in an island)
- No math, geometry, geology, or complicated things to master (hehe)
- ...

4.4.3 Training

- ...

Use of Synthetic Data

+ Problem with synthetic data
- ...

Data Augmentation

- ...

4.4.4 Model Usage

- ...

Generation from Sketch

- ...

Interactive Times

- ...

Realism

- ...

CHAPTER 5

Generation of Karst Networks

Contents

Back to summary

- ...

5.1 Geology

- ...

5.1.1 Formation

- ...

5.1.2 Significance

- ...

5.1.3 Role of Karsts in the Project

- ...

5.2 User Control

- ...

5.2.1 Existing Methods

- ...

5.3 My Method

- ...

5.3.1 Karst as a Tree?

- ...

5.3.2 Vegetation Generation -> Space Colonization

- ...

Part III

Erosion Simulation

Abstract

- Work carried out at the beginning of the thesis
- Representation choice still uncertain
- Search for abstraction of the representation
** Drives the generalization of the erosion system

CHAPTER 6

Érosion par particules

Contents

Back to summary

6.1 Introduction

Automated terrain generation is a key component of natural scene digital modeling for animated movies and video games. A standard approach is to first generate a base terrain geometry using noise to define the height on the input domain (Musgrave et al., 1989; Olsen, 2004; Roudier, 1993), the result will most likely lack realism and feel synthetic. Erosion simulation algorithms are applied, to simulate thousands of years of ageing by reproducing physical phenomena - i.e. effects of the elements (rain, wind, running water...) - affecting the terrain making it more believable (Galin et al., 2019; Smelik et al., 2009; Stachniak and

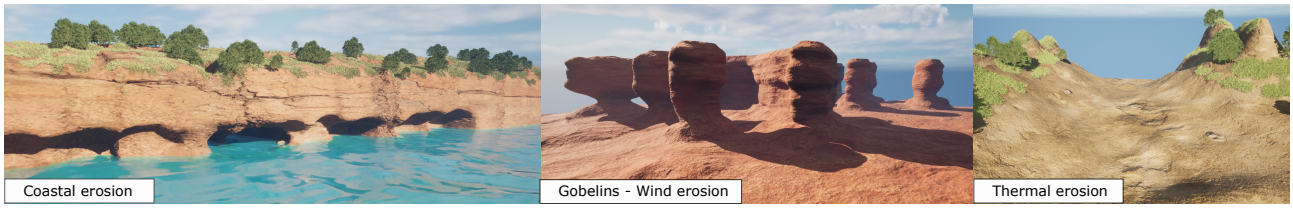


Figure 6.1: Applying shading and textures on the generated geometry can produce a plausible aspect of a coast eroded by waves on a long timespan, or a desertic landscape eroded by wind, or a mountainous area flatten by thermal erosion.

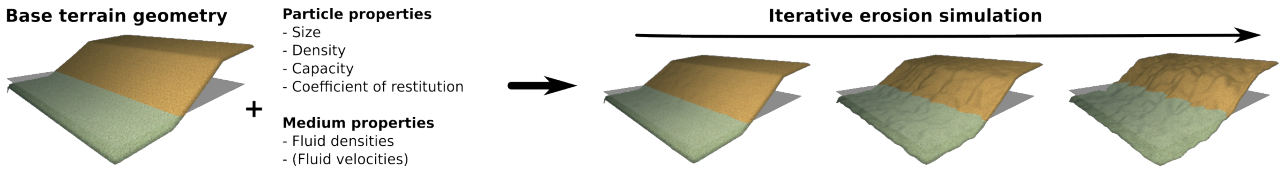


Figure 6.2: Our method require a base geometry, a small number of parameters for the particles and the medium used for the erosion simulation. It can be easily adapted to be compatible with different mediums and terrain representations.

Stuerzlinger, 2005).
The process of terrain alteration caused by the effect of water, air, or any other element - natural or not - over time is usually performed in three steps (Neidhold et al., 2005): **detachment** - pieces of the ground of variable dimensions, ranging from complete ledges to grains of sand, are removed from the terrain depending on the simulated meteorological phenomenon - **transport** - pieces of ground fallen from their initial position are moved to a different one (e.g. a cornice falls down a slope or a grain of sand is thrown into the air) - and **deposition** - transported pieces of land are accumulated at a new part of the landscape. Various phenomena can cause these alterations: **thermal erosion** (bursting of rocks caused by expansion of water under frost, then falling of debris to the bottom of a slope), **hydraulic erosion** (detachment caused by the impact of water particles on surfaces and the transport of sediments by the flow of runoff), **wind erosion** (fine particles carried away in the wind and hit surfaces on their way, creating new fine particles which then also fly away), **chemical erosion** (chemical decomposition of rocks caused by rainwater or other fluids), other exceptional phenomena such as avalanches, animals, lightning, etc... modify the terrain (Argudo et al., 2020; Cordonnier et al., 2017b; 2017a; 2018; 2023).

In practice, the core idea to simulate erosion is to add or remove material from the terrain at given positions on the interface between the terrain and fluid eroding it (e.g. air or water). Hence, the two major problems to tackle are: how to locally alter the terrain geometry for material detachment and deposition and where to perform these alteration given the properties of the environment (terrain slope, fluid density and velocity). A terrain is more than often represented in 2.5D using a 2D image called a heightmap which grey scale values define terrain elevation. While being the major terrain representation, only a limited number of environments can be modeled. Indeed, natural landscapes are intrinsically 3D (overhangs, cavities or geological structures such as arches or gobelins), this is particularly true for underwater environments generation. Alternate representation such as voxel grids, material layers or implicit surfaces can be used. A wide variety of method have been proposed to simulate natural erosion phenomena on heightmaps as the partial differential equations to model

erosion can be discretized and solved in 2D and the material detachment and deposition at a given point of the terrain surface can be easily performed by elevating or lowering the ground level i.e. changing locally pixel intensities. For volumetric representations, the alteration of the terrain is not as trivial. To define where to perform the erosion process the local slope variations are more than often used combined with eroding medium information. This fluid can be simulated using particle systems, Smoothed Particle Hydrodynamics (SPH) (Krištof et al., 2009) or approximated using a simple vector field. Proposed methods offer a specific erosion effect tailored to a single terrain representation and fluid simulation.

In this work we propose an approach to simulate a large part of the geomorphological and meteorological phenomena present in the literature of terrain generation (including 3D and volumetric effects). We introduce a generalized algorithm performing the three stages of erosion on surface and volume representations alike, and expose very few intuitive parameters to be adjusted by the user (Figure 6.2). We propose to tackle separately the material variation and the fluid simulation. Our method relies on a particle system to simulate eroding agents, each thrown particle will collide with the terrain, perform terrain alteration at the collision point and transport material along its path. Their motion is computed using simple particle physics accounting for the medium density and particle properties (buoyancy and gravity forces). We consider each particle as independent, hence, they do not interact with each other, no collision detection or response. This simplification allows for efficient parallel computation. When more accuracy or control is needed, we propose to provide a vector field used to modify the particle speed at each time step. The nature of this vector field is flexible, it can be computed using a more or less accurate fluid simulation (SPH, FLIP,...) or be manually defined by the user. We propose a particle-based strategy for material alteration that can be applied on surface and volumetric representation.
The main contributions of this paper are:

- a generalized particle-based algorithm performing the three stages of erosion on surface and volume representations,
- decoupling the erosion system from the fluid simulation, making the process more flexible in its usage and implementation and opening the door for richer effects that can easily be produced.

6.2 State of the art

In this section, we first present the major terrain representations (height fields, layered representations, voxel grids, and scalar functions) and a subset of the major simulated phenomena used to erode terrains. We highlight the fact that, in the literature, a specific erosion method tailored to a given terrain representation is proposed for given phenomena which might lead to limitation in term of terrain modeling. Indeed, changing representation costs information and precision loss.

6.2.1 Terrain Representations

A terrain can be represented in various ways, each of them suited for a given application of which we give an brief overview, more details can be found in (Galin et al., 2019).
Height Fields represents the surface of the terrain by defining the elevation at each point in a 2D grid. This representation is simple, regular, and fast to process allowing for easy manipulation, such as raising or lowering the terrain (Emilien et al., 2015; Gain et al., 2009).

Layered Representations are an extension of height fields using a 2D grid where each cell represents a stack of different materials instead of a simple height (Beneš and Forsbach, 2001; Peytavie et al., 2009) allowing for memory efficient representation of volumetric terrains. To create complex structures, arches or caves, solid materials can be transformed into more granular ones, that can be stabilized (Peytavie et al., 2009).
Voxel Grids are regular, uniform volumetric grids that encode information on the presence or absence of material for each 3D point in the domain. Voxel grids are advantageous due to their regularity (Dey et al., 2018) and ability to represent volumetric models at the cost of high memory footprint, which has limited their use in terrain generation (Ito et al., 2003; Kaufman et al., 1993; Lengyel, 2010). We consider two voxel grid representations : *density-voxel* grids for which each voxel contains a scalar value, for instance the occupation percentage (Eisemann and Decoret, 2008) and *binary voxel* grids that can be seen as a mask containing the presence of material information.
Implicit terrains represent landscapes as an implicit surface defined by a scalar function. This allows for high definition large terrain modeling. The application of combinable scalar function overlays (É. Guérin et al., 2016) or the definition of user-defined gradients (E. Guérin et al., 2022) can be used to create complex terrain features. Altering a implicitly defined surface is a challenging task hence limited option exist for erosion simulation (Paris et al., 2019b).

6.2.2 Erosion Processes

Erosion processes play a crucial role in shaping landscapes over time. We present different kind of erosion and how they apply to given terrain representations. Note that using existing methods all erosion methods can not be used on all representation.
Thermal Erosion is driven by large temperature shifts, transferring material based on slope thresholds. The process is iterative, redistributing material until slopes stabilize. It can be computed efficiently on height fields and layered terrains due to their manipulable height nature (Beneš and Forsbach, 2001; Musgrave et al., 1989; Peytavie et al., 2009). However, its application on voxel grids is challenging due to limited Z-axis resolution.
Hydraulic Erosion stems from water movement, eroding and depositing sediment based on water flow intensity. 2.5D terrains are widely studied for this simulation, using either water slope velocities (Neidhold et al., 2005) or water simulations for erosion effects (Mei et al., 2007). For smaller scales, 3D fluid simulations on voxel grids have been proposed (Beneš et al., 2006). Kristof et al (Krištof et al., 2009) used SPH (Smoothed Particle Hydrodynamics) for meshless erosion simulations on various terrains. Their method involves numerous particle interactions, demanding significant computational power. Our approach draws inspiration from this but enhances efficiency by removing certain particle interactions.
Wind Erosion shifts material through wind force, notably impacting areas with fine surface particles like deserts. It has been modeled on discrete height fields (Paris et al., 2019a; Roa and Benes, 2004) by mimicking sand's wind-driven trajectory and using thermal erosion for corrections. This process is simulated by iteratively displacing small amounts of matter, which make it less suitable for representations with discrete height resolution.
Erosion by Other Forces includes influences like glaciers, snow, tectonic movements, and fauna, each introducing distinctive terrain patterns, enriching its intricacy (Argudo et al., 2020; Cordonnier et al., 2016; 2017b; 2017a; 2018). However, most methods are tailored by a given terrain representation, often the height fields, and might not be applicable to other representations due to their intrinsic properties.

6.3 Particle erosion

Erosion occurs in three stages: material detachment, transport and deposition (respectively in red, black and green in Figure 6.3). In our approach, particles move through the medium following its flow (i.e. wind in air or currents in water) and then absorb or deposit a small amount of material upon contact with the land surface, effectively fulfilling the three stages of erosion.

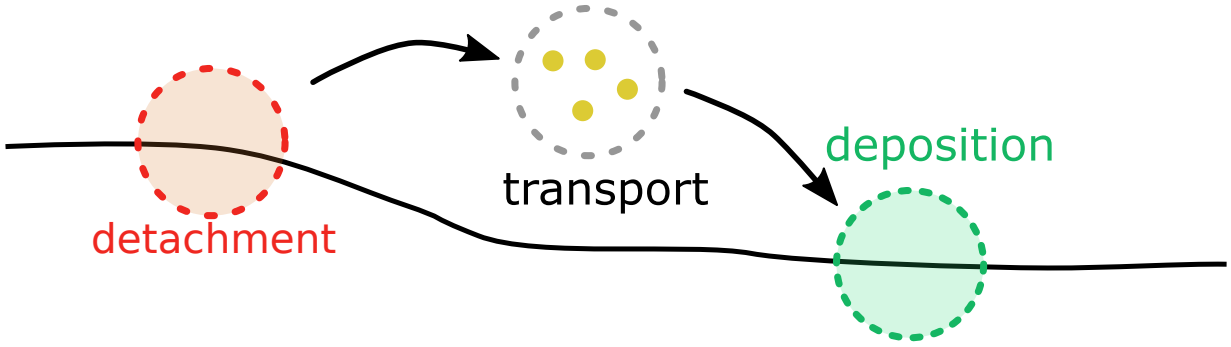


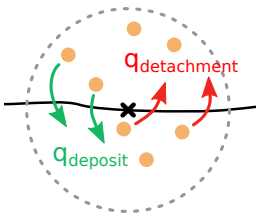
Figure 6.3: Three steps of the erosion process from the sediment point of view: detachment from its original location - dotted red circle -, transport in a fluid - dotted black circle -, deposition at a new location - dotted green circle.

6.3.1 Overview

Particles are transported through the medium and can pass through several different media. Each medium is defined by a density and a flow. Consider, for example, water density to be $1000\,\mathrm{kg\,m^{-3}}$ and that of air to be $1\,\mathrm{kg\,m^{-3}}$. The gravity applied to the particles is then very different between open and submerged environments due to the difference in buoyancy, while the process remains similar. Using a pre-calculated flow field to guide particle movement simplifies the simulation by treating particles as independent entities, eliminating the need for inter-particle calculations. This not only reduces significantly the overall execution time but also offers users high flexibility over the quality of the simulation and simplify the implementation.

6.3.2 Erosion process

Every time the particle hits the ground, a given amount $q_{\text{detachment}}$ of sediment is detached from the ground (red arrows) while another amount q_{deposit} of sediments is deposed at this location (green arrows). Our erosion model is based on the work of Wojtan et al where regular 3D grids are used to estimate the fluid velocity and sediment transport (Wojtan et al., 2007). In the spirit of (Krištof et al., 2009), we transposed their method into a particle-based erosion simulation, but, in our proposition, we decouple the particle system from the fluid simulation, making the process more flexible and opening the door for richer effects that can easily be produced.



Detachment. As a particle approaches the surface of the terrain, its motion applies friction at the interface between fluid and ground, causing bedrock to dislocate microscopic parts, that we call abrasion. We use pseudoplastics model to approximate the amount of matter

removed due to the shear forces while considering the physical properties of the fluid and the ground (Wojtan et al., 2007).

The shear rate θ is approximated by the relative velocity of the fluid to the solid boundary v_{rel} over a short distance l. We approximate the shear stress τ at the solid boundary by a power-law:

$$\tau = K\theta^n \tag{6.1}$$

where $\theta = v_{rel}/l$, K is the shear stress constant (often set to 1) and $n \in [0,1]$ is the flow behaviour index. Shear-thinning models typically assume n close to $\frac{1}{2}$, which is why we used this value as a constant.

We can then compute the erosion rate ε at any contact point between a fluid and a solid boundary using (6.1) by

$$\varepsilon = K_\varepsilon(\tau - \tau_{\text{critical}})^a \tag{6.2}$$

with $K_\varepsilon \in [0,1]$ a user-defined erosion constant, τ_{critical} the critical shear stress value for which the matter starts to behave like a fluid and a a power-law constant, typically considered as $a = 1$.

In our method, the eroded quantity is approximated as the material contained in the half sphere, of radius R, in the normal opposite direction at the particle impact point (Figure 6.6). We then use (6.2):

$$q_{\text{detachment}} = \varepsilon\frac{2\pi R^3}{3} \tag{6.3}$$

to get the final eroded amount $q_{\text{detachment}}$. The particle is also defined by a maximal amount of sediments that can be contained in its volume before being saturated noted $C_{\max}$. Note that this constant will be used for the settling velocity computation (6.7).

Deposition. The eroded sediments are considered in suspension in a fluid and are affected by its velocity. A fluid particle then transports the sediments in its flow until gravity settles it onto the ground again. The effect of gravity is modeled by a settling velocity w_s defined in Eq (6.7). We consider that the amount of sediment settled is proportional to the norm of the settling velocity as proposed in (Wojtan et al., 2007) with $\omega \in [0,1]$:

$$q_{\text{deposit}} = \omega||w_s||. \tag{6.4}$$

6.3.3 Transport

Our simulation is computed by integrating the full trajectory of multiple particles at each iteration unlike most other erosion methods. This allows to constantly have a terrain in a plausible state, while giving the possibility to increase the aging effect by running more iterations. Note that, reducing progressively the overall erosion strength can be used as a strategy to adapt the computation time to a chosen level-of-details.

We first present how to compute the particle speed using particle's physics then how to add optional medium velocity field to add a fluid simulation or user control.

Particle's physics. From its independence with other particles: we consider each particle following Newton's laws of motion.

First, we define the external forces $\vec{F}_{\text{ext}}$ applied on each particle, we consider gravity and buoyancy. We calculate the buoyancy force $\vec{F}_{\text{buoyancy}} = -\rho_{\text{fluid}}V\vec{g}$ with ρ_{fluid} the density of the fluid, V the volume of the particle and $\vec{g}$ the gravitational acceleration, but we can also

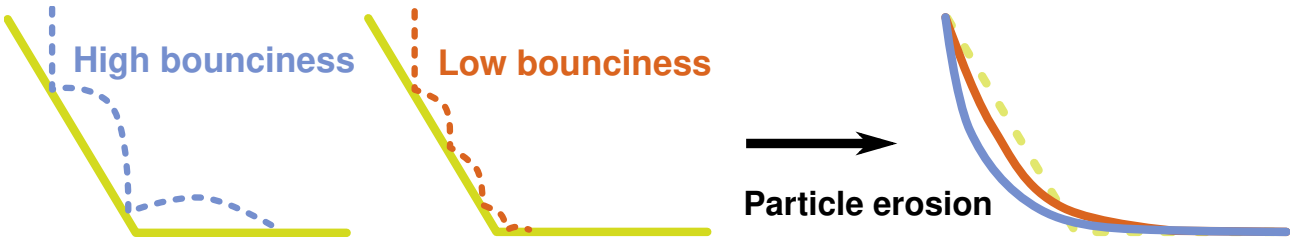


Figure 6.4: The coefficient of restitution affects the amount of energy absorbed from the particle when hitting the ground. Here, rain is applied on an initial slope (yellow). Only two particles are displayed, with a high (blue) and low (red) coefficient of restitution. The resulting slope after erosion is displayed in blue and red (right).

calculate the force of gravity $\vec{F}_{\text{gravity}} = m_{\text{particle}}\vec{g}$ with m_{particle} the mass of the particle. We then have the final external force $\vec{F}_{\text{ext}} = \vec{F}_{\text{gravity}} + \vec{F}_{\text{buoyancy}} = m_{\text{particle}}\vec{g} - v_{\text{fluid}}V\vec{g}$ knowing the density of an object $\rho_{\text{particle}} = \frac{m_{\text{particle}}}{V}$, we have:

$$\vec{F}_{\text{ext}} = V\vec{g}(\rho_{\text{particle}} - \rho_{\text{fluid}}). \tag{6.5}$$

The particle velocity v can be integrated from (6.5) by:

$$v = \int \vec{F}_{\text{ext}}\, dt + w_s + v_0, \tag{6.6}$$

with w_s the settling speed of sediments in a fluid with a viscosity μ given by Stoke's Law (Stokes, 2009):

$$w_s = \frac{2}{9}\vec{g}R^2\frac{(\rho_{\text{particle}} - \rho_{\text{fluid}})}{\mu}f(C). \tag{6.7}$$

We use the Richardson-Zaki relation as the hindered settling coefficient:
$f(C) = 1 - \left(\frac{C}{C_{\text{max}}}\right)^n$
with C and C_{max} respectively the fraction of volume of sediments contained and the maximal fraction of sediments the particle can contain, and n an exponent typically 4–5.5, which we set to 5 (Richardson and Zaki, 1954; Wojtan et al., 2007).

Finally, the particle position can be integrated as:

$$\mathbf{p} = \int v\, dt + \mathbf{p}_0.$$

When the particle hits the ground, a coefficient of restitution affects its behaviour by reducing its velocity post-collision. This value depends on ground material as it is influenced mainly by the material's particle shape, coefficient of friction and density (Yan et al., 2020). Less bouncy particles lose speed quickly and settle down sooner, forming a steeper pile (Figure 6.4 blue), or a higher talus angle like chalk. On the other hand, more bouncy particles disperse more widely upon hitting a surface, resulting in a gentler accumulation like clay (Figure 6.4 red).

Velocity field. In our model, we allow the user to add a velocity field to the environment that influences particles motion. This velocity field can be the result of a complex fluid simulation, a uniform vector field, or an artistic motion field. We modify Equation (6.6) such that the particle's speed will be influenced by the velocity field as follows:

$$v = \int \vec{F}_{\text{ext}}\, dt + w_s + \alpha v_{\text{fluid}} + v_0, \tag{6.8}$$

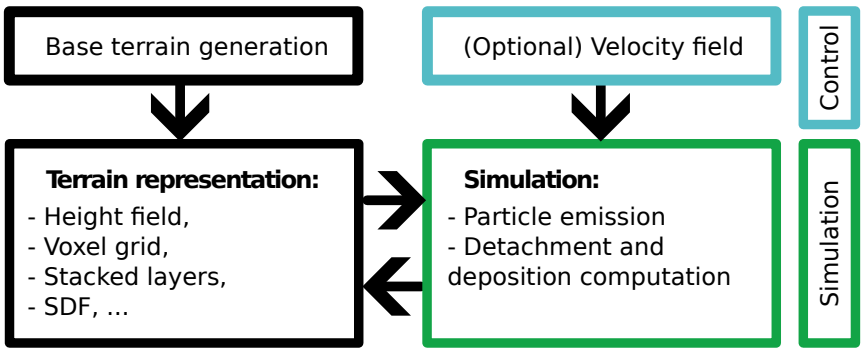


Figure 6.5: Our overall pipeline: our erosion process compute matter displacement of a terrain using an arbitrary representation as long as intersections between particles and the ground can be detected. An optional velocity field, provided by the user, guides the particles trajectories. We propose surface alteration methods to apply the erosion to the terrain in a coherent way between possible representations.

with v_{fluid} medium velocity field modulated by $\alpha \in [0, 1]$.

Our particle system can model intricate scenarios, like the erosion caused by water currents on the seabed or aeolian erosion. The velocity field remains static during the erosion, which may cause inconsistencies in the fluid velocity field. However, minor changes can be overlooked to maintain a balance between realism and computational efficiency (Tychonievich and Jones, 2010). We offer several velocity improvement methods:

-Fluid simulation refinement: Many erosion systems incorporate fluid simulation, requiring regular updates for erosion and velocity (Krištof et al., 2009; Wojtan et al., 2007). Our method can use fluid simulations with multi-resolution refinement, with the possibility to focus the velocity field adjustments near the updated boundaries of the surface (Roose et al., 2011).

-Particle velocities in fluid simulation: With a Lagrangian fluid simulation relying on particle systems (Koschier et al., 2022), our particle velocities can be incorporated in its computation. This approach is only a provisional solution due to potential parameter mismatches with main fluid simulation.

-Velocity field diffusion: Given the minor changes to the surface level at each erosion iteration, which reflect the gradual alterations in terrain surface, we can estimate that the velocity at a fixed point transitioning between the inside and outside of the terrain closely mirrors the velocities observed in its surrounding area. In this context, we can simply interpolate the velocity field at any transitioning point. This simple method, as used in Figure 6.9, allows us to find a balance between achieving realistic flow simulations and maintaining computational efficiency.

6.4 Our erosion method

In this section, we describe how to apply detachment and deposition to different terrain representations with our method (Figure 6.5). We cover the most commonly used representations namely height fields, layered terrains, voxel grid and implicit surfaces, note that our work could be extended to additional representations. Two conditions need to be satisfied for a representation to be eligible for our erosion method: being able to evaluate the intersection of a particle with the ground and compute the normal of the terrain at this point. To the best of our knowledge, all representation do.

We use Verlet integration for the particle's physics (Verlet, 1967), with low error rate and

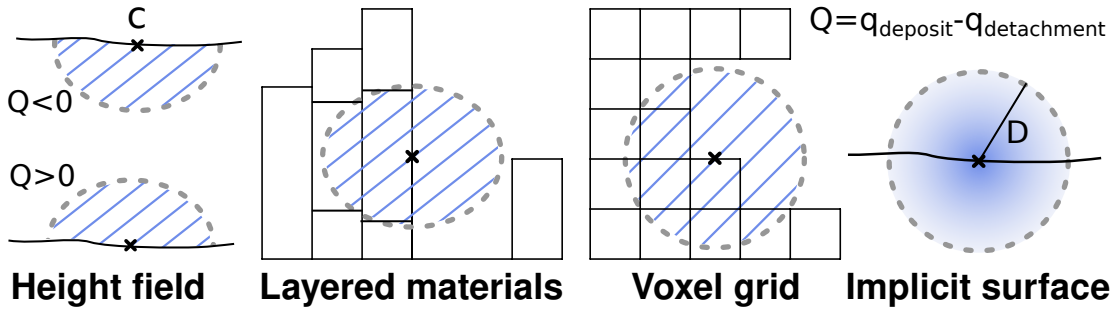


Figure 6.6: Illustration of the material detachment in the (half-)sphere at contact point C (cross) on different representations. (height field) When $Q < 0$ material detachment happen in the bottom scaled half sphere of the particle's contact with the ground, while the deposition is applied on the upper half sphere of volume when $Q > 0$. Unlike the height field, for 3D terrains detachment and deposit are applied in the full sphere around the contact point.

stability even for high dt, reducing computation time for negligible imprecision (Baraff and Witkin, 1998; Swope et al., 1982).

For all the representations, the amount of material absorbed by the particle, i.e. the erosion value $q_{\text{detachment}}$ from (6.3), is taken around the particle at a radius R, meaning that the modification of the terrain by a particle at position c will only occur for the positions $\mathbf{p}$ satisfying $||\mathbf{p} - c|| < R$. At the same time, the amount q_{deposit} from (6.4) is deposited, resulting in a change $Q = q_{\text{deposit}} - q_{\text{detachment}}$
In our simulation, while the dynamics are informed by physical principles, the particle size is conceptualized within a dimensionless framework. This provides the flexibility to adapt our results to various real-world scales, ensuring the applicability of our model across diverse scenarios. Note that, for a 2.5D terrain, we can consider that half of the sphere surrounding the particle is affected which has a volume of $V_{2.5D} = \frac{2\pi R^3}{3}$ while a 3D terrain is affected by the full sphere $V_{3D} = \frac{4\pi R^3}{3}$ (as illustrated Figure 6.6). In the following sections, we will describe the strategies used to modify the amount of matter for different representations.

6.4.1 Application on height fields

On a height field defined by $h(\mathbf{p}) = z$, the intersection point with the surface is verified at $\mathbf{p}_z = h(\mathbf{p})$, and the normal can be computed at the intersection point.

For this representation, the half sphere is scaled in the z direction to fit $\alpha V = Q$ using $\alpha = \frac{Q}{V}$. We then can decrease the height $h'(\mathbf{p})$ at all points $\mathbf{p}$ by the height of the scaled half sphere at position $\mathbf{p}$. Given the height of the scaled half sphere of center c and the distance of the particle to the center $d = ||\mathbf{p} - c||$ by $h_{\text{half sphere}}(\mathbf{p}) = \alpha\sqrt{R^2 - d^2}$ for all $\mathbf{p}$ such that $d \leq R$ the radius around a particle.

This change of height can be sampled at all points of the 2D grid by reducing the height by

$$\Delta h(\mathbf{p}) = \frac{\sqrt{R^2 - d^2}}{\alpha} = \frac{Q}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2} \tag{6.9}$$

The height at each point after an erosion is then computed as $\tilde{h}(\mathbf{p}) = h(\mathbf{p}) + \Delta h(\mathbf{p})$.

6.4.2 Application on layered terrains

Layered terrains are defined as $\mu : \mathbb{R}^3 \rightarrow \mathbb{N}$ assigning a discrete material index μ for any point in space (Beneš and Forsbach, 2001; Peytavie et al., 2009). In the original work, outer borders stack elements of the terrain are transformed into density-voxels to enable global erosion through height changes. We enable the erosion/deposition process directly on the layers hence removing the need for representation changes.
When intersecting the terrain, the amount eroded for each material stack should be the integration of the volume of the intersection between the sphere surrounding the particle and the cubicle represented by the stack. Since there is no easy solution (Jones and Williams, 2017), we approximate the volume of the stack we need to alter using the previously defined height field equation (6.9). At a distance d from the particle, the height is defined as:

$$H(d) = \frac{|Q|}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2}. \tag{6.10}$$

If $Q > 0$ (more deposition is applied that detachment) then we transform the materials in the stack contained in the sphere to become ground material. For $Q < 0$ the materials are transformed in background material.

6.4.3 Application on implicit terrains

Implicit terrain are defined using a function $f(\mathbf{p})$ and its variation resulting from the erosion process using $\Delta f(\mathbf{p})$. We propose a strategy to compute $\Delta f(\mathbf{p})$ at any point of the sphere surrounding the erosion point based on metaball primitives. At each contact point a metaball is added to create a hole or a bump in the terrain. A metaball is defined as:

$$\Delta f(\mathbf{p}) = \frac{3Q}{\pi}\frac{(1-d)}{R} \tag{6.11}$$

with d the distance of the point $\mathbf{p}$ to the sphere center. For all point $\mathbf{p}$ for which $d \geq R$, $\Delta f(\mathbf{p}) = 0$ (see 6.9).

As they are the most commonly used representations, we propose a formulation to erode implicit terrains defined by Signed Distance Functions (SDF) and by gradient or vector fields.

Signed Distance Functions Considering SDF, the terrain is defined as the 0-set of the signed distance function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, hence, for $f(\mathbf{p}) = 0$, the inside as $f(\mathbf{p}) < 0$ and outer-part (i.e. air or water) as $f(\mathbf{p}) > 0$.
The particle erosion applies at impact points at discrete positions, so we propose to add or subtract metaballs defined using equation (6.11) to respectively deposit or erode material using a composition tree:
$metaball(\mathbf{p}) = -\Delta f(\mathbf{p})$.
Now the eroded terrain function $\tilde{f}(\mathbf{p})$ will be evaluated at each point $\mathbf{p}$ from the initial terrain value $f(\mathbf{p})$, the erosion function $metaball(\mathbf{p})$ and the composition function $g(f_1, f_2)$:

$$\tilde{f}(\mathbf{p}) = g(f(\mathbf{p}), metaball(\mathbf{p})).$$

As a metaball is added for each particle bounce on the terrain space partitioning optimization algorithms such as k-d trees, BSP trees or BVH can easily be used to improve performances.

Other implicit terrains are present in the literature, notably a 2.5D representation based on the surface gradient (E. Guérin et al., 2022) and a 3D representation based on curves (Becher et al., 2017) for which the trajectory of each particle projected to the closest surface

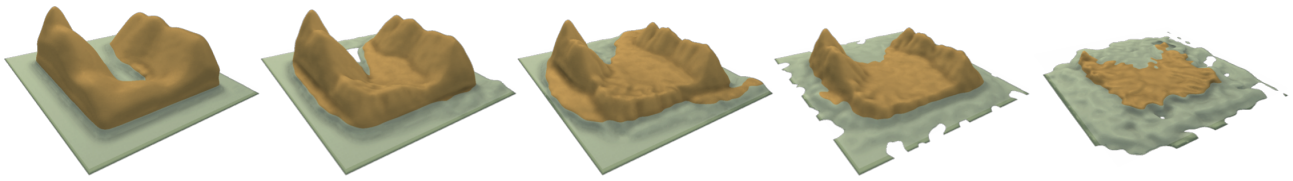


Figure 6.7: Our erosion method is applied iteratively on a completely synthetic island, the terrain is altered to obtain a plausible shape by forming rills. The use of particles with hydraulic densities dropped from the sky results in a strong erosion on the sides of the mountains, and the particles that slide to the sea are mainly drifting offshore resulting in the formation of small beaches and a weaker erosion on the bottom of the water body. Repeating the process causes the island height to decrease progressively up to the point where only the submerged part of the terrain is sheltered from erosion.

could be used to define the alteration of the terrain.
In the case of gradient-based representation, we propose to use the partial derivative from the equation of the 2D scalar fields (6.9) that gives:

$$\nabla h' = -\frac{Q}{\frac{2}{3}R^3}\frac{1}{\sqrt{R^2-d^2}}\vec{CP} \tag{6.12}$$

with $\vec{CP}$ the vector from the position $\mathbf{p}$ to evaluate to the center of the erosion point c. Now the new gradient field can be computed as:

$$\nabla\tilde{h}(\mathbf{p}) = \nabla h(\mathbf{p}) + \nabla h'(\mathbf{p}).$$

6.4.4 Application on voxel grids

We consider two of the voxel grids representations: density-voxel grids and binary voxel grids for which we present our material alternation strategy.

Density voxels. We consider "density-voxel" grids defined on $f : \mathbb{Z}^3 \to [-1, 1]$ for which a voxel is be full for $f(\mathbf{p}) = 1$, partially full for $-1 < f(\mathbf{p}) < 1$ or empty for $f(\mathbf{p}) \leq -1$. This definition allows us to erode them smoothly. Since this kind of grid is a discretizaion of a scalar function, We could directly use (6.11), as described previously, but we take advantage of the discrete nature of the representation to avoid expensive computation.
We apply the erosion from a particle at position c on all points $\mathbf{p}$ in the volume proportionally to the distance from the center of the sphere $d = ||\mathbf{p} - c||$ to find an approximation to the real erosion value per voxel $Q_{approx} = Q\frac{1-d}{R}$. Using their discrete nature, we rectify this value to sum up the total erosion value to Q by dividing each value by the sum of the distances. We now consider eroding the "empty" voxels since their density can drop until -1. We then have for all surrounding voxels:

$$\Delta f(\mathbf{p}) = Q\frac{(1-\frac{d}{R})}{\sum(1-\frac{d}{R})}. \tag{6.13}$$

Resulting voxel value is computed as $\tilde{f}(\mathbf{p}) = f(\mathbf{p}) + \Delta f(\mathbf{p})$. In our implementation, when $f(\mathbf{p}) > 1$, we simply transport the density excess to the above voxel, giving it a very close analogy to height fields as long as $|\Delta f| < 1$.

Binary voxels The terrain can be represented using an occupancy function as $f : \mathbb{Z}^3 \rightarrow \{0, 1\}$ where a voxel $f = 1$ defines the ground and $f = 0$ the background.
We propose to apply particle erosion by assigning voxels a number of hits, and transform them as air or as ground when this number reaches a critical value C that is proportional to the particle's strength parameter K_ε (Beardall et al., 2010).
On a hit, all voxels in a radius R receive a hit number:

$$\Delta hits = \lfloor \alpha \Delta f \rfloor \tag{6.14}$$

with Δf the erosion per voxel computed using (6.13) and α a coefficient high enough to obtain values above 1.
All voxels with $\#hits > C$ are transformed to background and voxels with $\#hits < -C$ are transformed to ground.
Note that, a binary voxel grid can also be transformed into a density-voxel grid to be eroded smoothly.

Our formulation for height fields (6.9), can be used to erode 2D scalar field-based representations. Similarly, our proposition for SDF (6.11) enables erosion for continuous 3D scalar fields and voxels (6.13) for discrete 3D scalar fields respectively.

6.5 Results

Our erosion process enables the simulation of a wide range of erosion effects on the major terrain representations alike. In this section, we present applications that demonstrate the versatility of our method by changing the particle's effect size, quantity, density, maximum capacity, deposition factor and the velocity fields. The results of each process are presented in Figure 6.14, parameters used are available at Table 6.1. It is important to note that all erosion examples presented in this section are available for any 3D terrain representation. However, we cannot create volumetric structure, such as overhangs, using 2.5D representations (height fields).

Environment density ρ_{fluid} is set to $1\,\text{kg}\,\text{m}^{-3}$ above water level (terrain blue part) and to $1000\,\text{kg}\,\text{m}^{-3}$ below it. Velocity field's refinement is done by using the presented diffusion strategy.

Rain. Hydraulic erosion from rain is the most common process used in terrain generation. In this case, particles are seen as water droplets falling from the sky and rolling downhill due to the gravitational force of Earth. No velocity field is required from fluid simulation. These parameters result in a detailed geometry of the rills on the side of mountains that quickly emerge and deposit many sediments in the valley. We demonstrate the result of rain erosion in *Figure 6.14: Rain* with a computation time of 4 seconds.
Using this erosion parameters in combination with water bodies results in different outcomes (Figure 6.7). The terrain above water is directly affected by the erosion process while particles colliding with the underwater part of the terrain are slowed down and filled with sediments, leading to mainly apply deposition. The result is a typical hydraulic erosion on mountains and the formation of slopes and beaches near water level.

Coastal erosion. Waves repeated motion creates coastal erosion, that can be seen as cliffs with holes at the water level.
We apply a uniform velocity field in the water pointing towards the coast to simulate waves and emit particles from the water area with a large size, a density between air and water densities, a high capacity factor and a low deposition factor ω. Using these parameters, the erosion process is focused at the interface of air and water, and apply a coarse detachment

Name	Rep.	Dimensions	Res	$\#P$	$\#N$	R	COR	ρ_{particle}	C_{factor}	ε	ω	Vel f
Rain	H	100x100	20	100	10	1.0	1.0	1000	10.0	2.5	0.3	None
Coastal	DV	100x100x30	10	80	3	5	0.1	500	10.0	5.0	0.5	Uniform
Meanders	I	N/A	N/A	10	20	5.0	1.0	1000	1.0	1.0	1.0	(1)
River	H	100x100	5	100	50	1.5-5	0.5	900	0.1	1.0	1.0	None
Landslide	H	100x100	20	200	10	2.5	0.2	500	0.1	1.0	1.0	None
Volcano	DV	100x100x40	50	150	30	1.0	5.0	2000	1.0	1.0	5.0	None
Karst	BV	100x100x50	2	1000	40	5	0.5	500	10.0	5.0	0.5	Uniform
Tunnel	DV	100x100x50	1	100	100	2.5	0.1	500	1.0	1.0	1.0	None
Wind	DV	100x100x50	0.2	100	10	1.5	0.9	1.5	1.0	1.0	1.0	(Paris e
Underwater	H	100x100	10	100	50	2.5	0.9	1000	1.0	1.0	1.0	(Stam, 2

Table 6.1: Parameters used for the generation of the terrains presented in Figure 6.14, with "Rep" the representation (H: Heightmap, DV: Density-voxels, BV: Binary voxels, I: Implicit) "Res" the resolution in meter per voxel or cell, $\#P$ the number of particles per iteration, $\#N$ the number of iterations, R the particles radius (in voxel or cell unit), COR the coefficient of restitution, ρ_{particle} the particle density in $\text{kg}\,\text{m}^{-3}$, C_{factor}, ε and ω respectively the capacity, erosion and deposition factors, "Vel field" the type of velocity field used and t the computation time of the simulation in seconds on CPU.
(1) The velocity field is a vector field defined as $v_{\text{fluid}}(\mathbf{p}) = [0\ sin(\mathbf{p}_x)\ 0]^T$.

while depositing a very small quantity of sediments, simulating the corrosive effect of water on limestone.
This effect can only be simulated on 3D terrain representations, but will create cliffs on a 2D representation. *Figure 6.14: Coastal* presents the result of coastal erosion on a density-voxel grid that creates overhangs around sea level using a small amount of particles. Note that, the same effect using an alternate implicit representation based on SDF is displayed in Figure 6.10. A shaded version of this effect is presented in Figure 6.1.

Rivers. Given a source point, we generate particles that run downhill, simulating the formation of a river. More complex erosion simulation using fluid simulations like SPH (Krištof et al., 2009) would create realistic results at the cost of high processing time. Our method offers the flexibility to be applied either with a velocity field (simple, used given or resulting from a fluid simulation) or without allowing for simplicity and efficiency.
When provided with a hand-made or procedural velocity field, our particle system can reproduce simple river meanders (*Figure 6.14: Meanders*).
Figure 6.14: River presents a river that has been modeled by emitting water particles with different sizes that ranges from 1.5 m to 5 m, a high coefficient of restitution and a low capacity factor. Random sizes are used to simulate a river for which the flow rate had fluctuated over formation time, while the low capacity ensure that the banks of the river stays smooth. A high coefficient of restitution is a strategy that let the particles flow with low friction, approaching a water behaviour. Our particles are affected only by gravity, without fluid simulation.

Landslide are mainly caused by large amount of water saturating the ground and flowing downhill, transporting matter in its path.
By using water particles with a medium size, a low coefficient of restitution and a low capacity factor but a high deposition factor ω, they transport sediments on short distances as the velocity quickly drops to 0, and ground material is completely spread along its path since it is easier to deposit the same amount of sediment than the eroded amount at each collision point. Reducing the density of the particle simulates a rise of viscosity in the settling velocity formula, increasing again the quantity of matter to deposit at contact with the ground. By

this means, we can simulate landslides as illustrated on *Figure 6.14: Landslide*. A smoother surface is resulting, compared to the rain erosion as the rills are filled with sediments as soon as they begin to form. By setting the initial capacity of the particle equal to 10% of its max capacity, the mass of the terrain increases, simulating a volcano eruption as illustrated on *Figure 6.14: Volcano*.

Karsts networks are created over hundreds of years from the corrosion of water on the limestone in the ground. A limited number of methods have been proposed for the procedural generation of karsts (Paris et al., 2021).
By reducing the deposition factor ω, the particles simulate corrosion (without mass conservation). We can use the same particle parameters than the coastal erosion (big size, a density between air and water densities, a high capacity factor and a low ω) and optionally provide a 3D shear stress map. The karst will automatically follow the softest materials, which is geologically coherent as given in example in *Figure 6.14: Karst*, where we can observe a "pillar" that is formed in the center, and thus the karst forms two corridors that finally merge partially. Underground results are only available for representations allowing 3D structures. Another underground terrain simulation is shown in *Figure 6.14: Tunnel* in which a water runoff is eroding a tunnel without the use of a fluid simulation. Here, when particles bounce often on the terrain surface, the coefficient of restitution may be seen as a viscosity parameter.

Wind erosion is a significant process in desertscapes shaping since there are no obstacles on the airflow path. Air particles can reach high velocities, transporting sand over long distances forming either dunes or are blasted into rocks, eroding into goblins.
By setting the density of our particles close to $1\,\mathrm{kg\,m^{-3}}$, two erosion simulations can be applied at once. Air particles follow closely the flowfield given by the user in air. This flowfield can be given from a complex simulation, a user-defined wind rose (Paris et al., 2019a) or a random flowfield with a general direction.
The generation of the different sand structures depends on the velocity field provided, and a simple field will easily generate linear dunes. On contact with a rock block, the simulation will automatically erode block borders, creating shapes looking like gobelins.
Figure 6.14: Wind gives an example of wind erosion on a flat surface with rock columns being eroded. Given a strong 2D velocity field computed by the high wind simulation proposed in (Paris et al., 2019a) is used on light particles, the simulation is fast thanks to the low number of collisions each particle has with the ground.

Multiple phenomena A terrain eroded with multiple erosion phenomena applied on a 500x500x50 density-voxel grid is illustrated in Figure 6.8. Here, water-density particles are applying rain on the terrain while the coasts of the river are being eroded thanks to a velocity

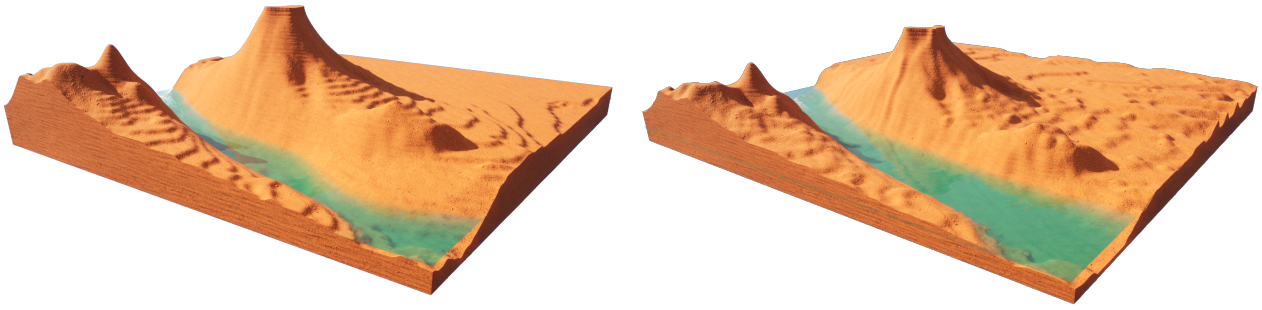


Figure 6.8: Multiple erosion types can be combined. On an initial synthetic 500x500x50 density voxel grid, the a wind erosion is applied on the surface of the terrain while hydraulic erosion shapes the rills and the base of the mountains. A water current digs its borders and spreads sediments at the bottom.

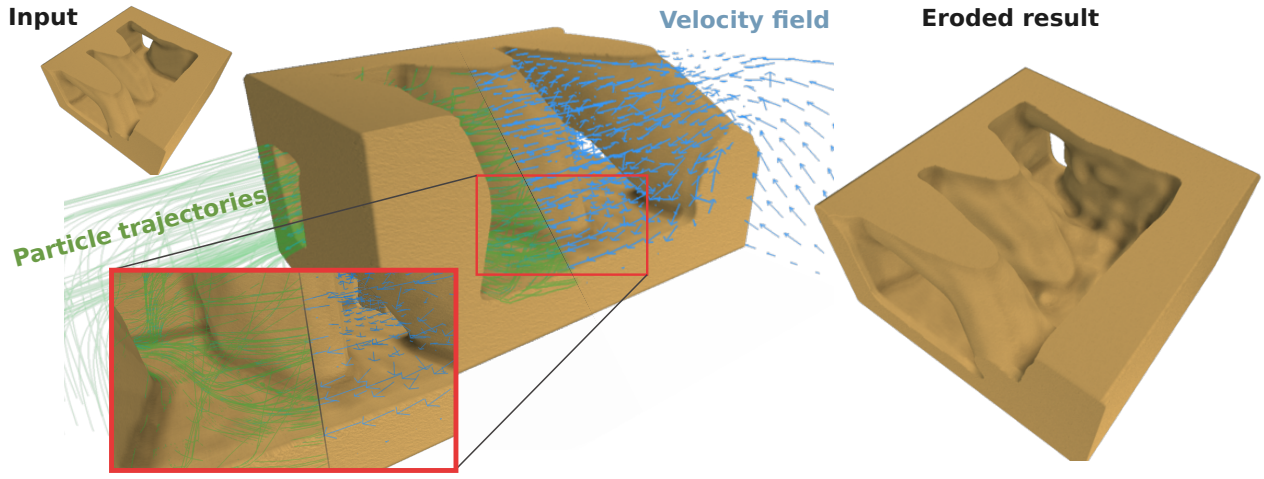


Figure 6.9: A complex water flow simulation is computed using OpenFOAM. Particle trajectories (green) are highly affected by the fluid velocity (blue). Most the terrain exposed surfaces is eroded (bottom).

field defined at the water level. The velocity field defined in the air mainly affects particles with air-density, such that wind erosion can be applied at the same time. The computation of these effects took 7 seconds on CPU.

Underwater currents Procedural generation of underwater 3D terrains has received little attention. The difference between the underwater and the surface rely on the buoyancy force that is much stronger, meaning that the water flow has a much more impacting effect on erosion than wind. Taking into account the density of the environment and the velocity field of water in our formulas are the keys to be able to apply any erosion in this environment. Our method works in a water environment by giving at least water density to particles. Given a velocity field describing underwater currents from a complex simulation or from a sketch, the particle system erodes the terrain.
In the example presented in *Figure 6.14: Underwater*, the velocity field is given by a simple 3D fluid simulation (Stam, 1999) applied on the terrain.

A complex water flow simulation is computed using SIMPLE (Caretto et al., 1973) fluid simulation with OpenFOAM. The resulting erosion can then follow complex water movement and erode the terrain at the most affected parts of the 3D terrain as the trajectories of the particles (green) is highly affected by the fluid velocity (blue). The density of the particles and the environment being close, the buoyancy cancels most of the gravity force, leaving the velocity of the particles computed by the fluid velocity v_{fluid} and settling velocity w_s from (6.8) (Figure 6.9).

6.6 Comparisons

In the following section, we compare our method with existing ones to show that while we are versatile on the terrain representation, we are also able to reproduce various effects without applying specific algorithms. The other works are displayed in blue to distinguish them from ours.

Coastal erosion on implicit terrain representation: Paris et al present an erosion simulation method applied to implicit terrains able to create coastal erosion, karsts and caves by adding negative sphere primitive in the terrain's construction tree (Paris et al., 2019b). The

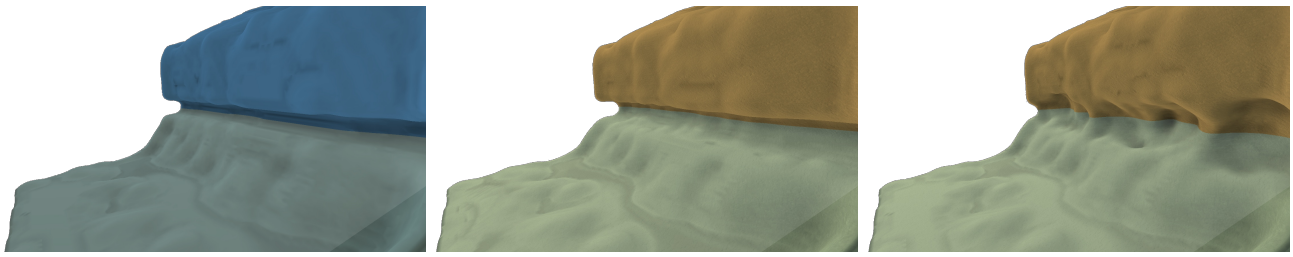


Figure 6.10: The algorithm proposed by Paris et al (Paris et al., 2019b) allows for the simulation of coastal erosion (left) that we can reproduce almost identically by allowing our particles to collide only once with the ground and applying only erosion (center). If we apply our erosion with the full tracking of our particles and using deposition, we can achieve more diverse results (right).

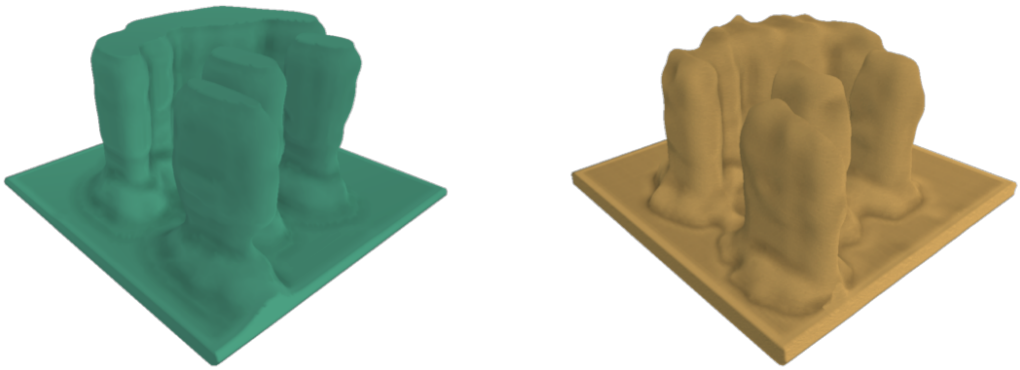


Figure 6.11: The algorithm proposed by Jones et al (Beardall et al., 2010) allows for an efficient simulation of the spheroidal erosion, making the creation of gobelins on voxel grids in a plausible way (left). Our algorithm naturally erodes the most exposed areas of the terrain when particles are affected by the wind (right).

positions of the spheres are determined using a Poisson disk sampling at the weakest terrain area defined by the Geology tree of their model. They are simulating the corrosion effect of water on the rocks. Our work is also able to approximate this phenomena by defining the position of these sphere primitives at the position where the water particles hit the surface. While the computation time of the positions of the sphere is higher due to the fact that we are evaluating the position of our particles at every time step in the implicit model (which could be improved by the triangulation of the implicit surface, or better, a dynamic triangulation), the distribution of our erosion primitives is based on a physical model instead of a mathematical model, meaning that we can integrate more easily the direction and strength of the waves for example. The management of their sphere primitives can be replicated with our method by considering that a particle exists until a collision occurs, at which point it disappears. Their method is not conserving the mass of the terrain, which is acceptable for the corrosion simulation, but limits its validity for other erosion simulations. In our method, the particle can be tracked until it settles, ensuring mass conservation (Figure 6.10);

Wind erosion on voxel grid representation: Jones et al propose a weathering erosion on voxel grids by approximating and eroding continuously the most exposed voxels (Beardall et al., 2010). When a solid voxel is decimated, it is considered deposit and is displaced down the slope until a minimal talus angle in the terrain is reached and if the deposition

is eroded again, it disappears. Our work is able to reproduce their algorithm by sending our particles from a close distance to the terrain surface. By doing so, we reproduce the erosion process as much as the deposition process since the air particles, filled with sediments, is falling automatically towards the local minimum of the erosion point. Just like in their work, we can easily define the resistance value of the materials to add diversity in the results. By adding the possibility of a wind field, even a very simple uniform vector field, to the simulation, we naturally add the wind shadowing effect that protects a gobelin surrounded by bigger gobelins, and also allows the deposit slope to fit more closely to the wind direction (Figure 6.11).

Hydraulic erosion on height field representation: Mei et al integrate and adapt to the GPU the pipe model proposed in (O'Brien and Hodgins, 1995) for the fluid simulation (Mei et al., 2007). This simulation is simple but efficient enough to approximate the Shallow-Water equations in real time and use the speed of columns of water to compute the erosion and deposition rate on the 2D grid of the terrain at each time step. Using columns of water even allows the flow to overpass small bumps on the terrain over time. Our method initially rely on a stable fluid flow that is consistent during the whole life time of a particle, but by refining the simulation at each time step instead of at the end of the particles lifetime, our erosion model is able to reproduce this effect, allowing the terrain to have a single batch of fluid going through it. Our method can be seen as a generalization of Mei at al. that can then be used on more than discrete 2D grids (Figure 6.12).

Wind erosion on stacked materials representation: Paris et al (Paris et al., 2019a) simulate the effect of wind over sand fields defined on stacked materials, creating dune structures, even taking into account obstacles like (Roa and Benes, 2004) and different material layers like vegetation (Cordonnier et al., 2017a) that are not affected by abrasion(Paris et al., 2019a). A wind field simulation is required to produce results, and while (Roa and Benes, 2004) and (Onoue and Nishita, 2000) consider a uniform vector field, this work consider a dynamic vector multi-scaled warped field from the terrain height. The sand grains then apply multiple moves: sand lift, bounces, reptation and avalanching. Once the sand is lifted by the wind, the trajectory of the grains can be seen as the displacement of particles, fitting completely with our model as illustrated Figure 6.13.

6.7 Discussion

This work is a generalization of erosion that is applicable to any terrain representation. In practice, while similar particle physics is used on different terrain representations, using similar parameters does not ensure resulting in the same eroded terrain. Surfaces and normals being approximated differently have rippling effect on particle trajectories. Note that, not all effects can be applied to all representations, forinstance, karsts generation on 2.5D data structures.

Realism Realism of the erosion simulation is highly correlated to the size and quantity of particles used and their distribution. Using too few or distributing them too sparsely will result in a terrain that is unrealistic since the alteration will have localized effects, breaking process homogeneity.
The resolution is also limited by the number and size of the particles, which can be problematic on implicit terrains that can theoretically have a infinite resolution.
Our method allows to perform erosion on implicit terrains. However, in its current form, our algorithm is time expensive on implicit representations since a large number of primitives are added in the composition tree. Using skeletons-defined primitives (Hong, 2013; Rigaudière

et al., 2000)from particles trajectories and erosion/deposition values could be a solution to optimize the computation time.

Usage of velocity fields In our erosion algorithm, we simplify particle physics to enhance computational efficiency and facilitate parameterization. We use the velocity field from fluid simulations to approximate particle velocities. Sediment mass is harnessed to compensate for this approximation, allowing compatibility with various fluid simulation algorithms. Velocity fields can be recomputed at a frequency meeting the applications needs, ranging from "classic erosion simulation" (recomputed at each time step) to "simple simulation" (never recomputed). We addressed provisional adjustments to mitigate discrepancies when terrain changes due to erosion are not reflected in a static velocity field in section “3.3 Transport”. However, it is important to note that these are expedient solutions and may not fully capture precise dynamics of an evolving terrain.

Performances To facilitate parallelization, we intentionally overlook particle interactions and sediment exchanges, albeit at the expense of achieving smoother results. Surface collisions are simplified to basic bounces with a damping parameter instead of relying on complex particles and ground properties (Young’s modulus, friction, material, ...)(Yan et al., 2020), further easing the parameterization process. However, these simplifications, combined with the inherent discrete nature of particles, as opposed to the continuous nature of erosion, result in a correlation between realism and particle count.
The performance of our method is influenced by the time required for collision detection. Consequently, we mainly observe better performances with explicit terrain models than with implicit models.

Particle’s atomicity While we can replicate various effects, the "fan" shape commonly observed in natural erosion patterns is not perfectly represented. This limitation arises because we do not account for the splitting of a particle, a process that significantly influences the multidirectional dislocation and trajectory of individual particles (Ranz et al., 1960). Additionally, we acknowledge an issue where particles may collide with the ceiling and the deposition is stuck. While a potential resolution involves splitting particles upon impact rather than simply depositing sediments, this introduces complexities to the parallelization layer of the method. Allowing particles to split introduces unpredictability in the total number of particles that will exist in the simulation. This unpredictability can complicate the use of multi-threading. Future works includes finding a data structure allowing this splitting efficiently, leading to more realistic erosion patterns.

Simulation with multiple materials One aspect we haven’t addressed is a layered terrain with multiple materials. In the native way our method is done, we do not consider the transport of different materials (all sediments are considered as sand), but by storing a list of the different materials and the quantity transported by each particle, the same simulation process could be done at the cost of some memory and performance overhead.
Another possible adaptation of the erosion strategy for material voxels is to extend the erosion computation from binary voxels by define transformation rules from one material to another when a voxel is eroded a number $\#hits < -C$ or $\#hits > C$. For example, the material "clay" may transform to "sand" when eroded or to "rock" when many depositions occurred.

6.8 Conclusion

We introduced a flexible particle-based erosion system that is easy to use and simple to implement. We have presented how to adapt the process for various terrain representations and generate a variety of erosion phenomenon due to rain, wind, water bodies... by adjusting

intuitive parameters hence generate automatically realistic 2.5D and 3D terrains. The use of external velocity fields provides a high flexibility i.e. using the simulations that best fits the user's needs (precision, control, implementation efficiency...). Our method can also be applied to underwater environments with identical physics simulation since our erosion method can be applied on 3D representations. Erosion algorithms are often limited to the use of height fields, but by finding more generalized methods, we can go toward a global use of 3D terrains, which can offer richer and more diverse landscapes.

6.9 Computation of a metaball

We use the following formula to evaluate a metaball in space with a center c and of radius R:

$$g(\mathbf{p}) = 1 - \frac{||\mathbf{p} - c||}{R}$$

using the euclidean distance.

We have a total amount Q to define in this space, so the final metaball function f needs to satisfy the equations (6.15) and (6.16):

$$f(\mathbf{p}) = \lambda g(\mathbf{p}) \tag{6.15}$$

$$\int_{\mathbf{p} \in V_{3D}} f \, dp = Q \tag{6.16}$$

First, let's exploit the radial symmetry of the metaball and rewrite $g(\mathbf{p}) = 1 - r$ by using the polar coordinates of the point $\mathbf{p} - c$.

We can then integrate g over the volume V_{3D} as

$$\begin{aligned} &\int_0^1 \int_0^\pi \int_0^{2\pi} g(r) r^2 \sin(\theta) \, dr \, d\theta \, d\phi \\ =&\int_0^1 \int_0^\pi \int_0^{2\pi} (1 - r) r^2 \sin(\theta) \, dr \, d\theta \, d\phi \\ =&\int_0^1 (1 - r) r^2 \, dr \times \int_0^\pi \sin\theta \, d\theta \times \int_0^{2\pi} 1 \, d\phi \end{aligned}$$

We then break down the integrals one by one such as

$$\int_0^1 (1 - r) r^2 \, dr = \frac{1}{12}$$

$$\int_0^\pi \sin\theta \, d\theta = 2$$

$$\int_0^{2\pi} 1 \, d\phi = 2\pi$$

By combining all these integrals, we get $\int g = \frac{1}{12} \times 2 \times 2\pi = \frac{\pi}{3}$.

So given $\int f = q_{\text{detachment}}$ and $\int f = \lambda \int g$, we can deduce that $\lambda = \frac{Q}{\int g} = \frac{3}{\pi} Q$.

From (6.15) we finally get

$$f(\mathbf{p}) = \frac{3Q}{\pi} \left(1 - \frac{||\mathbf{p} - c||}{R}\right) \tag{6.17}$$

, representing the rate of change on the evaluation function of the terrain surface.

The integration in the voxel space is out of the scope of this paper and a numerical solution is instead proposed in Section 6.4.4.

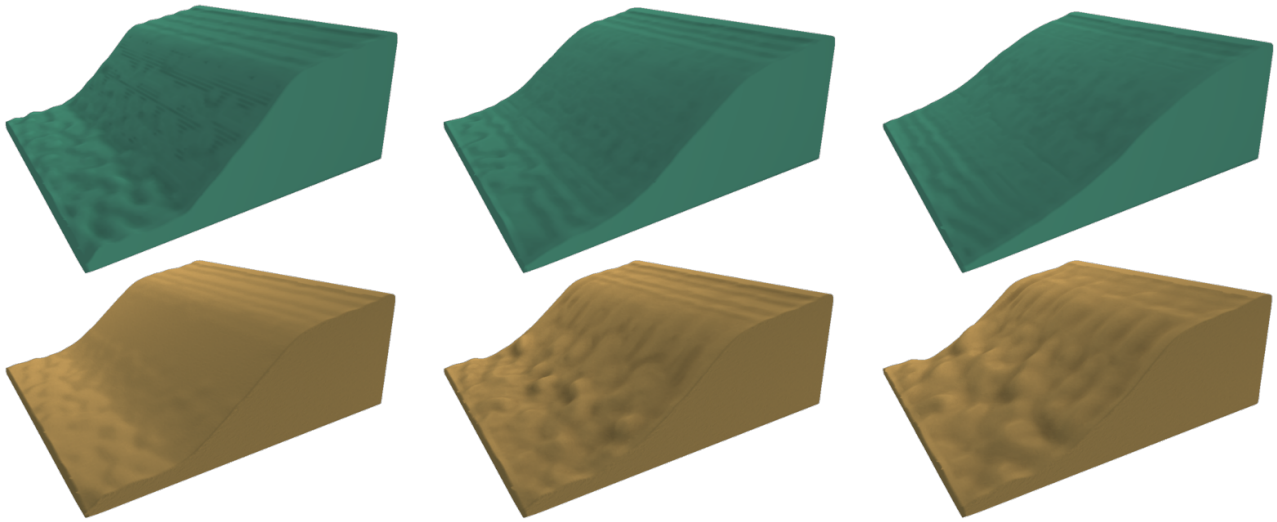


Figure 6.12: While our resulting geometry on the hydraulic erosion (bottom) is less smoothed than the one proposed proposed by Mei et al. (Mei et al., 2007) (top), our method allows the application on more terrain representations than the height fields only.

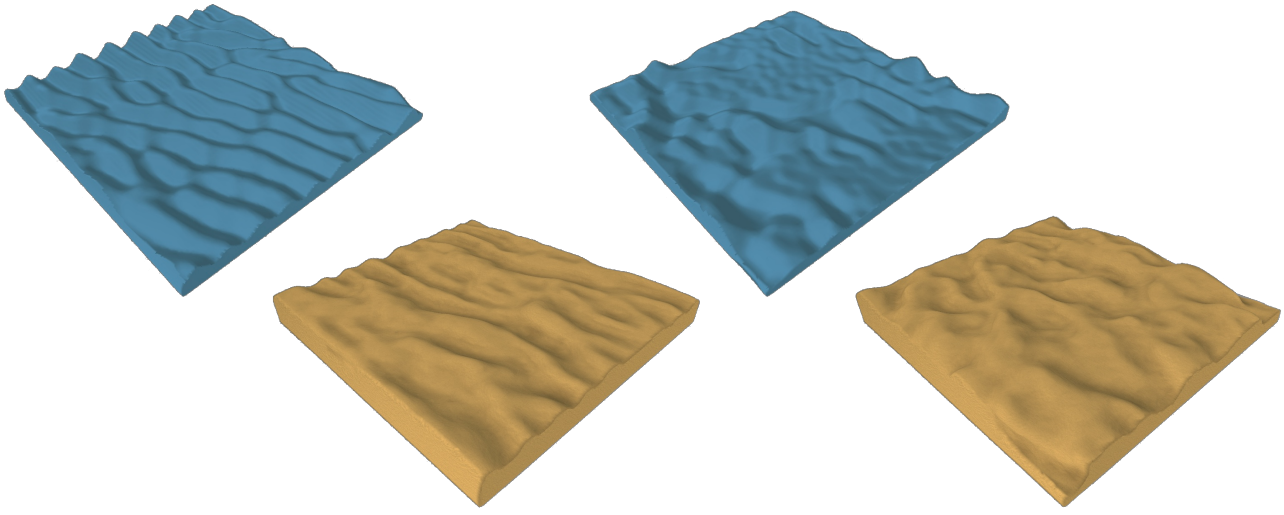


Figure 6.13: The algorithm from Paris 2020 allow the generation of desertscapes (top), which we can (at least partially) reproduce with our erosion simulation (bottom). The different effects are achieved by affecting the wind direction and strength.

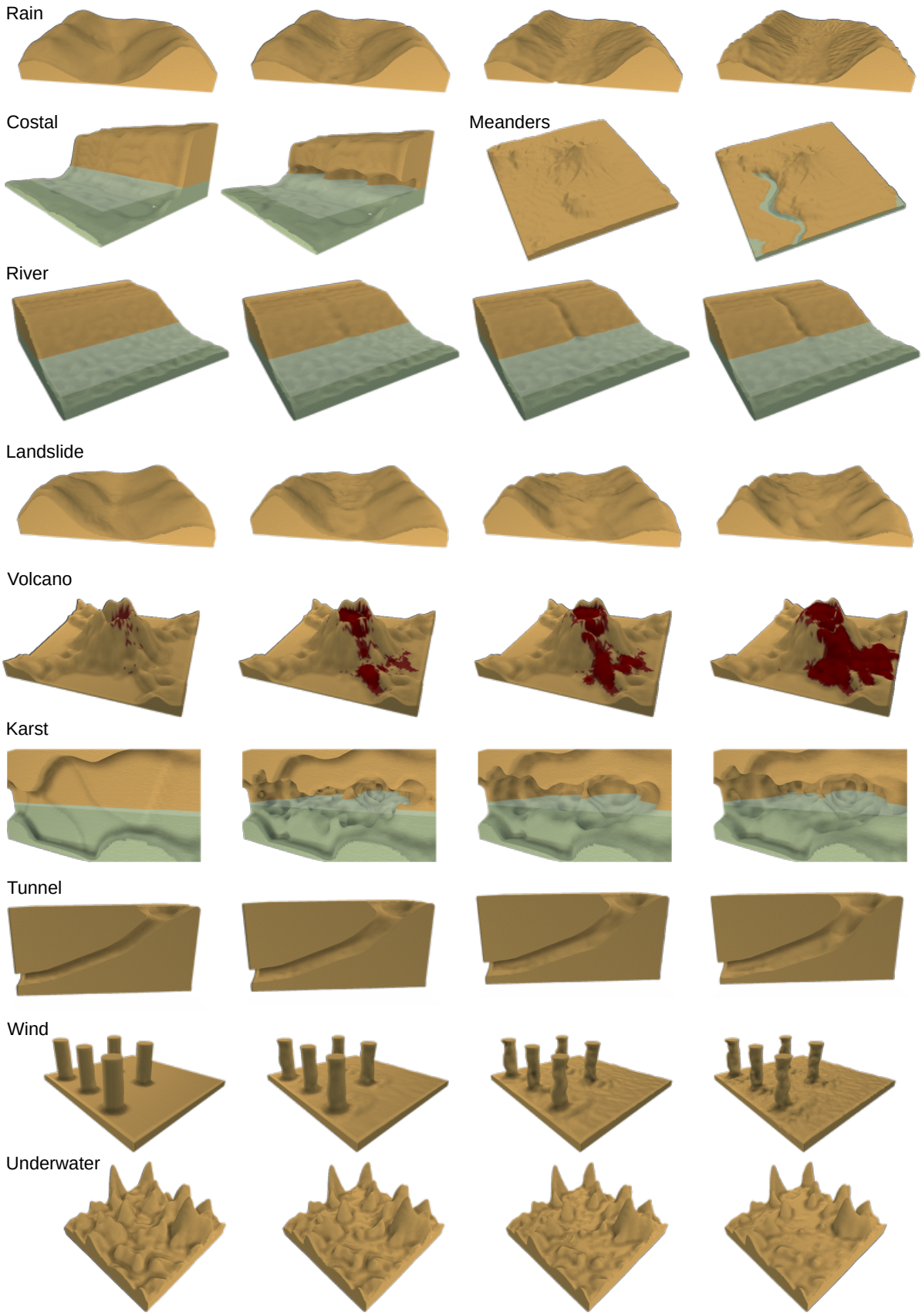


Figure 6.14: Erosion processes results on various representations presented in section 6.5. Used parameters used are detailed in Table 6.1.

CHAPTER 7

Conclusion

- ...

References

Abela, R., Liapis, A., & Yannakakis, G. N. (2015). A constructive approach for the generation of underwater environments. *Proceedings of the FDG workshop on Procedural Content Generation in Games* (cit. on p. 15).

Argudo, O., Galin, E., Peytavie, A., Paris, A., & Guérin, E. (2020). Simulation, modeling and authoring of glaciers. *ACM Transactions on Graphics*, *39*, 1–14. https://doi.org/10.1145/3414685.3417855 (cit. on pp. 48, 50).

Baraff, D., & Witkin, A. (1998). Large steps in cloth simulation. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998*, 43–54. https://doi.org/10.1145/280814.280821 (cit. on p. 55).

Beardall, M., Butler, J., Farley, M., & Jones, M. D. (2010). Directable weathering of concave rock using curvature estimation. *IEEE Transactions on Visualization and Computer Graphics*, *16*, 81–94. https://doi.org/10.1109/TVCG.2009.39 (cit. on pp. 58, 62).

Becher, M., Krone, M., Reina, G., & Ertl, T. (2017). Feature-based volumetric terrain generation. *Proceedings - I3D 2017: 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. https://doi.org/10.1145/3023368.3023383 (cit. on p. 56).

Beneš, B., & Forsbach, R. (2001). Layered data representation for visual simulation of terrain erosion. *Proceedings - Spring Conference on Computer Graphics, SCCG 2001*, 80–86. https://doi.org/10.1109/SCCG.2001.945341 (cit. on pp. 50, 56).

Beneš, B., Těšínský, V., Hornyš, J., & Bhatia, S. K. (2006). Hydraulic erosion. *Computer Animation and Virtual Worlds*, *17*, 99–108. https://doi.org/10.1002/cav.77 (cit. on p. 50).

BOSSCHER, H., & SCHLAGER, W. (1992). Computer simulation of reef growth. *Sedimentology*, *39*, 503–512. https://doi.org/10.1111/j.1365-3091.1992.tb02130.x (cit. on p. 15).

Brosz, J., Samavati, F. F., & Sousa, M. C. (2007). Terrain synthesis by-example. *Communications in Computer and Information Science*, *4 CCIS*, 58–77. https://doi.org/10.1007/978-3-540-75274-5_4 (cit. on p. 15).

Caretto, L. S., Gosman, A. D., Patankar, S. V., & Spalding, D. B. (1973). Two calculation procedures for steady, three-dimensional flows with recirculation. *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, *19*, 60–68 (cit. on p. 61).

Cordonnier, G., Braun, J., Cani, M.-P., Benes, B., Galin, É., Peytavie, A., & Guérin, É. (2016). Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum*, *35*, 165–175. https://doi.org/10.1111/cgf.12820 (cit. on pp. 37, 50).

Cordonnier, G., Cani, M.-P., Beneš, B., Braun, J., & Galin, É. (2017a). Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE Transactions on Visualization and Computer Graphics*, *24*. https://doi.org/10.1109/TVCG.2017.2689022 (cit. on pp. 14, 37, 48, 50, 63).

Cordonnier, G., Ecormier-nocca, P., Galin, É., Gain, J., Beneš, B., & Cani, M.-P. (2018). Interactive generation of time-evolving, snow-covered landscapes with avalanches. *Computer Graphics Forum*, *37*, 497–509. https://doi.org/10.1111/cgf.13379 (cit. on pp. 48, 50).

Cordonnier, G., Galin, É., Gain, J., Beneš, B., Guérin, É., Peytavie, A., & Cani, M.-P. (2017b). Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3072959.3073667 (cit. on pp. 48, 50).

Cordonnier, G., Jouvet, G., Peytavie, A., Braun, J., Cani, M.-P., Benes, B., Galin, E., Guérin, E., & Gain, J. (2023). Forming terrains by glacial erosion. *ACM Transactions on Graphics*, *42*, 14. https://doi.org/10.1145/3592422\"{\i} (cit. on pp. 14, 15, 48).

Cortial, Y., Peytavie, A., Galin, É., & Guérin, É. (2019). Procedural tectonic planets. *Eurographics Computer Graphics Forum*, *38*, 1–11. https://doi.org/10.1111/cgf.13614 (cit. on pp. 14, 15).

Cowart, L., Walsh, J. P., & Corbett, D. R. (2010). Analyzing estuarine shoreline change: A case study of cedar island, north carolina. *Journal of Coastal Research*, *265*, 817–830. https://doi.org/10.2112/jcoastres-d-09-00117.1 (cit. on p. 23).

Daly, R. A. . (1915). The glacial-control theory of coral reefs. *Proceedings of the American Academy of Arts and Sciences*, *51*, 157–251 (cit. on p. 36).

de Carpentier, G. J. P., & Bidarra, R. (2009). Interactive gpu-based procedural heightfield brushes. *Proceedings of the 4th International Conference on Foundations of Digital Games*, *8*, 55–62. https://doi.org/10.1145/1536513.1536532 (cit. on p. 14).

Dey, R., Doig, J. G., & Gatzidis, C. (2018). Procedural feature generation for volumetric terrains using voxel grammars. *Entertainment Computing*, *27*, 128–136. https://doi.org/10.1016/j.entcom.2018.04.003 (cit. on p. 50).

Domínguez, L., Anfuso, G., & Gracia, F. J. (2005). Vulnerability assessment of a retreating coast in sw spain. *Environmental Geology*, *47*, 1037–1044. https://doi.org/10.1007/s00254-005-1235-0 (cit. on p. 23).

Droxler, A. W., & Jorry, S. J. (2021). The origin of modern atolls : Challenging darwin's deeply ingrained theory. *Annual Review of Marine Science*. https://doi.org/https://doi.org/10.1146/annurev-marine-122414-034137 (cit. on p. 36).

Dunbabin, M., Manley, J., & Harrison, P. L. (2020). Uncrewed maritime systems for coral reef conservation. *2020 Global Oceans 2020: Singapore - U.S. Gulf Coast*. https://doi.org/10.1109/IEEECONF38699.2020.9389173 (cit. on p. 14).

Eisemann, E., & Decoret, X. (2008). Single-pass gpu solid voxelization for real-time applications. *Proceedings - Graphics Interface*, 73–80 (cit. on p. 50).

Emilien, A., Poulin, P., Cani, M.-P., & Vimont, U. (2015). Interactive procedural modelling of coherent waterfall scenes. *Computer Graphics Forum*, *34*, 22–35. https://doi.org/10.1111/cgf.12515 (cit. on p. 49).

Gain, J., Marais, P., & Straßer, W. (2009). Terrain sketching. *Proceedings of I3D 2009: The 2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, *1*, 31–38. https://doi.org/10.1145/1507149.1507155 (cit. on pp. 15, 37, 49).

Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.-P., Benes, B., & Gain, J. (2019). A review of digital terrain modeling. *Computer Graphics Forum*, *38*, 553–577. https://doi.org/10.1111/cgf.13657 (cit. on pp. 15, 47, 49).

Goes, F. D., & James, D. L. (2017). Regularized kelvinlets: Sculpting brushes based on fundamental solutions of elasticity. *ACM Transactions on Graphics*, *36*, 401–411. https://doi.org/10.1145/3072959.3073595 (cit. on p. 20).

Grosbellet, F., Peytavie, A., Guérin, É., Galin, É., Mérillou, S., & Benes, B. (2016). Environmental objects for authoring procedural scenes. *Computer Graphics Forum*, *35*, 296–308. https://doi.org/10.1111/cgf.12726 (cit. on p. 15).

Guérin, E., Peytavie, A., Masnou, S., Digne, J., Sauvage, B., Gain, J., & Galin, E. (2022). Gradient terrain authoring. *Computer Graphics Forum*, *41*, 85–95. https://doi.org/10.1111/cgf.14460 (cit. on pp. 14, 50, 56).

Guérin, É., Digne, J., Galin, É., & Peytavie, A. (2016). Sparse representation of terrains for procedural modeling. *Computer Graphics Forum*, *35*, 177–187. https://doi.org/10.1111/cgf.12821 (cit. on pp. 15, 50).

Guérin, É., Digne, J., Galin, É., Peytavie, A., Wolf, C., Beneš, B., & Martinez, B. (2017). Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3130800.3130804 (cit. on p. 15).

Hong, Q. (2013). A skeleton-based technique for modelling implicit surfaces. *Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013*, *2*, 686–691. https://doi.org/10.1109/CISP.2013.6745253 (cit. on p. 63).

Ito, T., Fujimoto, T., Muraoka, K., & Chiba, N. (2003). Modeling rocky scenery taking into account joints. *Proceedings of Computer Graphics International Conference, CGI*, *2003-Janua*, 244–247. https://doi.org/10.1109/CGI.2003.1214475 (cit. on p. 50).

Jones, B. D., & Williams, J. R. (2017). Fast computation of accurate sphere-cube intersection volume. *Engineering Computations*, *34*, 1204–1216. https://doi.org/10.1108/EC-02-2016-0052 (cit. on p. 56).

Kapp, K., Gain, J., Guérin, E., Galin, E., & Peytavie, A. (2020). Data-driven authoring of large-scale ecosystems. *ACM Transactions on Graphics*, *39*. https://doi.org/10.1145/3414685.3417848 (cit. on p. 15).

Kaufman, A., Cohen, D., & Yagel, R. (1993). Volume graphics. *Computer*, *26*, 51–64. https://doi.org/10.1109/MC.1993.274942 (cit. on p. 50).

Koschier, D., Bender, J., Solenthaler, B., & Teschner, M. (2022). A survey on sph methods in computer graphics. *Computer Graphics Forum*, *41*, 737–760. https://doi.org/10.1111/cgf.14508 (cit. on p. 54).

Krištof, P., Beneš, B., Křivánek, J., & Šťava, O. (2009). Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, *28*, 219–228. https://doi.org/10.1111/j.1467-8659.2009.01361.x (cit. on pp. 49–51, 54, 59).

Lengyel, E. (2010). Voxel-based terrain for real-time virtual simulations, 148 (cit. on p. 50).

Li, W. (2021). Procedural modeling of the great barrier reef. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13017 LNCS*, 381–391. https://doi.org/10.1007/978-3-030-90439-5_30 (cit. on p. 15).

Mareschal, J. C. (1989). Fractal reconstruction of sea-floor topography. *Pure and Applied Geophysics PAGEOPH*, *131*, 197–210. https://doi.org/10.1007/BF00874487 (cit. on p. 15).

Maslin, M., Louis, S., Dejean, K. G., Lapierre, L., Villéger, S., & Claverie, T. (2021). Underwater robots provide similar fish biodiversity assessments as divers on coral reefs. *Remote Sensing in Ecology and Conservation*, *7*, 567–578. https://doi.org/10.1002/rse2.209 (cit. on p. 14).

Mei, X., Decaudin, P., & Hu, B. G. (2007). Fast hydraulic erosion simulation and visualization on gpu. *Proceedings - Pacific Conference on Computer Graphics and Applications*, 47–56. https://doi.org/10.1109/PG.2007.27 (cit. on pp. 14, 50, 63, 66).

Michel, E., Emilien, A., & Cani, M.-P. (2015). Generation of folded terrains from simple vector maps. *Eurographics 2015 short paper proceedings*, 4–8. https://doi.org/10.2312/egsh.20151019 (cit. on p. 15).

Musgrave, F. K., Kolb, C. E., & Mace, R. S. (1989). The synthesis and rendering of eroded fractal terrains. *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989*, 41–50. https://doi.org/10.1145/74333.74337 (cit. on pp. 14, 15, 47, 50).

Neidhold, B., Wacker, M., & Deussen, O. (2005). Interactive physically based fluid and erosion simulation. *Natural Phenomena*, 25–32 (cit. on pp. 48, 50).

O'Brien, J. F., & Hodgins, J. K. (1995). Dynamic simulation of splashing fluids. *Proceedings Computer Animation, CA 1995*, 198–205. https://doi.org/10.1109/CA.1995.393532 (cit. on p. 63).

Olsen, J. (2004). Realtime procedural terrain generation. *Department of Mathematics And Computer Science (. . .*, 20 (cit. on p. 47).

Onoue, K., & Nishita, T. (2000). A method for modeling and rendering dunes with wind-ripples. *Proceedings - Pacific Conference on Computer Graphics and Applications*, *2000-Janua*, 427–428. https://doi.org/10.1109/PCCGA.2000.883978 (cit. on p. 63).

Oron, S., Akkaynak, D., Tchernov, B. N. G., & Shaked, Y. (2023). How monster storms shape fringing reefs: Observations from the 2020 middle east cyclone. *Ecosphere*, *14*. https://doi.org/10.1002/ecs2.4602 (cit. on p. 23).

Palmer, M. R., Shagude, Y. W., Roberts, M. J., Popova, E., Wihsgott, J. U., Aswani, S., Coupland, J., Howe, J. A., Bett, B. J., Osuka, K. E., Abernethy, C., Alexiou, S., Painter, S. C., Kamau, J. N., Nyandwi, N., & Sekadende, B. (2021). Marine robots for coastal ocean research in the western indian ocean. *Ocean and Coastal Management*, *212*. https://doi.org/10.1016/j.ocecoaman.2021.105805 (cit. on p. 14).

Paris, A., Guérin, E., Peytavie, A., Collon, P., & Galin, E. (2021). Synthesizing geologically coherent cave networks. *Computer Graphics Forum*, *40*, 277–287. https://doi.org/10.1111/cgf.14420 (cit. on p. 60).

Paris, A., Peytavie, A., Guérin, E., Argudo, O., & Galin, E. (2019a). Desertscape simulation. *Computer Graphics Forum*, *38*, 47–55. https://doi.org/10.1111/cgf.13815 (cit. on pp. 20, 50, 60, 63).

Paris, A., Galin, E., Peytavie, A., Guérin, E., & Gain, J. (2019b). Terrain amplification with implicit 3d features. *ACM Transactions on Graphics*, *38*, 1–15. https://doi.org/10.1145/3342765 (cit. on pp. 15, 50, 59, 61, 62).

Patel, D., Natali, M., Lidal, E. M., Parulek, J., Brazil, E. V., & Viola, I. (2021). Modeling terrains and subsurface geology. *Interactive Data Processing and 3D Visualization of the Solid Earth*, 1–43. https://doi.org/10.1007/978-3-030-90716-7_1 (cit. on pp. 15, 37).

Peytavie, A., Galin, E., Grosjean, J., & Merillou, S. (2009). Arches: A framework for modeling complex terrains. *Computer Graphics Forum*, *28*, 457–467. https://doi.org/10.1111/j.1467-8659.2009.01385.x (cit. on pp. 50, 56).

Prusinkiewicz, P., & Hammel, M. (1993). Fractal model of mountains with rivers. *Proceedings - Graphics Interface*, 174–180 (cit. on p. 14).

Ranz, W. E., Talandis, G. R., & Gutterman, B. (1960). Mechanics of particle bounce. *AIChE Journal*, *6*, 124–127. https://doi.org/10.1002/aic.690060123 (cit. on p. 64).

Richardson, J. F., & Zaki, W. N. (1954). The sedimentation of a suspension of uniform spheres under conditions of viscous flow. *Chemical Engineering Science*, *3* (cit. on p. 53).

Rigaudière, D., Gesquière, G., & Faudot, D. (2000). Shape modelling with skeleton based implicit primitives. *Methods* (cit. on p. 63).

Roa, T., & Benes, B. (2004). Simulating desert scenery. *Winter School of Computer Graphics SHORT communication Papers Proceedings*, 17–22 (cit. on pp. 50, 63).

Roose, D., Leuven, K. U., & López, Y. R. (2011). Dynamic refinement for fluid flow simulations with sph particle refinement for fluid flow simulations with sph. (Cit. on p. 54).

Roudier, P. (1993). Synthèse de paysages réalistes par simulation de processus d'érosion (cit. on p. 47).

Schott, H., Paris, A., Fournier, L., Guérin, E., & Galin, E. (2023). Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, *42*, 1–15. https://doi.org/10.1145/3592787 (cit. on p. 15).

Smelik, R. M., Kraker, K. J. D., Groenewegen, S. A., Tutenel, T., & Bidarra, R. (2009). A survey of procedural methods for terrain modelling. *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)* (cit. on p. 47).

Smelik, R. M., Tutenel, T., Bidarra, R., & Benes, B. (2014). A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, *33*, 31–50. https://doi.org/10.1111/cgf.12276 (cit. on p. 14).

Stachniak, S., & Stuerzlinger, W. (2005). An algorithm for automated fractal terrain deformation. *In Proceedings of Computer Graphics and Artificial Intelligence*, 64–76 (cit. on p. 47).

Stam, J. (1999). Stable fluids. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999*, 121–128. https://doi.org/10.1145/311535.311548 (cit. on p. 61).

Stam, J. (2003). Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics*, *22*, 724–731. https://doi.org/10.1145/882262.882338 (cit. on p. 59).

Stokes, G. G. (2009, July). On the effect of the internal friction of fluids on the motion of pendulums. Cambridge University Press. https://doi.org/10.1017/CBO9780511702266.002 (cit. on p. 53).

Swope, W. C., Andersen, H. C., Berens, P. H., & Wilson, K. R. (1982). A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics*, *76*, 637–649. https://doi.org/10.1063/1.442716 (cit. on p. 55).

Talgorn, F. X., & Belhadj, F. (2018). Real-time sketch-based terrain generation. *ACM International Conference Proceeding Series*, 13–18. https://doi.org/10.1145/3208159.3208184 (cit. on p. 15).

Tychonievich, L. A., & Jones, M. D. (2010). Delaunay deformable mesh for the weathering and erosion of 3d terrain. *Visual Computer*, *26*, 1485–1495. https://doi.org/10.1007/s00371-010-0506-2 (cit. on p. 54).

Verlet, L. (1967). Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical Review*, *159*, 98–103. https://doi.org/10.1103/PhysRev.159.98 (cit. on p. 54).

Vila-Concejo, A., & Kench, P. (2016, August). Storms in coral reefs. Wiley Blackwell. https://doi.org/10.1002/9781118937099.ch7 (cit. on p. 23).

Wejchert, J., & Haumann, D. (1991). Animation aerodynamics. *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991*, *25*, 19–22. https://doi.org/10.1145/122718.122719 (cit. on pp. 15, 20).

Williams, S. B., Pizarro, O., Steinberg, D. M., Friedman, A., & Bryson, M. (2016). Reflections on a decade of autonomous underwater vehicles operations for marine survey at

the australian centre for field robotics. *Annual Reviews in Control*, *42*, 158–165. https://doi.org/10.1016/j.arcontrol.2016.09.010 (cit. on p. 14).

Wojtan, C., Carlson, M., Mucha, P. J., & Turk, G. (2007). Animating corrosion and erosion. *Natural Phenomena*, 15–22 (cit. on pp. 51–54).

Yan, P., Zhang, J., Kong, X., & Fang, Q. (2020). Numerical simulation of rockfall trajectory with consideration of arbitrary shapes of falling rocks and terrain. *Computers and Geotechnics*, *122*. https://doi.org/10.1016/j.compgeo.2020.103511 (cit. on pp. 53, 64).