Contents

Chapter 1

Erosion simulation

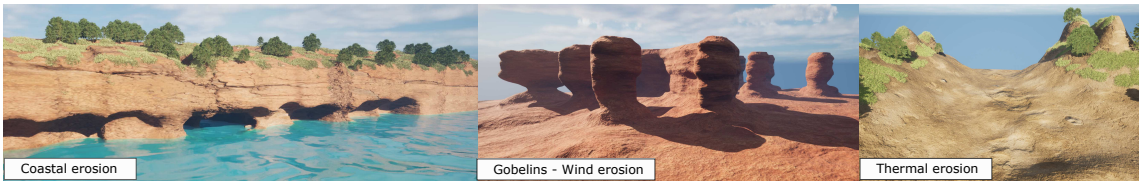

Figure 1.1: *Applying shading and textures on the generated geometry can produce a plausible aspect of a coast eroded by waves on a long timespan, or a desertic landscape eroded by wind, or a mountainous area flatten by thermal erosion.*

Abstract

In this chapter, we present a novel particle-based method for simulating erosion on various terrain representations, including height fields, voxel grids, material layers, and implicit terrains. Our approach breaks down erosion into two key processes (terrain alteration and material transport) allowing for flexibility in simulation. We utilise independent particles governed by basic particle physics principles, enabling efficient parallel computation. For increased precision, a vector field can adjust particle speed, adaptable for realistic fluid simulations or user-defined control. We address material alteration in 3D terrains with a set of equations applicable across diverse models, requiring only per-particle specifications for size, density, coefficient of restitution, and sediment capacity. Our modular algorithm is versatile for real-time and offline use, suitable for both 2.5D and 3D terrains.

Contents

↑ **Back to summary**

1.1 Introduction

We finally turn to the problem of terrain evolution, focusing on erosion as a key process shaping both aerial and underwater landscapes.

A standard approach for realistic terrain modelling is to first generate a base terrain geometry using noise to define the height on the input domain [MKM89, Ols04, RPP93] or to use the methods proposed in our previous chapters (**??** and **??**), 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 rain, wind, and running water erosion agents) affecting the terrain making it more believable [SS05, SKG*09, GGP*19].

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 [NWD05]:

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 such as a cornice falling down a slope or a grain of sand thrown into the air,

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 [CCB*17, AGP*20, CEG*18, CGG*17, CJP*23].

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 alterations 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 whose greyscale values define terrain elevation. While being the major terrain representation, only a limited number of environments can be modelled. Indeed, natural landscapes are intrinsically 3D (overhangs, cavities or geological structures such as arches or goblins), this is particularly true for underwater environments generation. Alternate representations such as voxel grids, material layers or implicit surfaces can be used. A wide variety of methods have been proposed to simulate natural erosion phenomena on heightmaps as the partial differential equations to model erosion can be discretised 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) [KBKŠ09] 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 generalised 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. 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 representations.

The main contributions presented in this chapter are:

- a generalised 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.

1.2 State of the art

In this section, we first review a subset of the major simulated phenomena used to erode terrains, we then cover different fluid simulation algorithms used in procedural terrain generation. 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.

1.2.1 Erosion processes

Driven by an array of natural forces and processes, erosion varies significantly across environments, from the intense carving of river valleys to the subtle reshaping of slopes in arctic regions. In this section, we present the primary types of erosion (thermal, hydraulic, and wind erosion) alongside other significant processes that contribute to landscape change. Each erosion type not only influences distinct terrain forms but also varies in applicability depending on terrain representation in simulations. Notably, not all erosion types are easily adaptable to all forms of terrain representation due to inherent limitations in data resolution and computational methods.

Gravity-driven erosion

Gravity-driven erosion encompasses processes that involve the downslope movement of soil, rock, or debris due to the force of gravity. These processes, including landslides and talus slope formation, play a crucial role in reshaping landscapes by redistributing materials across slopes, valleys, and cliffs, influencing terrain stability and morphology.

Thermal erosion

Freeze-thaw weathering, also known as frost wedging, frost shattering, or more simply thermal erosion in Computer Graphics, is a process that occurs when water infiltrates cracks and pores within rocks and then freezes (Figure 1.2). As the water freezes, it expands and exerts pressure on the rock, causing it to fracture and break apart over time. This cycle of freezing and thawing is especially prevalent in regions with large temperature variations between day and night or between seasons, such as alpine and polar climates. Over time, freeze-thaw weathering contributes to the breakdown of large rocks into smaller fragments, creating loose rock material that can accumulate and gradually move downslope.

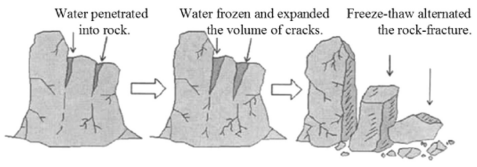

Figure 1.2: *Freeze-thaw process [WXF*17]*

Freeze-thaw weathering plays an important role in shaping landscapes, as it weakens rock faces and cliff edges, contributing to the formation of loose rock debris that eventually becomes part of other erosive processes, such as the development of talus slopes. This process, involving freeze-thaw cycles, can quickly fragment rock surfaces, with cumulative landscape impacts such as the formation of talus slopes observable over decades to centuries.

Figure 1.3: *Talus cones on north shore of Isfjord, Svalbard, Norway - Photo credit: Mark A. Wilson*

Talus slopes

Talus slopes, also known as scree slopes, are accumulations of loose, angular rock debris at the base of cliffs, steep slopes, or mountainous areas. These slopes form as fragments of rock break off due to weathering processes such as freeze-thaw, and gravity pulls them downslope, where they accumulate in a cone-shaped deposit (Figure 1.3). Talus slopes are common in high-altitude or cold regions where physical weathering of rock faces is intense, and they contribute to the visual ruggedness of mountainous landscapes. Formed by the accumulation of rock debris from higher elevations, talus slopes develop gradually as materials accumulate, influencing terrain over centuries.

Landslides

Landslides encompass a range of processes where rock, soil, or debris moves downslope due to gravity. These events vary in scale and speed, ranging from rapid, sudden rockfalls to slow, gradual soil creep (Figure 1.4 illustrate different landslide processes). They can be triggered by factors such as heavy rainfall, seismic activity, or thawing permafrost, which destabilise slopes and initiate movement. Key types of landslides include:

Rockfalls: Sudden detachment of rock from steep faces, often triggered by weathering, freeze-thaw cycles, or seismic activity, leading to rapid downslope movement.

Soil creep: Slow, continuous downslope movement of soil and rock, caused by repeated cycles of expansion and contraction due to changes in moisture and temperature, often imperceptible over short timescales.

Mudflows and debris flows: Rapid flows of water-saturated soil and debris, typically triggered by heavy rainfall or snowmelt, which transport large volumes of material downslope in a short period.

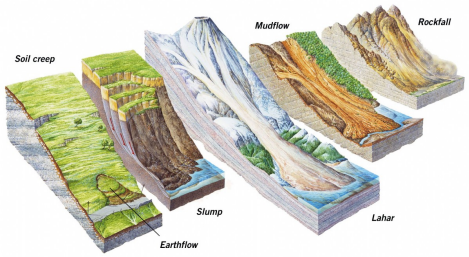

Figure 1.4: *Different landslide processes: soil creep, slump, lahar, mudflow, and rockfall.*

Landslides are a major force in landscape evolution, rapidly reshaping terrain and redistributing materials across slopes and valleys. Triggered by factors such as heavy rain or seismic activity, landslides can reshape landscapes almost instantaneously, though their frequency and impact may vary widely.

Realistic simulation of these effects can be achieved by applying multi-flow computational fluid dynamics on the internal rock fragments or sediments, considering them as fluid particles with an evaporating viscosity [FFRL24, HD01, LD09] or as inelastic frictional spheres [Wal93], but at the cost of very high computation times.

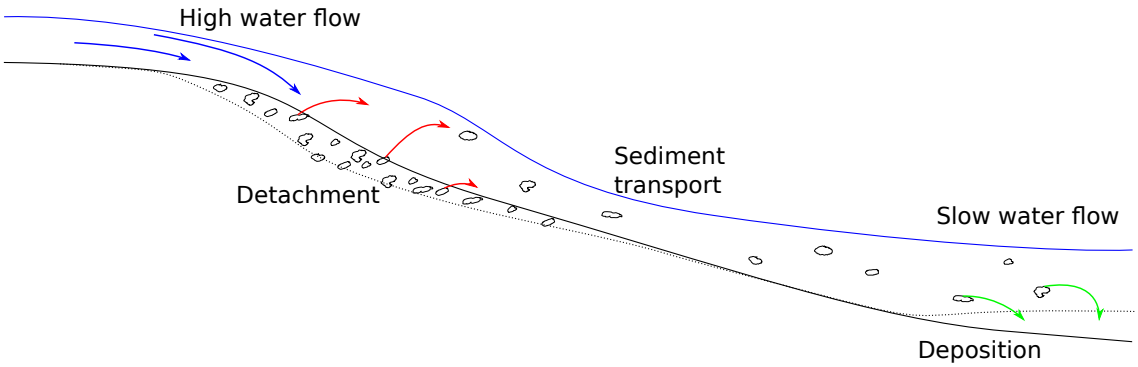

Figure 1.8: *Hydraulic erosion is caused by the friction of water displacing sediments on a slope from intense water flow regions to calmer downstream areas.*

In procedural terrain generation, however, freeze-thaw, talus slopes, and landslides are all similarly considered and generalised as "thermal erosion". The use of "thermal erosion" or "thermal weathering" in procedural generation, introduced by [MKM89], is a misnomer initially used in opposition to "hydraulic erosion". However, the effect of freeze-thaw can be seen as a reduction in the ground resistance to detachment, as the critical shear stress is highly reduced in cold weather regions, while talus slopes and landslides involve finer soil particles or the mixing of liquid fluids, introducing viscosity into the system, resulting in a quite similar output [HĎ11].

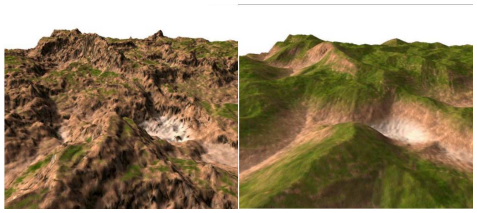

Figure 1.5: *Left: initial fractal terrain; right: thermal erosion simulation from [Ols04].*

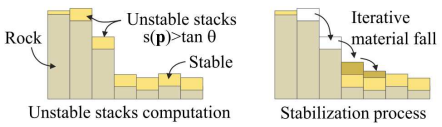

Figure 1.6: *Discrete height field erosion simulation iteratively distribute material from a cell to its neighbours until all slopes satisfy a maximum repose angle θ [GGP*19].*

Thermal erosion can be simulated by redistributing rock fragments or particles to accumulate at the base of cliffs or steep inclines, taking into account only the surface of the terrain. This effect can be achieved by applying gravity-based algorithms that allow loose materials to fall and settle, forming natural slopes of debris at the base of rocky terrain [MKM89, BBFJ10]. Initially proposed for discrete height field terrain representations, the thermal erosion simulation proposed by [MKM89] and improved for GPUs by [JT11] iteratively displaces a small amount of the height at each cell of the terrain and redistributes it to its direct neighbours if a repose angle is not satisfied (see Figure 1.6, used in Figure 1.5).

[BBFJ10] adapts this definition of thermal erosion for density-voxel representations, and [BF01] for layered representations. The importance of keeping track of scree areas allows for more detailed modelling such as the addition of geometry of rock piles illustrated in Figure 1.7, mimicking talus blocks at the bottom of a canyon [PGGM09, PPG*20].

Figure 1.7: *Procedural rock piles from [PGGM09].*

Hydraulic erosion

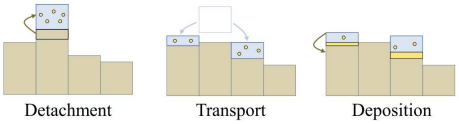

Figure 1.9: *Hydraulic erosion simulations uses water as a medium to move sediments from one position to another.*

Hydraulic erosion is the process by which moving water dislodges and transports soil, sediment, and rock from the Earth's surface (Figure 1.9). Occurring in multiple forms, including river-based, rainfall-induced, and coastal erosion, hydraulic erosion is driven by factors such as water velocity, volume, and surface composition. This process plays a primary role in reshaping landforms, forming valleys, river channels, and coastlines, and significantly contributes to sediment redistribution in terrestrial and coastal environments.

Fluvial erosion

Fluvial erosion is the process by which rivers and streams reshape the landscape by eroding, transporting, and depositing sediment. This phenomenon occurs as the kinetic energy of moving water exerts mechanical forces on the riverbed and banks, dislodging soil, rock, and sediment particles (Figure 1.8). The intensity of fluvial erosion is influenced by factors such as water velocity, discharge (volume of water flowing per unit time), channel slope, and the composition of the riverbed and banks.

In steep, fast-flowing sections of a river, higher water velocities generate turbulent flow, which increases the river's capacity to dislodge and carry large particles. These particles, including gravel and pebbles, collide with the riverbed in a process called abrasion, grinding and wearing down the bedrock over time. Additionally, water exerts direct hydraulic pressure, especially in areas where currents are swift, prying apart rocks and sediment through hydraulic action. This is especially effective in widening channels and undercutting banks.

Fluvial erosion processes contribute to the dynamic reshaping of river channels, forming distinct landforms such as V-shaped valleys, canyons, and river meanders. Over time, rivers naturally balance their erosive energy with sediment transport and deposition, forming floodplains where sediment is deposited during seasonal overflows. In meandering rivers, erosion typically occurs on the outer curves of bends, where flow velocity is highest, while sediment deposition takes place on the inner curves, forming point bars. This continual interaction between erosion and deposition drives the lateral migration of meanders, altering the river's course across the landscape. The river's competence, or its ability to transport particles of a certain size, depends on the flow's velocity and discharge. During periods of high flow, such as after heavy rainfall or snowmelt, rivers gain greater erosive power, enabling them to transport larger particles and increase their erosion rates. River systems reshape landscapes methodically over years to millennia, deeply engraving river paths and altering regional topographies.

Rainfalls

Rainfall-induced hydraulic erosion begins as raindrops strike exposed soil surfaces, causing splash erosion, where particles are dislodged and displaced by the impact of individual raindrops, creating tiny craters [VHLL05, LFW*24]. As rainfall accumulates and flows overland, it transitions into sheet erosion, where a thin layer of water, known as sheet flow, moves across the land surface. This process is often intensified on sloped terrain, where the water gains momentum as it descends, picking up and carrying loose particles downslope. Sheet erosion can remove a uniform layer of soil across a large area, gradually depleting soil fertility and weakening the structure of the soil surface.

Figure 1.10: *Rill and gully erosion along the Chilcotin River [GSP*10]*

On steep or prolonged slopes, sheet flow may concentrate into small channels, initiating rill erosion [Gat00]. Rills are narrow, shallow channels that cut into the soil as water flow converges, carving miniature stream-like paths down the slope visible in Figure 1.10. As rills deepen and widen, they can evolve into larger channels in a process called gully erosion, where channels become deep and wide enough that normal agricultural or natural processes cannot easily repair them. Gullies disrupt the landscape, fragmenting ecosystems, and accelerating the removal of topsoil.

The extent of erosion depends on factors such as rainfall intensity and duration, soil type, vegetation cover, and slope steepness. Sandy soils, for instance, are more prone to erosion due to their low cohesion, whereas clay-rich soils, while more resistant to initial splash erosion, are highly susceptible to rill and gully formation once water begins to concentrate. Splash erosion and sheet erosion can cause significant soil degradation and landscape changes over months to decades.

Hydraulic erosion simulation dominates procedural terrain work because it produces large, coherent landforms and can be simulated efficiently with surface-based models using simple rainfall fields.

In procedural terrain generation, fluvial erosion is focused on the generation of rivers and cascades defined as feature-curves [EPCV15]. First, the path of the river is drawn either by the user [HGA*10] or through simulation [Par23]. Second, the shape of the rivers' beds are modelled [GGG*13]. It is then finally possible to provide a geometric representation of the water surface to take into account flow rate, depth, obstacles, and user-defined details [PDG*19].

The computation of rivers' properties is defined by understanding the flow of water, which is directly related to the amount of rainfall at each point of the terrain [KMN88], then approximated by simplified Shallow Water modelling [MDH07], but is mainly achieved in acceptable computation time by considering all rivers as a drainage network, an oriented graph of rivers as introduced in [RPP93]. Recent works focus on the acceleration of the computation of these graphs and the drainage area associated at each point [CBC*16, SPF*23].

In most works, the amount of water falling at each point of the terrain can be taken into account. However, this strategy is divided among works that consider that rain falls uniformly on the whole terrain or using a random noise function, that rain is more present in regions of high altitude [NWD05], or can use a more accurate weather simulation to define areas more prone to rainfall [PMG*22].

As the motion of water is easier to model on the surface of the terrain, almost all algorithms consider the terrain with a 2.5D representation. On a large scale, this assumption is beneficial but limits their scope to this type of representation, making them unavailable to volumetric models. Particle-based methods are then proposed to overcome this issue for smaller-scale terrains, using 3D Eulerian fluid simulations on voxel terrain representations [BTHB06] or Lagrangian simulations on TIN terrains [KBKŠ09], at the expense of much higher computational time.

Figure 1.11: *Akun Island basalt sea cave - Photo credit: Steve Hillebrand*

Chemical erosion and caves

Chemical erosion, also known as chemical weathering, involves the chemical reactions between water and rock that lead to the dissolution and alteration of minerals within the rock. This process is especially significant in river and coastal environments, where water, often containing dissolved carbon dioxide, interacts with rocks such as limestone and dolostone to form weak carbonic acid. This acid reacts with carbonate minerals, gradually breaking down the rock structure through a process called dissolution. Over time, this breakdown weakens rock formations, making them more susceptible to mechanical erosion by water.

Chemical erosion is particularly influential in the formation of karst landscapes, where the dissolution of carbonate rocks creates unique features such as caves, sinkholes, and underground drainage systems.

As acidic water seeps into fractures within the rock, it slowly enlarges these cracks, leading to the creation of hollow spaces and intricate cave networks. In addition to carbonic acid, other dissolved ions, such as sulphur and organic acids, can also contribute to the chemical weathering of rocks in various environments. The dissolution of soluble rocks such as limestone forms extensive karst landscapes, including networks of underground caves, over thousands to millions of years.

Sea caves can form through the mechanical force of hydraulic action as waves continuously impact the shore. These sea caves develop along coastlines with cliffs, where wave energy focuses on weak points in the rock, such as fractures or softer rock layers. Over time, the pressure from waves and tides pries apart rock fragments, carving out hollow spaces within the cliffside or even a whole arch as presented in Figure 1.11. Formed by the erosive power of waves against rock cliffs containing zones of weakness, development can be observed over hundreds to thousands of years.

In desert environments, caves can also form through aeolian erosion, where wind-driven sand particles abrade rock surfaces. These aeolian caves are typically found in sandstone or other softer rocks, where strong, consistent winds gradually wear away the rock. Unlike chemical or hydraulic caves, aeolian caves are usually shallower and smaller, as the erosive force of wind is less powerful than water or acid-driven processes. However, these caves add unique features to desert landscapes, creating sheltered hollows that sometimes serve as habitats for desert wildlife. Created by wind erosion primarily in softer rock formations, these caves can develop over centuries, depending on the consistency and strength of wind.

In procedural terrain generation, chemical erosion has been rarely studied, as this interaction mostly occurs in shallow to deep waters and requires a volumetric representation, while most terrain generation algorithms are tailored to aerial landscapes and are limited by 2.5D representations. However, as visible in Figure 1.12, using 3D cellular automata can provide an explicit representation of this effect [MKL20, WCMT07] (and can be extended to a simulation of coastal erosion [Haw14]).

Figure 1.12: *[WCMT07] simulates chemical erosion on a regular grid.*

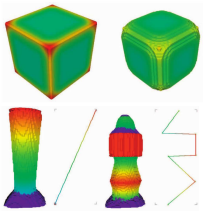

Figure 1.13: *Top: spheroidal weathering proposed in [BBFJ10] on voxel grids iteratively removes most exposed voxels (red). Bottom: modulating exposition with a resistance map (red/green curve) creates specific weathering patterns.*

An approximation of this phenomenon is proposed in [BFO*07, BBFJ10] on voxel grids by considering that the most vulnerable voxels of the terrain are the ones that have the most neighbours with air voxels, in a very similar manner to the computation of voxel-grid ambient occlusion (Figure 1.13). The simulation of sea caves for implicit terrains has been proposed by [PGP*18] by considering a resistance function in the bedrock, usually a function of height, and adding spherical holes in the construction tree at the least resistant areas of the terrain. The inverse percolation method enables the simulation to dig longer cavities, resembling coastal karsts.

Wind erosion

Aeolian erosion, also known as wind erosion, is the process by which wind transports, dislodges, and deposits particles of soil, sand, and rock, particularly in arid and semi-arid regions where vegetation is sparse. This form of erosion is driven by the movement of loose particles across the surface and the abrasive impact of wind-driven sand. Aeolian erosion leads to distinctive landforms that shape desert landscapes, coastal areas, and regions downwind of deserts. Wind erosion typically occurs through three main processes illustrated in Figure 1.14: deflation (removal of fine particles), saltation (bouncing movement of medium-sized particles), and abrasion (wearing down of rock surfaces by

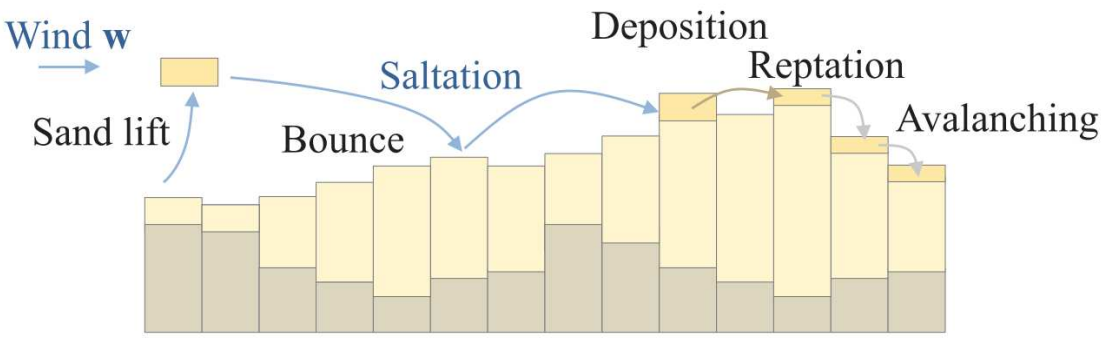

Figure 1.14: *Wind erosion includes the lifting of the sand, the transport through the wind, and its deposition [PPG*19].*

wind-driven particles).

Sand dunes

One such feature is sand dunes, mounds or ridges of sand formed by the accumulation of wind-transported particles (Figure 1.15). As wind moves sand across a surface, obstacles such as rocks or vegetation can slow the particles, causing them to settle and accumulate into dunes. The shape and type of dune depend on wind direction, strength, sand availability, and landscape features [PPG*19]. Common types of dunes include barchan dunes, which are crescent-shaped with tips pointing downwind; transverse dunes, long ridges perpendicular to the wind; and star dunes, which have multiple arms formed by shifting winds. These dunes are dynamic, migrating over time as wind continues to move sand particles, contributing to the constantly evolving desert landscape.

Figure 1.15: *Sand dunes [SM06]*

Figure 1.16: *A yardang near Meadow, Texas - Photo credit: U.S. Departement of Agriculture*

Yardangs

Yardangs are elongated ridges formed where softer material is eroded faster than more resistant rock. The wind-carved ridges align with the prevailing wind direction, creating streamlined shapes with steep sides facing into the wind and gentler slopes on the leeward side as visible in Figure 1.16. Yardangs vary widely in size, from small ridges a few metres long to massive formations stretching for kilometres, as seen in desert regions of Iran and Egypt. They illustrate the power of wind erosion in shaping landscapes over long periods, with their formation largely dependent on rock hardness, wind intensity, and particle size. These streamlined rock formations are carved by persistent wind eroding softer material faster than harder material, typically forming over centuries.

Ventifacts

Ventifacts are rocks that have been shaped and polished by the abrasive action of wind-driven sand. These rocks typically exhibit flat, smooth surfaces that are often oriented in the same direction as the prevailing winds. Their formation occurs predominantly in arid environments where loose sand is available to act as a natural sandblasting tool. This erosive process highlights the power of wind as a geomorphic agent, capable of sculpting rocks into distinctive shapes over time as illustrated in Figure 1.17. Rocks shaped by wind-driven sand, ventifacts can take decades to centuries to form, depending on local wind conditions and rock exposure.

Figure 1.17: *Ventifact at White Desert National Park, Egypt - Photo credit: Christine Schultz*

Wind erosion shifts material through wind force, notably impacting areas with fine surface particles such as deserts. Sand dune generation has been modelled on discrete height fields [RB04] by mimicking sand's wind-driven trajectory and using thermal erosion to correct the slope. These algorithms consider the wind coming from a unique direction, but with a little more complexity in the wind simulation, new dune formations can emerge. [PPG*19], adapted for layer-based representations in which the layers of sand and the layers of bedrock are defined, includes the definition of a wind simulation consisting of warping a uniform flow with the terrain slopes. This allows for the fast generation of large-scale desertic landscapes. More recently, [RDBC24] associated a fine-resolution 3D fluid simulation for wind modelling, enabling the simulation of dune formation at a small scale with high fidelity, albeit at a higher computational cost due to fluid dynamics.

While most aeolian terrain models account for the effect of wind on sand transport, far fewer simulate the feedback of transported material abrading obstacles. Early voxel-based approaches such as [BFO*07] and [BBFJ10] applied analytical heuristics (respectively spheroidal weathering and curvature-driven removal) to progressively reshape volumetric rock, in the spirit of exposure-based methods on voxels and implicit fields [PGP*19]. By contrast, [KHM*20] adopt a physics-inspired formulation: they convert polygonal meshes into a layered-terrain representation, erode its layer stacks when flow stresses exceed material resistance, advect the detached mass as particles within an SPH fluid, and finally redeposit it into the volume. This particle-fluid coupling parallels hydraulic erosion work [KBKŠ09], but targets wind-driven abrasion and deposition. Despite their different strategies (analytic voxel criteria versus particle-fluid transport) all three approaches ultimately rely on volumetric discretizations to modify geometry. Taken together, they outline a design space where voxel analytics offer controllable stylization, while physics-based enables obstacle-sand interactions.

Glacial erosion

Glacial erosion is a dominant geomorphic force in cold regions, characterised by massive ice sheets and glaciers that sculpt the Earth's surface over thousands of years. As glaciers advance and retreat, they transform landscapes through two primary mechanisms: plucking and abrasion.

Plucking occurs when glaciers freeze to the bedrock and, as they move, exert tremendous force that tears away blocks of rock. This process is facilitated by water that infiltrates cracks in the rock, freezes, and expands, helping to dislodge rock pieces as the glacier flows. Abrasion happens concurrently as rocks and debris embedded in the ice act as sandpaper, grinding and smoothing the rock surface beneath the glacier. This abrasion not only polishes the rock but also carves deep grooves and striations (linear marks that align with the direction of ice movement).

These erosive processes give rise to distinctive glacial landforms. U-shaped valleys, unlike the V-shaped valleys formed by river erosion, are wide and deep with a flat bottom, shaped by the broad, sweeping movement of glacier ice. Fjords are deep, narrow inlets of the sea set between high cliffs,

created by the deepening of U-shaped valleys by glaciers that then become submerged as sea levels rise. Cirques are amphitheatre-like hollows situated at the heads of glacial valleys, formed by the erosion caused by the rotational movement of ice within them. Moraines are accumulations of dirt and rocks scraped up and deposited by moving glaciers, typically appearing as ridges along the sides of glaciers or as mounds of debris left behind after a glacier retreats.

The timescale of glacial processes spans over thousands to millions of years, allowing glaciers to leave a lasting impact on the landscape.

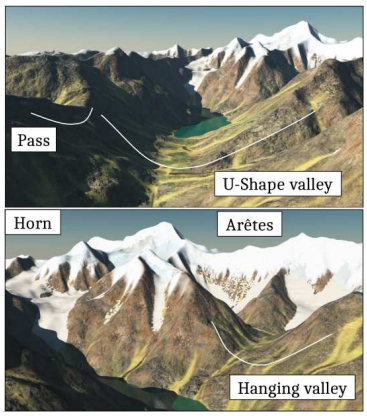

Figure 1.18: *Glacial erosion from [CJP*23] erodes smoothly valleys, shaping the terrain with natural looking valleys and mountain peaks.*

Few works have focused their effort on the understanding and modelling of glacial erosion phenomena. In these conditions, [AGP*20] consider the glacier as a fluid with no inertia, called the Shallow-Ice Approximation, taking only the terrain surface gradient and ice thickness to deduce the velocity of an ice column. The shear stress applied beneath the ice sheet is directly computed by the estimation of the ice column and its velocity. This method results in V-shaped valleys, but when integrating more factors into the simulation, such as debris flow, fluvial erosion and talus slopes, the improved method presented in [CJP*23] generates a large variety of features present in glacial landscapes as presented in Figure 1.18.

Volcanic activity

Volcanic activity is a powerful tectonic process that alters landscapes by creating new landforms through the eruption of magma from the Earth's mantle [RQT*13]. Volcanoes form primarily at tectonic boundary zones, either where plates diverge, allowing magma to rise and fill the gaps, or where one plate subducts beneath another, melting into magma due to high pressure and temperature.

The surface changes caused by volcanic activity are substantial and varied. When a volcano erupts, it deposits layers of lava that solidify into new rock formations, gradually building the volcanic cones and mountainous structures that are characteristic of volcanic islands and mountain ranges [Woo03].

Pyroclastic flows are another aspect of volcanic activity that dramatically changes the terrain. These fast-moving currents of hot gas and volcanic matter can race down the sides of a volcano, destroying everything in their path and laying down thick deposits that can form natural barriers or change drainage patterns by damming rivers and creating lakes almost instantaneously (see Figure 1.19).

Figure 1.19: *Piton de la Fournaise, Réunion Island, in eruption - Photo credit: Richard Roquejoffre*

Volcanic activity also leads to the formation of calderas, large depressions that occur when the summit of a volcano collapses into the emptied magma chamber below after an eruption. Calderas can be several kilometres in diameter and significantly alter the local landscape.

Although lava and pyroclastic flows have been studied in Computer Graphics, resulting in realistic eruption simulations in recent works, these simulations are not used in the case of procedural terrain modelling but focused on rendering and animation [ZKL*17, CBL*09, SAC*99, Las23].

Biological processes

Biological processes play a crucial role in shaping terrestrial and aquatic environments through mechanisms that either promote or inhibit erosion. Known collectively as bioerosion, the activities of various organisms have significant impacts on the stability and durability of habitats.

In terrestrial ecosystems, animals contribute to erosion. Burrowing animals, such as rodents, moles, and earthworms, play a significant role by disturbing soil structures. Their burrowing actions create tunnels and voids in the soil, increasing porosity and altering water infiltration rates, which can accelerate soil erosion under certain conditions. These disturbances also bring subsurface soils to the surface, making them more vulnerable to wind and water erosion.

Grazing animals including deer, cattle, and sheep can have dual effects on the landscape. Indeed, these animals reduce vegetation cover, which increase the soil's exposure to erosive forces such as rain splash and surface runoff. Additionally, their trampling compacts the soil, reducing its porosity and permeability, which increases runoff and potential erosion.

Very few mathods have focused on the effects of desire paths from human trailing and vehicle paths [CEG*18, JKA*19], human footsteps [AAR*24] or animal paths [ECC*21] as the resulting spatial scale may become very small in comparison to other types of erosion previously cited.

In coastal and marine environments, bioerosion is a significant force affecting rocky shores and coral reefs. Organisms such as barnacles, sea urchins, and various molluscs attach to rocks and coral structures, physically breaking down these materials as they feed. Parrotfish, for example, erode coral structures by scraping off coral polyps to ingest the algae living on them. Over years, this contributes to the gradual wearing down of coral reefs, influencing the structural complexity and ecological dynamics of these habitats.

Vegetation profoundly influences erosion control, hydrological cycles, and soil stability across diverse landscapes. The mechanisms through which plants impact erosion are varied and complex, ranging from the stabilisation provided by root systems to the protection offered by canopy cover.

Root systems form the foundation of soil stability. Deep-rooted plants, including many trees and shrubs, secure themselves deep within the soil, anchoring it firmly and preventing landslides and soil slips on slopes. These roots extend into subsoil layers, binding the soil particles together and maintaining the integrity of the slope beneath the surface. In contrast, grasses with fibrous root systems spread a dense network of fine roots through the topsoil, effectively creating a mat that holds the soil in place. This root mat not only prevents topsoil erosion but also enhances soil structure, promoting water infiltration and reducing runoff.

The canopy of vegetation acts as a first line of defence against the erosive force of rain. By intercepting raindrops, tree canopies distribute the water's impact over a wider area and decrease the velocity at which water hits the ground. This dispersion reduces the kinetic energy of raindrops, which lessens their ability to dislodge soil particles. Beneath the main canopy, smaller shrubs and herbaceous plants catch residual droplets, providing a secondary layer of protection that minimises splash erosion and soil displacement.

The presence of vegetative cover significantly modifies surface runoff dynamics. Dense plant life adds physical barriers that increase the land's surface roughness and impede the flow of runoff water. This roughness slows down water movement, allowing more time for the water to seep into the soil rather than washing it away. Additionally, the process of transpiration, where plants absorb water from the ground and release it into the atmosphere, reduces the soil's moisture level, thereby increasing its capacity to absorb further rainfall. This cycle plays a critical role in maintaining soil hydration balance and reducing the volume of runoff during precipitation events.

In conclusion, various erosion processes occur on terrains over very different timescales. However,

Process	Height field	Layered	Voxels	3D implicit	Specific representation
Thermal	[MKM89]	[BF01]	[BBFJ10]		
Landslide	[CEG*18]				[HĎ11]
Hydraulic	[KMN88, GGG*13]	[ŠBBK08]	[BTHB06]		[KBKŠ09]
Chemical			[WCMT07]	[PGP*19]	
Aeolian	[PPG*19]				
Glacial	[CJP*23, AGP*20]				
Volcanic					
Biological	[ECC*21, AAR*24]				

Table 1.1: Mapping of erosion processes with terrain representations. Most works target height fields; volumetric and implicit approaches exist but are rarer and typically more computationally demanding.

each procedural generation algorithm is mimicking a small number of these processes at once. The choice of the terrain representation is a limiting factor in the possibilities offered to the developer. Table 1.1 lists different erosion processes applicable on different terrain representation; we note that height fields have received the most attention but many processes are not proposed for all representations. However most of the erosion processes are very similar and are composed of three main steps, even if their names differ depending on the phenomenon: the detachment of matter, its displacement, and the deposition in smaller sediments. The displacement is guided by the environment in which the erosion occurs. It may be only the force of gravity, the trajectories of glaciers, the water flow, or the wind. Water and wind forces require fluid simulations to simulate accurately.

1.2.2 Fluid simulations

Fluid simulation constitutes an important aspect in Computer Graphics and in terrain generation, providing a physical basis for modelling water behaviour and dynamics for real-time applications and offline rendering [WXL*24]. The mathematical foundations of fluid simulations lie in the Navier-Stokes equations, or in the case of hydrodynamics, the incompressible Navier-Stokes equations. An accurate computation of these dynamics implies an expensive computational cost; thus, various solver variants have been proposed, relying on grids, particles, or hybrid frameworks, each balancing trade-offs between simulation stability, dissipation control, computational scalability, and user control. In this section we propose an overview of different solvers and their characteristics.

Governing equations

The governing equations for fluid simulation are rooted in the conservation laws of mass and momentum for an incompressible Newtonian fluid. In three dimensions, these are expressed by the incompressible Navier-Stokes equations, which consist of the momentum equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{g} \tag{1.1}$$

and the incompressibility constraint:

$$\nabla\cdot\mathbf{u} = 0 \tag{1.2}$$

Here $\mathbf{u}(\mathbf{x}, t)$ denotes the velocity field, $p(\mathbf{x}, t)$ the pressure field, ρ the fluid density, μ the dynamic viscosity, and $\mathbf{g}$ the gravitational acceleration vector. The momentum equation enforces conservation of momentum under advective transport, pressure gradient forces, viscous diffusion, and body forces; the divergence-free condition enforces mass conservation by ensuring volume preservation of fluid elements. In computer graphics contexts, these equations form the basis for physically grounded fluid behaviour, although various approximations or discretisation strategies are typically applied to achieve stability, efficiency, or control in practice.

When altitude variations are negligible compared to horizontal scales, a depth-averaged approximation known as the Shallow Water Equations (SWE) is often employed [Par19]. Under the assumption that the fluid column height $\eta(x, y, t)$ varies slowly in the vertical direction, the (non-conservative) SWE are expressed as

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\mathbf{g} \nabla \eta \tag{1.3}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (\eta \mathbf{u}) = 0 \tag{1.4}$$

where $\mathbf{u}(x, y, t)$ is the horizontal velocity at the surface, and $\mathbf{g}$ denotes gravitational acceleration. The first equation enforces conservation of mass in the depth-integrated sense, and the momentum equation balances advective transport and pressure forces arising from fluid depth. This model offers substantial computational savings and is widely used for real-time or large-domain simulations when three-dimensional effects such as vertical vortices and breaking waves are not critical [Par19].

Modelling assumptions are made explicit in selecting the governing equations. The fluid is typically assumed to be Newtonian and incompressible, with constant density and viscosity. Body forces are often limited to gravity, and surface tension is neglected. The incompressibility assumption simplifies the mathematical treatment and enhances numerical stability; its enforcement commonly involves a pressure projection step that ensures the velocity field remains divergence-free. Viscosity may be incorporated implicitly or explicitly depending on stability requirements, and external forces or boundary conditions are specified to match the intended scenario.

Numerical solvers

Hydrodynamics are continuous but numerical solvers are commonly organised according to their discretisation paradigm, with each category offering different trade-offs in stability, adaptivity, and computational cost. The principal categories comprise grid-based Eulerian methods, particle-based Lagrangian techniques, hybrid schemes that combine grid and particle representations, and reduced models tailored for simplified scenarios or interactive control.

On one hand, grid-based Eulerian methods discretise the fluid domain on a fixed grid and approximate the incompressible Navier-Stokes equations via operator splitting or projection schemes. Historically, the Marker-and-Cell (MAC) approach established the staggered-grid representation [HW65], storing velocities on cell faces and pressure at cell centres to enforce divergence-free constraints accurately in free-surface flows. Subsequent developments in graphics introduced semi-Lagrangian advection with implicit viscosity integration [Sta99], which achieves unconditional stability permitting large time steps at the expense of increased numerical dissipation. Lattice Boltzmann methods (LBM) offer a mesoscopic viewpoint on fluid dynamics, leveraging local collision and streaming operations on a lattice to recover macroscopic behaviour [CD98]; they are notable for parallel efficiency and natural handling of moderate boundary complexity, though three-dimensional or highly irregular domains carry significant computational overhead. Finite-volume or finite-element variants appear in engineering CFD frameworks such as OpenFOAM and can be adapted for graphics applications, but typically require careful optimisation to remain feasible for large-domain or interactive contexts.

On the other hand, particle-based Lagrangian methods represent fluid as discrete particles carrying mass, momentum, and other properties. Smoothed Particle Hydrodynamics (SPH) discretises the

continuum via kernel-weighted interactions among particles [Mon05], naturally handling free surfaces and complex boundaries without explicit interface tracking; however, SPH demands high particle counts for fidelity, and stability challenges (e.g., tensile instability, pressure oscillations) necessitate specialised corrective formulations [Mon05, KBST22]. Pure Particle-In-Cell (PIC) schemes map particle information to a grid for pressure projection but tend to introduce substantial numerical dissipation through frequent interpolation [Har62]. The Fluid-Implicit Particle (FLIP) method mitigates dissipation by updating particle velocities using grid-derived increments rather than full replacements, thereby preserving kinetic energy and small-scale turbulence [BKR88]; FLIP is well suited for detailed free-surface phenomena such as splashes yet requires careful tuning near boundaries to avoid instability. Other particle variants, including Moving Particle Semi-implicit (MPS) methods, extend the Lagrangian paradigm with implicit pressure solves, but share similar demands for neighbour search and stabilisation [KO96].

Hybrid approaches aim to combine the stability and incompressibility enforcement of grid-based solvers with the adaptivity and natural boundary handling of particles. In common hybrid pipelines, particles carry momentum and advect passive quantities, while a background grid enforces the divergence-free condition via projection. Affine Particle-in-Cell (APIC) refines the transfer between particles and grid by representing particle velocity fields affinely, improving momentum conservation and reducing dissipation relative to FLIP [JSS*15]. Such hybrids leverage the strengths of each paradigm but introduce overhead in particle-grid transfers and require careful design to maintain consistency and numerical robustness in dynamic domains.

Reduced fluid models are employed when full three-dimensional simulation is unnecessary or prohibitively expensive. Depth-averaged shallow water equations capture large-scale horizontal flows over terrain under the assumption of negligible vertical variations; their lower-dimensional form yields significant computational savings and is widely used in real-time or large-domain scenarios where vertical vortices or breaking waves are not critical [Vre94, PHTB12]. Potential flow approximations or other simplified models may be invoked for wave-like phenomena where vorticity and viscous effects can be ignored. Procedural or heuristic approximations, such as cellular automata-based flow or ad hoc velocity fields, support highly interactive or stylised effects by sidestepping full PDE solves, at the cost of physical fidelity. These reduced methods serve both as initial terrain-shaping tools and as fallback options for real-time applications where performance constraints dominate.

Solver characteristics

Numerical characteristics of solvers critically influence the realism, stability, and performance of fluid simulation. These include time integration and stability properties, processing of free surfaces and boundary conditions, control of numerical dissipation to preserve detail, and computational efficiency and scalability strategies.

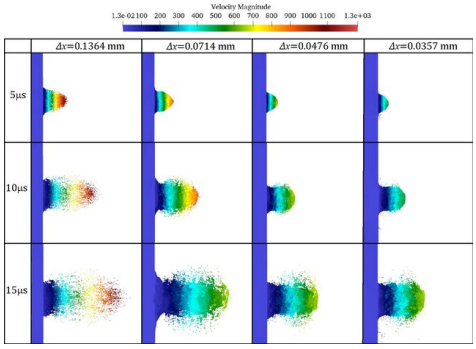

Figure 1.20: *Results of a fluid simulation after 5µs, 10µs and 15µs with different spatial resolutions (from left to right: 0.14mm, 0.07mm, 0.05mm and 0.04mm) shows large discrepancies.*

Time integration schemes must balance stability and accuracy. Explicit methods compute updates directly from known states but impose restrictive time-step limits proportional to grid spacing ($\Delta x, \Delta y, \Delta z$) or particle spacing in relation with the computed velocity, rendering fine resolutions costly. The CFL condition states that the dimensionless Courant number C should not exceed a maximal value C_{max} given depending on the PDE to solve (typically $C_{max} = 1$ for explicit integrations):

$$C = \Delta t \left(\sum_{i=1}^{n} \frac{u_i}{\Delta x_i} \right) \leq C_{max}$$

Thus, when looking at 2D and 3D explicit methods, the main constraint is finding a valid time-step Δt:

$$\Delta t_{2D} \leq \left(\frac{\Delta x}{u_x} + \frac{\Delta y}{u_y} \right) C_{max} \quad \Delta t_{3D} \leq \left(\frac{\Delta x}{u_x} + \frac{\Delta y}{u_y} + \frac{\Delta z}{u_z} \right) C_{max} \tag{1.5}$$

Implicit or semi-implicit treatments of diffusion or pressure terms allow larger time steps by solving linear or nonlinear systems ($C_{max} > 1$), as exemplified by semi-Lagrangian advection with implicit viscosity in [Sta99]. Pressure projection typically entails solving a Poisson equation, for which direct solvers offer accuracy but scale poorly to large grids, while iterative solvers (e.g., conjugate gradient, multigrid) afford scalability at the expense of convergence concerns that must be managed via preconditioning or adaptive tolerance. Time-stepping strategies may incorporate adaptive substepping or projection frequency adjustments to maintain stability without excessive computation.

Accurate handling of free surfaces and boundary conditions is essential for plausible fluid behaviour. Interface-capturing methods in Eulerian solvers, such as level-set or volume-of-fluid, track the free surface implicitly but can suffer volume loss or smearing; corrective reinitialisation or particle-based markers near the interface often mitigate these deficits. Particle-based schemes represent the free surface naturally through particle distribution but require surface reconstruction for rendering and may exhibit clumping or void regions without density control.

Solid boundary conditions in both paradigms demand robust collision and pressure treatment: Eulerian grids enforce no-slip or free-slip via ghost cells or immersed boundary techniques, while particles interact with geometry via repulsion forces or dynamic boundary particles. Hybrid methods must synchronise interface representation between particles and grid, ensuring that evolving boundaries remain consistent with divergence-free constraints. Dynamic domains, such as moving obstacles or adaptive meshes, necessitate regridding or particle reseeding procedures that preserve mass and momentum.

Numerical dissipation and detail preservation influence the visual richness of simulated flows. Semi-Lagrangian advection and grid-particle interpolation in PIC introduce artificial smoothing that dampens small-scale vortices and surface detail. Techniques to mitigate dissipation include the FLIP update, which applies only the change in grid velocity to particles, and higher-order advection schemes that reduce numerical diffusion. Vorticity confinement or turbulence-enhancement terms may reintroduce fine-scale structures lost to dissipation. In Eulerian solvers, divergence-free interpolation schemes and improved projection methods help maintain kinetic energy. SPH and other particle methods can preserve detail inherently but may suffer from noise or instability if neighbour sampling is irregular; stabilisation strategies, such as density reinitialisation, kernel correction, or pressure regularisation, seek to retain fidelity without sacrificing stability. Machine learning-based super-resolution methods have recently emerged to reconstruct fine details atop coarse simulation outputs, however, care must be taken when applying them to simulations that differ significantly from the data they were trained on, as they may not produce reliable results.

Non-physical fluid simulations

Recent advances in fluid simulation increasingly integrate machine learning techniques to accelerate or augment traditional numerical methods. Surveys of machine learning applications in computational fluid dynamics identify roles for data-driven methods, physics-informed models, and ML-assisted numerical solvers that improve convergence or predict intermediate quantities such as pressure fields and subgrid turbulence closures [HFTL22].

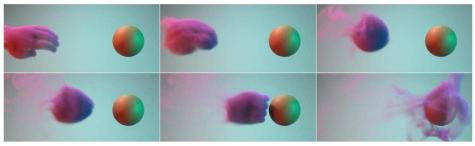

Figure 1.21: *An incompressible force field is computed to guide the smoke simulation from a user-defined shape to another [TACS21].*

Neural surrogates have been proposed to approximate costly components of the solver, yielding substantial speed-ups while maintaining acceptable fidelity in graphics contexts; such approaches

often leverage convolutional networks for pressure projection or learn residual corrections to coarse simulations [TSSP17, SRA24]. Physics-informed neural networks and related frameworks enable embedding governing equations into the learning process, facilitating generalisation and adherence to conservation laws, though their applicability to large-scale, real-time graphics remains an active research area [BNK25].

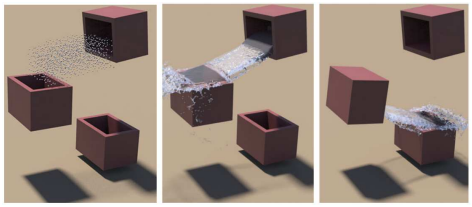

Figure 1.22: *[LCY*19] uses fluid rigging-skinning on water particle to guide the animation of particles.*

Procedural or artistic fluid control has been an emerging topic for the last few decades, striking a balance between user interaction and plausibility of the fluid behaviour without full PDE solving. **??** used the Kelvinlet paradigm [GJ17], an analytic formulation for material deformation derived from Kelvin's state using Green's function of linear elasticity of material, providing more control types than the fluid-specific Stokelets [CY76]. [WH91] shows that these analytic fluid primitives are an efficient and lightweight approximation of highly controlled fluid simulations. Stroke-based fluid models are closer to the storyboard design of animation artists, making them intuitive to use, but are usually for subsets of frames of an animation, which become time-consuming for the price of high-level control [XKG*16, PT05, YCYW20, PHT*13].

Procedural fluids are usually defined by noise functions to artificially introduce motion [BHN07], but may also be added after computing an accurate fluid simulation to introduce more vortices [WZF*25]. Vortices are usually too small for Eulerian simulations as the grid's cell width used may be larger than a vortex, and the dissipation removes their presence. Editing velocity fields procedurally may also come through the use of frequency-domain editing of forces as used in Figure 1.21 [FN20, TACS21], blending and interpolation techniques to merge a predefined fluid with another [RWTT14], or even morphing techniques to deform a velocity or force field as shown in Figure 1.22 [LCY*19, RTW*12, FEHM19].

Non-physical fluid simulations include procedural and artistic fluids during or after a more computationally expensive simulation, as they introduce variation on many levels: force field and velocity field of Eulerian simulation methods, or the force, velocity, and even position of simulation particles of Lagrangian methods [Sim90].

Fluid simulations come in a large variety of forms, each with advantages and inconveniences: accuracy, energy preservation, user control, and boundary representations. Erosion on terrains, and more importantly submerged terrains, has to compute the motion of fluid in the environment, whether it represents air for wind-driven phenomena or water for hydraulic erosions. Erosion simulation algorithms proposed in literature do not offer alternatives to the method-specific fluid simulation, constraining the user to only a subset of possibilities for a given terrain representation.

1.3 Particle erosion

Erosion occurs in three stages: material detachment, transport and deposition (respectively in red, black and green in Figure 1.23). 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, separating the transport process from the detachment and deposition processes (Figure 1.24). The overall algorithm presented in Algorithm 1 illustrate the separation of these processes.

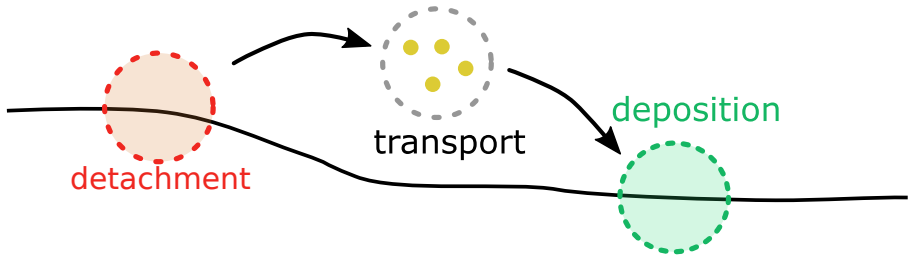

Figure 1.23: *Three steps of the erosion process from the sediment point of view: detachment from its original location (dotted red circle), transport in a fluid (dotted black circle), deposition at a new location (dotted green circle).*

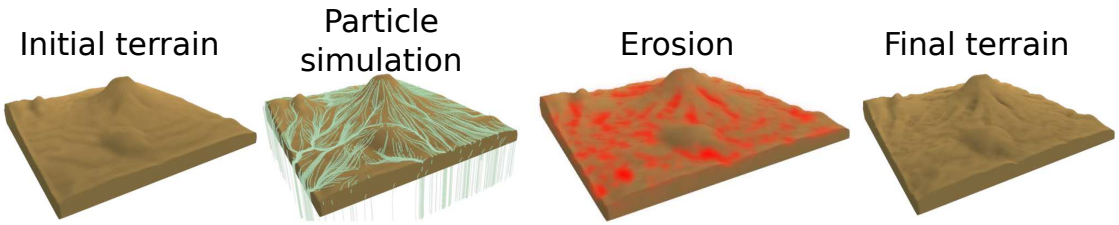

Figure 1.24: *An iteration of the erosion pipeline consist of an initial terrain onto which we simulate a particle system. The collisions are tracked and we model the detachment and deposition at each point, resulting in a final eroded terrain.*

1.3.1 Overview

Input: Terrain $\mathcal{T}$, particle parameters $(R, \rho_{\text{particle}}, COR, C_{\max}, K_\varepsilon, \omega)$, medium params $(\rho_{\text{fluid}}, \mu)$, velocity field v_{fluid}
Output: Updated terrain $\widetilde{\mathcal{T}}$

```
for iter ← 1 to N_iters do
    C ← ∅                                                  // Collisions list
    // (1) Emit particles in domain
    P ← SampleParticles(T, params)
    // (2) Particle transport
    foreach p ∈ P do
        while p in terrain do
            integrate motion                  // Equations (1.18) and (1.20)
            if p intersects T then
                add collision data to C       // particle + terrain properties
                apply collision response                          // Algorithm 2
    // (3) Terrain updates
    foreach collision in C do
        accumulate terrain change Δh, Δf, orΔμ                    // Section 1.4
    commit to T and recompute normals
return T̃
```

Algorithm 1: Particle-based erosion iteration

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 significantly reduces the overall execution time but also offers users high flexibility over the quality of the simulation and simplifies the implementation. Moreover, we consider that particles in a fluid do not influence the flow as it would become "overkill" for such lightweight objects [WZF*03].

1.3.2 Erosion process

Every time the particle hits the ground, a given amount $q_{\text{detachment}}$ of sediment is detached from the ground while another amount q_{deposit} of sediment is deposited at this location (respectively red and green arrows in Figure 1.25). Our erosion model is based on the work of [WCMT07] where regular 3D grids are used to estimate the fluid velocity and sediment transport. In the spirit of [KBKŠ09], 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.

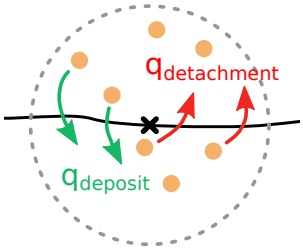

Figure 1.25: *When a particle is in contact with the terrain surface, an amount $q_{deposit}$ of sediments contained in the particle is released in the ground, and $q_{detachment}$ is absorbed.*

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, which we call abrasion. We use pseudoplastic models to approximate the amount of matter removed due to the shear forces while considering the physical properties of the fluid and the ground [WCMT07].

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

$$\tau = K\theta^n \tag{1.6}$$

where $\theta = \vec{v}_{rel}/l$, 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 Equation (1.6) by

$$\varepsilon = K_\varepsilon(\tau - \tau_{\text{critical}})^a \tag{1.7}$$

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 in similar manner as 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 hemisphere of radius R, in the normal opposite direction at the particle impact point (Figure 1.29). We then use Equation (1.7):

$$q_{\text{detachment}} = \varepsilon \frac{2\pi R^3}{3} \tag{1.8}$$

to get the final eroded amount $q_{\text{detachment}}$. The particle is also defined by a maximal amount of sediment that can be contained in its volume before being saturated, noted $C_{\max}$.

Deposition

The eroded sediment is considered to be in suspension in a fluid and is affected by its velocity. A fluid particle then transports the sediment in its flow until gravity settles it onto the ground again. The effect of gravity is modelled by a settling velocity w_s:

$$w_s = \frac{2}{9}\vec{g}R^2 \frac{(\rho_{\text{particle}} - \rho_{\text{fluid}})}{\mu} f(C) \tag{1.9}$$

We consider that the amount of sediment settled is proportional to the norm of the settling velocity as proposed in [WCMT07], with $\omega \in [0,1]$:

$$q_{\text{deposit}} = \omega \|w_s\| \tag{1.10}$$

1.3.3 Transport

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

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

Particle physics

Due to their independence from other particles, we consider each particle to follow Newton's laws of motion. Each particle has a volume V, an effective density ρ_{particle} (which may depend on the amount of carried sediment), and thus a mass $m_{\text{particle}} = \rho_{\text{particle}} V$. The surrounding medium has density ρ_{fluid}, viscosity μ, and velocity field $v_{\text{fluid}}(\mathbf{p}, t)$.

The exerced forces applied to a particle consist of external forces $\vec{F}_{\text{ext}}$ (gravity and buoyancy), and drag force $\vec{F}_{\text{drag}}$ (Figure 1.26). Gravity acts as

$$\vec{F}_{\text{gravity}} = m_{\text{particle}}\vec{g} \tag{1.11}$$

with $\vec{g}$ the gravitational acceleration vector. Buoyancy opposes gravity as

$$\vec{F}_{\text{buoyancy}} = -\rho_{\text{fluid}} V \vec{g} \tag{1.12}$$

The net body external force is then

$$\vec{F}_{\text{ext}} = V\left(\rho_{\text{particle}} - \rho_{\text{fluid}}\right)\vec{g} \tag{1.13}$$

In addition, drag relaxes the particle velocity $\vec{v}$ toward the local fluid velocity v_{fluid}. For small Reynolds numbers we use Stokes' drag formula for spheres in a fluid:

$$\vec{F}_{\text{drag}} = 6\pi\mu R\,(v_{\text{fluid}} - \vec{v}) \tag{1.14}$$

where R denotes the particle radius.

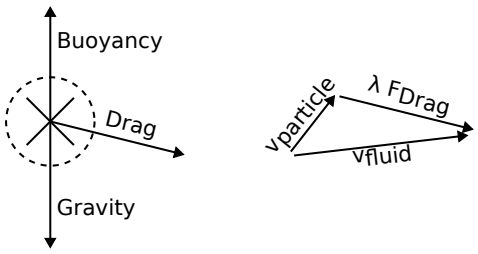

Figure 1.26: *Particle's motion in a fluid consist of three forces: gravity, buoyancy, and drag. The later is acting opposite direction to the relative particle velocity in the fluid.*

The full equation of motion is therefore

$$\dot{\vec{v}} = \frac{f(C)\vec{F}_{\text{ext}} + \vec{F}_{\text{drag}}}{m_{\text{particle}}} \tag{1.18}$$

where $f(C)$ is a hindered-settling factor that reduces the effective body force as the particle's carried sediment approaches its maximum capacity. Following Richardson-Zaki [RZ54], we take

$$f(C) = \left(1 - \frac{C}{C_{\max}}\right)^n, \tag{1.19}$$

with exponent n typically in the range 4-5.5, and we set $n = 5$, similarly as [WCMT07].

The particle position $\mathbf{p}$ evolves as

$$\dot{\mathbf{p}} = \vec{v} \tag{1.20}$$

When a particle collides with the ground, its post-impact velocity is modified by a coefficient of restitution COR. This value depends on the ground material as it is influenced mainly by the material's particle shape, coefficient of friction and density [YZKF20]. Less bouncy particles lose speed quickly and settle down sooner, forming a steeper pile (Figure 1.27 blue), or a higher talus angle such as chalk. On the other hand, more bouncy particles disperse more widely upon hitting a surface, resulting in a gentler accumulation such as clay (Figure 1.27 red). We propose this simple collision response in Algorithm 2.

Input: Particle $(\mathbf{p}, \vec{v}, C)$, terrain $\mathcal{T}$, restitution COR
Output: Updated particle state $(\vec{v}, C)$

$\vec{n} \leftarrow \mathcal{T}.\vec{N}(\mathbf{p})$
$\vec{v} \leftarrow COR\,(\vec{v} - 2\langle\vec{v}, \vec{n}\rangle\vec{n})$
`// Update sediment capacity at this point`
compute $q_{\text{detachment}}, q_{\text{deposit}}$ `// Equations (1.8) and (1.10)`
$C \leftarrow C + q_{\text{detachment}} - q_{\text{deposit}}$
return $(\vec{v}, C)$

Algorithm 2: Particle-terrain collision response

Velocity field

With our particle system, we propose to model intricate scenarios, such as 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 [TJ10]. We note several velocity improvement methods:

Fluid simulation refinement: Many erosion systems incorporate fluid simulation, requiring regular updates for erosion and velocity [KBKŠ09, WCMT07]. Our method can use fluid simulations with

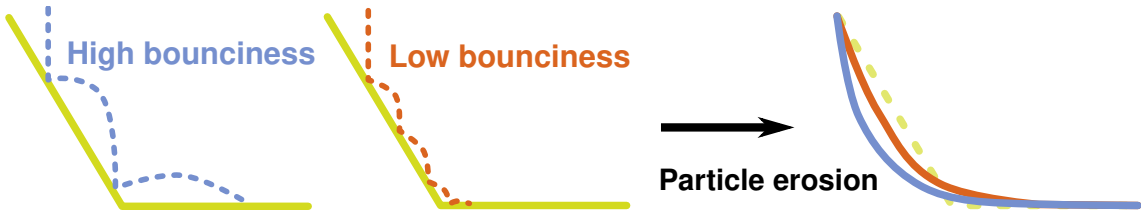

Figure 1.27: *The coefficient of restitution affects the amount of energy absorbed from the particle when hitting the ground. Here, rain is applied on an initial slope (yellow). Only two particles are displayed, with a high (blue) and low (red) coefficient of restitution. The resulting slope after erosion is displayed in blue and red (right).*

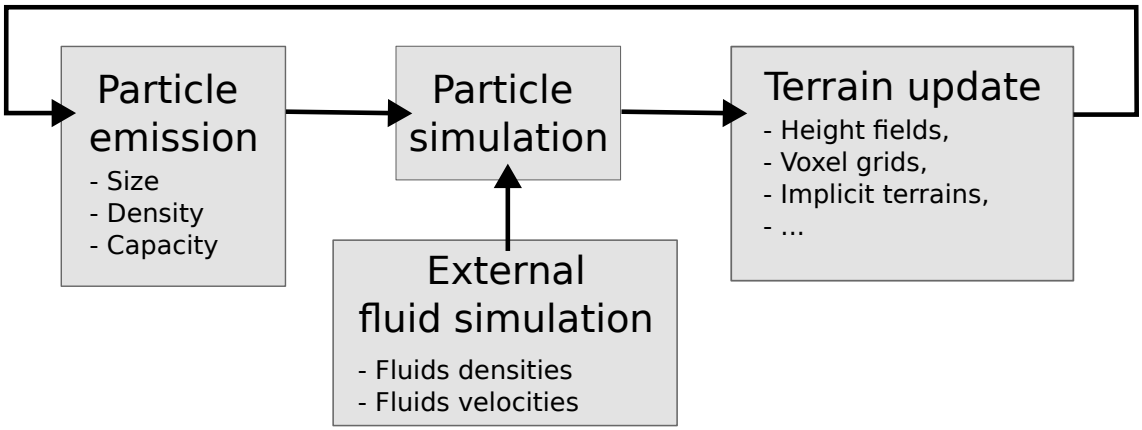

Figure 1.28: *Our method requires a base geometry, a small number of parameters for the particles and the medium used for the erosion simulation. It can be easily adapted to be compatible with different mediums and terrain representations.*

multi-resolution refinement, with the possibility to focus the velocity field adjustments near the updated boundaries of the surface [RLL11].

Particle velocities in fluid simulation: With a Lagrangian fluid simulation relying on particle systems [KBST22], our particle velocities can be incorporated into its computation. This approach is only a provisional solution due to potential parameter mismatches with the 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 1.46, allows us to find a balance between achieving realistic flow simulations and maintaining computational efficiency.

1.4 Our erosion method

In this section, we describe how to apply detachment and deposition to different terrain representations with our method (Figure 1.28). We cover the most commonly used representations, namely height fields, layered terrains, voxel grids 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: the ability to evaluate the intersection of a particle with the ground and to compute the normal of the terrain at this point. To the best of our knowledge, all representations do.

We use Verlet integration for the particle physics [Ver67], with low error rate and stability even for high dt, reducing computation time for negligible imprecision [BW98, SABW82].

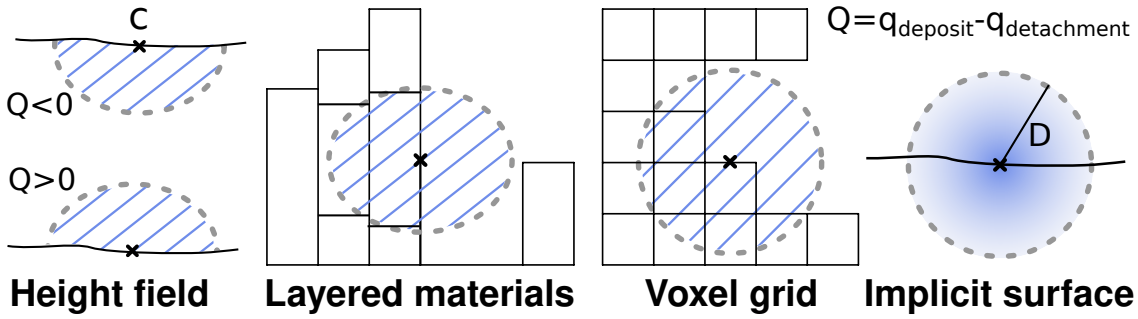

Figure 1.29: *Illustration of the material detachment in the (half-)sphere at contact point C (cross) on different representations. (height field) When $Q < 0$, material detachment happens in the bottom scaled half-sphere of the particle's contact with the ground, while the deposition is applied on the upper half-sphere of volume when $Q > 0$. Unlike the height field, for 3D terrains detachment and deposition are applied in the full sphere around the contact point.*

For all the representations, the amount of material absorbed by the particle, i.e. the erosion value $q_{\text{detachment}}$ from Equation (1.8), is taken around the particle at a radius R, meaning that the modification of the terrain by a particle at position $\mathbf{c}$ will only occur for the positions $\mathbf{p}$ satisfying $||\mathbf{p}-\mathbf{c}|| < R$. At the same time, the amount q_{deposit} from Equation (1.10) 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 conceptualised 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 in Figure 1.29). In the following sections, we will describe the strategies used to modify the amount of matter for different representations.

1.4.1 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 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 centre $\mathbf{c}$ and the distance of the particle to the centre $d = ||\mathbf{p}-\mathbf{c}||$ 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

$$\Delta h(\mathbf{p}) = \frac{\sqrt{R^2-d^2}}{\alpha} = \frac{Q}{\frac{2}{3}\pi R^3}\sqrt{R^2-d^2} \tag{1.21}$$

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

1.4.2 Application on layered terrains

Layered terrains are defined as $\mu : \mathbb{R}^3 \mapsto \mathbb{N}$, assigning a discrete material index μ for any point in space [BF01, PGGM09]. 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 [JW17], we approximate the volume of the stack we need to alter using the previously defined height field equation Equation (1.21). At a distance d from the particle, the height is defined as

$$H(d) = \frac{|Q|}{\frac{2}{3}\pi R^3}\sqrt{R^2 - d^2} \tag{1.22}$$

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

1.4.3 Application on implicit terrains

Implicit terrains 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

$$\Delta f(\mathbf{p}) = \frac{3Q}{\pi R^3}\frac{(1-d)}{R} \tag{1.23}$$

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

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

Signed Distance Functions

Considering SDFs, the terrain is defined as the 0-set of the signed distance function $f : \mathbb{R}^3 \mapsto \mathbb{R}$, hence, for $f(\mathbf{p}) = 0$, the inside is $f(\mathbf{p}) < 0$ and the outer part (i.e. air or water) is $f(\mathbf{p}) > 0$.

The particle erosion applies at impact points at discrete positions, so we propose to add or subtract metaballs defined using equation Equation (1.23) 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 optimisation algorithms such as k-d trees, BSP trees or BVH can easily be used to improve performance.

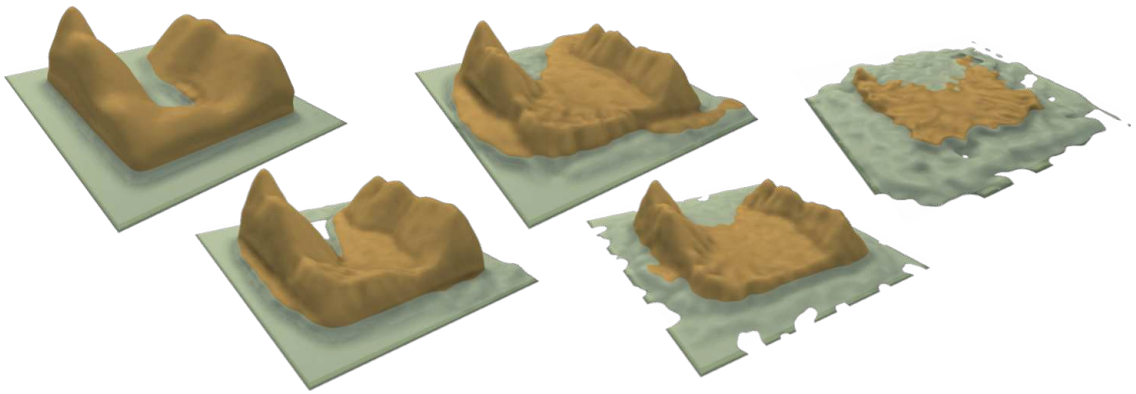

Figure 1.30: *Our erosion method is applied iteratively on a completely synthetic island; the terrain is altered to obtain a plausible shape by forming rills. The use of particles with hydraulic densities dropped from the sky results in strong erosion on the sides of the mountains, and the particles that slide to the sea mainly drift offshore, resulting in the formation of small beaches and 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.*

Other implicit terrains

Other implicit terrain models are present in the literature, notably a 2.5D representation based on the surface gradient [GPM*22] and a 3D representation based on curves [BKRE17], for which the trajectory of each particle projected to the closest surface 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 Equation (1.21), which gives, for $d \leq R$,

$$\nabla h' = -\frac{Q}{\frac{2}{3}\pi R^3} \frac{1}{\sqrt{R^2 - d^2}} \vec{CP} \tag{1.24}$$

with $\vec{CP}$ the vector from the centre of the erosion point $\mathbf{c}$ to evaluate to the position $\mathbf{p}$.

Avoiding the blow up for $R = d$ can be done using a regularisation by introducing $\varepsilon \ll R$ in Equation (1.24):

$$\nabla h' = -\frac{Q}{\frac{2}{3}\pi R^3} \frac{1}{\sqrt{R^2 - d^2 + \varepsilon^2}} \vec{CP} \tag{1.25}$$

Now the new gradient field can be computed as:

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

1.4.4 Application on voxel grids

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

Density voxels

We consider "density-voxel" grids defined on $f : \mathbb{Z}^3 \mapsto [-1,1]$, for which a voxel is 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 discretisation of a scalar function, we could directly use Equation (1.23), 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 $\mathbf{c}$ on all points $\mathbf{p}$ in the volume, proportionally to the distance from the centre of the sphere $d = ||\mathbf{p} - \mathbf{c}||$, 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 to -1. We then have for all surrounding voxels

$$\Delta f(\mathbf{p}) = Q\frac{(1-\frac{d}{R})}{\sum(1-\frac{d}{R})} \tag{1.26}$$

The resulting voxel value is computed as $\tilde{f}(\mathbf{p}) = f(\mathbf{p}) + \Delta f(\mathbf{p})$. In our implementation, presented in Algorithm 3, when $f(\mathbf{p}) > 1$ we simply transport the density excess to the voxel above, giving it a very close analogy to height fields as long as $|\Delta f| < 1$.

Input: Voxel grid $f : \mathbb{Z}^3 \mapsto [-1,1]$, contact point $\mathbf{c}$, erosion radius R, volume change Q
Output: Updated grid $\tilde{f}$

total $\leftarrow \sum_{\mathbf{q} \in \|\mathbf{q}-\mathbf{c}\| < R}(1 - \frac{\|\mathbf{q}-\mathbf{c}\|}{R})$
foreach *voxel* $\mathbf{p}$ *with* $\|\mathbf{p} - \mathbf{c}\| < R$ **do**
  $d \leftarrow \|\mathbf{p} - \mathbf{c}\|$
  $\Delta f(\mathbf{p}) \leftarrow Q \cdot \frac{(1-\frac{d}{R})}{\text{total}}$ `// Equation (1.26)`
  $f(\mathbf{p}) \leftarrow f(\mathbf{p}) + \Delta f(\mathbf{p})$
  if $f(\mathbf{p}) > 1$ **then**
    spill excess upward voxel
return $\tilde{f}$

Algorithm 3: Density-voxel terrain update

Binary voxels

The terrain can be represented using an occupancy function as $f : \mathbb{Z}^3 \mapsto \{0,1\}$, 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 transforming them into air or ground when this number reaches a critical value C that is proportional to the particle's strength parameter K_ε [BBFJ10].

On a hit, all voxels in a radius R receive a hit number

$$\Delta hits = \lfloor \alpha \Delta f \rfloor \tag{1.27}$$

with Δf the erosion per voxel computed using Equation (1.26) and α a coefficient high enough to obtain values above 1.

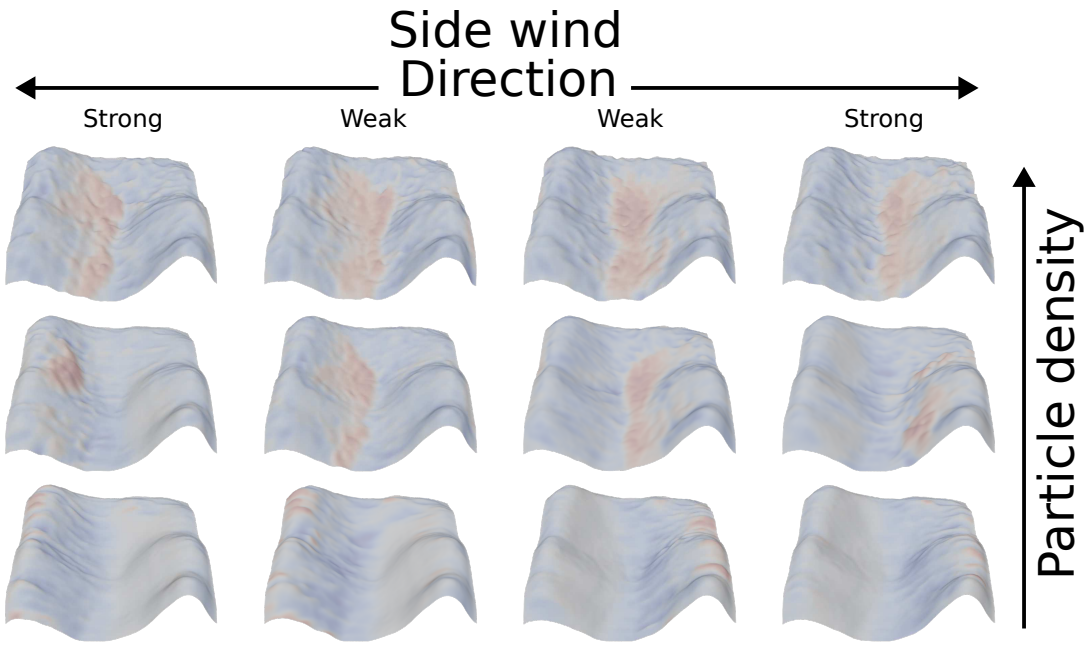

Figure 1.31: *Uniform wind field applied laterally on a terrain with different particle densities. Deposition and erosion is visible in red and blue respectively.*

All voxels with $\#hits > C$ are transformed into background, and voxels with $\#hits < -C$ are transformed into 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 Equation (1.21) can be used to erode 2D scalar field-based representations. Similarly, our proposition for SDFs Equation (1.23) enables erosion for continuous 3D scalar fields, and voxels Equation (1.26) for discrete 3D scalar fields, respectively.

1.5 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 effect of each parameter individually are displayed in Figures 1.31 to 1.34. Parameters used for each result are available in Table 1.2. 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 structures, 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 refinement is done using the presented diffusion strategy.

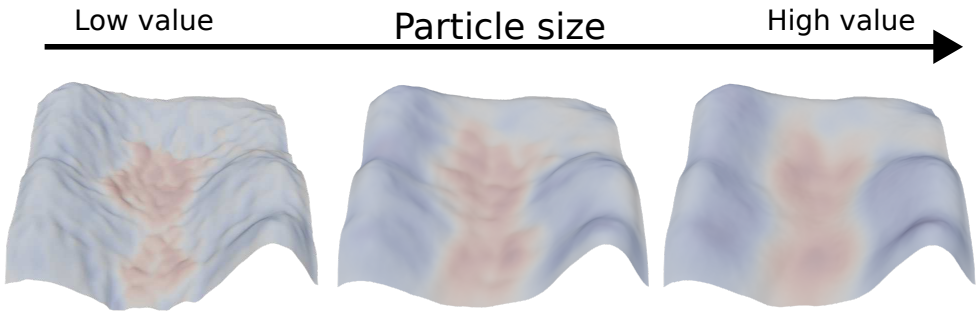

Figure 1.32: *Particle radius influence the smoothness over the final result. Deposition and erosion is visible in red and blue respectively.*

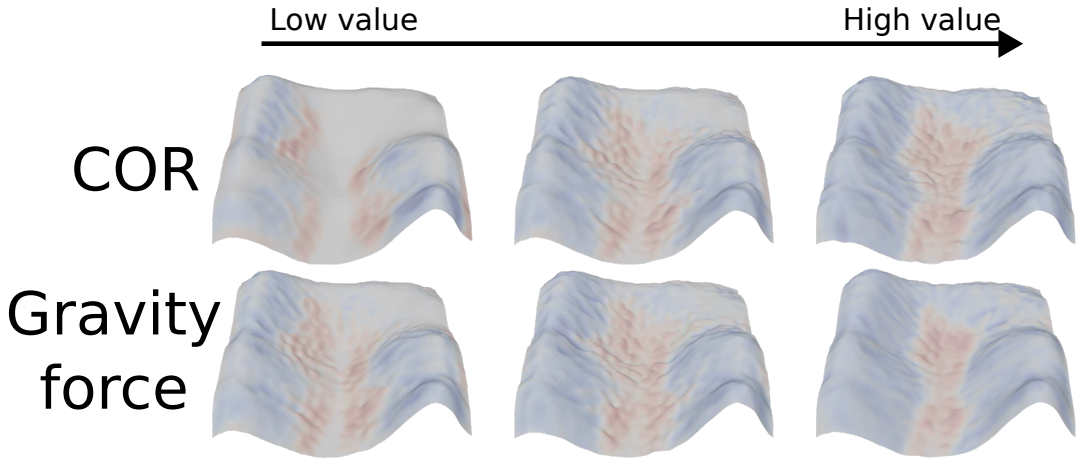

Figure 1.33: *The coefficient of restitution (COR) and the force of gravity influence similarly the localisation of deposits. A low COR emulate debrit scree at the bottom of slopes; high COR values approximate viscous transports such as hydraulic processes. Deposition and erosion is visible in red and blue respectively.*

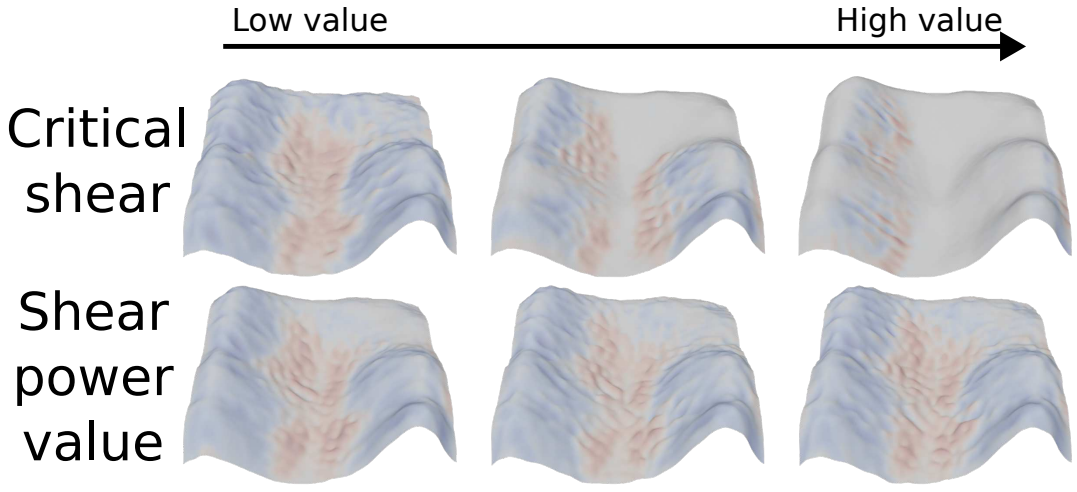

Figure 1.34: *Large critical shear values restric erosion to steep slopes, while large shear power values increase the effect of erosion on the whole terrain. Deposition and erosion is visible in red and blue respectively.*

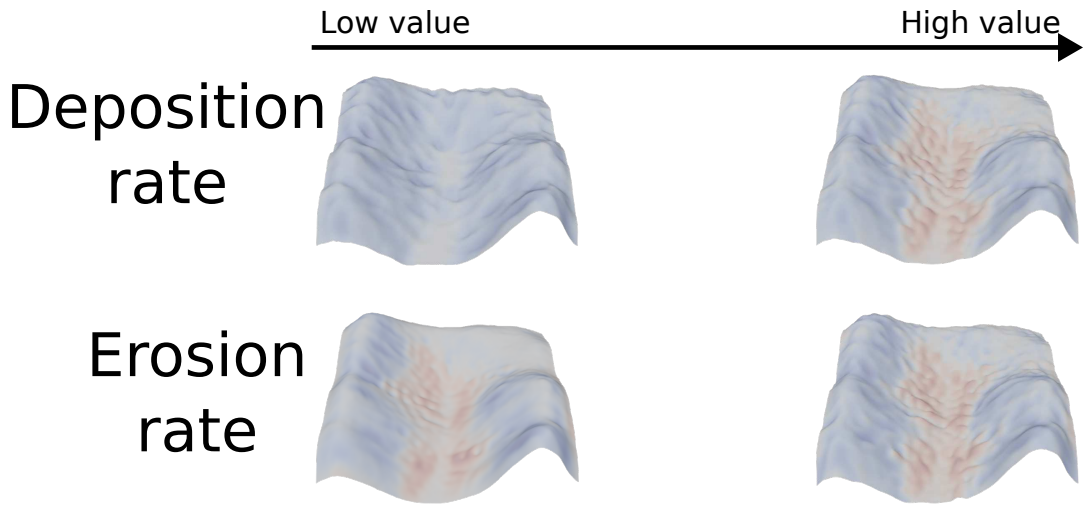

Figure 1.35: *Deposition and deposition rates directly influence the displacement of matter from the particles on the terrain. Deposition and erosion is visible in red and blue respectively.*

1.5.1 Rain

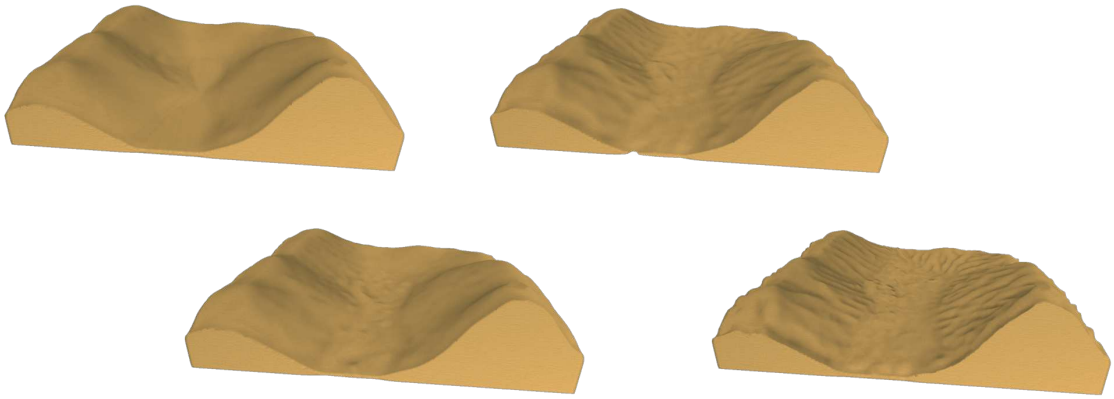

Figure 1.36: *Rain erosion on a synthetic heightmap generated by Perlin noise using small water particles.*

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 Earth's gravitational force. No velocity field is required from fluid simulation. These parameters result in detailed geometry of the rills on the sides of mountains that quickly emerge and deposit many sediments in the valley. We demonstrate the result of rain erosion in Figure 1.36 with a computation time of 4 seconds.

Using these erosion parameters in combination with water bodies results in different outcomes (Figure 1.30). 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 sediment, leading mainly to deposition. The result is typical hydraulic erosion on mountains and the formation of slopes and beaches near the water level.

1.5.2 Coastal erosion

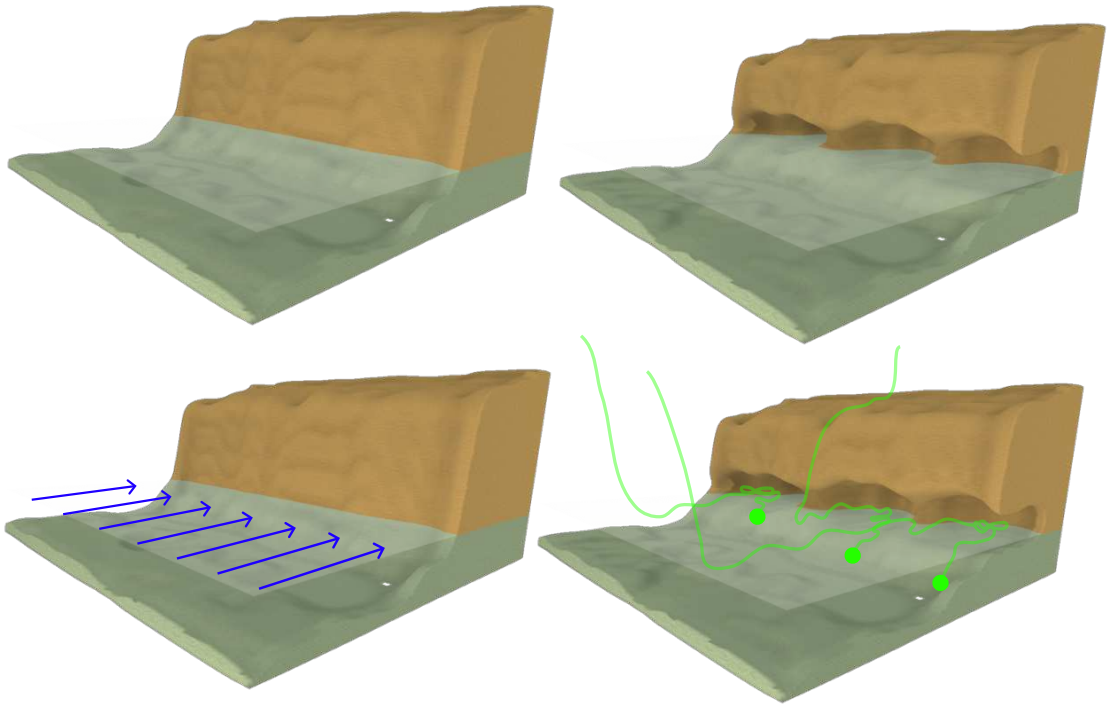

Figure 1.37: *Coastal erosion on cliff from density-voxel representation using large particles with density slightly lower than water to target target collision on the air-water interface. A uniform flow field (blue arrows on bottom left) force the particles towards the cliff (green paths of three particles visible in bottom right).*

The repeated motion of waves creates coastal erosion, which 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, applying coarse detachment while depositing a very small quantity of sediment, 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 1.37 presents the result of coastal erosion on a density-voxel grid that creates overhangs around sea level using a small number of particles. Note that the same effect using an alternate implicit representation based on SDF is displayed in Figure 1.48. A shaded version of this effect is presented in Figure 1.1.

1.5.3 Rivers

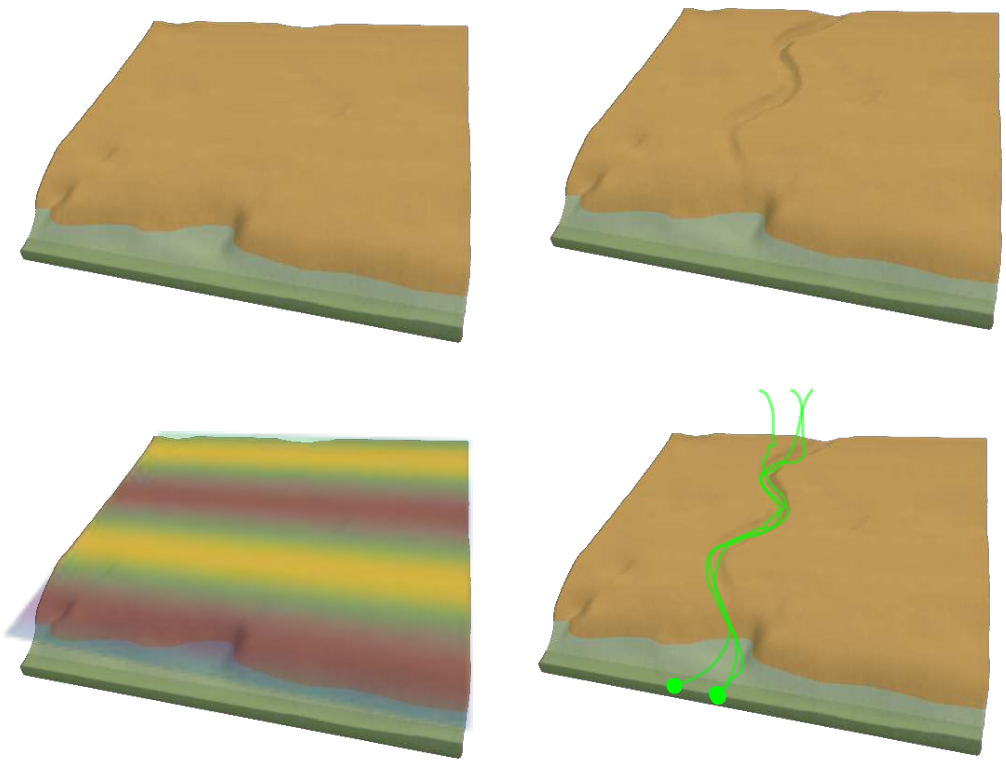

Figure 1.38: *Meandering river simulation on an implicit terrain representation using a user-defined sinusoidal flow field. Bottom left illustrate the alternating variation on the velocity field, guiding the particles as illustrated with green paths in bottom right.*

Given a source point, we generate particles that run downhill, simulating the formation of a river. More complex erosion simulations using fluid simulations such as SPH [KBKŠ09] 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, user-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 1.38).

Figure 1.39 presents a river that has been modelled by emitting water particles with different sizes ranging 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 fluctuated over formation time, while the low capacity ensures that the banks of the river stay smooth. A high coefficient of restitution is a strategy that lets the particles flow with low friction, approaching water-like behaviour. Our particles are affected only by gravity, without fluid simulation.

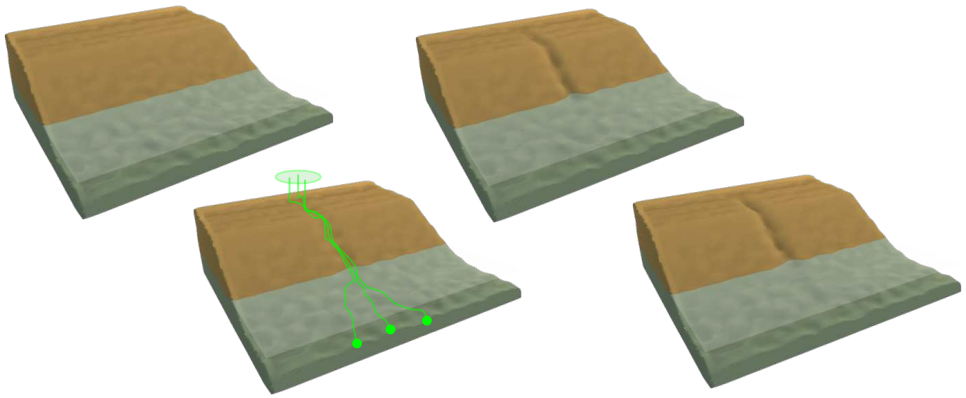

Figure 1.39: *A single river simulation on a layered representation with water particles emitted from a small disk (green).*

1.5.4 Landslide

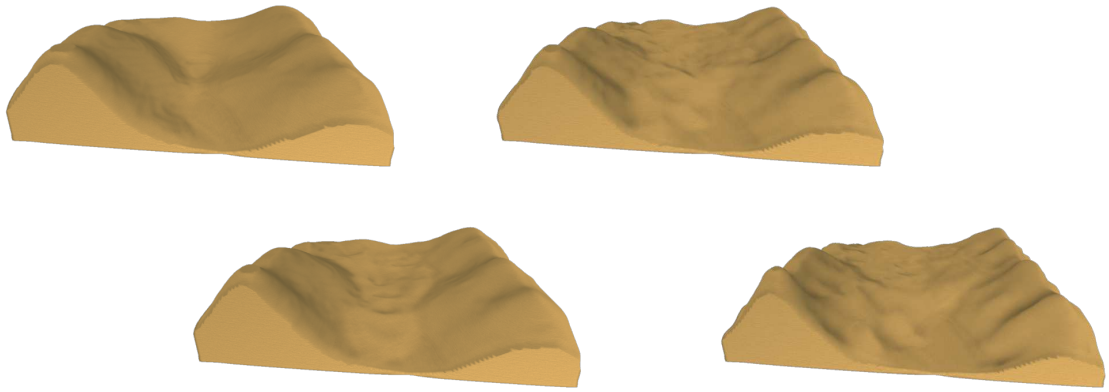

Figure 1.40: *Landslides on a height field representation is emulated by using large particles with high deposition factor and low capacity.*

Landslides are mainly caused by large amounts 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, a low capacity factor, and a high deposition factor ω, they transport sediment over short distances as the velocity quickly drops to 0, and ground material is completely spread along the path, since it is easier to deposit the same amount of sediment as the eroded amount at each collision point. Reducing the density of the particle simulates a rise in viscosity in the settling velocity formula, again increasing the quantity of matter to deposit at contact with the ground. By this means, we can simulate landslides as illustrated in Figure 1.40. A smoother surface results compared to rain erosion, as the rills are filled with sediment 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 volcanic eruption as illustrated in Figure 1.41.

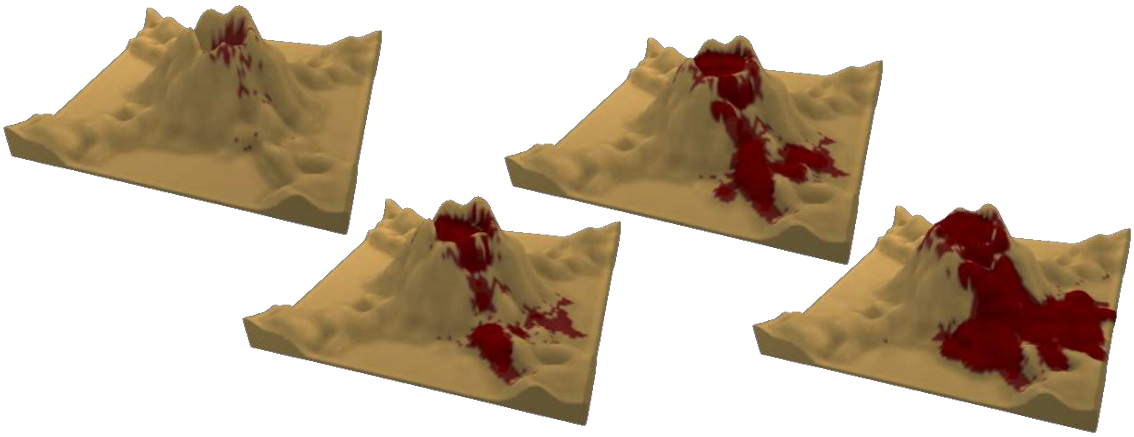

Figure 1.41: *A lava flow emulation by the emission of particles already containing matter from the crater center.*

1.5.5 Karsts

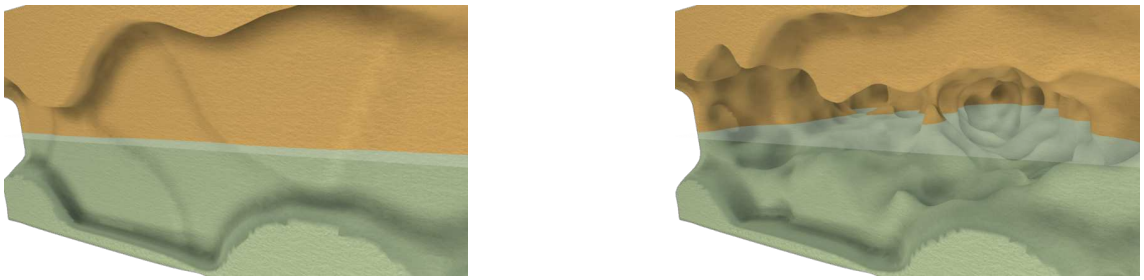

Figure 1.42: *Karstic cavities dug from a plain terrain on a binary-voxel grid representation.*

Karst 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 [PGP*21].

By reducing the deposition factor ω, the particles simulate corrosion (without mass conservation). We use the same particle parameters as in coastal erosion (large 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 the example in Figure 1.42, where we observe a "pillar" formed in the centre, 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 1.43, in which a water runoff is eroding a tunnel without the use of a fluid simulation. Here, when particles bounce frequently on the terrain surface, the coefficient of restitution may be interpreted as a viscosity parameter.

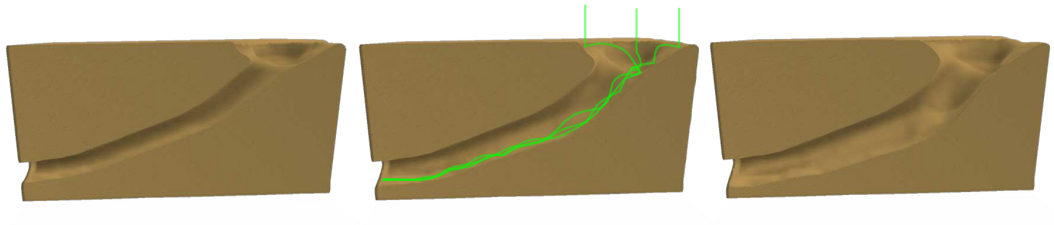

Figure 1.43: *Particles emitted from the entry of a tunnel dig their way inside the volumetric conduit. Green lines: example of particle trajectories.*

1.5.6 Wind

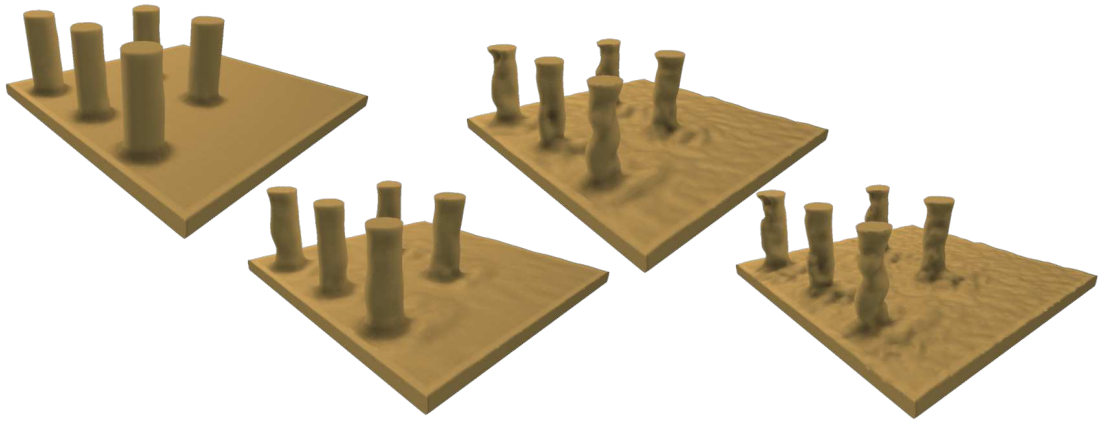

Figure 1.44: *High wind erosion applied on a density-voxel grid simulating abrasion on static pillars using a uniform flow field. Sand is naturally deposited at in front and on the sides of the obstacles.*

Wind erosion is a significant process in desertscape shaping, since there are no obstacles in the airflow path. Air particles can reach high velocities, transporting sand over long distances, forming either dunes or blasting into rocks, eroding them 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 flow field given by the user. This flow field can be provided by a complex simulation, a user-defined wind rose [PPG*19], or a random flow field with a general direction.

The generation of 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 resembling goblins.

Figure 1.44 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 [PPG*19] and used on light particles, the simulation is fast thanks to the low number of collisions each particle has with the ground.

1.5.7 Underwater currents

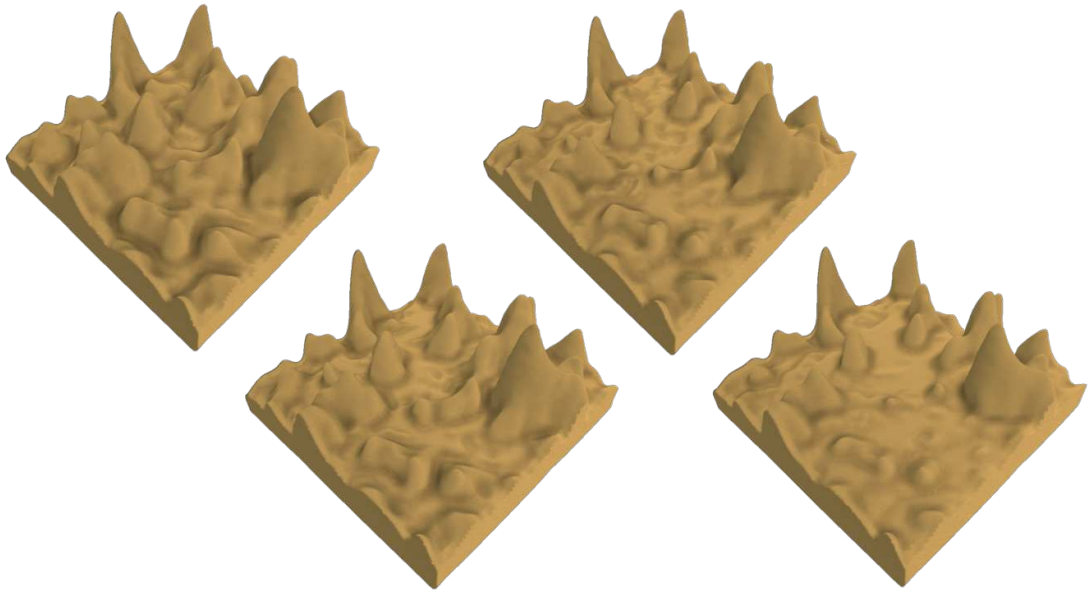

Figure 1.45: *Simulation of underwater erosion rapidly deposits sediments, filling in hollows and smoothing the surface on a height field.*

Procedural generation of underwater 3D terrains has received little attention. The difference between underwater and surface environments lies in the buoyancy force, which is much stronger, meaning that water flow has a much more significant effect on erosion than wind. Taking into account the density of the environment and the velocity field of water in our formulas is key to being able to apply erosion in this environment. Our method works in a water environment by assigning 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 1.45, the velocity field is given by a simple 3D fluid simulation [Sta99] applied to the terrain.

A complex water flow simulation is computed using SIMPLE [CGPS73] 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) are highly affected by the fluid velocity (blue). The densities 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} (Figure 1.46).

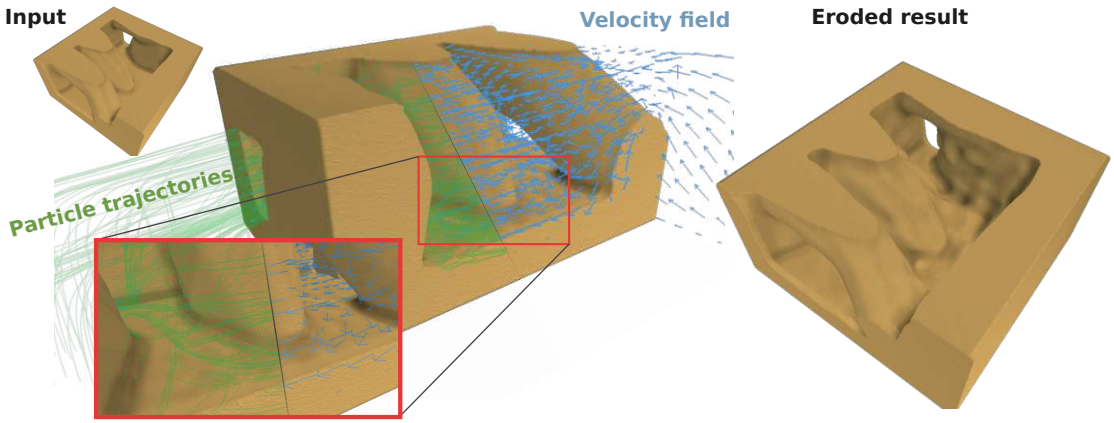

Figure 1.46: *A complex water flow simulation is computed using OpenFOAM. Particle trajectories (green) are highly affected by the fluid velocity (blue). Most of the terrain's exposed surfaces are eroded (bottom).*

Name	**Rep.**	**Dimensions**	**Res**	$\#P$	$\#N$	R	COR	$\rho_{\textbf{particle}}$	$C_{\textbf{factor}}$	ε	ω	**Vel field**	t	Figure
Rain	H	100x100	20	100	10	1.0	1.0	1000	10.0	2.5	0.3	None	4.0	Figure 1.36
Coastal	DV	100x100x30	10	80	3	5	0.1	500	10.0	5.0	0.5	Uniform	0.5	Figure 1.37
Meanders	I	N/A	N/A	10	20	5.0	1.0	1000	1.0	1.0	1.0	(1)	1	Figure 1.38
River	L	100x100	5	100	50	1.5-5	0.5	900	0.1	1.0	1.0	None	2.5	Figure 1.39
Landslide	H	100x100	20	200	10	2.5	0.2	500	0.1	1.0	1.0	None	4	Figure 1.40
Volcano	DV	100x100x40	50	150	30	1.0	5.0	2000	1.0	1.0	5.0	None	0.8	Figure 1.41
Karst	BV	100x100x50	2	1000	40	5	0.5	500	10.0	5.0	0.5	Uniform	20	Figure 1.42
Tunnel	DV	100x100x50	1	100	100	2.5	0.1	500	1.0	1.0	1.0	None	0.8	Figure 1.43
Wind	DV	100x100x50	0.2	100	10	1.5	0.9	1.5	1.0	1.0	1.0	[PPG*19]	0.5	Figure 1.44
Underwater	H	100x100	10	100	50	2.5	0.9	1000	1.0	1.0	1.0	[Sta03]	4	Figure 1.45

Table 1.2: Parameters used for the generation of the terrains presented in Section 1.5, with "Rep" the representation (H: Height field, DV: Density voxels, BV: Binary voxels, L: Layered, I: Implicit), "Res" the resolution in metres per voxel or cell, $\#P$ the number of particles per iteration, $\#N$ the number of iterations, R the particle 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 $v_{\text{fluid}}(\mathbf{p}) = [0\ \sin(\mathbf{p}_x)\ 0]^\intercal$.

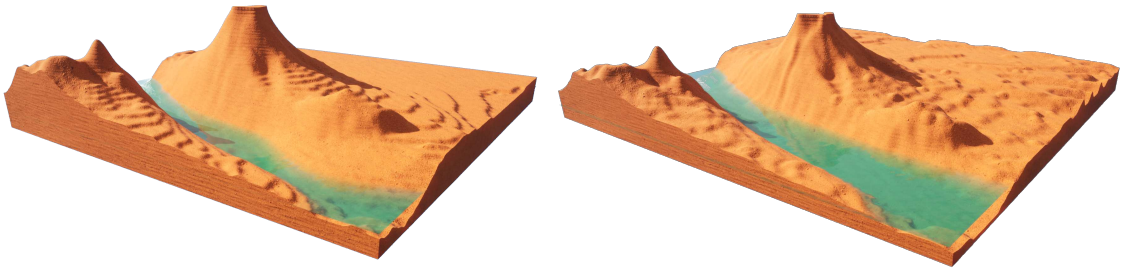

Figure 1.47: *Multiple erosion types can be combined. On an initial synthetic 500x500x50 density voxel grid, wind erosion is applied on the surface of the terrain while hydraulic erosion shapes the rills and the base of the mountains. A water current digs its borders and spreads sediment at the bottom.*

1.5.8 Multiple phenomena

A terrain eroded with multiple erosion phenomena applied on a 500x500x50 density-voxel grid is illustrated in Figure 1.47. Here, water-density particles apply rain on the terrain while the coasts of the river are eroded due to a velocity field defined at the water level. The velocity field defined in the air mainly affects particles with air density, allowing wind erosion to be applied at the same time. The computation of these effects took 7 seconds on CPU.

The particle density directly influence the impact of the velocity field on their trajectory, as we can see in Figure 1.31. Denser particles fall to the bottom of the basin while lighter particles are transported by the wind, increasing the abrasion on the windward slopes and leaving the shaded slopes untouched.

1.6 Comparisons

In the following section, we compare our method with existing ones to show that while we are versatile on 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.

1.6.1 Coastal erosion on implicit terrain representation

[PPG*19] present an erosion simulation method applied to implicit terrains able to create coastal erosion, karsts and caves by adding negative sphere primitives in the terrain's construction tree. The positions of the spheres are determined using Poisson disk sampling at the weakest terrain areas defined by the Geology tree of their model. They simulate the corrosion effect of water on rocks. Our work is also able to approximate this phenomenon by defining the position of these sphere primitives at the locations where the water particles hit the surface. While the computation time for the positions of the spheres is higher due to the fact that we evaluate the position of our particles at every time step in the implicit model (which could be improved by 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 one. This means we can more easily integrate the direction and strength of 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 does not conserve the mass of the terrain, which is acceptable for 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 1.48 displays a comparison of our method with [PPG*19]. Observing real world eroded cliffs in Figure 1.49, our method is able to emulate both plateforms and cavities.

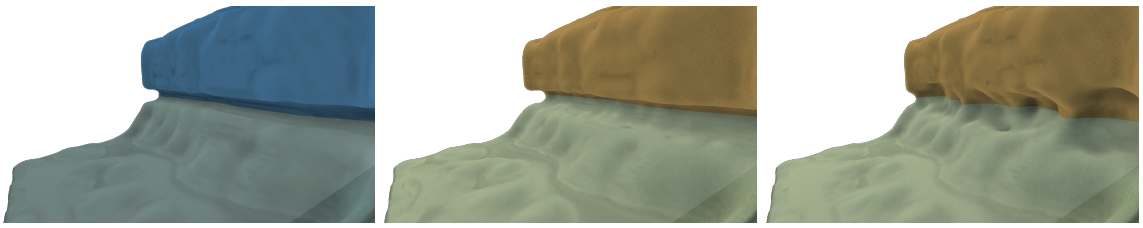

Figure 1.48: *The algorithm proposed by [PPG*19] allows for the simulation of coastal erosion (left), which we can reproduce almost identically by allowing our particles to collide only once with the ground and applying only erosion (centre). If we apply our erosion with full tracking of our particles and using deposition, we can achieve more diverse results (right).*

Abrasion platform associated with the present sea-level in Cyprus [GŞS*16]

John Smith's Bay cave in Bermuda (Photo credit: Akil Simmons)

Figure 1.49: *(Left) [PGP*19] emulate platform coastal erosion, (right) but our method is also able to reproduce the irregular wave-cuts in reef and cliff erosions.*

1.6.2 Weathering on voxel grid representation

[BBFJ10] propose a weathering erosion on voxel grids by approximating and eroding continuously the most exposed voxels. When a solid voxel is decimated, it is considered deposited and is displaced down the slope until a minimal talus angle in the terrain is reached. 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 both the erosion and the deposition processes since the air particles, filled with sediment, fall automatically towards the local minimum of the erosion point. Similarly as their work, we can easily define the resistance value of 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 goblin surrounded by larger goblins and also allows the deposit slope to align more closely with the wind direction (Figure 1.50).

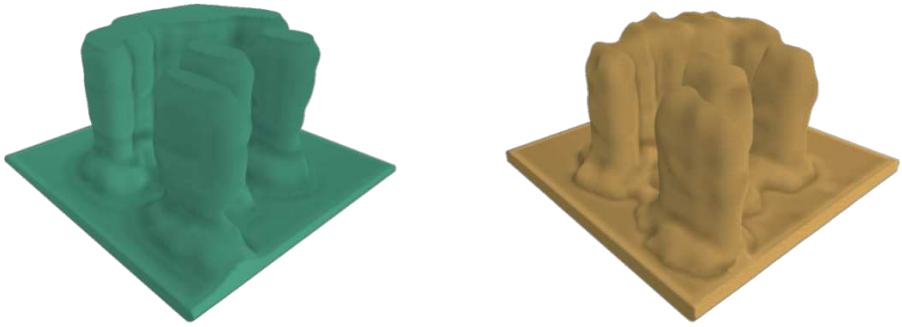

Figure 1.50: *The algorithm proposed by [BBFJ10] allows for an efficient simulation of spheroidal erosion, making the creation of goblins on voxel grids in a plausible way (left). Our algorithm naturally erodes the most exposed areas of the terrain when particles are affected by wind (right).*

1.6.3 Hydraulic erosion on height field representation

[MDH07] integrate and adapt to the GPU the pipe model proposed in [OH95] for fluid simulation. This simulation is simple but efficient enough to approximate the Shallow Water Equations in real time and uses 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 relies on a stable fluid flow that is consistent throughout the lifetime of a particle, but by refining the simulation at each time step instead of at the end of the particle's lifetime, our erosion model is able to reproduce this effect, allowing the terrain to have a single batch of fluid moving through it. Our method can be seen as a generalisation of Mei et al., which can then be used on more than just discrete 2D grids. Figure 1.51 presents a comparison between [MDH07] and our method on an identical terrain. We observe that our terrain surface is not completely smooth, but rather uneven as is the case in real slopes, with rills and gullies, as shown in Figure 1.52.

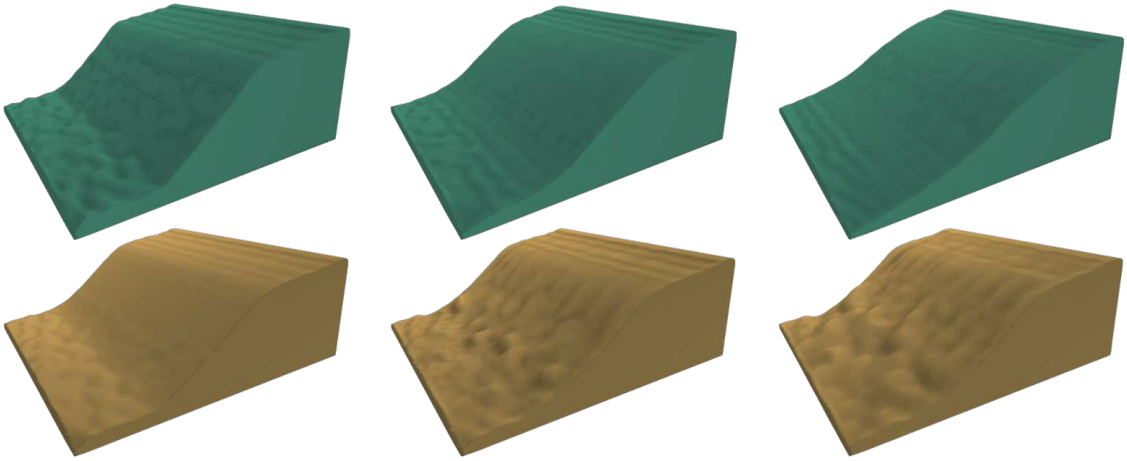

Figure 1.51: *While our resulting geometry for hydraulic erosion (bottom) is less smoothed than that proposed by [MDH07] (top), our method emulate the presence of rills at the ground surface, and allows its application on more terrain representations than just height fields.*

Figure 1.52: *Hydraulic erosion on slope without human intervention tends to create rills or gullies instead of smooth surfaces - Photo credit: [RSW*25] (left and center) and Geoparque Villuercas-Ibores-Jara, Portugal (right).*

1.6.4 Wind erosion on stacked materials representation

[PPG*19] simulate the effect of wind over sand fields defined on stacked materials, creating dune structures, even taking into account obstacles alike [RB04] and different material layers such as vegetation [CCB*17], which are not affected by abrasion [PPG*19]. A wind field simulation is required to produce results, and while [RB04] and [ON00] consider a uniform vector field, this work considers a dynamic vector multi-scaled warped field derived from 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 in Figure 1.53.

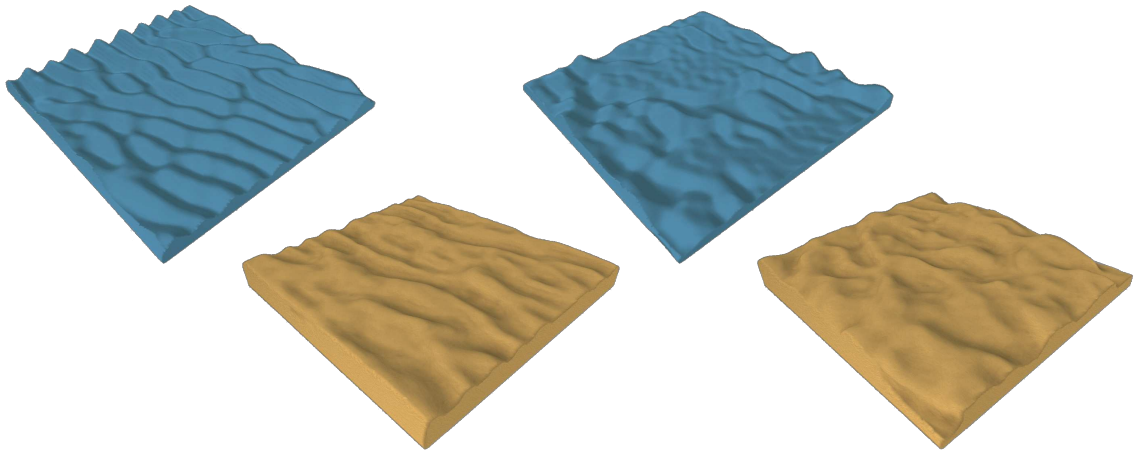

Figure 1.53: *The algorithm from [PPG*19] allows for the generation of desertscapes (top), which we can (at least partially) reproduce with our erosion simulation (bottom). The different effects are achieved by altering the wind direction and strength.*

1.7 Conclusion

We presented a general particle-based erosion framework that decouples terrain alteration from material transport and operates uniformly across height fields, layered models, voxel grids, and implicit 3D terrains. By driving independent particles with simple physics and optionally an external velocity field, our method reproduces a wide range of geomorphological effects using a single set of abstractions and a small, intuitive parameter set: rain-driven rilling, coastal undercutting, river meanders, landslides, aeolian abrasion, and underwater smoothing.

The key advantage is representation-agnosticism: the same erosion logic applies whether the terrain is a 2.5D image or a fully volumetric implicit and voxel field. This enables users and developers to switch representations without redesigning the erosion algorithm, and to trade speed for fidelity by swapping in different flow fields (from hand-authored vectors to CFD) without touching the erosion core.

In practice, while similar particle physics is used on different terrain representations, using similar parameters does not ensure the same eroded terrain. Surfaces and normals being approximated differently have a rippling effect on particle trajectories. Note that not all effects can be applied to all representations, for instance, karst generation on 2.5D data structures.

Realism The 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 localised 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 infinite resolution.

Our method allows 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 to the composition tree. Using skeleton-defined primitives [Hon13, RGF00] from particle trajectories and erosion/deposition values could be a solution to optimise computation time.

Usage of velocity fields In our erosion algorithm, we simplify particle physics to enhance computational efficiency and facilitate parameterisation. 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 that meets the application's 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 Section 1.3.3. However, it is important to note that these are expedient solutions and may not fully capture the precise dynamics of an evolving terrain.

Performances To facilitate parallelisation, 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 particle and ground properties (Young's modulus, friction, material, ...) [YZKF20], further easing the parameterisation 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 performance 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 [RTG60]. Additionally, we acknowledge an issue where particles may collide with the ceiling and the deposition becomes stuck. While a potential resolution involves splitting particles upon impact rather than simply depositing sediment, this introduces complexities to the parallelisation 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 work includes finding a data structure allowing this splitting efficiently, leading to more realistic erosion patterns.

Simulation with multiple materials One aspect we have not addressed is a layered terrain with multiple materials. In the native form of our method, we do not consider the transport of different materials (all sediment is 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 defining transformation rules from one material to another when a voxel is eroded a number of times $\#hits < -C$ or $\#hits > C$. For example, the material "clay" may transform to "sand" when eroded, or to "rock" when many depositions occur.