Contents

Chapter 1

Semantic terrain representation

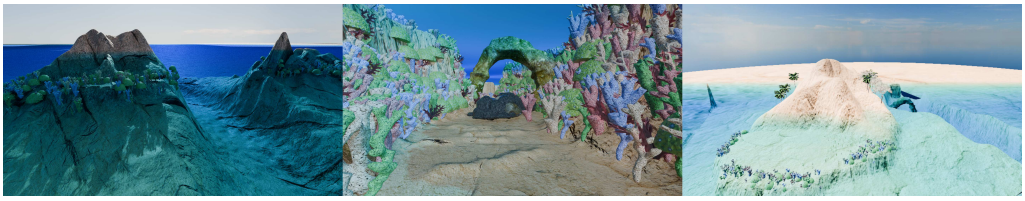

Figure 1.1: *Three underwater scenes using our environmental objects representation. From left to right: an island cut in half by a canyon, the inside of a canyon with rocks and corals, an island with corals, algae and a trench.*

Abstract

This chapter introduces a novel method for procedural terrain generation, which leverages a sparse representation of environmental features to produce landscapes that are lightweight, plausible and adaptable to user desires. The method differs from traditional terrain generation approaches by emphasising multi-scale user interaction and incorporating expert knowledge to model the evolution of terrain features over time. By representing terrain features as discrete entities, or "environmental objects", the method enables dynamic interaction between these entities and their surrounding environment, represented through continuous scalar and vector fields. The generation process is iterative and allows for user-guided modifications at any iteration, including the introduction of environmental events that can influence the terrain's evolution. The proposed approach is particularly flexible, capable of generating underwater landscapes with a focus on large-scale plausibility and detailed, localised feature representation.

Contents

↑ **Back to summary**

1.1 Introduction

While the previous chapter focused on generating the large-scale geomorphology of coral reef islands through sketch-based procedural rules and learning-based generation, these terrains remain purely geometric. In practice, convincing seascapes require not only landforms but also the biotic and abiotic elements that populate them (coral colonies, algae, sediments, rivers, and other ecological features that interact across scales). Addressing this broader challenge calls for a representation that integrates semantic structure, ecological constraints, and user control. To address this challenge, this chapter

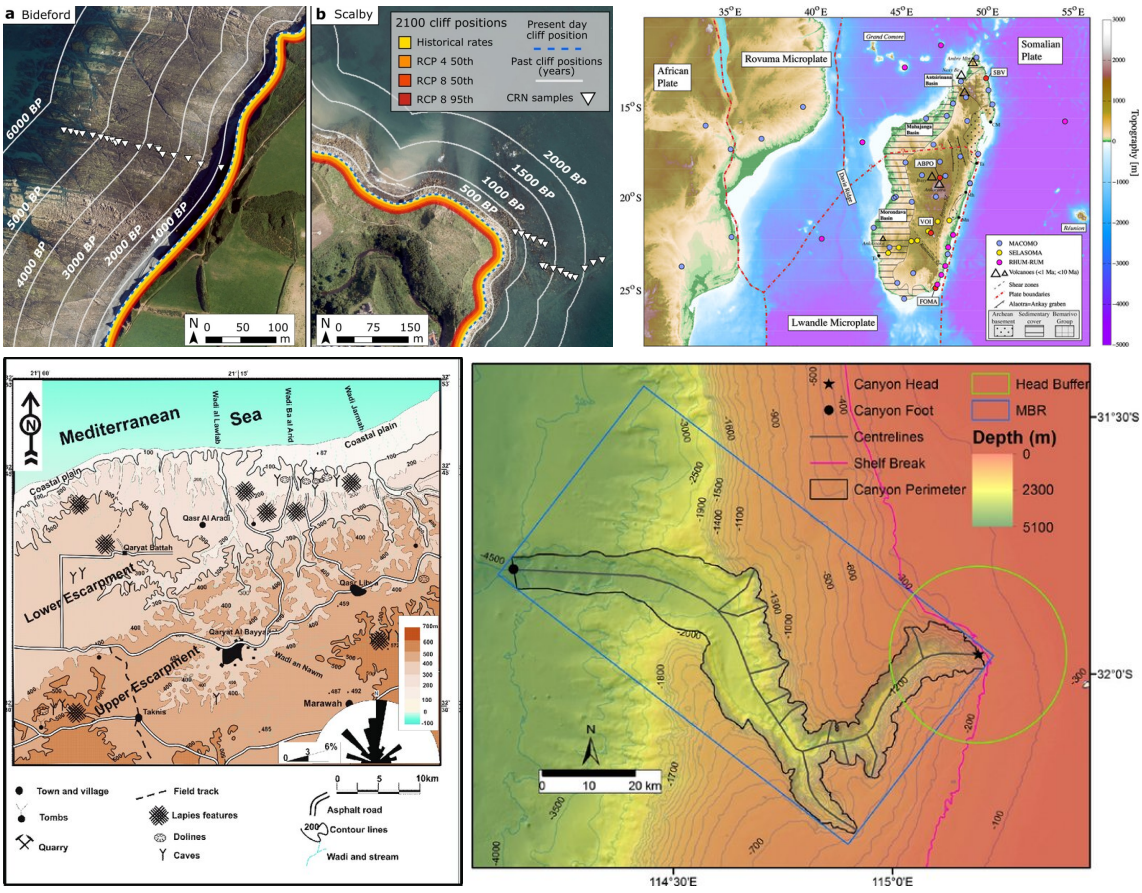

Figure 1.2: *Examples of cartographies used in different fields of natural science. From left to right, and top to bottom: Evolution of coastlines at Bideford, UK and Scalby, UK over the last 6000 years and 2000 years respectively (BP = Before Present) [SRH*22]; sedimentary distribution over Madagascar island [PWA*17]; geological features distribution of karstic landscape at Qasr Libya, Libya [AM09]; localisation of key parameters of Perth Submarine Canyon, Australia [HNHC14]*

turns to ecosystem generation, introducing a semantic framework that unifies terrain and ecology through symbolic primitives and environmental fields.

Procedural ecosystem generation in computer graphics remains largely bifurcated: classical terrain algorithms produce heightfields or meshes without semantic structure, and plant-placement systems adopt fixed-radius or zone-of-influence rules for foliage alone; full ecosystem simulators, though rich in biotic and abiotic processes, are too costly for real-time or interactive content creation. Consequently, no prior work combines sparse, semantic primitives for both living and non-living features, multi-material field coupling driven to steady state, and multi-scale, user-interactive authoring into a single, lightweight ecosystem generator.

To fill this gap, we introduce a pipeline that constructs both biotic (corals, algae, trees) and abiotic (islands, rivers, canyons) elements by instantiating environmental objects (points, curves, regions) which read from and write to continuous environmental fields (elevation, water flow, resource stocks) in an iterative, steady-state diffusion loop. At each step, environmental objects candidates spawn or die via expert-tuned generation and fitness rules, and users can inject global geomorphological events or manually refine any primitive, all while maintaining interactive performance for thousands of objects.

We draw qualitative inspiration from cartographic symbology, where topographic and bathymetric charts employ 0D points (volcanoes, observatories), 1D lines (canyon heads, reef crests), and 2D regions (soil or sediment covers) to strip away geometric complexity and preserve the spatial relationships that underpin expert reasoning (see Figure 1.2). Similarly to these map primitives, our environmental object abstractions enable a “sketch-first, refine-later” workflow: designers begin with a

coarse semantic layout, then progressively zoom in, from mountain ridges down to individual boulders or coral colonies, without ever losing global coherence.

In the following state of the art, we first review classical radius-based heuristics used for ecosystem generation in Computer graphics (FRN, ZOI), and ecological-field theories (EFT) and reaction-diffusion that inform our local interaction and diffusion mechanisms. We then present our environmental objects-environmental attributes pipeline in detail and demonstrate its use for underwater coral-reef environments.

Unlike full-fledged ecosystem simulations, which model temporal dynamics (population growth, predator-prey interactions, nutrient cycling, and fluid flows, ...) over successive time steps to study emergent behavior, our work centers on ecosystem generation, where the goal is to produce a plausible, editable snapshot of a landscape by iteratively placing and culling semantic primitives (environmental objects) that read from and write to steady-state environmental fields; this generation process trades off dynamic fidelity for interactive performance and multi-scale user control, delivering plausible biotic and abiotic scene layouts without the overhead of continuous process simulation.

Building on these observations, this chapter contributes:

- We introduce a sparse, domain-agnostic symbol set (points, curves, regions) that both read continuous environmental fields and write local modifications back to them, creating a closed feedback loop.
- A decay-stabilised reaction-diffusion solver for scalar "materials" and analytical deformation of the vector-flow field yield interactive, steady-state updates without heavy fluid or erosion simulation.
- A focus-and-refine pipeline lets users jump from kilometre-scale planning to sub-metre edits while preserving global coherence.

1.2 State of the art

Simulating ecosystems involves a range of modelling approaches, each balancing biological realism and computational efficiency. Broadly, biological models can treat populations as continuous fields using reaction-diffusion equations, or model each organism individually through agent-based methods. For clarity, we classify existing work in three categories: site-based (grid or tessellation) models, individual-based models, and continuous field models, each emphasising a different balance between local details and scalability. Site-based models discretise space to model local interactions efficiently. In contrast, individual-based models (IBMs) capture fine-scale behaviours and heterogeneity, often using spatial interaction rules (fixed-radius, zone-of-influence, or field of neighborhood methods). Ecological Field Models (EFMs) bridge these approaches by representing environmental properties as continuous fields influenced by organisms, enabling dynamic feedback between species and their surroundings. These diverse frameworks provide flexible tools for capturing complex ecological processes at varying scales and resolutions.

Two main modelling paradigms emerge in the literature: modelling dense populations as continuous media [Tur52], or representing each individual explicitly [Cza98]. The former excels in broad predictions; the latter in local realism.

From a large-scale perspective, where billions of individuals are indistinguishable (e.g., grass shoots, bacteria, whole populations), we treat species as continua and borrow reaction-diffusion models from physics and chemistry.

Conversely, when the number of individuals is smaller or when populations are sparse enough that individuals and their interactions must be tracked explicitly, Individual-Based Models become appropriate. These methods treat organisms as agents that sense and interact with neighbours, making decisions (spawning, growing, and dying for vegetation).

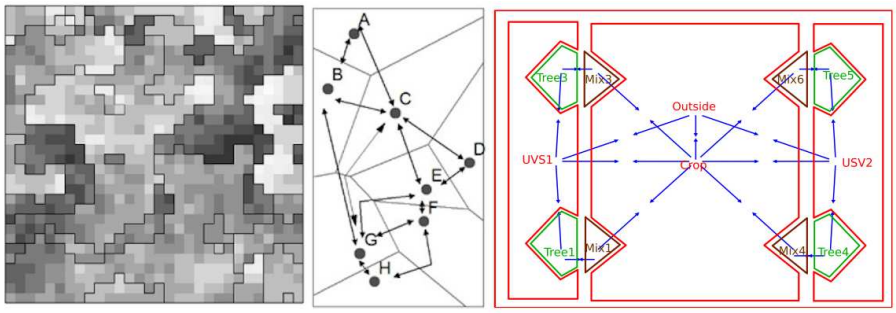

Figure 1.3: *Site-based models partition the environment into discrete regions with locally uniform properties. Examples include (left) regular grids, (middle) Voronoi cells, and (right) generalised topological maps [NR12, LJGS23].*

A hybrid family, the Ecological Field Models [WSWP85], describes how each individual modifies the state of its surrounding environment. Borrowing field theory from physics, they let organisms deposit or absorb continuous "influence fields", creating explicit feedback loops with the environment.

Below, we review the three modelling families (site-based, individual-based, field-based) because our method borrows the sparse symbolic control of IBMs yet retains the continuous feedback of EFMs. We then discuss diffusion models, the mathematical backbone for our environmental fields.

1.2.1 Site-based models

Site-based models represent the environment as discrete regions with uniform properties, interacting primarily with their immediate neighbours (Figure 1.3). The concept generalises to other tessellations such as Voronoi diagrams or arbitrary graphs.

Grid-based models

Grid-based models represent ecosystems by discretising continuous space into a regular arrangement of cells, typically squares or rectangles, each containing uniform environmental properties and ecological conditions (Figure 1.3, left). In this approach, each cell represents a spatially explicit patch hosting one or multiple populations, with interactions primarily occurring between neighbouring cells. The central ecological rationale for employing grid-based models is their conceptual simplicity, computational efficiency, and ease of representing spatial processes such as diffusion, dispersal, competition, or predation [GR05, CCR10, NR12].

Mathematically, grid-based ecological dynamics are commonly described using discrete-time state equations that represent how the state of a cell (e.g., population density, biomass, or nutrient availability) changes according to its local interactions. A general formulation of these dynamics can be expressed as

$$N_{t+1}(x,y) = f(N_t(x,y), \overline{N_t}(x,y), E_t(x,y)) \tag{1.1}$$

where $N_t(x,y)$ is the state variable at cell coordinates (x,y) at time t, $\overline{N_t}(x,y)$ is the neighbourhood of the cell at coordinates (x,y), $E_t(x,y)$ is the explicit environmental factors (e.g., precipitation and soil nutrients), and f is the update function that describes the biological changes given the current state, the neighbourhood and the environmental factors around a cell.

Grid-based models are particularly suited for simulations involving diffusion-like processes or phenomena with clear spatial boundaries. Classic examples in ecology include modelling vegetation dynamics in arid ecosystems, spatially explicit predator-prey interactions, forest-fire spread, and habitat fragmentation effects. Additionally, grid-based representations form the basis for Cellular Automata models,

where cells exhibit discrete states (e.g., occupied, empty, burning), updated synchronously according to predefined state-transition rules.

Grid-based models are commonly used due to computational simplicity, easy integration with geographic information systems (GIS), and suitability for parallel computing [Cza98, NR12]. They facilitate large-scale ecological modelling by clearly defining spatial relationships and boundary conditions, thus making them practical for exploring broad-scale ecological hypotheses or management scenarios.

However, significant limitations exist, as ecological processes may differ drastically at varying scales (e.g., local vs. landscape-level processes). Incorrect spatial resolution of the grid cells can lead to ecological inaccuracies or oversimplifications [Wie89]. Moreover, grid cells assume ecological homogeneity within each cell, which may poorly reflect natural ecological heterogeneity, especially in large cells [Lev92]. Complex ecological interactions, such as asymmetric competition or directional effects, are typically simplified or lost in grid-based discretisation [DL94]. Finally, regular grids impose artificial directional biases (e.g., horizontal/vertical) that can impact ecological interpretations, especially for processes that depend on continuous spatial gradients or isotropy.

Tessellation models

Tessellation models generalise the idea of grid-based approaches by partitioning continuous ecological space into discrete regions based on proximity rather than regular shapes (Figure 1.3, centre). Among these methods, Voronoi diagrams (also known as Thiessen polygons in ecological studies) and their dual, Delaunay triangulations, are particularly relevant and widely applied [LJGS23, Fol19, MU08].

Voronoi diagrams partition a plane based on a set of points (e.g., tree positions, animal territories, or resource centres). Each region in a Voronoi diagram is defined as the set of all points closer to a specific focal point (seed) than to any other point. Formally, given a set of points $p_1, p_2, ..., p_n$, the Voronoi region V_i associated with the point p_i is defined as

$$V_i = x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j), \forall j \neq i \tag{1.2}$$

with $d(x, p_i)$ the distance between the point x and the seed p_i.

Conversely, the Delaunay triangulation is the dual graph structure of a Voronoi diagram. It connects points whose Voronoi regions share a common boundary, thereby explicitly defining neighbourhoods among entities (e.g., individuals or resource patches). This duality between Voronoi and Delaunay representations provides two complementary perspectives: territorial partitioning (Voronoi) and direct adjacency or connectivity (Delaunay).

In ecological modelling, these tessellation methods have significant implications and applications. Voronoi diagrams naturally represent resource partitioning among individuals or populations, capturing territorial or competitive interactions without the artificial constraints imposed by fixed grids [CC06]. For instance, they have been effectively employed in forest ecology to model individual-tree resource competition, where the area of a tree's Voronoi cell directly correlates to available resources and competitive interactions.

The use of tessellated models may become very useful as they adapt to irregular spatial distributions, accurately modelling ecological territories and resource accessibility based on actual entity locations. These models inherently define spatial neighbourhoods without arbitrary distances or predefined shapes, enabling more precise representation of interactions among entities.

Unlike grids, Voronoi cells automatically adjust spatial resolution according to entity density, allowing finer resolution in densely populated areas and coarser resolution in sparse ones.

However, Voronoi-Delaunay models also present several limitations and challenges:

- Dynamic updating of Voronoi diagrams and Delaunay triangulations is computationally more

intensive than fixed-grid models, especially in large-scale or highly dynamic ecosystems where individuals frequently appear, disappear, or move.

- Voronoi models implicitly assume isotropic interactions (competition or resource usage is equal in all directions) and strictly distance-based territories, neglecting directional biases such as prevailing winds, slopes, or anisotropic resource gradients.

Generalised topological maps represent ecosystems using abstract spatial relationships and connectivity structures, emphasising the importance of topological adjacency or interactions rather than explicit geometric positions or shapes [UMTS09]. Unlike grid-based or Voronoi-Delaunay tessellation models, generalised topological maps prioritise topological relationships over exact geometric distances or spatial coordinates (Figure 1.3, right). Formally, these maps can be defined as graphs $G = (V, E)$, where vertices V represent spatial entities (e.g., habitat patches, resources, individuals, or territories), and edges E define ecological interactions, such as connectivity, dispersal pathways, competitive influence, or predator-prey relations [HACV21, MU08, PSP*24].

By abstracting space into relational networks, these maps enable efficient representation and analysis of ecological processes in scenarios where exact geometry may be unknown, less relevant, or overly complex. They are particularly valuable in modelling metapopulation dynamics, habitat connectivity, or animal movement patterns, where the critical ecological factor is the presence or absence of connectivity rather than explicit spatial positions. Furthermore, generalised topological approaches facilitate integration with network theory, providing tools to analyse ecological resilience, robustness, fragmentation, and connectivity dynamics directly through graph-based metrics (e.g., connectivity indices, centrality measures, or modularity analyses) [LJGS23, GBH*12].

However, the use of generalised topological maps also presents several limitations and ecological assumptions:

- Removing explicit geometry sacrifices detailed spatial realism, potentially limiting the ability to accurately model distance-dependent ecological processes such as diffusion or continuous dispersal.
- Ecological outcomes can be highly sensitive to how vertices and edges are defined, necessitating careful selection or validation of the underlying graph structure to avoid artificial ecological patterns or biased results.
- Empirical calibration or validation may be difficult, as topological maps represent abstract ecological relationships rather than measurable spatial distances or tangible boundaries.

Despite these limitations, generalised topological maps remain valuable tools, especially when combined with explicit spatial approaches [ECC*21] or when explicit geometry is unavailable or unneeded such as in metapopulation studies where only patch connectivity is known [DARB18] or where movement follows network-like pathways shaped by the environment rather than simple Euclidean distance [BP22, MCB*14, CARR12].

While site-based models (grids, tessellations, topological maps) provide an efficient and conceptually simple way to represent ecosystems, they impose discretisations that may introduce artificial scales, isotropy assumptions, or directional biases. Our approach differs in that it does not partition space into fixed cells. Instead, we represent the environment as continuous fields updated by sparse environmental objects (EnvObj) anchored on points, curves, or regions. This hybrid representation avoids the rigidity of predefined spatial partitions: continuous fields capture diffusion- and transport-like processes at any resolution, while objects encode semantic features (rivers, reefs, sand patches) that interact with those fields. Conceptually, this places our method between grid-based and individual-based models: similarly as site-based approaches, it supports efficient simulation of field processes, but in the same manner as IBMs, it maintains the individuality and heterogeneity of discrete entities.

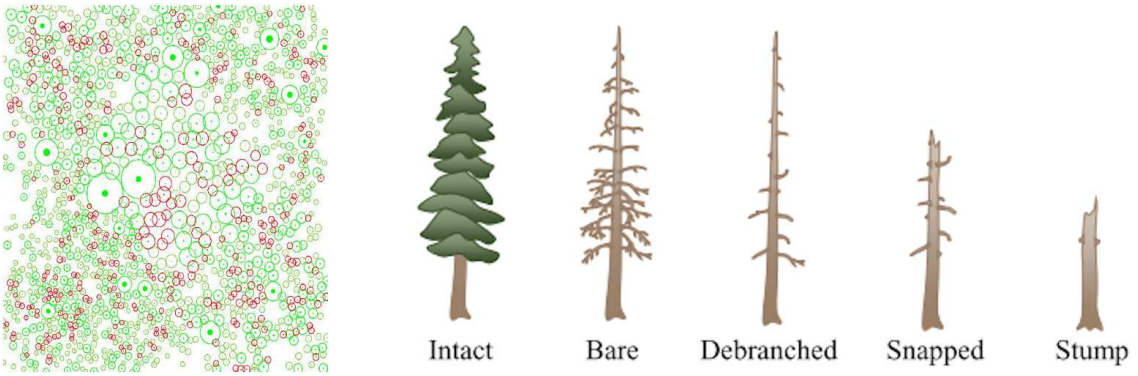

Figure 1.4: *Individual-based models focus attention on each element of the scene, with interaction between neighbouring entities. This perspective allows for more precision on (left) the positioning of each tree [AD06], but also on (right) the life cycle and geometry of the individuals [PGG*24].*

1.2.2 Individual-based models

In contrast to aggregated or spatially discrete models, Individual-Based Models (IBMs) explicitly represent each organism as a distinct unit [CHM17]. IBMs have gained popularity in vegetation modelling because they capture variability in size, spatial positioning, and competitive interactions among individuals. Each tree or plant can be modelled independently, with processes such as growth, mortality, and reproduction shaped by local environmental conditions and the presence of neighbours [Chn13, PGG*24].

This fine-grained representation enables IBMs to reproduce emergent stand-level dynamics from simple local rules. They are particularly effective for modelling competition for light, nutrients, and space, where outcomes depend on relative size or distance among neighbours [MSMM11, ZD20]. To reduce computational costs while maintaining ecological realism, vegetation IBMs often employ distance-based competition methods (e.g., FRN, ZOI, FON) or stochastic simplifications. Recent advances in parallel computing and GPU simulation have further extended the applicability of IBMs to large-scale virtual ecosystems.

Although our discussion is focused on vegetation, IBMs are a general ecological framework. They are applied to animal movement and foraging, predator-prey dynamics, and even disease spread in heterogeneous populations [GR05, Chn13, PD09]. These examples underline the versatility of IBMs, but here we emphasise their role in vegetation modelling and spatial competition, which is most relevant for our purposes.

Fixed-Radius Neighbourhood

In the Fixed-Radius Neighbourhood (FRN) method, the first appearance of a distance-based method, each tree has a radius inside which we consider that there is shade and competition for nutrients. If two trees' radii intersect, a competition allows the stronger tree to survive while the weaker one dies (Figure 1.4, left).

In graphics, FRN-like exclusion is realised by Poisson-disk sampling (Figure 1.5) or blue noise approximations [DHOS00] to generate entities as points that do not fall within another entity's radius. The algorithm can be accelerated using a regular grid structure as proposed by [Bri07]: each cell of the grid has a width of $\frac{r}{\sqrt{n}}$ with n the number of dimensions (usually $n = 2$). Each cell stores at most one sample, limiting neighbour checks to adjacent cells. Each point added to the grid recursively instantiates new random samples in a spherical annulus between radius r and $2r$. Variants of this method guarantee maximal coverage [EDP*11] or GPU parallelisation [Wei08]. However, in the presence of at least two species with different radii, the circle packing problem becomes more complicated and requires a larger number of collision checks, while still relying on the spatial indexing method from Bridson's algorithm.

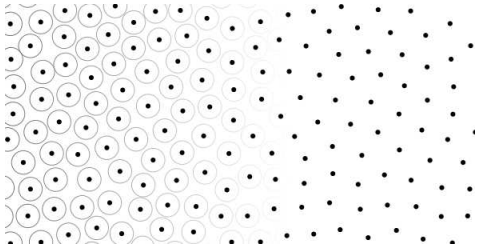

Figure 1.5: *Poisson Disk Sampling iteratively adds points in space and rejects a candidate if it is within radius r of any existing point.*

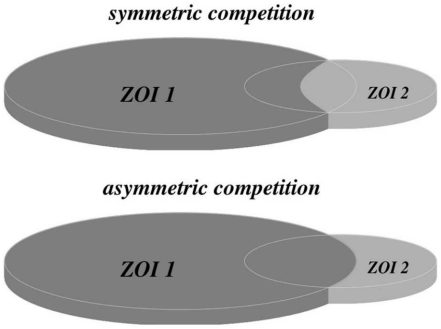

Figure 1.6: *ZOI is represented by regions around entities. The intersection area between two discs defines the competition between the two entities. Top: Symmetric competition (equal resource sharing, with the competition at the intersection divided equally); bottom: Asymmetric competition (larger entity captures more resources, such as light) [PWL*20]*

Zone of Influence

The Zone of Influence (ZOI) model is an ecological method designed to quantify competition among trees by modelling their interactions as gradual rather than binary processes [Cza98]. In contrast to a simple binary approache proposed with Fixed-Radius Neighbourhood, the ZOI model assigns each tree a circular influence zone, typically defined based on measurable attributes such as diameter at breast height (DBH), crown dimensions, or species-specific ecological characteristics.
In this framework, competition between adjacent trees is quantified by calculating the fractional overlap between their influence zones (visible in Figure 1.6).

Mathematically, the intensity of competition C_{ij} exerted by a neighbouring tree j on a focal tree i is expressed as

$$C_{ij} = \frac{A_{\text{overlap},ij}}{A_{\text{ZOI},i}} \tag{1.3}$$

where $A_{\text{overlap},ij}$ is the area of overlap between the radii of trees i and j, and $A_{\text{ZOI},i}$ is the total area of the influence zone of tree i. This intersection area can be computed directly with

$$\begin{aligned} A_{\text{overlap},ij} =& r_i^2 \cos^{-1}\left(\frac{d^2 + r_i^2 - r_j^2}{2dr_i}\right) + r_j^2 \cos^{-1}\left(\frac{d^2 + r_j^2 - r_i^2}{2dr_j}\right) \\ & - \frac{1}{2}\sqrt{(-d + r_i + r_j)(d + r_i - r_j)(d - r_i + r_j)(d + r_i + r_j)} \end{aligned} \tag{1.4}$$

assuming r_i and r_j are the radii of trees i and j respectively, and d the distance between the two centres.

The biological consequences of competition quantified by ZOI, such as reduced growth or increased mortality risk, can be represented through growth-response equations. For instance, tree growth under competition might be modelled as in [Das12, UCCH04]:

$$G_i = G_{\max,i}\left(1 - \alpha \sum_j C_{ij}\right) \tag{1.5}$$

with $G_i \in [0,1]$ the growth of the tree, $G_{\max,i}$ the maximal potential growth if no competition affects it, and α a sensitivity parameter determining how competition impacts growth.

The ZOI model typically assumes symmetric (isotropic) competition, meaning all trees compete equally in every direction, and the competition effect between two trees is reciprocal if their zones and dimensions are similar. However, ecological realism can be increased by adopting an asymmetric (anisotropic) version of the model, which acknowledges that competition is often directionally biased or size-dependent (Figure 1.6). For asymmetric competition, the equation is modified by introducing weighting based on differences in tree size or dominance, for example:

$$C_{ij} = \frac{A_{\text{overlap},ij}}{A_{\text{ZOI},i}} f(DBH_i, DBH_j) \tag{1.7}$$

with $f : \mathbb{R}^2 \mapsto [0,1]$ weighting function that introduces asymmetry between individuals based on size or dominance traits. For example, $f(\text{DBH}_i, \text{DBH}_j)$ can bias competition towards larger trees by giving them a higher share of contested resources, while $f(h_i, h_j)$ with tree heights h_i and h_j can represent light competition by favouring taller individuals. The exact form of f depends on the ecological process of interest (e.g., size dominance, shading, or nutrient uptake).

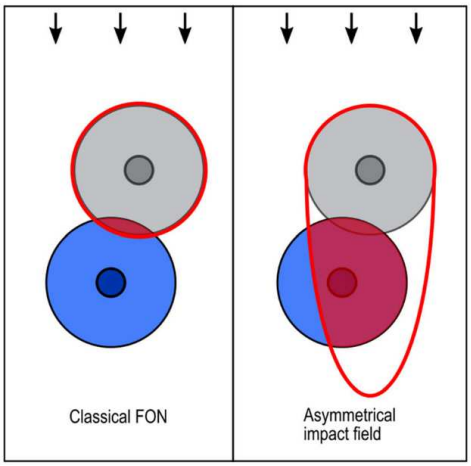

Figure 1.7: *A common simplification is isotropy; real stands often show directional effects (slope, wind, light). Left: competition in classical FON models is identical for each entity; right: a flow direction (black arrows) induce a deformed competition for resources. Asymmetric interaction area is much more complex to compute [PWL*20].*

The ZOI model is theoretically rooted in the understanding that trees compete gradually and continuously, rather than abruptly. However, it does rely on simplifying assumptions (circular, isotropic zones, homogeneous environmental conditions, and static radii within measurement intervals) which can limit its applicability to ecological scenarios involving directional resource gradients or heterogeneous environments (Figure 1.7).

Empirical studies across diverse forest ecosystems have demonstrated that the ZOI approach significantly improves predictions of tree growth, survival, and mortality compared to simpler binary models such as FRN. The nuanced representation of competition offered by ZOI models better captures complex ecological dynamics, particularly in mixed-species stands, uneven-aged forests, or structurally diverse ecosystems [Bel72, BD92].

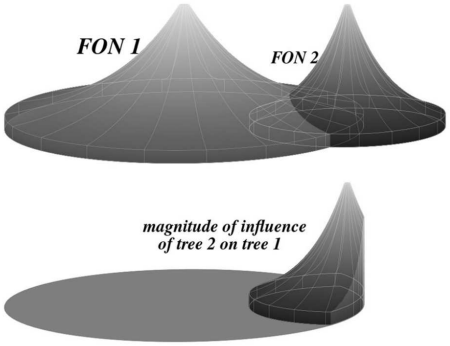

Figure 1.8: *The FON model proposes a smoother version of the ZOI, where the intersection between two radii also accounts for the distance from the tree trunk [PWL*20]*

Field of Neighbourhood

The Field of Neighbourhood (FON) approach provides an alternative method to quantify tree competition by explicitly incorporating distance-dependent and size-dependent effects of neighbouring trees on a focal tree's growth or survival (the tree being evaluated) [BH00]. Unlike the ZOI model, which uses circular zones and geometric overlaps, FON models typically rely on explicit indices that incorporate both the distance and the relative sizes of neighbouring trees, without necessarily defining explicit geometric overlap areas.
In the FON approach, competition intensity affecting a given focal tree is calculated by summing the competitive effects of neighbouring trees, typically as a weighted function of their size and distance (Figure 1.8).

A commonly used formulation of competition intensity FON_i for a tree i at distance r from the trunk is expressed as

$$FON_i(r) = \begin{cases} 1 & \text{for } 0 \leq r < \text{RBH}, \\ e^{-C(r-\text{RBH})} & \text{for RBH} \leq r \leq R, \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

with RBH the radius at breast height (RBH $= \frac{1}{2}$DBH), R the radius of influence of the tree (usually proposed as $R = a\sqrt{\text{RBH}}$ with a scaling parameter a), and C an attenuation coefficient, defined for each species.

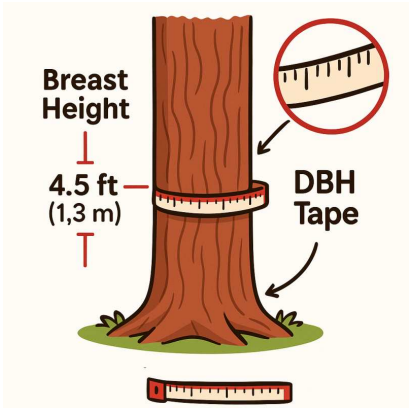

Figure 1.9: *DBH (and RBH) is measured at 1.3 m above ground level.*

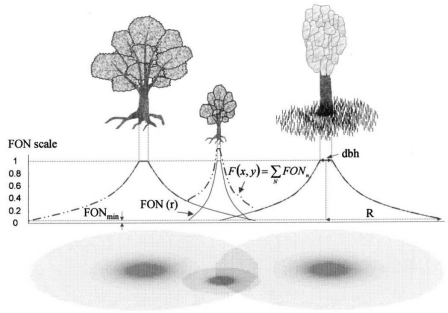

Figure 1.10: *The evaluation of the FON value at any point is done by accumulating all individual fields [BH00]*

The overall competition at any point in space can now be described as

$$F(x,y) = \sum_i FON_i(x,y) \tag{1.12}$$

The competition perceived by a tree i is the average of all FON values intersecting its area A:

$$F_A^i = \frac{1}{A} \sum_{j \neq i} \int_{A_{\text{overlap},ij}} FON_j(x,y)\, da \tag{1.13}$$

Unlike the ZOI method, the FON model does not assume a hard boundary around each tree, instead explicitly capturing a smooth decline in competitive impact with increasing distance. It also inherently accommodates asymmetric competition, since larger and closer neighbours exert stronger competitive influences than smaller or more distant trees (Figure 1.10).

The FON method does not rely explicitly on geometric overlap and is thus flexible and particularly useful in heterogeneous forests. However, to capture the self-thinning property of silviculture, an empirical law in forestry describing how tree density decreases as average tree biomass increases [MHS*19], satisfying $W = CN^{-\frac{3}{2}}$ (with W the average biomass of an individual, N the density, and

C a constant) [Wes84], the generation of the ecosystem should be iteratively simulated with small time steps [AD05], which is computationally expensive. Rather than simulating the full growth model of vegetation, most ecosystem generation algorithms focus on the distribution of the elements of the terrain, abstracting the competition into distance function deformations or statistical optimisation.

By designing non-monotonic field functions around trees, emergent behaviours can appear. [LP02] proposed to apply what they call a deformation kernel on the distance function, resulting in a larger variety of patterns, such as the clustering of bushes and the competition of larger trees (Figure 1.11, top left). Building on this idea, defining deformation kernels for each pair of species (Figure 1.11, top right, presents a kernel matrix for two species) allows the system to include ecological interactions between different species in the terrain. As a result, the simulation can introduce intra-species clumping while maintaining inter-species segregation.

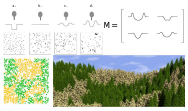

Figure 1.11: *The deformation of neighbourhood kernels enables modelling of different behaviours for species and their interactions. Top-left: different kernels generate different layouts; Top-right: a matrix M encodes a distance kernel for each pair of species; Bottom: using a single matrix provides a way to control the interaction of two species in a scene [LP02]*

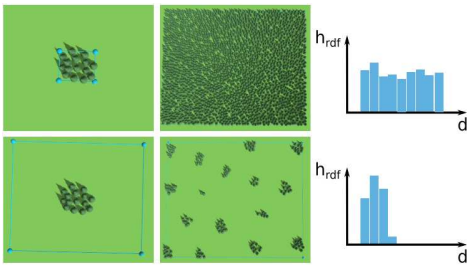

Figure 1.12: *Using statistical distribution from an input terrain, [EVC*15] synthesise a different terrain with similar distributions. Two examples are given (delimited by blue rectangular boundaries): uniform distribution (top) and clustered distribution (bottom). Differences in distribution arise from the voids in the regions of interest.*

Alternatively, learning the spatial distribution of distances between individuals from examples can be used as a statistical metric to introduce interactive tools for ecosystem generation [EVC*15] (Figure 1.12). In this work, a user-provided patch serves as an example: the system extracts descriptors such as histograms of inter- and intra-species distances, or correlations with environmental variables such as slope, and reproduces these distributions in other regions. Unlike kernel-based models, which require explicit design of influence functions, this approach learns statistical descriptors directly from data.

This method is not limited to vegetation but generalises to any scene elements (e.g., houses, rocks, roads), making it highly flexible for procedural content generation. From an ecological standpoint, it resonates with point-process modelling, where distributions of distances or clustering statistics are fitted to observed spatial patterns [Ben21, Bad10]. For computer graphics, the strength of this approach is interactivity and scalability: complex, realistic layouts can be synthesised quickly without costly simulation, while retaining ecological plausibility through statistical consistency.

While the FON model provides a biologically detailed representation of competition, its computational complexity makes it significantly more challenging to implement in large-scale simulations. Unlike the ZOI model, which relies on predefined geometric overlaps that can be calculated efficiently, the FON approach requires integrating a continuously decaying competition function over an arbitrary area of influence. This necessitates evaluating the function at multiple points across overlapping zones, increasing computational demand. Moreover, the additive nature of FON, where each tree contributes to the total competition field at every spatial point, further exacerbates the computational cost, scaling poorly in dense forest stands. The need for numerical integration or spatial discretisation makes FON computationally expensive compared to ZOI, which only requires simple geometric calculations. Consequently, while FON is useful for small-scale or highly detailed ecological studies, it is generally impractical for large-scale forest simulations.

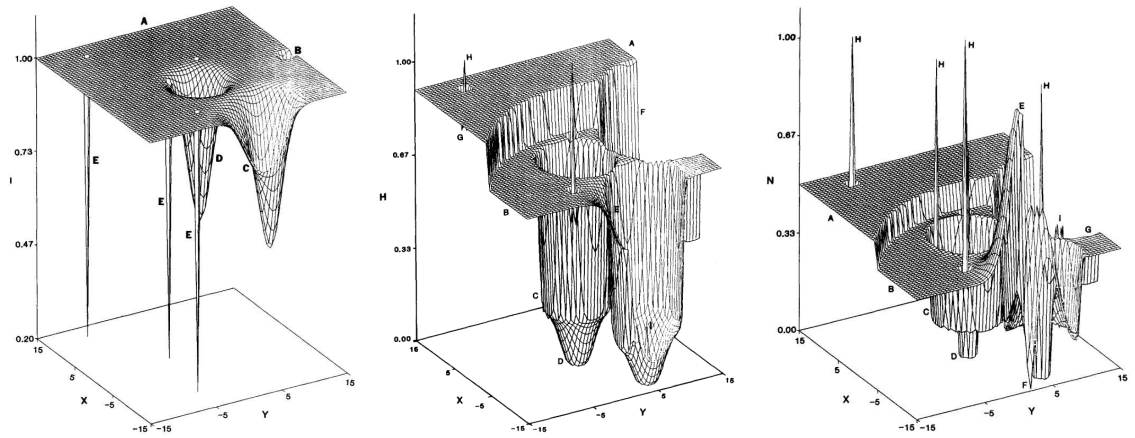

Figure 1.13: *Ecological Field Models represent the environment as a mathematical function where environmental properties such as humidity, shade, and nutrients are seen as distinct fields (from left to right respectively) created by the presence of trees, bushes, and flowers [WSWP85]*

Whereas FRN, ZOI, and FON define interactions through explicit neighbour relations, our approach shifts this logic into continuous environmental fields. Individuals still act as discrete units, but they interact indirectly by modifying and sensing shared fields, aligning IBMs with the Ecological Field Model (EFM) perspective.

1.2.3 Ecological Field Models

While the site-based and individual-based approaches have been successfully used in various ecological studies, many real-world ecological processes do not fit neatly into a discrete grid or a set of independent agents. Instead, they operate in a continuous spatial domain, where organisms interact with smooth environmental gradients rather than discrete neighbours. For example, competition for light and nutrients is not limited to direct neighbours.

To address these challenges, Ecological Field Models (EFMs) provide an alternative framework where environmental properties are represented as continuous, spatially explicit fields [WSWP85]. Rather than defining interactions through direct contact (as in IBMs) or discrete adjacency (as in SBMs), EFMs describe how organisms modify and respond to smooth environmental gradients [Chn11, SRSS12].

From a computer graphics perspective, EFMs share similarities with Signed Distance Fields (SDFs). Just as SDFs represent geometry through continuous distance fields, EFMs represent organism-environment interactions through continuous ecological fields, allowing interactions to be modelled as smooth gradients rather than discrete neighbour effects.

EFMs are based on the fundamental idea that organisms are not isolated agents but rather integral components of their environment, influencing and being influenced by it in a spatially explicit manner (Figure 1.13 shows the effect of multiple vegetation entities on the humidity, shade, and nutrient fields of the environment). The core ecological principles behind EFMs include two key points:

Continuous organism-environment interactions: Organisms modify their surroundings beyond their immediate location. Environmental factors such as soil nutrients, water availability, light exposure, and chemical signals influence other species gradually across space.

Environmental feedback: Species respond to spatially varying gradients rather than direct neighbour-based interactions, as seen in phototropism (guiding growth toward light) [PSK*12] or root development towards nutrient-rich soils [LKM*23].

The combination of these two principles creates natural feedback loops that are explicitly modelled,

allowing a more realistic representation of ecological interactions compared to static environmental assumptions in traditional models.

EFMs provide a continuous representation of environmental factors, bridging the gap between grid-based and individual-based models by modelling smooth ecological gradients, instead of forcing interactions into discrete cells. This method allows organisms to influence and respond to fields dynamically, while providing a scalable framework that avoids the exponential computational cost of tracking every individual separately.

While EFMs provide a biologically grounded approach to modelling spatial interactions, their implementation requires an analytical framework to define how organisms modify and respond to environmental fields.

An ecological field represents a spatially varying environmental property across a landscape. Mathematically, we can define an n-dimensional ecological field as a scalar field F for each point $\mathbf{p} \in \mathbb{R}^n$:

$$F(\mathbf{p}) : \mathbb{R}^n \mapsto \mathbb{R} \tag{1.14}$$

Any organism i in the scene has an influence on the ecological fields through an influence kernel K_i. These functions describe how an organism's presence affects its surroundings. We can then evaluate the ecological field with N entities:

$$F(\mathbf{p}) = F_0(\mathbf{p}) + \sum_{i=0}^{N} K_i(\mathbf{p}) \tag{1.15}$$

with $F_0(\mathbf{p})$ the baseline environmental state.

The primary challenge in computer graphics is how to efficiently store, sample, and modify continuous environmental fields. Since EFMs define spatially varying properties, they require a structured representation that balances accuracy, efficiency, and scalability.

Several strategies have been proposed to implement ecological fields in practice, ranging from grid discretisations [WSWP85, SRSS12] to procedural functions inspired by noise and implicit surfaces [Per85, FPRJ00]. In computer graphics, field-based representations are long established as a means to control geometry or appearance, for instance through implicit modelling, distance fields, or scalar textures. Building on these ideas, the Environmental Objects framework [GPG*16] can be seen as a concrete instantiation of EFM principles: fields are directly attached to objects, enabling them both to emit and sense spatial influences. Although originally designed for urban scenes, its conceptual alignment with EFMs illustrates how continuous organism-environment feedbacks can be embedded into procedural modelling systems.

Environmental Objects

The Environmental Objects system [GPG*16] implements this principle in practice by extending scene assets with scalar fields (e.g., heat, humidity, occlusion). These fields govern local interactions, such as snow, leaves, or soot accumulating on surfaces, and can in turn influence the appearance and behaviour of nearby objects. Originally developed for interactive urban set dressing, the method introduces a hierarchical framework for procedural scenes in which object-level fields control both visual detail and environmental coherence (Figure 1.14).

- Environmental Objects in a virtual scene dynamically adjust their appearance based on surrounding scalar fields, allowing for realistic procedural effects illustrated with snow deposition, leaf accumulation, and icicle formation. Unlike global simulations, which are computationally demanding, this model uses local environmental fields to procedurally generate details without requiring extensive physics-based calculations.
- Users can modify environmental parameters, such as temperature fields, to influence scene aesthetics in an intuitive and predictable manner. The model enables scene composition through

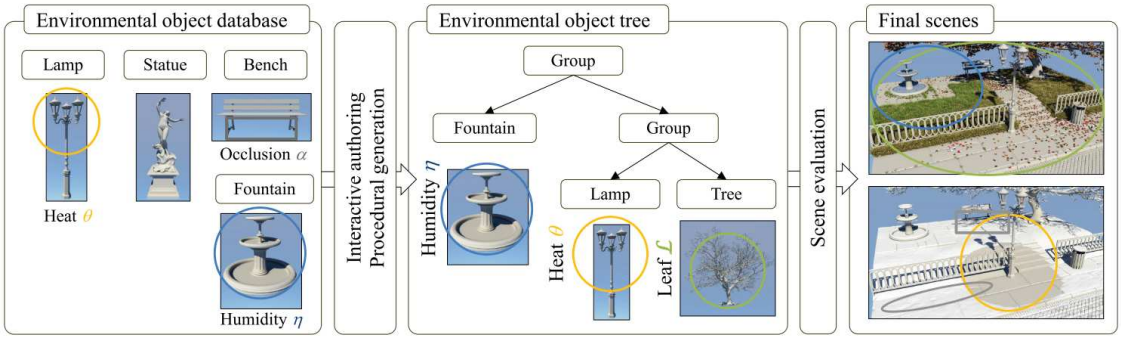

Figure 1.14: *The Environmental Objects framework [GPG*16] presents a practical methodology for integrating spatially explicit environmental interactions into procedural modelling. The Environmental Objects concept aligns closely with EFMs, providing a computationally feasible way to embed field interactions into simulated ecosystems.*

level of details using object groupings, optimising the computation of large-scale environments while maintaining local control over environmental interactions.

By employing implicit primitives (to define the environmental fields) and procedural blending techniques, the framework enables the efficient simulation of environmental changes while allowing for real-time scene modifications.

In this method, each 3D model can be associated with "procedural effects", which regroup scalar fields representing changes in the environment (Figure 1.14, left). These fields are defined using implicit primitives (point-, spline-, surface- or volume-based) allowing the environment properties to be changed along a path, a surface or a volume, instead of only using points as in the initial EFM. This enables the system to represent, for example, the diffusion of heat around a pipe, which is not possible with the latter. The computation of an environment property is then an agglomeration of all the scalar fields associated with the given property (Figure 1.14, middle). The authors propose computing all scalar fields, such as the heat field θ, at any position $\mathbf{p}$ by the accumulation of objects' fields, in a similar manner as Equation (1.15):

$$\theta(\mathbf{p}) = \theta_0(\mathbf{p}) + \sum_{i=1}^{n} \theta_i(\mathbf{p}) \tag{1.16}$$

and the occlusion field α as the product of all objects' occlusion fields:

$$\alpha(\mathbf{p}) = \alpha_0(\mathbf{p}) \prod_{i=1}^{n} \alpha_i(\mathbf{p}) \tag{1.17}$$

assuming θ_0 and α_0 to be global scalar fields for temperature and shade, symbolising the baseline environmental state in the ecological domain.

Using the field values, the system instantiates small details such as leaves by sampling the space with candidate "anchor positions" (Figure 1.14, right). If the field values satisfy certain conditions at the anchor position $\mathbf{p}_i$, the small-scale objects can be instantiated at this position (for example, for leaves: $\alpha(\mathbf{p}_i)l(\mathbf{p}_i) < d_i$ using the occlusion field α, the leaves deposition field l and the distance of the anchor position to the depositing surface d_i as shown in Figure 1.15). This method was extended to modify the geometry or texture of the object based on the field values (in Figure 1.15, the heat field and the humidity field), to manually move or rotate each of the objects, or even integrate user-defined density fields to populate entangled details on the ground [GGG*16].

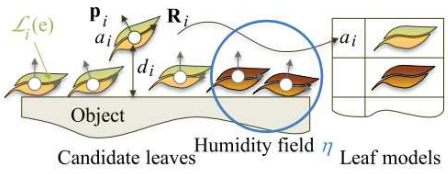

Figure 1.15: *Leaves are instanced if the field values are fitted for leaves presence, and the geometry and texture are altered depending on the environment [GPG*16].*

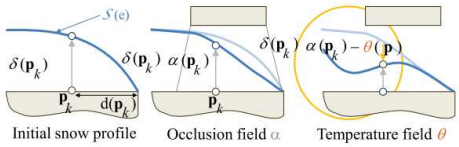

Figure 1.16: *Snow deposition is a displacement of vertices along the surface normals of the mesh (left), modulated by the value of fields such as occlusion (middle) and heat (right) [GPG*16].*

On the other hand, the geometry of large objects can be altered using the scalar fields. The authors present a simulation of snow deposition on the objects by displacing the vertices i in the direction of the original surface normals $\vec{n}_i$ by an amount depending on the environment (Figure 1.16):

$$\mathbf{p}'_i = \mathbf{p}_i + (\alpha(\mathbf{p}_i)s(\mathbf{p}_i)g(d(\mathbf{p}_i)) - \theta(\mathbf{p}_i))\,\vec{n}_i \quad (1.19)$$

with s the snow field, and g a snow elevation function based on the distance between the original vertex and the border of the mesh.

In addition to the EFM framework, environmental effects from environmental objects are not limited to point sources (fields can be emitted by splines and volumes) as the scalar functions may use a spline primitive or be extended to volume primitives (Figure 1.17). Translated to the ecological domain, this methodology extends the EFM as the inclusion of water bodies, essential for realistic vegetation simulation, can be taken into account by considering environmental effects from rivers and lakes which are often modelled as splines and regions in 3D modelling [EPCV15, GGG*13, GGP*15].

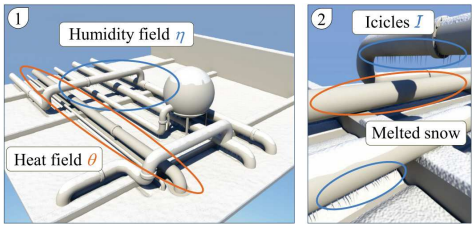

Figure 1.17: *The heat field produced by a pipe uses a distance function from a line instead of a single point in space [GPG*16].*

The Environmental Objects method can be regarded as a variant of EFMs, since both rely on scalar fields that objects emit and sense. They share a common substrate of composable fields, but their purposes differ: Environmental Objects mainly enrich visual appearance, while EFMs model ecological dynamics. This overlap suggests cross-overs: Environmental Objects techniques can improve the scalability and visualisation of EFMs, while EFMs can provide Environmental Objects biologically grounded feedback loops.

Our approach adopts this fields-attached-to-objects view, but upgrades it to a bi-directional simulator: objects both read fields to adapt and write fields to decide where/whether they should exist, yielding emergent spatial distributions.

However, classical EFMs assume static fields, whereas many ecological processes evolve dynamically (e.g., soil moisture depletion, growth-induced shading, or decaying scent trails). By integrating biologically inspired feedback loops, procedural generation could extend beyond static environmental effects, enabling simulations of growth, decay, and species interactions [OL01, PMG*22]. Capturing such time-dependent feedbacks requires a mathematical framework for field dynamics. The next section therefore turns to reaction-diffusion equations, which provide the computational backbone for

modelling growth, depletion, and decay in ecological fields.

Advection-Reaction-Diffusion

Reaction-diffusion models describe how a quantity $u(x,t)$ (e.g., population density, nutrient concentration, sediment stock) evolves under two coupled processes: local reactions and spatial diffusion. In its classical isotropic form [Tur52, OL01]:

$$\frac{\partial u}{\partial t} = D\nabla^2 u + R(u), \tag{1.20}$$

where D is the diffusion coefficient and $R(u)$ encodes local growth, decay, or interactions. Adding transport by external flows yields the full advection-reaction-diffusion formulation.

Diffusion

Diffusion models the random spread of material, following Fick's law:

$$\text{Flux} = -D\nabla u. \tag{1.21}$$

This produces isotropic spreading, equivalent to convolving point inputs with a Gaussian kernel. In heterogeneous environments, anisotropic diffusion generalises D to a tensor $\mathbf{D}$ [Ram24], introducing directional bias (e.g., slope-driven seepage, anisotropic dispersal in currents). In graphics, diffusion has long been exploited for mesh smoothing and fairing [Tau95], anisotropic image filtering [PM90], and texture synthesis based on reaction-diffusion patterns [Tur91]. GPU acceleration allows large grids to be updated interactively, which directly informs our use of diffusion for transporting environmental materials.

Reaction

The reaction term governs local dynamics:

Linear reaction: $R(u) = \lambda u$, describing exponential growth ($\lambda > 0$) or decay ($\lambda < 0$).

Nonlinear reaction: logistic growth, predator-prey interactions, or Lotka-Volterra competition terms capture resource limitation or trophic interactions [BC12, Ver44].

In our framework, the reaction term corresponds directly to the production or consumption of environmental materials by environmental objects. For example, a coral colony deposits calcium carbonate ($\lambda > 0$), while erosion removes sediment ($\lambda < 0$). In graphics, reaction-diffusion systems have been widely used for texture and pattern synthesis, where extending $u(x,t)$ to multiple interacting fields enables the emergence of Turing-like patterns such as stripes or spots [Tur91, WK91, SKJY06].

Advection

Advection introduces directed transport under an external velocity field $\mathbf{v}(x)$ (e.g., currents, wind, gravity-driven runoff):

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D\nabla^2 u + R(u). \tag{1.22}$$

Unlike diffusion's random spread, advection shifts material coherently along streamlines, aligning with our treatment of water currents as vector fields [BGI*20, OAL*17]. In graphics, this formulation underpins fluid solvers such as Stam's "stable fluids" [Sta99], where advection steps move smoke, fire, or water density across a grid. GPU-accelerated advection is also widely used in real-time effects such

as smoke trails or rivers, making it a natural tool to simulate ecological transport processes, which will be presented in depth in **??**'s state of the art.

Decay and steady state

To avoid indefinite accumulation, a decay term $-\mu u$ is added:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D\nabla^2 u + R(u) - \mu u. \tag{1.23}$$

The decay term $\mu > 0$ guarantees that even with continuous deposition, the system relaxes toward a steady state where inputs balance dispersal and losses. This makes advection-reaction-diffusion models particularly suitable for efficient simulations of environmental materials, where explicit time-stepping can be replaced by directly solving for equilibrium. In graphics, decay is often added for numerical stability and realism: without it, transported quantities such as heat, smoke, or density fields accumulate without bound. In our case, decay serves the same dual purpose: guaranteeing stability and providing interpretable equilibrium states.

Green's function solutions

For point sources, the advection-reaction-diffusion equation admits analytic solutions via Green's functions. For example, in 2D with isotropic diffusion, constant advection $\mathbf{v}$, and decay μ, the fundamental solution is a shifted, damped Gaussian:

$$G(x,t) = \frac{1}{4\pi D t} \exp\left(-\frac{\|x - \mathbf{v}t\|^2}{4Dt} - \mu t\right). \tag{1.24}$$

This solution represents a spreading plume: diffused by D, advected along $\mathbf{v}$, and exponentially damped by μ.

For extended sources such as curves (Γ) or regions (Ω), the solution is the convolution of the point-source Green's function with the source geometry:

$$u(x,t) = \int_\Gamma G(x-y,t)\, dy, \tag{1.25}$$

$$u(x,t) = \int_\Omega G(x-y,t)\, dy. \tag{1.26}$$

Closed-form solutions exist only for simple shapes such as infinite lines and disks. For arbitrary geometries, these integrals must be evaluated numerically. This distinction is essential in our framework: point-like environmental objects can exploit analytic kernels, but curve- and region-based environmental objects require numerical integration.

The advection-reaction-diffusion formalism provides a compact vocabulary to describe four fundamental mechanisms of ecosystem dynamics: diffusion (random spread), reaction (local production or consumption of resources as environmental materials deposition and uptake), advection (directional transport by flows) and decay (loss mechanisms ensuring stability).

These mechanisms are the backbone of how we model the interaction between discrete environmental objects and continuous ecological fields, combining analytic tractability for simple cases with numerical flexibility for complex geometries.

Figure 1.18: *Environmental objects can be visualised by 3D triangular meshes. From left to right: rocks, trees, corals, arches.*

1.3 Core concepts

Our approach introduces the concept of environmental objects, simplified, scale-agnostic terrain features that interact with their surroundings to simulate complex ecosystem dynamics. These environmental objects, which represent natural features (corals, algae, islands, rocks, ...) influence and are influenced by environmental fields (temperature, humidity, elevation, ...).

In this section, we present the different related concepts that underpins our method. We first introduce the continuous environmental fields (environmental attributes), then the discrete actors that perceive and affect these fields (environmental objects), before detailing how they modify the fields (environmental modifiers) and how the full pipeline orchestrates their interaction.

1.3.1 Environmental attributes

In geography, and in the Ecological Field Models reviewed earlier, a "field" is a spatially continuous variable defined at every point of the domain, encompassing scalar fields (elevation, temperature) and vector fields (wind, current). In the following, we name every such quantity an environmental attribute.

In an ecosystem simulation, each actor has an impact on all other actors, which results in an exponentially growing computation effort which becomes problematic as the number of elements of the terrain increases. Direct pairwise interaction between n organisms scales as $\mathcal{O}(n^2)$. Instead, we adopt the standard EFM strategy: every environmental object writes to local environmental attributes through its environmental modifier. Each object then reads them, turning the global complexity into a linear complexity per simulation step and thus remaining interactive even for thousands of objects.

We bundle the currently supported fields under the unified term "environment", denoted as $\mathcal{E} = (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$, where $\mathcal{H}$ (height) and $\mathcal{L}$ (water level) are scalars, $\mathcal{W}$ (water currents) is a 2D vector field, and $\mathcal{M}$ (environmental materials) is a vector of per-point resource stocks (sand, salt, moisture, limestone, ...).

1.3.2 Environmental objects

A geographical feature, also called object or entity, is defined as a discrete phenomenon located at or near the Earth's surface. It represents geographic information that can be depicted in maps, geographic information systems (GIS), and other forms of geographic media. This term includes both natural and human-made objects, ranging from tangible items (e.g., buildings or trees) to intangible concepts (e.g., neighbourhoods or savanna). Features are distinct entities with defined boundaries, differentiating them from continuous geographic masses or processes occurring over time. They can be categorised as natural features, such as ecosystems, biomes, water bodies, and landforms, or artificial features

(e.g., settlements, administrative regions, and engineered constructs). We use the term environmental object in this work to avoid the ambiguous term "feature".

Each environmental object is anchored by a skeleton (a point, curve, or region in the cartographic sense) which captures its footprint while postponing heavy geometry generation (Figure 1.19).

Point-based: isolated boulders, individual trees, individual corals, ...

Curve-based: river, coral reefs, fault scarps, ...

Region-based: islands, forests, sediment patches, ...

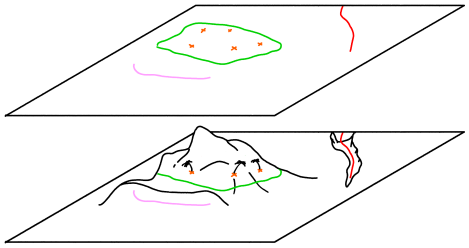

Figure 1.19: *Environmental objects, from 2D (top) to their 3D geometry (bottom)*

Via its specified environmental modifiers, an environmental object locally deposits, removes, or diverts environmental materials, thereby updating the surrounding environmental attributes. Conversely, its viability is re-evaluated through a user-defined fitness function that samples those same fields. If viable, a skeleton fitting function refines the skeleton (e.g., a river streams downhill, and seagrass meadow patches are limited to luminous areas).

1.3.3 Environmental modifiers

The environment determines if an environmental object can survive at a certain position. When an environmental object is placed, its surrounding environmental attributes are affected through environmental modifiers $\mathcal{E}^+ = (\mathcal{H}^+, \mathcal{W}^+, \emptyset, \mathcal{M}^+)$ where $\mathcal{H}^+$ defines a change of height, $\mathcal{W}^+$ the modifications of the water current field, and $\mathcal{M}^+$ the environmental materials alterations. The water level $\mathcal{L}$ is not modified by an object, hence $\emptyset$.

Environmental objects are subject to altitude constraints. However, to maintain a clear distinction between semantic modelling and 3D modelling, we do not compute the exact physical shape or detailed height field of each environmental object. Instead, we define a coarse height function $\mathcal{H}^+$, a parametric surface derived from the skeleton, that provides a coarse estimate of local elevation changes. This coarse proxy lets us evaluate altitude-dependent rules without the cost of globally computing the height field.

Altering the vector field $\mathcal{W}$ is performed by composing the individual contributions $\mathcal{W}^+$ of each object at position $\mathbf{p}$, following [WH91]. For efficiency we adopt an analytical deformation kernel inspired by Kelvinlet solutions [GJ17] to aggregate these local disturbances.

Each environmental object possesses intrinsic environmental materials that both spread and are absorbed around its skeleton over time. For example, a coral colony deposits limestone while simultaneously slowing currents; the reef then uptakes that limestone for growth. In our model the colony contributes a deposition term $\mathcal{M}^+_{\text{limestone}}$, and the reef absorbs it. No direct object-object communication is required.

Scalar environmental attributes are updated by adding or removing mass around the skeleton and then diffusing it, biased by $\mathcal{W}$. We assume a steady state regime, guaranteed by a decay rate $\mu \in [0, 1[$ that prevents unbounded accumulation during advection-diffusion.

1.4 Pipeline

The method is structured into three key phases:

initialization sets up the foundational terrain elements and environmental objects definitions;

iterative generation process instantiates environmental objects and updates their surrounding environment;

output stage that yields a sparse representation of the scene's features.

This section details each phase and explains how the system dynamically adapts to user input and environmental changes. Algorithm 1 summarise the overall process of our method.

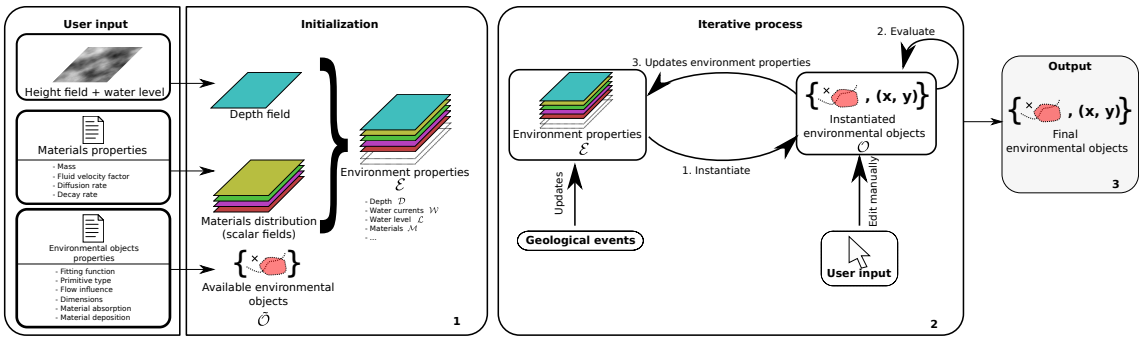

Figure 1.20: *Overview of the pipeline of the method. Three different inputs are required at initialization (left): a base terrain and a given water level directly given by the user for each generated scene, a set of properties for the environmental materials available (Section 1.3), and the properties and generation rules of environmental objects available (Section 1.3.2), established with experts. The generation process (middle) iteratively proceed in three steps: instantiating new environmental objects (Section 1.5), evaluating the fitness of each entity with respect to user modifications (Section 1.7), and updating the environment until stabilisation (Section 1.6) with user-defined geomorphological events (Section 1.7.2). When the scene suits the user, the sparse representation of environmental objects in the scene is returned and can be rendered (right).*

1.4.1 Initialization

We initialise the terrain with a height field h and a water level $\mathcal{L}$. During this chapter we will include many environmental objects depending on altitude or depth, so we will use the shorthand notations $\mathcal{H} = h - \mathcal{L}$ (signed height above water) and $\mathcal{D} = -\mathcal{H}$ (signed depth below water). The height field provides variation in altitude, which can influence the generation process of the scene.

A catalogue of available environmental objects $\widetilde{\mathcal{O}}$ is supplied by field experts; the user provides an optional target list of environmental objects $\hat{\mathcal{O}}$ used as a stop condition of the process.

Finally, different environmental materials are defined with properties: diffusivity, density, decay rate, and advection sensitivity. These parameters are chosen with field experts to reflect plausible physical behaviour.

An initial environment configuration, resulting from the initial height field h and water level $\mathcal{L}$, the environmental materials distribution $\mathcal{M}$ represented as scalar fields $\mathcal{M} : \mathbb{R}^2 \mapsto \mathbb{R}$, and water currents $\mathcal{W}$ as a vector field $\mathcal{W} : \mathbb{R}^2 \mapsto \mathbb{R}^2$, is then available for all environmental objects to evaluate growth and spawning. The environmental attributes is noted $\mathcal{E} = (\mathcal{H}, \mathcal{W}, \mathcal{L}, \mathcal{M})$ (Section 1.3.1).

The definition of environmental objects properties and environmental attributes is done with field experts, who first decide which environmental materials layers are relevant for the target biome and then anchor the corresponding parameters in the system. The generation phase starts from this environment plus an optional seed set of environmental objects.

Input: Base height field h, water level $\mathcal{L}$, user flow $\mathcal{W}_{\text{user}}$, catalogue $\widetilde{\mathcal{O}}$, target multiset $\hat{\mathcal{O}}$, material params for $\mathcal{M}$
Output: Sparse set of instantiated environmental objects

Initialisation:
$\mathcal{H} \leftarrow h - \mathcal{L}$ `// signed height`
Initialise $\mathcal{M}$ scalar fields (zeros or expert priors)
$\mathcal{W}_{\text{simulation}} \leftarrow$ terrain-aware flow from $\mathcal{H}$ $\mathcal{W} \leftarrow \mathcal{W}_{\text{user}} + \mathcal{W}_{\text{simulation}}$
$\mathcal{E} \leftarrow (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$
$\mathcal{O} \leftarrow \varnothing$
Generation loop:
while *not stop (max iters / $\hat{\mathcal{O}}$ reached / user approval)* **do**
  `// (1) Instantiate new objects (Section 1.5)`
  foreach *class* $\in \widetilde{\mathcal{O}}$ **do**
    sample position: $\mathbf{p}_s \leftarrow \arg\max_{\mathbf{p}_k}\ \omega_{\text{class}}(\mathbf{p}_k, \mathcal{E})$ `// Best position Section 1.5.1`
    if $\omega_{class}(\mathbf{p}_s, \mathcal{E}) > threshold_{class}$ **then**
      compute object's skeleton: $skeleton_o \leftarrow \Gamma_{\text{class}}(\mathcal{E})$ `// Section 1.5.2`
      add new object o to $\mathcal{O}$
  `// (2) Environment update from environmental modifiers (Section 1.6)`
  $\mathcal{H} \leftarrow$ recompute from all objects height modifiers $\mathcal{H}_o^+$ `// Section 1.6.2`
  $\mathcal{W}_{\text{simulation}} \leftarrow$ recompute from $\mathcal{H}$ `// Section 1.6.3`
  $\mathcal{W}^+_{\text{objects}} \leftarrow \sum_{o\in\mathcal{O}} \lambda_o\, \mathbf{u}_o$
  $\mathcal{W} \leftarrow \mathcal{W}_{\text{user}} + \mathcal{W}_{\text{simulation}} + \mathcal{W}^+_{\text{objects}}$
  $\mathcal{M} \leftarrow$ advection-reaction-diffusion `// Section 1.6.1`
  $\mathcal{E} \leftarrow (\mathcal{H},\ \mathcal{W},\ \mathcal{L},\ \mathcal{M})$
  apply direct user edits to skeleton `// Section 1.7.1`
  apply geomorphic events analytically on $\mathcal{E}$ `// Section 1.7.2`
  $\mathcal{O} \leftarrow$ remove unfit objects
return $\mathcal{O}$ `// sparse environmental objects`

Algorithm 1: environmental objects generation pipeline

1.4.2 Generation process

After initialization, the system enters an incremental loop with two stages:

- instantiate candidate environmental objects, then
- update the environment.

The loop terminates when user-defined criteria are met (e.g. an iteration count threshold, the target set of environmental objects $\hat{\mathcal{O}}$, or manual approval).

Figure 1.21: *While the fitness function guides the position of the environmental objects, the skeleton fitting function provides an indication of the suitability of the environmental object in its surroundings. This information is used to determine if the environmental object should be removed, but may also be used for visual purposes to indicate unhealthy corals or eroded rocks.*

Instantiation

Each iteration proposes new environmental objects at locations sampled stochastically and ranked by their generation rules (Section 1.5). Every instantiated object immediately evaluates its own viability through the user-supplied fitness function and adjusts shape and pose with the skeleton fitting function.

Environment update

After instantiation, every environmental object applies its environmental modifiers: perturbing the flow field $\mathcal{W}$ (Section 1.6.3), and deforming the coarse height field $\mathcal{H}$, then depositing or absorbing environmental materials $\mathcal{M}$ (Section 1.6) as an advection-diffusion solver relaxes environmental materials under current-driven transport and gravity until a local steady state is reached.

At any iteration, the user may edit individual environmental objects interactively (Section 1.7.1) or schedule geomorphic events, such as storms or ground subsidence, that modify environmental attributes globally (Section 1.7.2).

1.4.3 Output

The output of our system is a set of environmental objects stored as 2D skeletons in the plane. We deliberately stop at this sparse, symbolic level; meshing, texturing, and final rendering are left to downstream tools chosen by the end-user. For each object, we export its skeleton geometry, semantic class, and environment-modifier parameters, enough to drive terrain synthesis or placement of assets in a game engine or simulator. Figures in this chapter illustrate possible renderings generated with a mix of implicit surfaces and triangular meshes (Figure 1.18).

1.5 Placement of environmental objects in an environment

At each iteration of our algorithm, we aim to place new environmental objects at plausible locations. Because we do not enforce physical time continuity, our goal is to progressively enrich the scene according to user intent while maintaining ecological and geological plausibility. Since each environmental object modifies the environment when instantiated, potentially destabilising previous placements, the goal is to insert new elements where they minimally disturb the existing scene.

Our placement algorithm proceeds in two steps: first, it identifies the globally most suitable position using the fitness function ω, which depends on environmental attribute (altitude $\mathcal{H}$, water flow $\mathcal{W}$, water level $\mathcal{L}$, and environmental materials availability $\mathcal{M}$). Second, we refine the object's shape and pose using its skeleton fitting function Γ.

1.5.1 Fitness function

Our placement method take inspiration from "Darwinian fitness", refering to an organism's ability to survive and reproduce in its environment. It is a measure of how well-suited an organism is to its surroundings, and those with higher fitness are more likely to pass on their genes to the next generation. In a similar manner, the fitness function of an environmental object evaluates its suitability at a location in the terrain, extending the meaning to living features (e.g., forests and corals) and non-living elements (e.g., reefs and lagoons).

The fitness function $\omega(\mathcal{E}) : \mathbb{R}^2 \mapsto \mathbb{R}$ quantitatively evaluates how well a given environment location satisfies the requirements of environmental objects. In other words, ω returns a numeric score representing the suitability of local environmental conditions for each object. This function considers various environmental variables such as altitude, the presence of specific materials, water current strength, and their respective gradients. Defining separately ω for each environmental object class allows the algorithm to be tailored to the unique needs and tolerances of different object types. High ω values indicate favourable sites for the object, while low values correspond to unsuitable or inhospitable locations.

Different types of environmental objects require different criteria for their fitness evaluation. For example, a river might prioritise lower altitude and proximity to water currents, while a forest might prioritise higher altitude and specific material availability. The flexibility of the fitness function allows it to be customised for each environmental object, ensuring that the generated terrain remains coherent and realistic.

In practice, to find suitable locations for an object, we evaluate its ω across the environment's spatial domain by randomly sampling ω values over the terrain, effectively producing a fitness map for that object class, where peaks indicate highly suitable locations. Each object class thus yields its own fitness map.

The value of the fitness function can be used later on to modify the geometric representation or the appearance of the environmental object to appear more eroded (Figure 1.21).

1.5.2 Skeleton fitting function

A candidate object is first seeded at the position $\mathbf{p}_s = \arg\max_{\mathbf{p}} \omega(\mathbf{p})$, i.e. the global maximum of the fitness map. This seed acts as the initial guess for the subsequent shape optimisation for the skeleton fitting function $\Gamma(\mathcal{E}) : \mathbb{R}^2 \mapsto \mathbb{R}$ described below. Moreover, this two-step optimisation allows a coarse approximation of the global maximum of the fitness function with few samples, reducing its computational demand, as the skeleton fitting function optimisation is focused on a single seed.

Point-based skeleton

For objects whose skeleton reduces to a single point (boulders, individual corals, etc.), we refine the seed by hill-climbing on a (possibly different) fitting field Γ. In most cases we simply set $\Gamma = \omega$, but the user may supply a sharper or multi-objective field when needed. Optimisation proceeds by gradient ascent on $\nabla\Gamma$ with a small step size and terminates when $|\nabla\Gamma| < \varepsilon$.

Curve-based skeleton

The skeleton of a curve-based environmental object is determined by the shape that fits best given the environment it is added to. Using an Active-Contour ("snake") formulation [KWT88], we seek a parametric curve $C(s) : [0,1] \mapsto \mathbb{R}^2$ that minimises the composite energy:

$$E(C) = E_{\text{internal}} + E_{\text{external}} + E_{\text{shape}} + E_{\text{gradient}} \tag{1.27}$$

with

$$E_{\text{internal}} = \alpha_{\text{i}} \left(\int_C \left\| C'(s) \right\|^2 ds + \int_C \left\| C''(s) \right\|^2 ds \right) \tag{1.28}$$

$$E_{\text{external}} = -\alpha_{\text{e}} \int_C \Gamma(C(s))\, ds \tag{1.29}$$

$$E_{\text{shape}} = \alpha_{\text{s}} \left(L - \int_C ds \right)^2 \tag{1.30}$$

$$E_{\text{gradient}} = \alpha_{\text{g}} \int_C -\langle C'(s), \nabla\Gamma\left(C(s)\right)\rangle^2 ds \tag{1.31}$$

Here $\alpha_\bullet$ are user-tunable weights and L is an optional target length. The minus sign in E_{external} converts the integrand into an attraction term (high Γ lowers the energy).

In this configuration, E_{internal} induces a smooth continuity of the curve by reducing the spacing of each point while reducing the curvature. Another energy, E_{external}, integrates the skeleton fitting function over the curve, often seen as an attractor of the points, that tries to descend the gradient to find local minima. At the same time, E_{shape} applies constraints on the curve shape, which, in this case, is to target a specific length L. As such, the curve searches for an optimised shape given constraints in E_{shape} for minimising E_{external}. We introduce a new term E_{gradient} in the energy computation that pushes the points of the curve in the direction of the slope of the skeleton fitting function, also providing an orientation for the curve.

If a steep coast can be found where the terrain slope is significant near the water level, we can define $\Gamma = |\mathcal{H} + 1| \,/\, \|\Delta\mathcal{H}\|$, but no orientation is needed, thus we set $\alpha_{\text{g}} = 0$. In this case, the curve will follow a path at the water level and spread its extremities over areas with a steep slope. A river may also be symbolised as a parametric curve, but we need to add information about the direction and magnitude of the slope. As such, we can use $\Gamma = \mathcal{H}$, which forces the direction of the curve to fit with the terrain slope. The introduction of the gradient component also provides an orientation to shapes.

Internal energy

$$E_{\text{internal}} = \alpha_{\text{i}} \left(\int_C \left\| C'(s) \right\|^2 ds + \int_C \left\| C''(s) \right\|^2 ds \right)$$

The internal energy introduced in [KWT88] is composed of two components imposing penalties on the local properties of the points of the curve. The first derivative forces the points along the curve to be evenly spaced by minimising the curve tension, while the second derivative restricts it from forming sharp corners, increasing the rigidity of the curve. As our aim is to represent natural elements, we rarely find sharp features and thus we use the original definition from the Snake formulation.

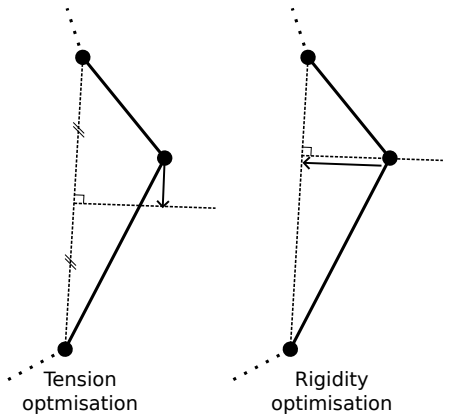

Figure 1.22: *Internal energy minimisation for one vertex is done by reducing the difference of length with the next and with the previous vertices (tension), while reducing the curvature at this point (rigidity).*

In Figure 1.22 we see the two different components of this energy minimisation: averaging the vector formed by the previous segment with the vector formed by the next segment, we can move the vertex to a position for which the two new segments are equal in length. On the other hand, translating the vertex towards its projection minimises the curvature at this vertex.

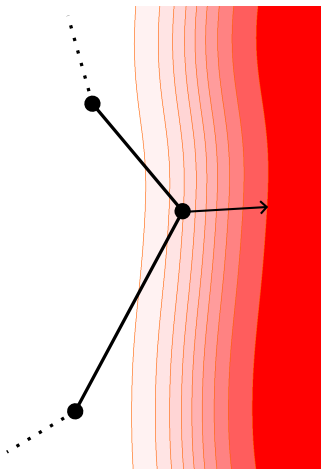

Figure 1.23: *The central vertex can optimise its external energy by translating in the same direction as the function gradient $\nabla\omega$.*

External energy

$$E_{\text{external}} = -\alpha_{\text{e}} \int_C \Gamma(C(s))\, ds$$

The external energy is also present in the original work. Using an external scalar field, each point of the curve is forced to follow the steepest slope of the field. The external field can be seen as an attractor for the curve (Figure 1.23).

The energy is minimised when the vertex is positioned at the global maximum of the scalar field.

Shape energy

$$E_{\text{shape}} = \alpha_{\text{s}} \left(L - \int_C \|C'\|\, ds \right)^2$$

We introduce the shape energy, an energy defined on the whole curve to apply constraints on its final shape. As natural features often have a given dimension, we propose to add a constraint on the length of the skeleton (Figure 1.24).

The original Snake algorithm is biased, forcing the curve to shrink as the internal forces attract the vertices towards their neighbours. The introduction of a target length cancels this effect.

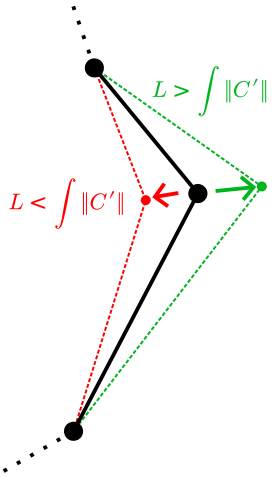

Figure 1.24: *The shape energy is minimised when the arc length of the curve C is equal to a parameter L, obtainable by moving towards or in opposition to its neighbours.*

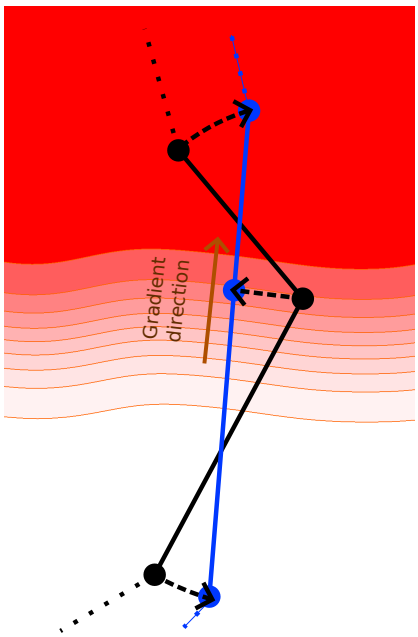

Figure 1.25: *The gradient energy is minimised when the curve crosses perpendicularly the isolines of the function. Black: the initial curve, Blue: a possible curve crossing the gradient obtained by displacing the vertices.*

Gradient energy

$$E_{\text{gradient}} = \alpha_g \int_C -\langle C'(s), \nabla\Gamma\left(C(s)\right)\rangle^2 ds$$

In our application, having information about the orientation of environmental objects may be essential. For this purpose, we introduced a gradient energy component in the formulation. The equation imposes that the direction of the curve at any point should be directed towards the gradient of the scalar field (Figure 1.25). As the external energy pushes points towards the lowest point of the scalar field, the gradient field restricts the gradient descent for the global curve into a specific way, which may feel more natural. Figure 1.26 presents the effect of gradient energy with three identical initial curves: a strong gradient energy factor (top) drives the curve uphill while a negative coefficient (bottom) forces the curve to avoid steep slopes.

During the optimisation process, the gradient of the scalar field is already evaluated by the external energy optimisation, so the addition of the gradient component is almost free.

Region-based skeleton

When the skeleton is a closed boundary (island, forest patch, lagoon), we adapt Chan-Vese segmentation [CV01]: the gradient term vanishes and the external energy integrates over the enclosed domain.

The resulting energy E to minimise for a closed region whose borders are defined by the curve C is then expressed as $E(C) = E_{\text{internal}} + E_{\text{external}} + E_{\text{shape}}$.

The internal energy is expressed identically as for curve-based environmental objects. The external energy, however, is modified to take into account the interior of the region instead of only the borders. We use the idea of the Chan-Vese algorithm, separating the energy value for the inside Ω and the borders C of the region [CV01, Get12]:

$$E_{\text{external}} = \lambda_1 \int_\Omega \Gamma(s)\, ds + \lambda_2 \int_C \Gamma(C(s))\, ds \tag{1.32}$$

λ_1 emphasises interior homogeneity (e.g. constant humidity for a forest), while λ_2 attracts the boundary to strong gradients (reef rim at the lagoon edge).

Finally, the shape constraint energy E_{shape} can target an area A, for example:

$$E_{\text{shape}} = \left(A - \int_\Omega 1\, ds\right)^2 \tag{1.33}$$

In our implementation, each environmental object's skeleton is a connected component, as we define the boundaries by the connected curve C, but it can be convex or concave. An infinite penalty is added for the curve's self-intersection.

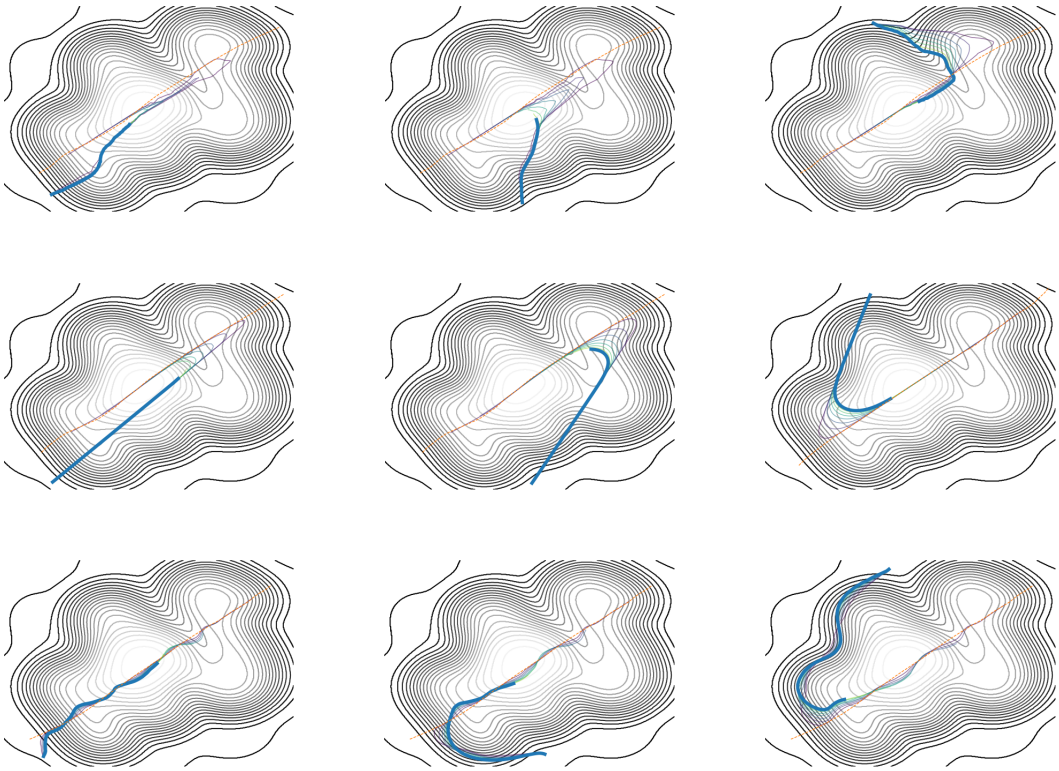

Figure 1.26: *Three examples of Snake optimisation from an initial curve (orange line). One vertex of the curve is fixed. Top has a strong gradient energy coefficient $\alpha_g > 0$, middle has no gradient energy coefficient $\alpha_g = 0$, and bottom has negative gradient energy coefficent $\alpha_g < 0$.*

At every global iteration, each object re-evaluates $\omega(C, \mathcal{E}) < T_{\text{fit}}$ with T_{fit} a threshold for which the object is removed, fixed for each object class; otherwise we take advantage of the iterative nature of the Snake algorithm and improve the shape of the skeleton by optimising their skeleton fitting function again. If the user fixed the position of vertices, we simply set $\nabla E = \vec{0}$ at the given vertices.

1.6 Environmental modifiers

Every instantiated environmental object contributes a local environmental modifier tuple $\mathcal{E}_o^+ = (\mathcal{H}_o^+,\ \mathcal{W}_o^+,\ \mathcal{M}_o^+)$, affecting elevation, flow, and resource fields respectively. At the end of each iteration we assemble the provisional environment:

$$\mathcal{E}^\star = \mathcal{E} + \sum_{o \in \mathcal{O}} \mathcal{E}_o^+, \tag{1.34}$$

before advection-diffusion relaxation and viability tests.

1.6.1 Environmental material modifiers

For every material layer $m \in \mathcal{M}$ (sand, limestone, nutrients, ...), we store a scalar availability field $c_m(\mathbf{p}, t)$ defined on the plane.

Each material has four transport parameters $(\rho_m, \nu_m, D_m, \mu_m)$: mass density ρ_m, flow-coupling factor ν_m, molecular diffusivity D_m, and first-order decay rate μ_m.

In our method, we consider that living or abiotic objects deposits (positive source) or absorbs (negative

source) material, thereby mediating indirect interaction with its neighbours. For object o deposing or absorbing the material m, we denote the net source term $S_{m,o}(\mathbf{p})$ at a given point $\mathbf{p}$:

$$S_{m,o}(\mathbf{p}) = (D_{m,o} - A_{m,o})\, G(d(\mathbf{p})), \tag{1.35}$$

where G is a Gaussian kernel centred on the skeleton, $d(\mathbf{p})$ is the shortest distance of $\mathbf{p}$ to the skeleton, and $D_{m,o}$ and $A_{m,o}$ are the deposition and absorption rates.

Material advection combines gravity-driven downslope drift and water-borne transport. We approximate the velocity field:

$$\Phi(\mathbf{p}, t) = \rho_m \nabla \mathcal{H}(\mathbf{p}, t) + \nu_m \mathcal{W}(\mathbf{p}, t) \tag{1.36}$$

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

$$\frac{\partial c_m}{\partial t} + \Phi \cdot \nabla c_m = D_m \nabla^2 c_m - \mu_m c_m + \sum_{o \in \mathcal{O}} S_{m,o} \tag{1.37}$$

We solve Equation (1.37) numerically using an Euler integration:

$$\begin{aligned} c_m(\mathbf{p}, t + dt) = \max\big(0, c_m(\mathbf{p}, t) \\ + dt\big(D_m \nabla^2 c_m(\mathbf{p}, t) - \mu_m c_m(\mathbf{p}, t) - \Phi(\mathbf{p}, t) \cdot \nabla c_m(\mathbf{p}, t) + \sum_{o \in \mathcal{O}} S_{m,o}(\mathbf{p})\big)\big) \end{aligned} \tag{1.38}$$

The linear decay μ_m prevents indefinite mass build-up and drives the system toward a bounded equilibrium, following the practice of ecological field models [SRSS12]. Figure 1.29 shows the stability of a diffusion simulation over 200s with and without decay, with and without a flow field. A steady state is visible as the MAE (Mean Average Error, i.e. discrete approximation of L^1-norm) from one time step to another drops at $t = 50s$ in the case of a small decay, while the MAE blows up in the case without decay. The pseudo code presented in Algorithm 2 illustrate conciseness of the implementation.

An example of deposition of sand from an environmental object is provided in Figure 1.27, which is mainly affected by the water flow generated by the islands.

The effects of varying the different material properties of sand spread from rocks in a canyon with high water currents are presented in Figure 1.44. The special case of shade and coral calcareous deposition (reef-building material) only relies on diffusivity and decay to conserve a static modification on the surface, without the need for another framework for this unique type of environmental change (Figures 1.28 and 1.45).

Input: $c(\cdot)$, $\mathcal{W}$, $\mathcal{H}$, params (ρ, ν, D, μ), sources and sinks $S(\cdot)$, step size dt
Output: Steady state environmental material

$\Phi(\mathbf{p}) \leftarrow \rho\nabla\mathcal{H}(\mathbf{p}) + \nu\mathcal{W}(\mathbf{p})$ for all $\mathbf{p}$
$\hat{c} \leftarrow c$
do
  $c \leftarrow \hat{c}$
  foreach p do
    $\hat{c}(\mathbf{p}) \leftarrow \max\{0,\ c(\mathbf{p}) + dt\,[D\nabla^2 c - \mu c - \Phi \cdot \nabla c + S(\mathbf{p})]\}$
while $\frac{1}{|\Omega|}\sum_{\mathbf{p}} |\hat{c}(\mathbf{p}) - c(\mathbf{p})| > \tau$ // MAE < τ

return $\hat{c}$

Algorithm 2: Advection-diffusion-reaction for a single environmental material

Figure 1.27: *(Left) A single island deposits sand over its region, which is displaced by the water flow. (Centre) With two adjacent islands, we see an accumulation of sand (green) in the valley they form, as we would see in nature. (Right) This phenomenon is directly due to the cancelling of their respective water flow influence, producing a region with low currents, allowing sand to settle, without any need for complex computation.*

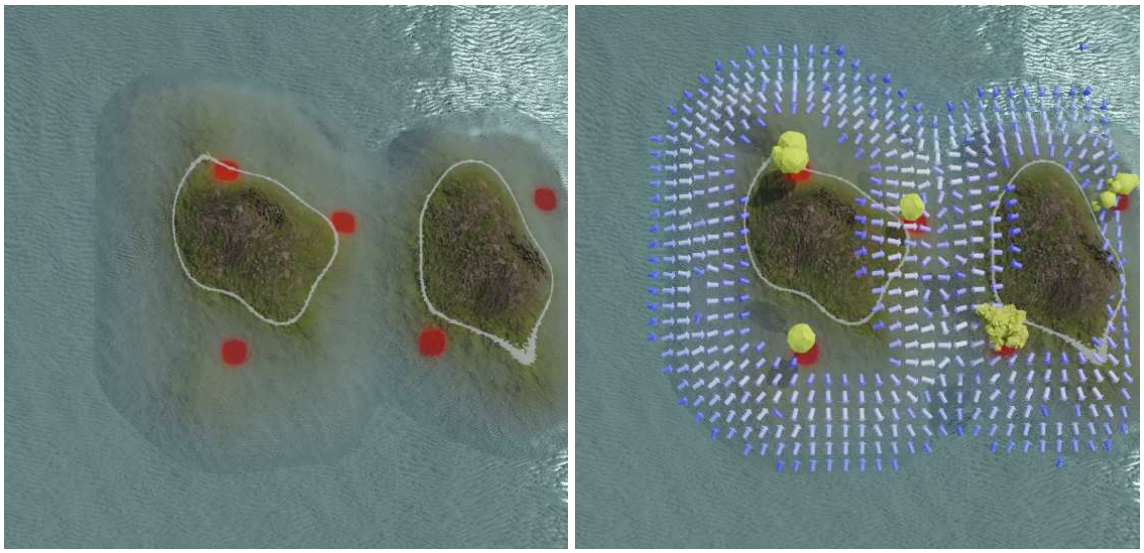

Figure 1.28: *While we do consider shade (red) as being a material in the same manner as sand or pebbles, setting the influence of water currents and slope to null makes it independent of the environment, staying at the position of the environmental object that generates it.*

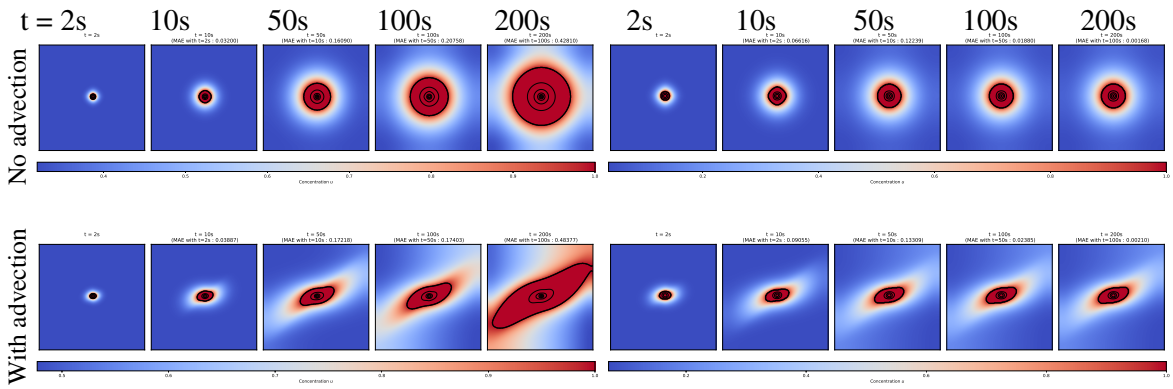

Figure 1.29: *With the addition of a decay term in the advection-diffusion-reaction equation, the spread of the environmental materials tends towards stability after a few iterations (right), whereas the total mass would otherwise increase indefinitely (left). Concentration values are clamped to 1 for visualization. This behavior also holds in the presence of a flow field (bottom), here with bounded diagonal vector field oriented along $x = y$. Moreover, decay reduces boundary-related instabilities; bottom left uses a zero-gradient boundary condition, which leads to a blow-up from the borders.*

Approximation of distances from environmental materials

The use of environmental materials provides an efficient, on-demand approximation of the distance to the closest instance of an object type $\mathcal{O}$, with computational complexity linear in the number of objects in the scene. For each object type (e.g., coral, rock, reef, island, ...), we maintain a separate environmental material field $\mathcal{M}_{\text{object}}$ that evolves under diffusion-reaction only (advection disabled by setting $v = 0$ and $\rho = 0$). The object's skeleton acts as a Dirichlet boundary: for every skeleton point $\mathbf{q}$, the concentration is fixed to a constant value $c_{\text{object}}(\mathbf{q}) = C_s$ with $C_s > 0$.

Let Γ denote the skeleton of all objects of a given type and $d = \|\mathbf{p} - \mathbf{q}\|$ with $\mathbf{q} \in \Gamma$ the closest skeleton point to $\mathbf{p}$. The diffusion-reaction field is updated using forward Euler steps of Equation (1.38):

$$c_{\text{object}}(\mathbf{x}, t + \Delta t) = c_{\text{object}}(\mathbf{x}, t) + \Delta t \left(D\nabla^2 c_{\text{object}}(\mathbf{x}, t) - \mu\, c_{\text{object}}(\mathbf{x}, t) \right) \tag{1.39}$$

$$c_{\text{object}}(\mathbf{q}, t) = C_s, \quad \mathbf{q} \in \Gamma. \tag{1.40}$$

To approximate the distance from a query point $\mathbf{p}_0$ to the nearest object, we perform gradient ascent on the concentration field: starting at $\mathbf{p}_0$, we follow the discrete gradient ∇c_{object} until reaching a location $\mathbf{p}$ where $c_{\text{object}}(\mathbf{p}) = C_s$. The distance estimate is then

$$d \approx \|\mathbf{p} - \mathbf{p}_0\|. \tag{1.41}$$

This setup allows direct queries for the distance to any specific type without additional preprocessing on the complete terrain, while alternative strategies such as using Euclidean distance transforms could also be applied to avoid the gradient ascent with a single lookup at the cost of heavier computation. The complete distance-query procedure is given in Algorithm 3.

1.6.2 Height modifiers

Terrain elevation is updated by blending the coarse height function functions of all objects that overlap a query point $\mathbf{p}$. We simplify the shape of environmental objects to analytical functions such as a mountain to a cone, a reef as a curved cylinder, or a coral boulder as a sphere, enabling fast, on-demand, and multi-scale computation of the altitude at any point. Because we work with a DEM, overhangs are ignored.

Input: Screened field $c_{\text{object}}(\cdot)$ with Dirichlet $c_{\text{object}} = C_s$ on skeleton Γ, query point $\mathbf{p}_0$
Output: Approximated distance $\hat{d}$

$\mathbf{p} \leftarrow \mathbf{p}_0$
while $c_{object}(\mathbf{p}) < C_s - \varepsilon$ **do**
  $\mathbf{p} \leftarrow \mathbf{p} + \eta \nabla c_{\text{object}}(\mathbf{p})$ `// gradient ascent`
return $\hat{d} \leftarrow \|\mathbf{p} - \mathbf{p}_0\|$

Algorithm 3: Distance query to nearest object of a given type

To preserve ordering constraints, we partition objects into three mutually exclusive groups:

- $\mathcal{G}$: grounded data (e.g. islands) defined from the absolute zero level,
- $\mathcal{A}$: altitude-relative shapes (depth-constrained corals, tidal flats),
- $\mathcal{S}$: fine surface details that sit on the current terrain, which can be seen as bumps or carves from an aerial view.

Groups $\mathcal{G}$ and $\mathcal{A}$ are combined with our smooth maximum/minimum operators as presented in the previous chapter to avoid C^0 discontinuities, while surface details in $\mathcal{S}$ are simply summed:

$$\mathcal{H}^+_{\mathcal{G}}(\mathbf{p}) = \text{smax}_{o\in\mathcal{G}}\, \mathcal{H}^+_o(\mathbf{p}) \tag{1.46}$$
$$\mathcal{H}^+_{\mathcal{A}}(\mathbf{p}) = \text{smin}_{o\in\mathcal{A}}\, \mathcal{H}^+_o(\mathbf{p}) \tag{1.47}$$
$$\mathcal{H}^+_{\mathcal{S}}(\mathbf{p}) = \sum_{o\in\mathcal{S}} \mathcal{H}^+_o(\mathbf{p}) \tag{1.48}$$

The final coarse elevation and its analytic gradient (for slope queries) are then given by the smooth maximum between $\mathcal{H}^+_{\mathcal{G}}$ and $\mathcal{H}^+_{\mathcal{A}}$, and the addition of $\mathcal{H}^+_{\mathcal{S}}$ (Figure 1.30):

$$\mathcal{H}^\star(\mathbf{p}) = \text{smax}\left(\mathcal{H}^+_{\mathcal{G}},\ \mathcal{H}^+_{\mathcal{A}}\right) + \mathcal{H}^+_{\mathcal{S}} \tag{1.49}$$

The gradient $\nabla\mathcal{H}^\star$ is used by fitness rules that depend on slope, as well as the analytic water flow simulation $\mathcal{W}_{\text{simulation}}$.

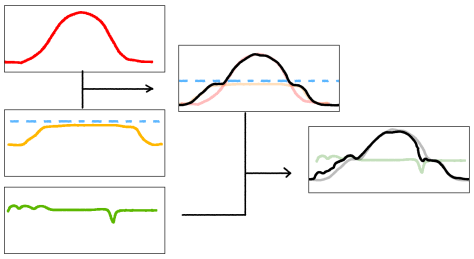

Figure 1.30: *The final coarse height function is a combination of the three groups of environmental objects: grounded data $\mathcal{G}$ (red) and altitude-relative shapes $\mathcal{A}$ (orange) are aggregated by the* smax *operator, then the surface elements $\mathcal{S}$ (green) are added.*

1.6.3 Influence on water currents

We define the global water current as a vector field composed of three components:

$$\mathcal{W}(\mathbf{p}) = \mathcal{W}_{\text{user}}(\mathbf{p}) + \mathcal{W}_{\text{simulation}}(\mathbf{p}) + \mathcal{W}^+_{\text{objects}}(\mathbf{p}) \tag{1.50}$$

Here, $\mathcal{W}_{\text{user}}(\mathbf{p})$ is a user-defined vector field that provides direct control over the flow direction and intensity using stroke paths, as presented in the previous chapter. The term $\mathcal{W}_{\text{simulation}}(\mathbf{p})$ is an analytically defined component inspired by the terrain-aware wind simulation approach introduced in [PPG*19], which we use for large water body fluid simulation. Finally, $\mathcal{W}^+_{\text{objects}}(\mathbf{p})$ represents the deformation introduced by environment objects.

The simulated component $\mathcal{W}_{\text{simulation}}$ captures how water flow is diverted and shaped by the underlying terrain. It starts from a uniform flow direction a and introduces warping based on the terrain gradient evaluated at multiple spatial scales. The resulting vector field is expressed as:

$$\mathcal{W}_{\text{simulation}}(\mathbf{p}) = \sum_{i=0}^{n} c_i \Phi_i \cdot v \tag{1.51}$$

In this formulation, v is a base velocity vector adjusted according to the terrain elevation:

$$v = a\,(1 + k_w\,|\mathcal{H}(\mathbf{p})|) \tag{1.52}$$

where a is the global flow direction and velocity, $|\mathcal{H}(\mathbf{p})|$ is the vertical distance from the terrain to the water level at point $\mathbf{p}$, and k_w is a scaling factor simulating the Venturi effect, which increases velocity in regions of compression.

Each $\Phi_i \cdot v$ term represents the warping of v at scale i, defined as

$$\Phi_i \cdot v = (1-\alpha)v + \alpha k_i \nabla\widetilde{\mathcal{H}_i}^{\perp}(\mathbf{p}) \quad \text{with} \quad \alpha = \left\| \nabla\widetilde{\mathcal{H}_i}(\mathbf{p}) \right\| \tag{1.53}$$

Here, $\widetilde{\mathcal{H}_i}$ denotes the smoothed terrain elevation at scale i, and $\nabla\widetilde{\mathcal{H}_i}^{\perp}(\mathbf{p})$ is the vector orthogonal to the terrain gradient, directing flow along paths that curve around elevation changes. The blending coefficient α controls the interpolation between the original flow and the deviated flow, based on the steepness of the terrain. The deviation coefficient k_i determines the strength of the flow deflection at each scale.

As proposed in the original method, we employ two spatial scales ($n = 2$), using Gaussian smoothing kernels with radii of 200m and 50m. The associated weights are set to $c_0 = 0.8$ and $c_1 = 0.2$, with deviation coefficients $k_0 = 30$ and $k_1 = 5$ respectively. This multi-scale formulation enables the simulation to account for both large topographic features and finer terrain details, resulting in a more nuanced and plausible flow field.

The terrain-based simulation component $\mathcal{W}_{\text{simulation}}$ is designed to model large-scale water current behaviour by leveraging an analytical approximation of flow deviation due to terrain. This solver is inspired by the work of [PPG*19], originally developed for wind field simulation, and has been adapted here to capture how elevation changes influence the direction and intensity of surface water flow.

The proposed formulation offers a terrain-aware approach to modelling water currents, recognising that topography is the dominant influence on flow direction in natural environments, particularly in the absence of dynamic fluid interactions. By analysing terrain gradients at multiple spatial scales, the solver captures both broad elevation patterns and fine landform features, resulting in a plausible, nuanced flow field that naturally conforms to hills, valleys, and river paths without requiring a full physical simulation.

Unlike physics-based solvers (discussed further in **??**), the analytical formulation of $\mathcal{W}_{\text{simulation}}$ exposes intuitive parameters such as flow direction, elevation sensitivity, and gradient deviation that can be directly adjusted by the user, making it especially suitable for terrain generation pipelines where user intent or artistic direction must be encoded. The solver is also computationally efficient and supports on-demand evaluation, avoiding the need for dense precomputed data and aligning well with the requirements of real-time systems, streaming terrains, and interactive editors.

While the model assumes steady-state flow and does not simulate dynamic phenomena such as rainfall, flooding, or erosion, these effects can be layered atop the static flow if needed, preserving a lightweight design that still conveys physically plausible behaviour. Finally, our system is purposefully modular: $\mathcal{W}_{\text{simulation}}$ addresses large-scale flow patterns, while small-scale variations (turbulence, vortices, and flow deviations caused by small-scale features) are handled separately by the $\mathcal{W}^{+}_{\text{objects}}$ term, allowing each component to specialise in its relevant spatial scale.

Vector field deformation through Kelvinlets

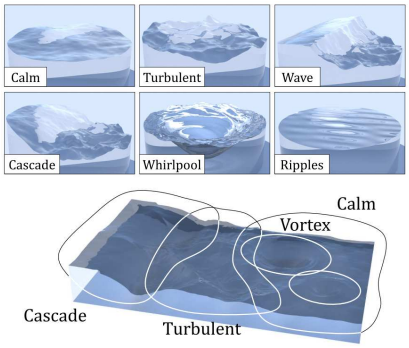

Figure 1.31: *[PDG*19] uses a sparse representation of a river surface to include details at the water surface that can be aggregated in a construction tree.*

$\mathcal{W}^{+}_{\text{objects}}$ is a deformation field defined as the accumulation of flow primitives. The use of analytical primitives to represent localised variations within large-scale scenes has been explored in various contexts through the edition of wind fields ([WH91], Figure 1.32) and authoring of rivers' water surface geometry ([PDG*19], Figure 1.31). In the latter, flow primitives are primarily used to generate the geometry of the water surface, but the same primitives can also be reused to produce a flow field, enabling effects such as floating debris or leaves drifting along the surface. Because geometry is sensitive to overlapping influences, the authors organise primitives into a blending tree, where merge and replace operations control how individual contributions interact.

On the other hand, [WH91] showed that simply summing flow primitives generates velocity fields that approximate Navier-Stokes dynamics (Figure 1.32, bottom). Because their method affects motion rather than visible geometry, this additive approach offers a good trade-off between plausibility, user control, and computation time. In our method, we adopt a similar additive approach to combine Kelvinlet-based primitives. Our goal is to define a deformation field, and as such, summation offers an effective and intuitive way to accumulate the influence of multiple localised deformations at various scales, without requiring additional structure to resolve overlaps.

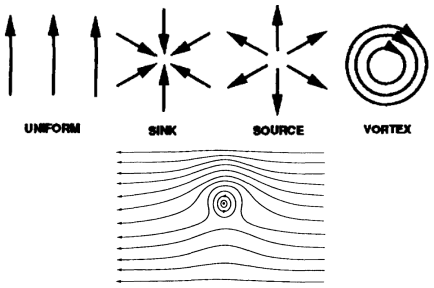

Figure 1.32: *[WH91] propose to describe the flow field of the environment by a sum of primitives, resulting in a simulation of wind field controllable by the user at very small memory cost.*

Kelvinlets were introduced to computer graphics as an efficient means of producing physically plausible and interactive deformations [GJ17]. In our context, they provide a smooth and compact representation of flow alterations, ideal for integrating environmental features into vector field modelling.

Kelvinlets are based on the Kelvin solution, which is the Green's function of the Navier-Cauchy equations of linear elasticity. It describes the displacement $\mathbf{u}(\mathbf{p})$ at a point $\mathbf{p} \in \mathbb{R}^3$ caused by a force applied at $\mathbf{q}$, in an isotropic, homogeneous elastic material:

$$\mu\nabla^2\mathbf{u} + (\lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) + \delta(\mathbf{p}-\mathbf{q})\mathbf{F} = 0 \tag{1.54}$$

where λ and μ are the Lamé parameters, with μ also known as the shear modulus, and $\mathbf{F}$ is the force vector applied at a single point $\mathbf{q}$ via the Dirac delta function $\delta(\mathbf{p}-\mathbf{q})$.

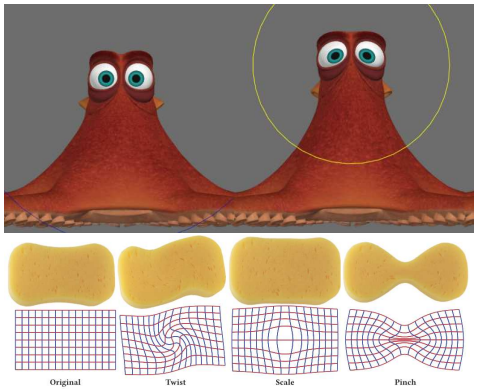

Figure 1.33: *[GJ17] presents four types of deformations using Kelvinlets, resulting in organic pinch-like interaction with the original field. Top: a grab on a ©Disney/Pixar character. Bottom, from left to right: a twist, a scale, and a pinch operation.*

To regularise the singularity at $\mathbf{p} = \mathbf{q}$, [GJ17] introduced a smoothed form known as the regularised Kelvinlets, which allow deformation effects to fall off smoothly with distance, and prevent numerical instabilities. Given a centre $\mathbf{q}$, an evaluation point $\mathbf{p}$, and a regularisation ε, we define $\mathbf{r} = \mathbf{p} - \mathbf{q}$ and the regularised distance $r_\varepsilon = \sqrt{\|\mathbf{r}\|^2 + \varepsilon^2}$.

We use the scale and grab formulations of the regularised Kelvinlets brushes (Figure 1.33), denoted as $s_\varepsilon(\mathbf{r})$ and $g_\varepsilon(\mathbf{r})$ respectively, to simulate obstruction and diversion, and define them as

$$s_\varepsilon(\mathbf{r}) = (2b - a)\left(\frac{1}{r_\varepsilon^3} + \frac{1}{2r_\varepsilon^5}\right)(s\mathbf{r})$$

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

with $a = \frac{1}{4\pi\mu}$ and $b = \frac{a}{4(1-v)}$, provided μ is a shear modulus and v a Poisson ratio specified for each Kelvinlet, s a scaling factor, and $\mathbf{F}$ the force vector of the grab operation. These values are tunable per object to simulate different material or flow resistance behaviours.

Deformations defined on curves use $\mathbf{q} = \mathbf{p}_C^*$ with $\mathbf{p}_C^*$ the closest point on the curve from the point $\mathbf{p}$ and $\mathbf{F} = C'(\mathbf{p})$. We can then define $\mathbf{u}_o(\mathbf{p}) = s_\varepsilon(\mathbf{q} - \mathbf{p}) + g_\varepsilon(\mathbf{p} - \mathbf{q})$.

Finally, we retrieve the velocity field from the objects, with a per-object coefficient λ_o for the strength of its flow effects (due to its size, porosity, surface roughness, etc.):

$$\mathcal{W}^+_{\text{objects}}(\mathbf{p}) = \sum_{o \in \mathcal{O}} \lambda_o \mathbf{u}_o(\mathbf{p}) \tag{1.55}$$

The use of regularised Kelvinlets is well-suited to our model. They offer a compact representation thanks to their closed-form definition, which allows each deformation to be described analytically without the need for precomputed data or large simulation grids. They maintain continuity, ensuring that the resulting vector field is smooth and free of visual or numerical artefacts, which is crucial when blending influences from multiple environment objects. Their behaviour is physically plausible, as they are derived from fundamental solutions to elasticity equations, and can convincingly simulate natural flow behaviours such as deflection around obstacles or local flow obstruction. They are also computationally efficient, since their evaluation is lightweight and independent of the size or complexity of the terrain, making them particularly suitable for integration into large-scale procedural generation pipelines and real-time interactive applications.

Figure 1.34 presents a rock acting as an obstacle to the incoming water flow, which guides the flow, and more importantly the environmental materials, in a plausible direction on the side of the obstacle, with the introduction of artificial vortices at its back. A curve-based water influence is presented in Figure 1.42 and Figure 1.43, which is a unique "grab Kelvinlet" in the direction of the curve skeleton of the canyon, and thus inserting new environmental objects downstream, where the environmental materials are transported.

Unfortunately, the Kelvinlet paradigm is not applicable to region-based environmental objects. In this case, the water influence is given using the vector towards the closest point on the outline, scaled by the distance between the evaluation point and the curve, as presented in Figure 1.35. Note that the vector field produced by the islands is directly computed by the distance field of its region's boundaries and is independent of the generated geometry, explaining the discrepancy between the emerged surface and the resulting flow field in Figure 1.35.

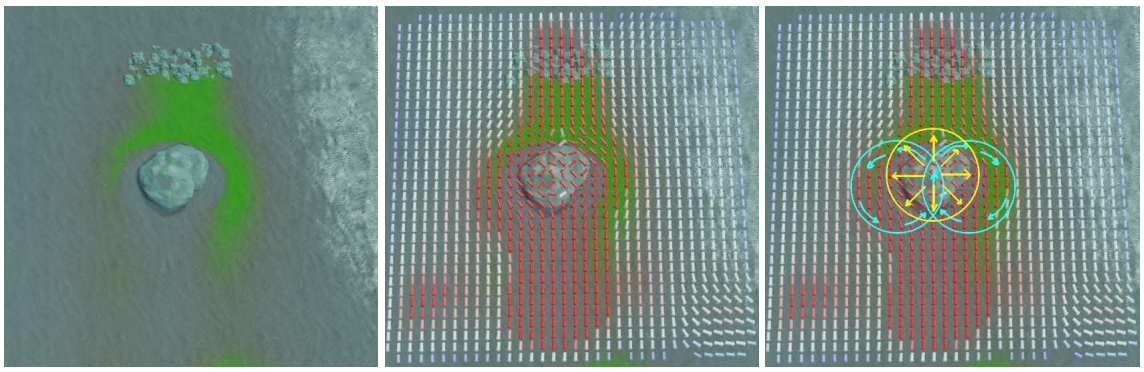

Figure 1.34: *A strong current is affected by a rock, visible by the deviation of sand (green) produced by smaller rocks. The flow is directly computed by the composition of three Kelvinlet objects assigned to the rock: a "scale Kelvinlet" (yellow) deflects the materials as an obstacle would; and two "twist Kelvinlets" (cyan) approximate the vortices of the flow, pushing the sand back behind the rock.*

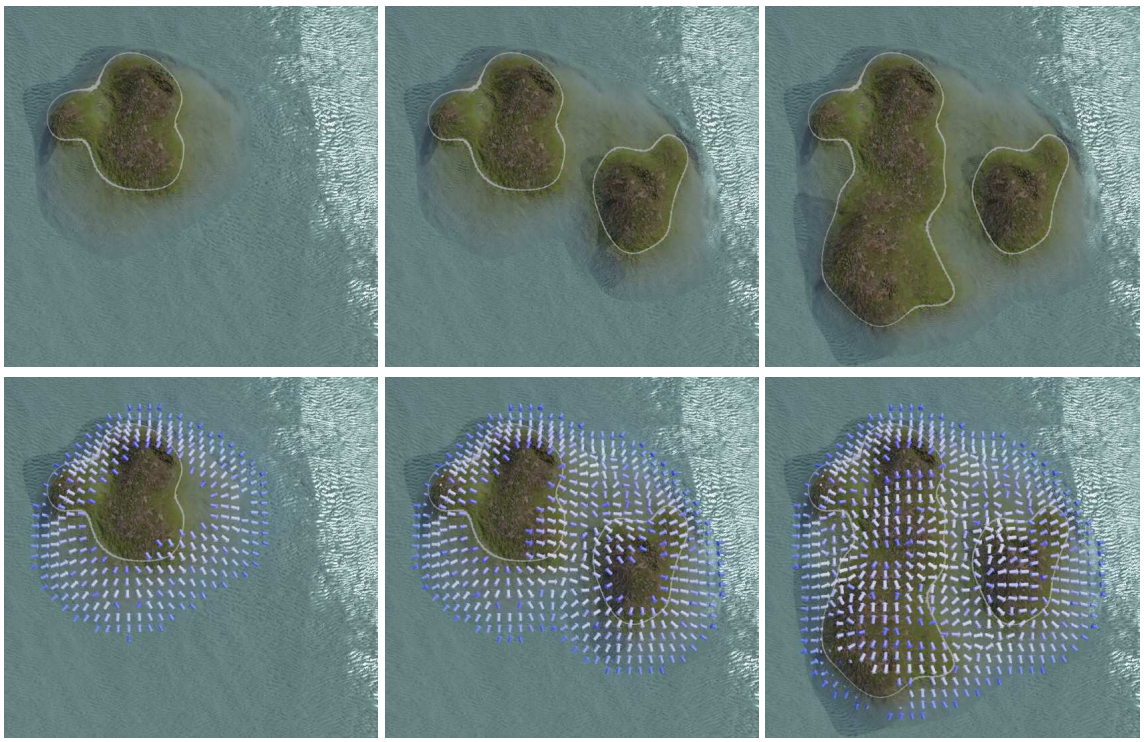

Figure 1.35: *Each environmental object has an influence on the water currents. Islands, defined with an area, add a force towards themselves (waves crashing on coasts) and simultaneously a force towards the ocean that acts as an obstacle flow. The resulting vector field of the scene is the sum of each environmental object's influence.*

1.7 User interactions

The user can guide the generation process. The use of simple shapes as environmental objects facilitates the edition of the simulation, as we can interactively add, remove or modify environmental objects, or focus the generation process in a restricted area. Interaction with the environmental attributes is also provided as geomorphic events, that the user can invoke during the simulation. While the direct interactions on the environmental objects are instantaneous, the geomorphic events are active over a given duration.

Figure 1.36: *Starting from a coral colony developed around a canyon (left), the user edits the shape of the canyon, resulting in a different configuration of the scene, killing the corals that end up too deep in the water (centre) and the development and growth of new corals at the previous location of the canyon (right).*

1.7.1 Direct interactions with the environmental objects

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

Translating an environmental object is trivial, it simply requires evaluating the fitness function of the environmental objects at a translated position to verify that the environment is still suitable for its survival.

The deformation of environmental objects can be applied on curve and region environmental objects by displacing directly the control points of the skeleton (Figure 1.36 deforming a canyon, and Figure 1.37 editing an island). Various methods are suitable for displacing a polygon, such as the use of Kelvinlets, cage deformation, displacement maps, or Gaussian influence. In the scope of this chapter, we will not cover the multiple aspects of curve and polygon deformation. However, as the closed or open geometry shape of the skeleton changes, it needs to run again the skeleton fitting function optimisation through our Snake algorithm (Section 1.5).

Locking the vertices from the deformed skeleton edited manually by the user can be achieved by cancelling the energy $E(C)_v$ from the Snake algorithm for this specific vertex v, disabling it from any displacement during the optimisation process. Finer control can be given by applying a coefficient on the energy gradient, such that the vertices are not locked but hindered.

By modifying an environmental object, the environmental attributes may change, which can result in the destruction of the now incompatible environmental objects in the scene (Figure 1.36, middle). The removal of environmental object is subject to chain reactions, often visible in nature. Reducing the reaction is achievable by alternating, until a steady state is reached again, a step in the environment update (Section 1.6) and a step of skeleton refinement for each environmental object. As a counterbalance to the removal, each deleted environmental object is reinserted in the scene (Figure 1.36, right) following the process described in Section 1.5.

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

Control over the region of the terrain that should be updated is given by adjusting all fitness functions through a scalar field $\lambda : \mathbb{R}^2 \mapsto \mathbb{R}$ such that the fitness function $\omega(\mathbf{p})$ of any new environmental object is evaluated as $\omega^*(\mathbf{p}) = \lambda\mathbf{p}\omega(\mathbf{p})$. This is especially useful in the planning of robotic simulations, as we can first generate the overall shape of our terrain and then focus the generation process around the areas that may be visited by the robot, avoiding useless simulations and computational power. Figure 1.47 shows an example of colonisation of the coral polyps that we limited manually into an annulus.

Figure 1.37: *Deformation of an environmental object is applied by displacing the control points of its outlines. All environmental objects in the scene evaluate their new fitness in the environment.*

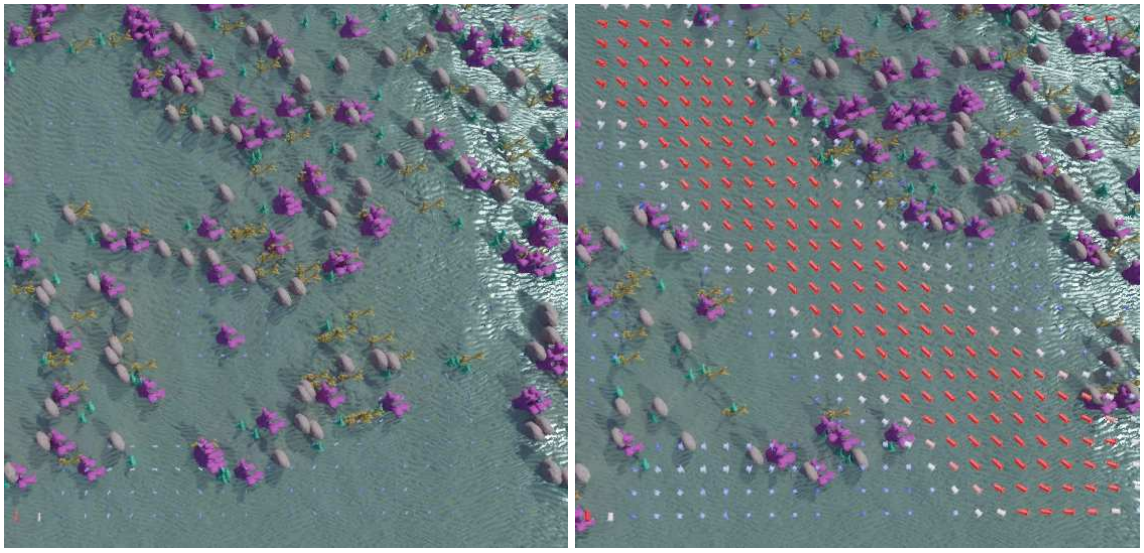

Figure 1.38: *A strong water current created by the user has a direct influence on the corals that are sensitive. Environmental objects that are far enough from the user stroke are unaffected.*

Our water current simulation is modelled as a simple vector field. As such, the user is able to interact with it at any moment of the simulation, allowing for the death of sensitive environmental objects while guiding the simulation into a new landscape. By modifying the water currents, the user also modifies the transport rate of environmental materials at this position. The modification of currents is given as a stroke, a parametric curve C for which we evaluate $\Delta\mathcal{W}_{\text{user}}(\mathbf{p})$ just as described in **??**'s user-driven wind velocity field construction. A simple user stroke shows the impact of a strong underwater current on coral colonies in Figure 1.38.

1.7.2 Indirect interaction with environmental objects

A configuration file can define in advance the different geomorphic events that should be triggered during the simulation. This enables the simulation to reproduce the geological evolution of real-world landscapes. Multiple geomorphic events can be triggered either as sudden or continuous environmental changes. These changes play a significant role in the morphology of landscapes. We define geomorphic

Figure 1.39: *Lowering the water level by a few metres caused most of the coral objects to satisfy $\omega \leq 0$, causing their death. Since the water level (blue) decreases slowly, new coral objects spawn progressively at a lower altitude.*

events with a starting point and an ending point, such that at any time of the simulation we can compute the progress of the geomorphic event as $t_e \in [0,1]$ and linearly interpolate the effects on environmental attributes.

Water level events

Water level changes are key geomorphic events that shape underwater landscapes. As previously submerged environmental objects become elevated above the water level, flora and fauna terrain features dry and die. Deprived of the living part of the features, everything is more affected by terrestrial erosion. By updating the value of the depth $\mathcal{D}$ evaluated in the fitness functions, any environmental object that is sensitive to depth and altitude will be impacted automatically, which may cause death (Figure 1.39). The modification of the water level is defined as

$$\mathcal{D}(\mathbf{p}) = \mathcal{D}_0(\mathbf{p}) + \sum_{e \in \text{events}} \Delta\mathcal{D}_e t_e \tag{1.56}$$

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

Subsidence and uplift events

Subsidence and uplift are the main geomorphic events that create or destroy islands in the long term, as presented in **??**. These geomorphic events are simulated as a simple factor on the height field of the generated terrain (Figure 1.40). Subsidence is not always uniform across the terrain. As such, the user can provide a position $\mathbf{q}$ at which the subsidence is strongest, the amount of subsidence applied $\Delta\mathcal{H}_e$ and a standard deviation σ, from which we can then compute, at any point in space and time of the simulation, the height of the terrain:

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

with $G_\sigma(x)$ the Gaussian function:

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{1.57}$$

Figure 1.40: *Simulating subsidence on a part of the terrain (brown area) causes the depth value to change locally, resulting in the death of coral objects that find themselves too deep to survive. Here two subsidence geomorphic events are triggered in parallel.*

The Gaussian analytic formulation guarantees continuity of the changes on the environmental attributes, with smooth transitions from the epicentre of the geomorphic events as the distance increases. Any other analytical formulation could be used, but we chose the Gaussian formulation for its simplicity of use as the only parameter needing parametrisation from the user is the standard deviation σ.

Storm events

Storms are significant factors in the geomorphology of coral reefs [VK16, OATS23] and coasts [DAG05, CWC10]. Due to the extreme wind and wave velocities, coasts are highly eroded in a short time period and the more fragile corals near the water surface are broken, possibly causing breaches in the reefs and spreading polyps in the current's direction. While there are many factors at play to understand the appearance of storms and the hydrodynamics affecting them, we simplified the model of storms for the user to a single epicentre $\mathbf{q}$ with a wind velocity v_{wind} and a standard deviation σ representing the spread around the epicentre (Figure 1.41). The computation of water currents is then given as

$$\mathcal{W}_{\text{user}}(\mathbf{p}) = \mathcal{W}^{*}_{\text{user}}(\mathbf{p}) + \sum_{e \in \text{events}} v_{\text{wind}} G(\|\mathbf{p} - \mathbf{q}\|)$$

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

Our framework can easily be extended as the geomorphic event system remains similar for all geomorphic events as we see in Algorithm 4. Including higher-level simulations in the geomorphic event system can be added, such as the simulation of tectonic activity, the use of fluid dynamics for tsunami geomorphic events, the integration of human activity to alter specific environmental materials, etc...

1.8 Results and discussion

Our method provides a way to generate scenes at different scales. We demonstrate this capacity with the generation of a large scene of an island (Figure 1.1 and Figure 1.49), after which we focused the generation process in a canyon (Figure 1.46), then a small-scale visualisation of coral colonies (Figure 1.47).

Figure 1.41: *The result of a storm localised on one side of the island (red area) modifies the result of the evaluation of environmental objects around its epicentre for a short period of time. Most of the coral objects died from the geomorphic event, except for a few environmental objects less sensitive to the strength of the water currents.*

Input: $\mathcal{H}, \mathcal{L}, \mathcal{W}_{\text{user}}$, events, t
Output: $\mathcal{H}, \mathcal{L}, \mathcal{W}_{\text{user}}$ after applying geomorphic events changes

foreach *e in events* **do**
  $t_e \leftarrow \frac{t - t_{e,\text{start}}}{t_{e,\text{end}} - t_{e,\text{start}}}$

$\mathcal{L} \leftarrow \mathcal{L} + \sum_{e \in \text{water events}} \Delta\mathcal{L} t_e$
$\mathcal{H}(\mathbf{p}) \leftarrow \mathcal{H}(\mathbf{p}) + \sum_{e \in \text{subsidence events}} \Delta\mathcal{H}_e \, G_{\sigma_e}(\|\mathbf{p} - \mathbf{q}_e\|) t_e$
$\mathcal{W}_{\text{user}}(\mathbf{p}) \leftarrow \mathcal{W}_{\text{user}}(\mathbf{p}) + \sum_{e \in \text{storm events}} v_e \, G_{\sigma_e}(\|\mathbf{p} - \mathbf{q}_e\|)$
return $\mathcal{H}, \mathcal{L}, \mathcal{W}_{user}$

Algorithm 4: Applying geomorphic events

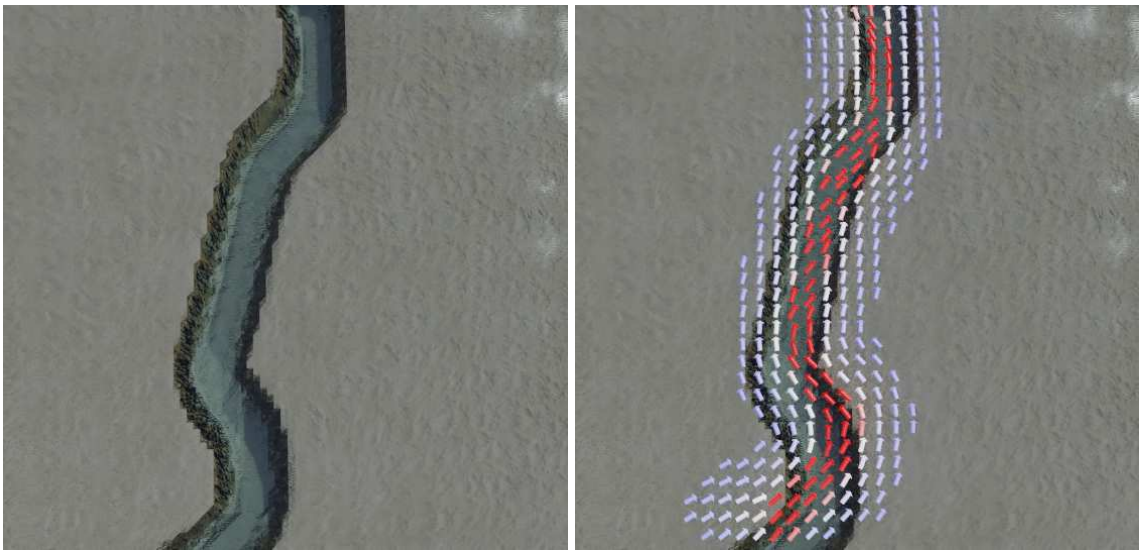

Figure 1.42: *A canyon creates a water current in its direction.*

Figure 1.43: *Inside view of a canyon with an arch. The arch generates shade (red) and pebbles (green), which is a beneficial environment for the appearance of rocks, which will in turn spread sand.*

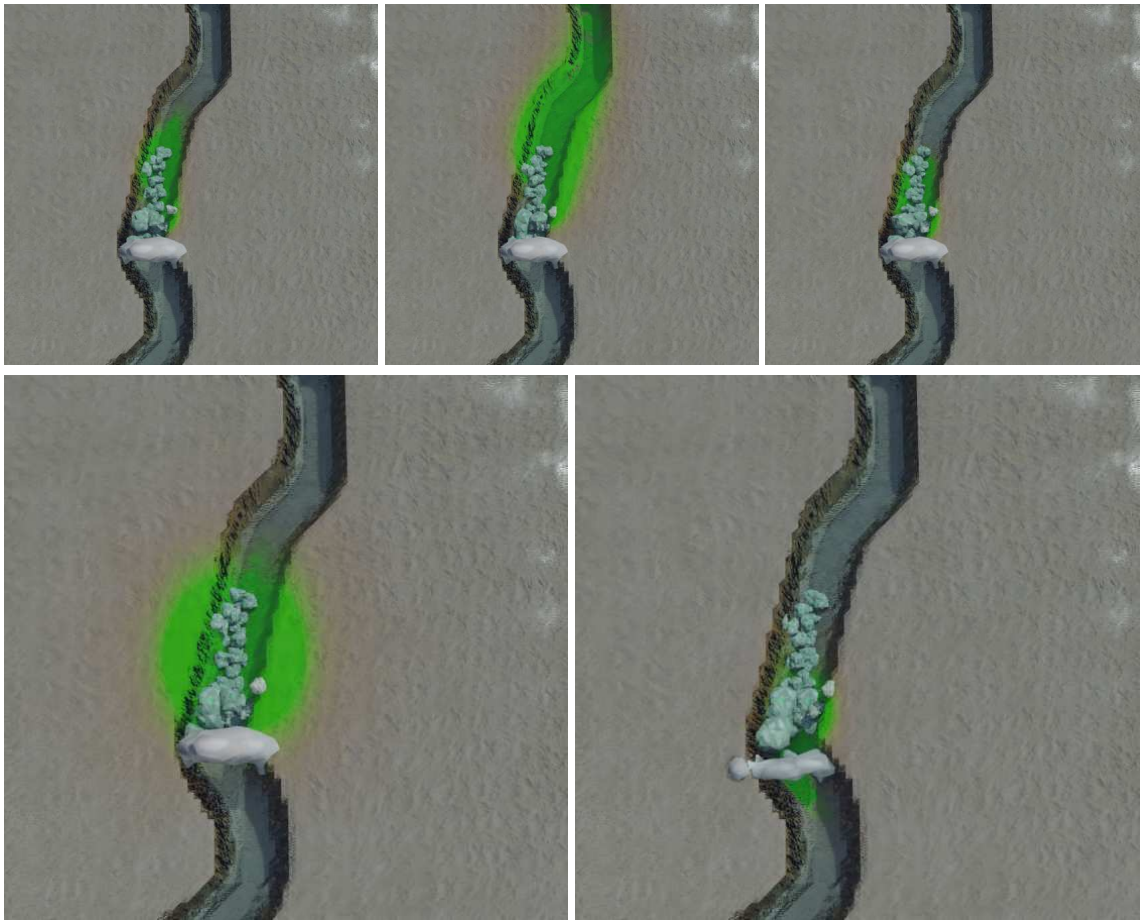

Figure 1.44: *Influence of material parameters on their spread (sand generated by the rocks). From left to right, and top to bottom: normal sand parameters, low decay, no influence of water flow, large diffusivity, inverse water influence (spread upwind in the canyon).*

In Figure 1.49, a canyon scene is generated using our method. The water flow is affected by the curve of the canyon such that the currents are oriented in the direction of the curve's tangent. In this example, we force the position of arches to be inside the canyon. The arches deposit a material "rock deposit", which is the main element of the fitness function of the Rock object. The "rock deposit" is slightly affected by water currents, but its mass makes it highly affected by gravity. As such, rocks will spawn underneath arches. In reality, an arch is often created as part of a large coral boulder that sees the calcareous bottom part detached by the water currents, often resulting in an arch surrounded by big rocks and smaller rocks from the erosion of the first rocks.

As such, we define an environmental object "Arch" with a fitness function $\omega_{arch}(\mathbf{p}) = 5 - d(canyon - \mathbf{p}) * \|\mathcal{W}(\mathbf{p})\|$, an environmental object "Rock" using $\omega_{rock}(\mathbf{p}) = \mathcal{M}_{rock\_deposit}(\mathbf{p})$ and Pebble using $\omega_{pebble}(\mathbf{p}) = \mathcal{M}_{smaller\_rock\_deposit}$

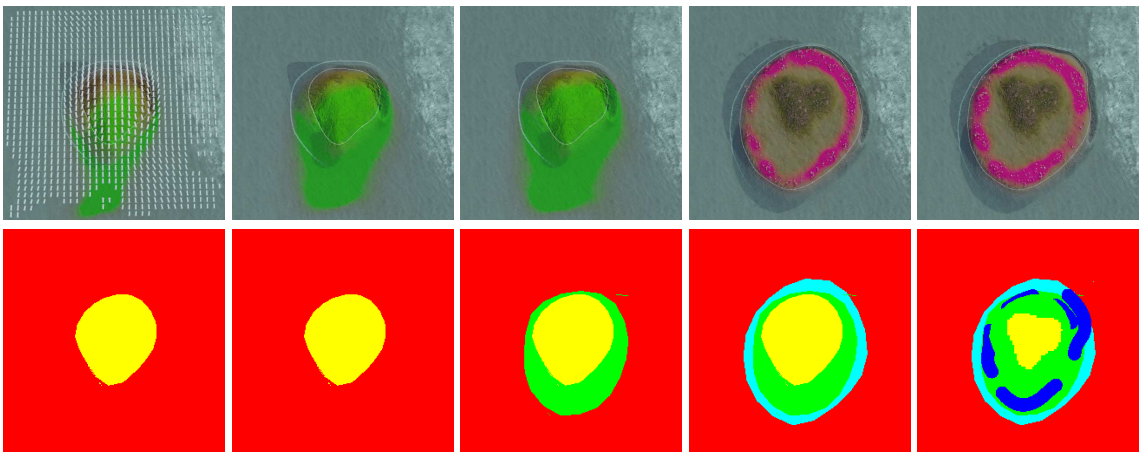

Figure 1.45: *Iterative construction of an island under high wind force, the deposition of sand is affected by the wind, giving an elongated shape to the beach. The large presence of sand due to the beach creates a lagoon, and quickly coral colonies spread on the borders of the lagoon to avoid the sand. Coral reefs appear from the deposition of calcareous material from coral. Bottom: The label map used in* **??** *can be extracted from the environmental objects of the scene.*

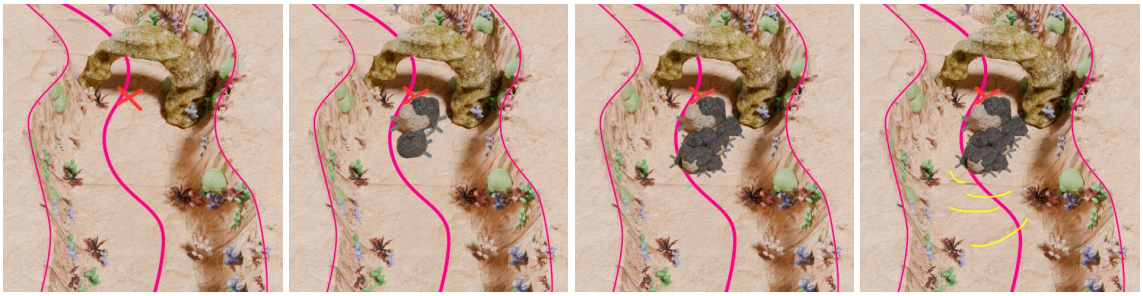

Figure 1.46: *Evolution of a canyon scene at different iterations of the simulation. The appearance of an arch causes the spawning of rocks, pebbles, and finally some deposition of sand at the bottom of the canyon, spawning ripples.*

Finally, sand ripples are simply described as curves appearing where there is a lot of sand available: $\omega_{ripple}(\mathbf{p}) = \mathcal{M}_{sand}(\mathbf{p})$.

Following these simple rules, Figure 1.46 shows the emergence of details in the scene.

In this example we defined three different types of corals, coralA, coralB and coralC, to illustrate the possibility to model behaviours from the choice of fitness functions. Each of the coral types deposits a material "coral polyp" and "coral polyp A" ("coral polyp B" and "coral polyp C" respectively). By considering a fitness function that minimises the ratio $\frac{\text{coral polyp}}{\text{coral polyp A}}$, we see an emergent behaviour of the three types of coral fighting for the space colonisation. Figure 1.47 shows the result of this simulation at three different iterations. At the border between two colonies, none of the colonies make progression due to the amount of coral polyp specific from the other colony.

The proposed method aims to generate plausible landscapes using simplified versions of the evolution of an ecosystem and of the 3D representation. The biological realism of the result is highly correlated to the amount of simplification and assumptions, while the visual realism is completely dependent on the geometric functions used for the 3D modelling of the environmental objects. While proposing a flexible method that proposes a generic approach for terrain generation, a close collaboration with field experts and with graphists is needed to achieve optimal results. Adding notions of slope and sensibility to water currents in Figure 1.48 for different coral morphologies (massive, branching, foliose and table corals), each coral entity finds a place that fits the distribution of species presented in **??**.

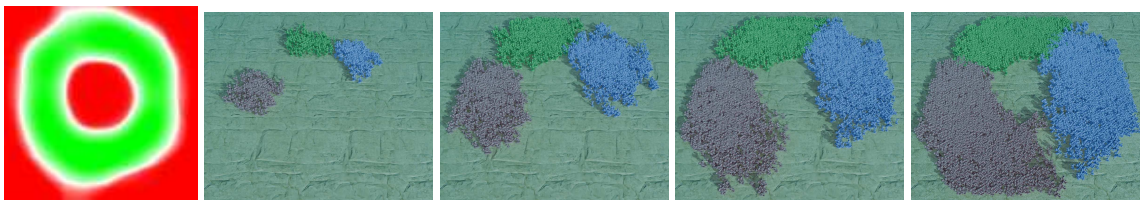

Figure 1.47: *Three colonies of coral (red, blue, green) restricted to an annulus (leftmost, red has negative impact on the fitness function while green has positive impact) in the middle section of the terrain fighting for the space.*

Figure 1.48: *Introducing penalities on water currents and terrain slope for different types of corals, we see naturally a layout emerging with branching corals (purple) in the back reef, foliose corals (cyan) mainly around the reef crest, and massive corals (brown) spreading from the back reef to the fore reef.*

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

1.9 Conclusion

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

included in the automatic simulation process.

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

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

While our semantic environmental object framework introduced sparse, interactive primitives for ecosystem generation without restraining to a unique terrain representation, the underlying environmental fields remain constrained to 2D scalar fields. This simplification ensured efficiency and interpretability, but it prevents the representation of intrinsically three-dimensional features such as overhangs, caves, and porous reef structures. Moreover, terrains in the real world are rarely static: they evolve continuously under the action of water, wind, and other erosive agents. Extending procedural methods toward volumetric models and terrain-altering processes is therefore essential to reach higher realism, especially in underwater environments where bioerosion and hydrodynamics shape the seascape.

The following chapter tackles this challenge by focusing on erosion simulation. We present a particle-based method that generalises detachment, transport, and deposition processes across both surfacic and volumetric representations, offering a flexible and efficient framework for integrating erosion into procedural terrain generation.

Figure 1.49: *A simple coral reef island is generated using an island, a lagoon, reefs, coral polyps, beaches, trees and algae environmental objects. Trees appear on beaches and algae grow in the lagoon's sand.*