Oasis: A Real-Time Hydraulic and Aeolian Erosion Simulation with Dynamic Vegetation

Alexander Maximilian Nilles[a], Lars Günther and Stefan Müller
Institute for Computational Visualistics, University of Koblenz, Universitätsstr. 1, Koblenz, Germany
{nillesmax, larsguenther98, stefanm}@uni-koblenz.de

Keywords: Real-Time Simulation, Aeolian Erosion, Hydraulic Erosion, Vegetation Simulation, Desert, Sand Dune Simulation, GPU, CUDA.

Abstract: We present a novel real-time combined simulation for aeolian erosion, hydraulic erosion and vegetation, capable of transforming barren deserts with sand dunes into lush forest landscapes and vice versa using simple user interaction. Existing aeolian and hydraulic erosion methods are extended and unified using a moisture model on a layered heightmap, supporting bedrock, soil and sand as terrain materials. Vegetation uses a 3D radius-based model and is efficiently rasterized to a 2D density map via a split-Gaussian model, inhibiting erosion. Abiotic factors such as moisture, terrain slope, surface water and illumination are considered in vegetation growth and vegetation can spread radially as well as with the wind. Each plant considers the position and size of all neighboring plants as a biotic growth factor, made possible through a set of uniform grids of varying resolutions. The user can freely model different plant species by defining their ecological niche and adaptability to changes in terrain elevation and competition with other plants. Even underwater vegetation is possible. Interspecies competition can be defined freely using a competition matrix. The resulting method runs in real-time at a terrain resolution of 2048^2 with 2,000,000.00 plants.

1 INTRODUCTION

Erosion processes play a key role in landscape formation, which is an important area of computer graphics. Real-time erosion simulations on heightmaps exist for both hydraulic (water-based) erosion (Benes, 2007; Mei et al., 2007; Krištof et al., 2009; Šťava et al., 2008) and aeolian (wind-based) erosion (Taylor and Keyser, 2023; Nilles et al., 2024a), however, to the best of our knowledge, no method currently exists that combines hydraulic and aeolian erosion with support for dune propagation in a single simulation.

Vegetation, which interacts bidirectionally with terrain, is also important to landscapes. Plenty of research in computer graphics has dealt with placing large amounts of vegetation in a plausible or realistic manner on existing landscapes. Some real-time erosion simulations also consider the effects of vegetation on erosion, but do not simulate the vegetation itself, such as in (Hawkins and Ricks, 2023). However, non-real-time methods combining erosion and vegetation bidirectionally have been proposed before (Cordonnier et al., 2017). This method in particular, while interactive and powerful, requires almost an entire minute for a simulation step at a low resolution. In the current time, higher resolutions are usually required and long waiting periods for a potentially undesired result from the viewpoint of an artist are likely to cause such methods to be rejected in this field.

In this paper, we propose a novel method that fills both gaps. Our method combines aeolian and hydraulic erosion with a vegetation simulation. We simulate vegetation growth and spread while considering biotic and abiotic factors. All simulation components affect each other bidirectionally, which is made possible using a terrain moisture model and a split-Gaussian rasterization of our novel radius-based vegetation model. Our method is fully interactive and the user can change parameters to transform a desert into a lush forest and vice versa, or create an oasis in a desert. The method is capable of simulating a 2048^2 sized terrain with two million plants in real-time. It is the first method in computer graphics that simulates aeolian erosion, hydraulic erosion and vegetation in a bidirectional manner in real-time at this scale. To facilitate reproduction of our results, the code is available open source (Nilles and Günther, 2025).

[a] https://orcid.org/0000-0002-4196-7424

Nilles, A. M., Günther, L. and Müller, S.
Oasis: A Real-Time Hydraulic and Aeolian Erosion Simulation with Dynamic Vegetation.
DOI: 10.5220/0013112100003912
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 1: GRAPP, HUCAPP and IVAPP*, pages 39-52
ISBN: 978-989-758-728-3; ISSN: 2184-4321
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

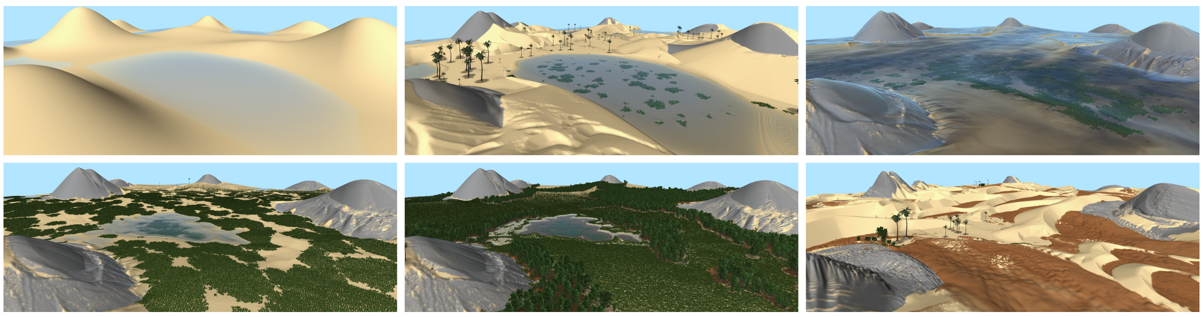
Figure 1: A terrain initialized with low-frequency noise is transformed by a user by changing the rain strength. Initially, rain is disabled, causing dune formation and aeolian bedrock abrasion. Adapted plants such as palm trees grow in the vicinity of water sources and seaweed grows under water. The terrain is then flooded with heavy rain, causing strong hydraulic erosion. Palm trees die in the process and only seaweed can grow, which transforms some sand to soil. Next, rain strength is reduced significantly and most water evaporates. Sufficient moisture prevents saltation and allows for bushes to grow across the entire scene, which transform sand to soil on a large scale. As soil is generated, trees can begin to grow in the new environment and start to spread throughout the scene, forming forests. Finally, the user disables rain again. The terrain dries out, causing most vegetation to die, returning to a desert. The entire process took roughly 15 minutes.

The remainder of this paper is structured as follows: In Section 2, we introduce relevant related work. Next, Section 3 introduces our method in detail, followed by our results in Section 4. We conclude with Section 5, discussing important directions for future work and limitations.

2 RELATED WORK

In this section, we will briefly review related work in computer graphics concerning aeolian erosion, hydraulic erosion, vegetation simulation and combined methods, while focusing on real-time methods.

2.1 Aeolian Erosion

The Desertscape Simulation (Paris et al., 2019) was the first method in computer graphics supporting dune formation and bedrock abrasion in desert environments. Barchans, linear dunes, star-shaped dunes and nabhka dunes are supported and vegetation is considered via a density map. The simulation is interactive, but not real-time. It has since been implemented in real-time on the GPU (Taylor and Keyser, 2023; Nilles et al., 2024a). (Taylor and Keyser, 2023) added echo dune support and compared their results with wind tunnel experiments (Tsoar, 1983) and an accurate offline method (Lü et al., 2018). In (Nilles et al., 2024a), the method was further optimized for the GPU using CUDA and made deterministic. A new reptation method produces results closer to the reference method in (Lü et al., 2018). The implementation has since been expanded with divergence-free wind fields and improved reptation (Nilles and Günther, 2024).

2.2 Hydraulic Erosion

Multiple real-time hydraulic erosion methods emerged since 2007. (Benes, 2007) used the shallow water equations. Water destroys the terrain and forms grit (regolith), simulated as a high viscosity fluid. (Mei et al., 2007) instead use the virtual pipes method (O'Brien and Hodgins, 1995), paired with a capacity-based dissolution/deposition model. In (Šťava et al., 2008), the virtual pipes method and erosion model from (Mei et al., 2007) are combined with the shallow water grit simulation from (Benes, 2007). The method is generalized to multiple different material layers with varying erosion resistance and materials age over time. (Krištof et al., 2009) used a 3D smoothed particle hydrodynamics simulation and perform particle-based erosion on a heightmap instead. (Hawkins and Ricks, 2023) extended (Šťava et al., 2008) with vegetation, modeled as an additional layer that can die as a result of erosion, forming another layer which can be transported similar to sediment. Vegetation reduces the impact of erosion. However, there is no simulation of vegetation growth and spread. Recently, (Nilles et al., 2024b) extended the approach of (Mei et al., 2007) to 3D using multi-layered heightmaps. Their method is capable of generating arches, overhangs and limited caves in real-time.

2.3 Vegetation Simulation

We focus on methods that only consider plant position, size and type (radius-based methods) instead of methods simulating detailed growth of individual plants. An early real-time method considers abiotic factors such as elevation and slope, which is paired with noise functions for procedural vegetation placement via an ecosystem probability (Hammes, 2001). This was extended by (Ch'ng, 2011), which introduce biotic factors for a more realistic placement of plants. The *Field of Neighborhood* (FON) method introduced in (Berger et al., 2002) uses a circular zone of influence around each plant, modeling plant competition that results in growth reduction of neighboring plants. This was further developed by (Alsweis and Deussen, 2006; Weier et al., 2013), where pregenerated tiles are used for real-time performance. *EcoBrush* (Gain et al., 2017) instead draws interpolated samples from a database to synthesize a full ecosystem quickly with interactive control by the user.

A new framework for real-time procedural plant distribution on large-scale terrains was developed in (do Nascimento et al., 2018). They consider abiotic and biotic factors and allow the user to define plant types via a set of parameters describing their adaptability, which resembles our approach. Vegetation is organized in a quadtree, where vegetation only affects plants in layers below it. Moisture is procedurally computed from a wide range of parameters using influence curves. Plant overlap is avoided using distance fields instead of directly checking plants for collisions.

Ecoclimates (Pałubicki et al., 2022) is the state-of-the-art in realism for outdoor landscapes with vegetation. The bidirectional relationship between vegetation and weather is modeled by combining a 3D weather simulation, a soil model and a vegetation model, simulating the entire water cycle. The simulation is not real-time, but interactive. It is the first method that captures forest edge effects, Foehn and spatial vegetation patterning.

2.4 Combined Methods

(Cordonnier et al., 2017) combined erosion simulation and vegetation simulation with bidirectional feedback, which is also the goal of our method. Their method uses a layered heightmap terrain representation and an event-based framework. Geomorphological events include rainfall, running water, temperature, lightning, gravity and fire. Ecosystem events deal with soil moisture, evapotranspiration, illumination and temperature, which define the vigor and stress of plants, resulting in germination, growth or death and the generation of humus (dark organic matter in soil). Notably, aeolian erosion or dune formation are not supported. The event-based nature is ill-suited to the GPU, and their CPU implementation scales poorly with resolution, requiring roughly $10\times$ as long when the number of cells is quadrupled. A simulation step at 1024^2 resolution requires $38s$. Recently, (Hartley et al., 2024) generalized erosion on various terrain representations, ranging from heightmaps over multi-layered heightmaps to 3D voxels, using particles as erosion agents. The framework can be used for a variety of erosion effects, such as hydraulic erosion, aeolian erosion and thermal erosion, reproducing the Desertscape Simulation results from (Paris et al., 2019) to some extent.

3 OUR METHOD

This section will explain our method in detail. The core components of the simulation are *vegetation simulation*, *aeolian erosion* and *hydraulic erosion*. A high-level overview of how these components are linked together can be found in Figure 2. Aeolian erosion has been adapted from the real-time desertscapes simulation in (Nilles et al., 2024a) and our implementation directly extends their code (Nilles and Günther, 2024). For hydraulic erosion, we have taken ideas from earlier as well as recent work (Mei et al., 2007; Šťava et al., 2008; Nilles et al., 2024b). Our vegetation model is a modified radius-based model that we designed specifically for our erosion simulation.

3.1 Overview

The terrain consists of a 2D grid of $N_x \times N_y$ cells. Each cell $\boldsymbol{i}$ is square with a side length of l_c, set to $1m$ for all scenes in this paper. The cells form a layered heightmap, containing bedrock $\mathcal{T}_B$, soil $\mathcal{T}_e$, sand $\mathcal{T}_s$ and water $\mathcal{T}_w$ in that order. Uppercase indicates absolute quantities and lowercase relative quantities, i.e.

$$\mathcal{T}_{W,\boldsymbol{i}} = \mathcal{T}_{B,\boldsymbol{i}} + \mathcal{T}_{e,\boldsymbol{i}} + \mathcal{T}_{s,\boldsymbol{i}} + \mathcal{T}_{w,\boldsymbol{i}} \tag{1}$$

is the absolute water height of cell $\boldsymbol{i}$. We define a vegetation height $\mathcal{T}_v$, which is relative to the sand height and overlaps with the water layer. The absolute vegetation height is calculated as

$$\mathcal{T}_{V,\boldsymbol{i}} = \mathcal{T}_{B,\boldsymbol{i}} + \mathcal{T}_{e,\boldsymbol{i}} + \mathcal{T}_{s,\boldsymbol{i}} + \mathcal{T}_{v,\boldsymbol{i}} \tag{2}$$

and is used for shadow calculation. The 3D position of the terrain surface is defined as

$$\boldsymbol{x_i} = \left(l_c \cdot \left(\boldsymbol{i}_x + \frac{1}{2}\right), \mathcal{T}_{S,\boldsymbol{i}}, l_c \cdot \left(\boldsymbol{i}_y + \frac{1}{2}\right)\right)^T \tag{3}$$

Each cell has a 2D wind velocity $\boldsymbol{w_i}$, calculated from a high-altitude wind velocity $\boldsymbol{w}_a$, and a 2D water velocity $\boldsymbol{u_i}$.

The different components of our simulation interact via an absolute terrain moisture level $\mathcal{T}_M$, modeled up to a depth of $1m$. The moisture capacity $\mathcal{T}_{M_c}$ is

$$\mathcal{T}_{M_c,\boldsymbol{i}} = c_M \cdot \min(\mathcal{T}_{e,\boldsymbol{i}} + \mathcal{T}_{s,\boldsymbol{i}}, 1), \tag{4}$$

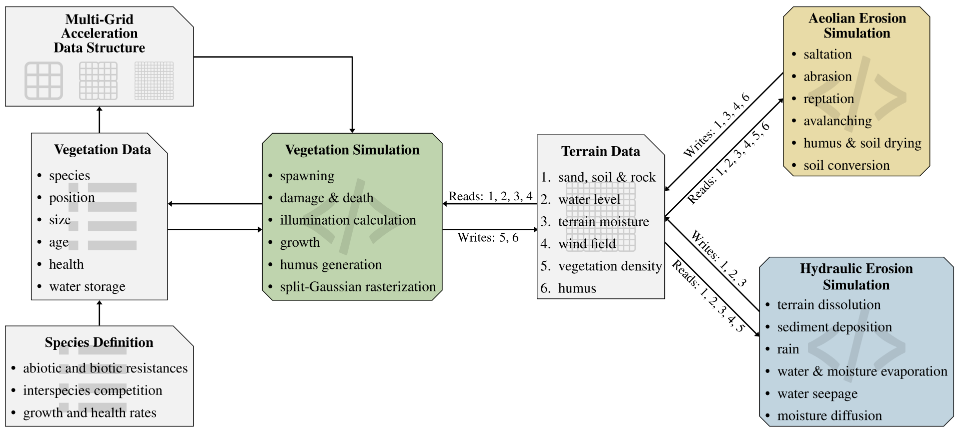

Figure 2: A high-level overview of our method. Terrain data is stored in a 2D grid which serves as the common interface of vegetation simulation, aeolian erosion and hydraulic erosion, which read and modify the terrain data. This enables bidirectional feedback between the three simulations. The vegetation data is stored as an array, which is only directly read and modified by the vegetation simulation. Neighborhood queries for plant competition are accelerated via our multi-grid data structure. The erosion simulations access the vegetation information via a vegetation density map, which is rasterized from the actual vegetation using our split-Gaussian model.

where c_M is a moisture capacity constant, typically set to 1. The relative moisture is defined as

$$\mathcal{T}_{m,\boldsymbol{i}} = \frac{\mathcal{T}_{M,\boldsymbol{i}}}{\mathcal{T}_{M_c,\boldsymbol{i}}}. \tag{5}$$

Moisture is used for plant growth and affects aeolian erosion and the angle of repose of soil and sand.

The aeolian erosion simulation from (Nilles et al., 2024a) requires a vegetation density $\mathcal{T}_{\mathcal{V}} \in [0,1]$ for each cell. Instead of using a static vegetation density, we create the vegetation density map dynamically in each simulation step, based on the current 3D positions and sizes of the plants in the scene. It impacts the angle of repose and aeolian erosion, which we extend to impact hydraulic erosion and moisture evaporation.

Lastly, aeolian and hydraulic erosion processes turn soil into sand. We use vegetation to make this reversible via a humus layer $\mathcal{T}_{h,\boldsymbol{i}}$. This layer does not contribute to the terrain height. It is generated by vegetation, decays if there is no moisture present and otherwise slowly transforms sand into soil.

3.2 Vegetation Model

We use a custom radius-based vegetation model tailored to our use case. It is not directly based on previous work and intentionally simple. More advanced ideas from related work could be integrated later to improve our model.

Each plant k has a 3D position $\mathcal{V}_{\boldsymbol{x},k}$, radius $\mathcal{V}_{r,k}$, health $\mathcal{V}_{h,k}$, water storage $\mathcal{V}_{w,k}$, age $\mathcal{V}_{a,k}$ and species $\mathcal{S}_k$. $\boldsymbol{\mathcal{S}}_j$ is the set of all plants that belong to species j. The species $\mathcal{S}_k \in \mathbb{N}$ of a plant determines how it is adapted to different environmental factors, how fast it grows, matures and withers, as well as the maximum radius and how the stem height and root depth relate to its radius. A competition matrix $\boldsymbol{D}$ encodes interspecies competition, where each entry $d_{ij} \in [0,\infty)$ defines how much species j reduces the growth of species i. This can be used to model species that can coexist easily, compete equally for resources, or to model an invasive species that completely dominates another.

Table 1 lists the 26 parameters defining a species and whether they affect growth, health or the spawning of new vegetation. We will explain the most important parameters in detail and refer to `vegetation.cu` in the source code for further information.

The age of each plant is incremented by Δt every time step and a plant is considered mature if it has reached the maturity percentage of its maximum radius in the maturity time. Only mature plants can reproduce and plants that fail to reach maturity are removed. Water resistance is used to differentiate underwater plants from regular plants. A resistance of 100% indicates a species that only grows under water.

The vegetation simulation is structured as follows: First, vegetation is rasterized into multiple 2D grids (Section 3.2.1), which bidirectionally links it to the erosion simulations and is used when spawning new vegetation (Section 3.2.2). Spawning of new vegetation depends on the local and windward density of each species, as well as compatibility with the environment. Next, a two-layered illumination map is calculated using the terrain elevation and rasterized veg-

Table 1: The parameters of a species. A cross marks whether they affect growth (G), health (H) or spawning of new vegetation (S). Indirect effects are marked with parenthesis.

Parameter	G	H	S
Maximum radius $r_{\max}$	×	×	(×)
Growth rate ϕ_g	×	×	
Position adjust rate ϕ_p		(×)	
Damage rate ϕ_d		×	
Shrink rate ϕ_s		(×)	
Maturity time		×	×
Maturity percentage		×	×
Relative stem height h_s	(×)	(×)	(×)
Relative root depth h_r	(×)	(×)	(×)
Water usage rate ϕ_w	×	×	
Water capacity w_c	×	×	
Water resistance	×	×	×
Minimum moisture			×
Maximum moisture		×	×
Soil compatibility	×		×
Sand compatibility	×		×
Maximum stem coverage		×	
Minimum root coverage		×	
Maximum slope	×	×	×
Base spawn probability p_b			×
Density spawn multiplier p_ρ			×
Wind spawn multiplier $p_{\mathcal{W}}$			×
Humus generation rate ϕ_h			
Minimum illumination	×	×	
Maximum illumination	×	×	
Density separation ρ_s			×

etation heights (Section 3.2.3), which is used as an abiotic growth factor as well as for visualization. We then calculate the growth of each plant (Section 3.2.4) depending on the local abiotic and biotic (plant competition) factors. Plants that are unable to grow due to incompatibility with the environment receive damage, potentially dying, and plants that were unable to mature in time are culled (Section 3.2.5). Lastly, the acceleration data structure for neighborhood lookups is updated (Section 3.2.6).

3.2.1 Rasterizing Vegetation

Our terrain simulation interacts with vegetation indirectly using the vegetation density map $\mathcal{T}_{\mathcal{V}}$. This needs to be rasterized from the current population of plants in every simulation step. We describe the density of a single plant using a split 3D Gaussian, with μ set to the position of the plant and diagonal covariance matrix $\Sigma(\boldsymbol{x},k)$, set to $4\mathcal{V}_{r,k}^2$ horizontally. Vertically, $4(h_{s,k}\cdot\mathcal{V}_{r,k})^2$ is used for positions above the plant, and $4(h_{r,k}\cdot\mathcal{V}_{r,k})^2$ below it.

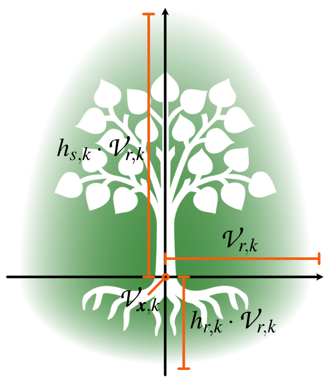
$h_{s,k}\cdot\mathcal{V}_{r,k}$
$\mathcal{V}_{r,k}$
$\mathcal{V}_{\boldsymbol{x},k}$
$h_{r,k}\cdot\mathcal{V}_{r,k}$

Figure 3: An illustration of our vegetation density model in 2D for a plant with shallow roots and high stem. The Gaussian vegetation density is shown in green.

The density of plant k at 3D position $\boldsymbol{x}$ is

$$\rho(\boldsymbol{x},k) = e^{-\frac{1}{2}(\boldsymbol{x}-\mathcal{V}_{\boldsymbol{x},k})^T\Sigma(\boldsymbol{x},k)^{-1}(\boldsymbol{x}-\mathcal{V}_{\boldsymbol{x},k})}. \tag{6}$$

We further modify this to ensure that the density is 0 if the distance in the 2D plane is equivalent to the radius, which is necessary for our acceleration data structure:

$$\bar{\rho}(\boldsymbol{x},k) = \max\left(\rho(\boldsymbol{x},k) - \frac{e^{-2}\|\boldsymbol{x}-\mathcal{V}_{\boldsymbol{x},k}\|_{xz}}{\mathcal{V}_{r,k}}, 0\right). \tag{7}$$

This is evaluated at the terrain surface $\boldsymbol{x_i}$, which allows us to properly consider height changes in terrain. If a plant is partially buried under terrain or has its roots exposed, it will have a lower density. The split model allows us to describe plants that do not grow spherically and to specify root depth separately from height. Figures 3 and 4 illustrate this density model.

The rasterized vegetation density is defined as

$$\mathcal{T}_{\mathcal{V},\boldsymbol{i}} = \min\left(\sum_k \bar{\rho}(\boldsymbol{x_i},k), 1\right). \tag{8}$$

We additionally rasterize the per-species vegetation density $\mathcal{T}_{\mathcal{V},\boldsymbol{i},j}$ by restricting the sum in Equation (8) to a given species j, which is used when spawning new plants, alongside a directional per-species vegetation density which takes the wind direction into account:

$$\mathcal{T}_{\mathcal{W},\boldsymbol{i},j} = \sum_{k\in\mathcal{S}_j}\left(\rho_{\mathcal{W}}(\boldsymbol{i},k)\cdot\left(1-\frac{\|\boldsymbol{x_i}-\mathcal{V}_{\boldsymbol{x},k}\|}{r_{\max,j}}\right)\right), \tag{9}$$

$$\rho_{\mathcal{W}}(\boldsymbol{i},k) = \max\left(\frac{\boldsymbol{w_i}}{\|\boldsymbol{w_i}\|}\cdot\frac{\boldsymbol{x_i}-\mathcal{V}_{\boldsymbol{x},k}}{\|\boldsymbol{x_i}-\mathcal{V}_{\boldsymbol{x},k}\|}, 0\right). \tag{10}$$

$\mathcal{T}_{\mathcal{V},\boldsymbol{i},j}$ and $\mathcal{T}_{\mathcal{W},\boldsymbol{i},j}$ only consider plants that have reached maturity, which can reproduce.

Vegetation height is rasterized as

$$\mathcal{T}_{v,\boldsymbol{i}} = \max_k\left(\mathcal{V}_{r,k}\cdot h_{s,k}\cdot\bar{\rho}(\boldsymbol{x_i},k)\right), \tag{11}$$

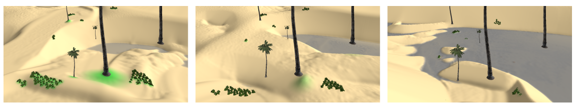
Figure 4: We visualize the rasterized vegetation density in green. Damage was disabled, allowing plants to survive if uprooted or buried. As the terrain changes, vegetation density adjusts, changing smoothly depending on the distance to the terrain surface from the plant origin. Plants that are buried deep or floating high do not contribute to density.

and the humus map is updated for the next time step by adding the humus generated by each plant k based on its humus generation rate $\phi_{h,k}$

$$\mathcal{T}_{h,\boldsymbol{i}}^{t+\Delta t} = \mathcal{T}_{h,\boldsymbol{i}}^{t} + \Delta t \cdot \sum_k \phi_{h,k} \cdot \bar{\rho}(\boldsymbol{x_i}, k). \qquad (12)$$

3.2.2 Spawning Vegetation

In each simulation step, new vegetation can spawn in any given cell. The species of a spawning plant is determined using a per species weight

$$\begin{aligned} p(\boldsymbol{i}, j) = p_{b,j} \cdot (1 + p_{\rho,j} \min(\mathcal{T}_{\mathcal{V},\boldsymbol{i},j}, 1) \\ + p_{\mathcal{W},j} \min(\mathcal{T}_{\mathcal{W},\boldsymbol{i},j}, 1)) \end{aligned}. \qquad (13)$$

In order to avoid vegetation spawning in incompatible environmental conditions, we reduce the weight of a species to 0 in those situations. Table 1 shows which environmental factors we consider. For example, if a species has a water resistance of 25%, we check the water level in the cell. If the water level exceeds 25% of the stem height, the species cannot spawn in this cell. A species with 100% water resistance is instead considered to be an underwater plant, which can only spawn and survive under water.

Furthermore, the density separation parameter $\rho_{s,j}$ of a species is used as a threshold. If the per-species density is larger than this, we set the weight to 0 to avoid plants spawning too close to each other.

The probability of a vegetation spawn event per cell and simulation step depends on the sum $p_{\text{sum}}(\boldsymbol{i})$ of the per species weights per cell, as well as the total amount of cells and the time step. As each cell can only contain a single plant at the same time, we consider $\frac{p_{\text{sum}}(\boldsymbol{i})}{N_x N_y}$ to be the probability of at least one plant spawning in the cell over the course of 1 second. This means that the vegetation spawn event has to happen with probability

$$p_{\text{spawn}}(\boldsymbol{i}) = 1 - \left(1 - \min\left(\frac{p_{\text{sum}}(\boldsymbol{i})}{N_x N_y}, 1\right)\right)^{\Delta t}. \qquad (14)$$

If a vegetation spawn event is triggered, the species of the plant is then selected randomly based on the per species weights.

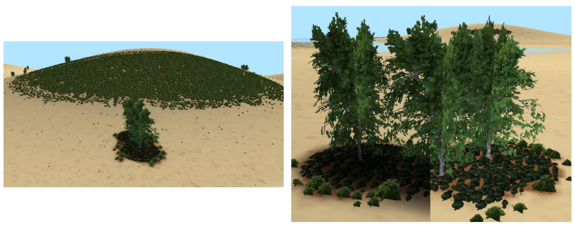
Figure 5: Left image: two types of bushes grow in a scene. One type cannot grow in direct sunlight or strong shadow and competes with trees, growing only on the shaded side of a hill and the outer edge of the tree shadow. The other type has no competition with trees and requires strong shade, so it grows underneath trees. A closeup is shown in the right image. The left half of the image is rendered using our volumetric shadow map, the right half has this disabled.

Plants spawn fully healthy with age 0 and no stored water. Their initial radius is set to 5% of the maximum possible radius, calculated as

$$r_{\max}(\boldsymbol{i}, j) = \min\left(\frac{\mathcal{T}_{s,\boldsymbol{i}} + \mathcal{T}_{e,\boldsymbol{i}}}{h_{r,j}}, r_{\max,j}\right). \qquad (15)$$

This considers the root depth of a plant and avoids plant growth into the bedrock layer.

3.2.3 Illumination

Light is one of the most important growth factors for plants. Instead of modeling a full day and night cycle with constantly changing shadows and light direction, we model the average illumination throughout the day. Our approach is motivated from ambient occlusion, where we introduce a directional bias toward a dominant light direction, usually south. We compute two illumination values per cell, one for the ground level and one for the vegetation level. These can then be interpolated for a given intermediate position. Figure 5 demonstrates the capabilities of this model.

Given a cell offset $\boldsymbol{o}$, we calculate the upward tangens angle to the cell at that offset from a given cell $\boldsymbol{i}$ at the two elevation levels:

$$\boldsymbol{\alpha}(\boldsymbol{i}, \boldsymbol{o}) = \frac{1}{l_c \|\boldsymbol{o}\|} \begin{pmatrix} \max(\mathcal{T}_{V,\boldsymbol{i}-\boldsymbol{o}} - \mathcal{T}_{S,\boldsymbol{i}}, 0) \\ \max(\mathcal{T}_{V,\boldsymbol{i}-\boldsymbol{o}} - \mathcal{T}_{V,\boldsymbol{i}}, 0) \end{pmatrix} \qquad (16)$$

and a weight considering light direction $\boldsymbol{l}$ and distance

$$d(\boldsymbol{i}, \boldsymbol{o}) = -2 \frac{\max(l_c(\boldsymbol{o} \cdot \boldsymbol{l}), 0)}{\|l_c \cdot \boldsymbol{o}\|^2}. \qquad (17)$$

Illumination values are then calculated using a 7×7 neighborhood by averaging an exponential function:

$$\boldsymbol{\sigma}(\boldsymbol{i}) = \sum_{\boldsymbol{o} \in \{-3 \ldots 3\}^2} \frac{2}{49} \begin{pmatrix} e^{d(\boldsymbol{i},\boldsymbol{o}) \cdot \boldsymbol{\alpha}_x(\boldsymbol{i},\boldsymbol{o})} \\ e^{d(\boldsymbol{i},\boldsymbol{o}) \cdot \boldsymbol{\alpha}_y(\boldsymbol{i},\boldsymbol{o})} \end{pmatrix} - 1. \qquad (18)$$

The reason for multiplying by 2 and subtracting 1 is that half of the samples evaluate to 1, so the illumination value would be in $[0.5,1]$ without this.

Using Equation (18), we can compute the illumination at height y via linear interpolation

$$\sigma(\boldsymbol{i},y) = (1-t)\boldsymbol{\sigma}_x(\boldsymbol{i}) + t\boldsymbol{\sigma}_y(\boldsymbol{i}), \text{ where} \tag{19}$$

$$t = \frac{y - \mathcal{T}_{S,\boldsymbol{i}}}{\mathcal{T}_{V,\boldsymbol{i}} - \mathcal{T}_{S,\boldsymbol{i}}}. \tag{20}$$

For underwater plants, we additionally apply an exponential decay of illumination with water depth.

3.2.4 Vegetation Growth

The rate at which a plant grows is determined by multiple growth factors g_i: Competition with surrounding plants $g_c \in [0,1]$, illumination $g_I \in [0,1]$, terrain slope $g_s \in [0,1]$, water availability $g_m \in [1,2]$, standing water $g_w \in [0,1]$ and ground composition $g_g \in [0,1]$.

These growth factors are determined based on the current environment and the species-specific parameters in Table 1. Further effects such as temperature are left for future work. We use a simplified model compared to the piece-wise linear hat-like functions used for plant response in (Cordonnier et al., 2017) and other previous work.

For illumination, each species has an interval of compatible illumination values. The illumination level is determined by evaluating Equation (19) at the top of the plant and g_I is interpolated based on the position in the valid interval, where the interval borders map to 0 and the center of the interval maps to 1. For example, if the interval is $[0.5,1.5]$, the associated species starts growing under medium illumination and growth increases up to the maximum illumination level of 1. The interval $[-0.5,0.5]$ describes a species that only grows in the shade and grows best at 0 illumination.

The growth factors g_s and g_w work similarly, the only difference is that the species determines the right end of the interval, which maps to no growth, while flat terrain and absence of standing water lead to the best growth for these factors. Underwater plants ignore g_w.

Ground composition works differently. It is intended to model nutrients as well as the ability of the plant's roots to grow in hard and soft materials. The sand and soil compatibility of the species are multiplied with the percentage of roots covered by that type of ground and added up. Consequently, the growth factor g_g decreases if the roots are partially exposed to air. A plant that has 0 compatibility with sand will thus not grow if the ground entirely consists of sand.

Competition is calculated based on all surrounding plants as

$$g_{c,k} = 1 - \sum_{i,i\neq k} \frac{1}{4} \cdot d_{\mathcal{S}_k \mathcal{S}_i} \cdot \left(\frac{\min(\|\mathcal{V}_{\boldsymbol{x},i} - \mathcal{V}_{\boldsymbol{x},k}\| - (\mathcal{V}_{r,k} + \mathcal{V}_{r,i}), 0)}{\mathcal{V}_{r,k}} \right)^2. \tag{21}$$

This roughly approximates how much the radii of two plants overlap, weighted with the competition relationship between their associated species. A plant that spawns in a region with high competition will grow slowly and potentially not reach maturity.

Lastly, we compute a growth factor for water availability. Using the volume of the roots $V_{r,k} = \frac{2}{3}\pi h_{r,k} \mathcal{V}_{r,k}^3$, stem $V_{s,k} = \frac{2}{3}\pi h_{s,k} \mathcal{V}_{r,k}^3$ and entire plant $V_k = V_{r,k} + V_{s,k}$, we determine water capacity, required water and available ground water as

$$W_{\text{cap}}(k) = w_{c,k} \cdot V_k \tag{22}$$

$$W_{\text{req}}(k) = \Delta t \cdot \phi_w \cdot V_{s,k} \tag{23}$$

$$W_{\text{avail}}(k) = \Delta t \cdot \mathcal{T}_{M,\boldsymbol{i}} \cdot V_{r,k} \tag{24}$$

If $W_{\text{avail}} < W_{\text{req}}$, the remaining water is taken from the plants own storage $\mathcal{V}_w$, otherwise, the plant can increase its water storage with superfluous ground water, up to W_{cap}. The required water models both water used by the plant's cells and water loss due to transpiration. The growth factor g_m is 1 if $W_{\text{req}}(k)$ is satisfied and increases up to 2 if there is an excess of water.

Aside from growth, competition and illumination also impact the maximum radius of a plant, causing plants to be smaller in the presence of competition and bad lightning conditions. The maximum radius $\bar{r}_{\max}(k)$ is further limited by the distance to the bedrock, similar to Equation (15). It is calculated as

$$\min\left(\frac{(\mathcal{V}_{\boldsymbol{x},k})_y - \mathcal{T}_{B,\boldsymbol{i}}}{h_{r,k}}, g_{c,k} \cdot g_{I,k} \cdot r_{\max,k} \right). \tag{25}$$

For underwater plants, we additionally consider the distance to the water surface as a limit.

The plant radius is updated as

$$\mathcal{V}_{r,k}^{t+\Delta t} = \mathcal{V}_{r,k}^{t} + \Delta t \cdot \phi_g \cdot \prod_i g_{i,k}. \tag{26}$$

If the maximum radius is exceeded, growth is set to 0.

3.2.5 Vegetation Health

The purpose of our vegetation health model is to cull plants that are no longer able to survive after the environmental conditions have changed. A plant receives damage if its radius is larger than 110% of the current maximum radius and if the water requirement cannot be satisfied. It is also damaged if illumination is outside the compatible interval and if relative ground moisture or terrain slope exceed a threshold. Lastly, it is damaged if too much of the plant is underground, exposed to air or standing water.

Each of these factors results in a damage value d_i, which is 0 at the border of the allowed range and grows larger the further the conditions deviate from allowed values. If any of these damage values exceeds 0, the plant cannot grow. The health of a plant is updated using

$$\mathcal{V}_{h,k}^{t+\Delta t} = \mathcal{V}_{h,k}^{t} + \Delta t \left(\phi_g \cdot \prod_i g_{i,k} - \phi_d \sum_i d_{i,k} \right). \quad (27)$$

A plant can thus recover its health as it grows. The maximum health is 1 and if the health reaches 0, the plant dies and is removed. Plants that fail the maturity condition have their health set to 0.

Some species are able to shrink, which can be controlled with the shrink rate ϕ_s and only happens if a plant's radius exceeds the current maximum by at least 10%. Similarly, some plants can adjust their position up or down toward the current ground surface, which is set via the species parameter ϕ_p. Plants that can shrink or adjust their position are able to adapt better to changing environments, avoiding damage.

3.2.6 Acceleration Data Structure

Our vegetation model needs to iterate over all plants in the scene at various points, which is not possible in real-time for large plant populations. However, the influence of a plant is 0 if it is far enough away. For the different rasterized quantities, the influence radius of a plant is equivalent to its radius $\mathcal{V}_{r,k}$. The influence for the competition calculation $g_{c,k}$ depends on the sum of the radius $\mathcal{V}_{r,k}$ and the radius of the other plant.

We thus need a suitable acceleration data structure for neighborhood lookups. Uniform grids are a good candidate because they are very efficient to create in parallel on the GPU. Their resolution has to be chosen to be optimal for a fixed search radius, but the plants in our scene have radii varying from 0 to $20m$. This leads to very poor performance in situations where large and small plants coexist.

Our solution to this problem is to use multiple uniform grids at the same time. Each grid covers a different range of radii. We use 4 different scales, with the respective radii being $0-2.5m, 2.5-5m, 5-10m$ and $10-20m$. The cell size of each uniform grid is twice the upper bound. Each plant belongs to exactly one of these grids, which is selected based on its radius. For rasterization of plant quantities, we only need to iterate over the plants in 2×2 cells of each uniform grid. Note that our approach is different from the quadtree used in (do Nascimento et al., 2018), as we use the data structure solely for a neighborhood search, whereas the quadtree in the previous work is used for plant interaction and plants in the same layer do not interact with each other, only affecting the layers below them. In our case, all plants interact with each other.

Competition is more expensive to calculate than rasterization. When computing the competition of a small plant, only a few cells in each grid have to be considered. Large plants have to iterate over a big area in the higher resolution grids. Our data structure is thus less optimal for competition calculation. As there are usually far fewer large plants than small plants and many more terrain cells than plants in a scene, rasterization performance was the more important factor.

Each uniform grid is allocated densely, so using multiple grids increases memory. This is negligible because each successive grid has a significantly smaller resolution. The time required to create multiple uniform grids is almost equivalent to creating a single grid in our implementation. In order to achieve this, we first calculate a key for each plant, where the grid index is encoded in the most significant bits and the least significant bits are set to the cell index inside that grid. We then sort the plants by their keys using radix sort. A single kernel with one thread per plant then compares keys of neighboring plants in order to sparsely fill all uniform grids at once with the respective start and end indices into the list of plants.

3.3 Aeolian Erosion

Our method directly extends the code of (Nilles et al., 2024a), so we refer to the original work for exact details. The object map and echo dunes implementation proposed by (Taylor and Keyser, 2023) was removed, as it was not important to our use case. We will give a brief overview of the method and then highlight the key changes we made to it.

3.3.1 Overview

The method by (Nilles et al., 2024a) is an enhanced real-time implementation of (Paris et al., 2019) using CUDA. It is limited to desertscapes environments and capable of simulating dune formation and propagation. Only a bedrock and sand layer are used in the method and there is no concept of moisture or water. Vegetation is supported using a static density map that is unaffected by changes in elevation during the simulation. A simulation step consists of *wind field calculation, wind shadow calculation, saltation, sand de-*

position/bouncing, abrasion, reptation and *avalanching*.

Wind field calculation takes a time-varying high altitude wind direction $\boldsymbol{w}_a$ and outputs the 2D wind field $\boldsymbol{w}_i$. This is done by first scaling wind strength with terrain height (venturi effects) and then warping the wind direction using the gradient of a set of Gaussian convolutions of the terrain, which was implemented efficiently with cuFFT. Since the original paper in (Nilles et al., 2024a), this has been extended by creating a divergence-free wind field via pressure projection in the frequency domain, which is scheduled to be published after the submission of our paper (Nilles and Günther, 2024).

Wind shadow calculation traces backward against the wind direction for each cell and finds the steepest angle up to a maximum distance. This angle determines a wind shadow value in $[0,1]$. Angles below 10° cause no shadow, angles above 15° cause full shadow.

Saltation is the lifting of sand by the wind, as well as the advection of lifted sand. The amount of sand that is lifted depends on the vegetation density $\mathcal{T}_\mathcal{V}$ and wind shadow, which protect from saltation. Advection was originally done using a forward scheme with atomics, but the divergence-free wind field added later allows for semi-Lagrangian advection that steps backward in the wind direction while conserving mass.

Sand deposition and bouncing happen as part of saltation. After advection, a percentage of lifted sand is deposited, while the remaining sand is considered to have bounced on the terrain and remains lifted to be advected in the next simulation step. The deposition probability increases with wind shadow and vegetation density and is also affected by the ground material.

Bedrock abrasion to sand happens due to sand that bounces on bare bedrock. The strength can be set by the user and is affected by wind speed, vegetation density and bedrock abrasion resistance.

Reptation describes movements of sand on the terrain triggered by sand particles colliding with the ground. (Nilles et al., 2024a) proposed a new method to support this effect which suffered from some artifacts. The current implementation uses an improved version that implements reptation by adaptively reducing the angle of repose based on the amount of deposited and bounced sand (Nilles and Günther, 2024).

Avalanching refers to the stabilization of sand slopes toward the angle of repose, set to 33° with no vegetation and 45° at full vegetation density. This is implemented using an iterative algorithm. Many iterations are necessary for scenes with strong saltation. A single avalanching iteration per simulation step is applied to the bedrock layer, using an angle of 68°.

3.3.2 Our Changes

As our vegetation model rasterizes the vegetation density $\mathcal{T}_\mathcal{V}$ needed for aeolian erosion, no further changes were necessary to support it. We extended the original method with an additional material layer (soil), which is fairly straightforward. The soil layer is avalanched with one iteration per frame, using angles 45° without vegetation and 68° at full vegetation density.

Similar to bedrock, soil can be abraded due to saltation, with a separate strength that can be set by the user. Soil abrasion additionally depends on moisture by interpolating the strength to 0 as $\mathcal{T}_{m,\boldsymbol{i}}$ reaches 50%. If moisture is below a threshold m_dry (2% by default), soil slowly dries out and is transformed into sand:

$$\Delta\mathcal{T}_{e,\boldsymbol{i}} = \phi_\text{dry}\Delta t \cdot (1 - \mathcal{T}_{\mathcal{V},\boldsymbol{i}}) \cdot \max(1 - \frac{\mathcal{T}_{m,\boldsymbol{i}}}{m_\text{dry}}, 0) \cdot e^{-10\cdot\mathcal{T}_{s,\boldsymbol{i}}}, \tag{28}$$

where $\phi_\text{dry} = 0.01$ is the dry erosion rate. Vegetation protects from this and dry erosion strength is reduced by the thickness of the sand above the soil. Equation (28) is also subtracted from the humus layer $\mathcal{T}_{h,\boldsymbol{i}}$.

Sand can transform back into soil using a combination of vegetation, humus and moisture, making the previous processes reversible:

$$\Delta\mathcal{T}_{s,\boldsymbol{i}} = \phi_\text{humus} \cdot \Delta t \cdot \mathcal{T}_{\mathcal{V},\boldsymbol{i}} \cdot \mathcal{T}_{m,\boldsymbol{i}}, \tag{29}$$

where $\phi_\text{humus} = 0.01$ is the humus conversion rate. The humus layer decreases by Equation (29) in the process.

Lastly, we consider the water layer and moisture throughout the saltation, deposition and avalanching process. Sand lifting and abrasion are disabled under water. Additionally, sand can only be lifted if the relative moisture is below 10%, reaching full strength at 0%. In order to avoid sand piling up around bodies of water, the deposition probability is clamped to 1% in cells with standing water and decreases with moisture, reaching 1% of the original values at 10% relative moisture. We slightly increase the angle of repose of soil and sand, reaching a peak at 50% moisture, at which point the angle of repose drastically decreases to almost 0° at 100%. This models grit/regolith, causing the terrain to smooth out under and around water.

3.4 Hydraulic Erosion

We combine several ideas from previous work in our method (Mei et al., 2007; Št'ava et al., 2008; Nilles

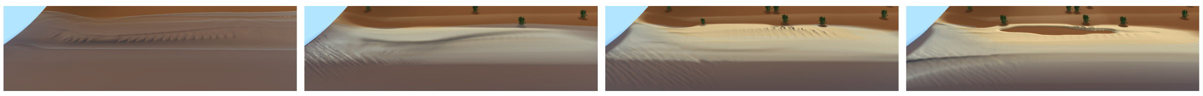
Figure 6: Strong waves erode the terrain and sediment is washed ashore as sand, forming a beach. After the wave strength is reduced, the sand that is further inland begins to dry out and is eventually transported away by the wind (last image).

et al., 2024b). The water layer is simulated using the virtual pipes method, with the GPU implementation proposed in (Mei et al., 2007). Hydraulic erosion largely follows the capacity-based dissolution/deposition model from (Mei et al., 2007), extended to multiple layers as in (Šť'ava et al., 2008).

3.4.1 Overview

Sediment capacity depends on terrain slope and water velocity as in previous work. We additionally decrease the capacity with increasing water depth, as proposed in (Nilles et al., 2024b). This is done because the water velocity calculated in the virtual pipes model describes the surface, while erosion happens at the bottom of each water column. Furthermore, the previous work does not use vegetation. We reduce the capacity to 50% with increasing vegetation density.

Sand, soil and bedrock can be dissolved into sediment, which works via user-defined strengths per material as in the previous work. The deposition of sediment as sand has been modified to account for vegetation and happens at twice the rate at 100% vegetation density. In (Mei et al., 2007), semi-Lagrangian advection was used for sediment due to its simplicity and ease of implementation with GPU texture fetches. We instead use forward advection with atomic adds, because the water velocity field from the virtual pipes method is not divergence-free, causing semi-Lagrangian advection to not conserve mass.

In (Šť'ava et al., 2008), the authors additionally simulate a regolith layer using the shallow water equations. (Nilles et al., 2024b) simplified this effect by modifying the angle of repose with water depth. In our implementation, this effect is instead controlled by the terrain moisture as described previously.

3.4.2 Waves

We modify the virtual pipes method such that the wind field from aeolian erosion can interact with it, generating waves. A sine function is used to create a time-varying wave strength that depends on the wind:

$$f_{\boldsymbol{w}}(\boldsymbol{i}) = \|\boldsymbol{w}_{\boldsymbol{i}}\| \cdot c_{\text{wave}} \cdot \max\left(\sin(\phi_{\text{wave}} \cdot t), 0\right) \tag{30}$$

with wave period ϕ_{wave} and wave strength c_{wave}. We apply exponential decay based on water depth, where wave strength decreases to 0 with decreasing depth. This force is then applied to the outflow flux in the virtual pipes method, using the dot product between outflow direction and wind direction.

3.4.3 Rain and Water Sources

The user can specify a minimum and maximum rain probability $p_{\text{rain}}^{\min}, p_{\text{rain}}^{\max}$, which are interpolated between based on height, where the maximum is used at a height $h_{\text{rain}}^{\max}$ set by the user:

$$p_{\text{rain}}(\boldsymbol{i}) = p_{\text{rain}}^{\min} + (p_{\text{rain}}^{\max} - p_{\text{rain}}^{\min}) \cdot \frac{\mathcal{T}_{W,\boldsymbol{i}}}{h_{\text{rain}}^{\max}}. \tag{31}$$

This probability functions as a threshold for a noise function $\eta(\boldsymbol{i}, t) \in [0, 1]$. The user can control the time and space frequencies of this noise function. If the noise value is less than or equal to the rain probability in a cell, we add $\Delta t \cdot \phi_{\text{rain}}$ to the water level.

A minimum water level can be set for the cells on the border of the terrain, or all cells in the scene. This will prevent the water level $\mathcal{T}_W$ from decreasing below that point and can be used to create oceans or lakes.

3.4.4 Water Seepage and Moisture Diffusion

Surface water in our simulation slowly seeps into the ground, turning into ground moisture. Sand and soil have different seepage rates $\phi_{\text{M}}^{s}, \phi_{\text{M}}^{e}$. The combined seepage rate is determined based on the composition of the terrain up to a depth of $1m$ as

$$\phi_M(\boldsymbol{i}) = \phi_{\text{M}}^{e} + (\phi_{\text{M}}^{s} - \phi_{\text{M}}^{e}) \cdot \min(\mathcal{T}_{s,\boldsymbol{i}}, 1). \tag{32}$$

This rate is multiplied by 0.02 if the relative ground moisture is above 50%. If the absolute moisture exceeds the current capacity $\mathcal{T}_{M_c,\boldsymbol{i}}$, the excess moisture is emitted as surface water. Otherwise, water seeps into the ground:

$$\Delta\mathcal{T}_{M,\boldsymbol{i}} = \min(\phi_M(\boldsymbol{i}) \cdot (\mathcal{T}_{M_c,\boldsymbol{i}} - \mathcal{T}_{M,\boldsymbol{i}}), \mathcal{T}_{w,\boldsymbol{i}}), \tag{33}$$

which is subtracted from the water layer and added to the absolute moisture.

We then apply diffusion to the moisture map, using a single forward iteration of a standard grid-based diffusion algorithm.

3.4.5 Evaporation

(Mei et al., 2007) implemented evaporation as a percentage loss per time step, which behaves inconsistent

Table 2: Timings in *ms* of our simulation at 2048^2 resolution, with a total of $0.25, 0.5, 1$ and 2 million plants. Including visualization, the required GPU memory was 1.6GB for the smallest and 1.8GB for the largest scene. The rows show timings for the full method and select components. In each scene, large trees are combined with small bushes that have no competition and can grow in the shade of trees, which is particularly challenging for our datastructure.

	250k	500k	1m	2m
Full Simulation	16.5	17.5	19.2	23.4
◦ Aeolian	11.7	11.7	11.7	11.6
◦ Sand Aval.	7.6	7.6	7.5	7.4
◦ Hydraulic	2.4	2.4	2.4	2.4
◦ Vegetation	2.4	3.3	5.1	9.3
◦ Data Structure	0.8	1.1	1.6	2.6
◦ Growth	0.7	1.2	2.1	4.6
◦ Raster	0.6	0.8	1.1	1.8
◦ Shadowmap	0.3	0.3	0.3	0.3

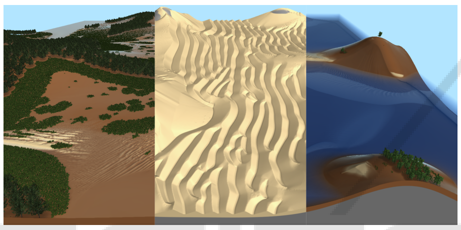
Figure 7: From left to right: wetlands, deserts and islands in an ocean, showcasing possible environments.

with different time steps. We instead use exponential decay as in (Nilles et al., 2024b):

$$\mathcal{T}_{w,\mathbf{i}}^{t+\Delta t} = \mathcal{T}_{w,\mathbf{i}}^{t} \cdot e^{-\phi_w^{\text{evap}} \cdot \Delta t}, \tag{34}$$

where ϕ_{evap} is the user-specified evaporation rate. A separate rate can be specified for ground moisture evaporation, which additionally depends on the ground material (see Equation (32)) and is reduced by vegetation:

$$\mathcal{T}_{M,\mathbf{i}}^{t+\Delta t} = \mathcal{T}_{M,\mathbf{i}}^{t} \cdot e^{-\phi_M^{\text{evap}} \cdot \phi_M(\mathbf{i}) \cdot \Delta t \cdot (1-\frac{3}{4}\mathcal{T}_{\mathcal{V},\mathbf{i}})}. \tag{35}$$

Evaporation of moisture can only happen if no surface water is present in a cell.

4 RESULTS

We include two videos that show our results in the supplementary material, recorded directly during simulation with a simple real-time renderer.

Our method allows to seamlessly transition between aeolian erosion with dune formation and hydraulic erosion by varying rain, water sources and evaporation parameters during the simulation, forming a unified real-time erosion framework, made possible due to the addition of our ground moisture

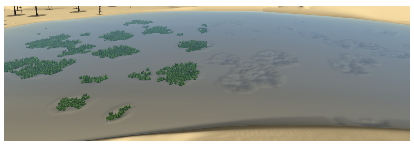
Figure 8: Plants protect against hydraulic erosion and encourage sediment deposition, demonstrated here with underwater plants. Mounds form around plants, similarly to nabhka dunes. The right half has vegetation hidden.

model. Using a noise function with a threshold for rain allows for different parts of the scene to be affected by either type of erosion at the same time. This can be further controlled by varying the rain probability with terrain height, allowing the user to restrict rain to mountains or valleys. We are thus able to support a wide range of scenes, ranging from completely dry deserts to wetlands and even islands in an ocean environment (see Figure 7). Parameters can be adjusted on the fly by the user, enabling transformation of a desert into a lush forest or an underwater environment and back.

By introducing waves due to wind to the hydraulic erosion simulation, it is possible to create beaches, demonstrated in Figure 6. Strong waves erode the terrain and wash sand ashore. If the wave strength is reduced, sand that has been washed further inland does not receive enough moisture from the water anymore, causing it to be transported away by the wind due to aeolian erosion if there is no rain.

The dynamic vegetation model proposed by us improves upon the original static vegetation density model used for aeolian erosion (Paris et al., 2019). All shortcomings of the original approach as mentioned in (Nilles et al., 2024a) have been addressed, the vegetation density now appropriately changes as vegetation is buried or uncovered by sand due to our 3D split-Gaussian model (see Figure 4) which is rasterized to a vegetation density in each simulation step. Additionally, the ideas from (Paris et al., 2019) generalized well to hydraulic erosion, allowing for the equivalent of a nabhka dune forming under water (see Figure 8).

Our vegetation model is coupled bidirectionally to the erosion simulation. Plants lessen the impact of erosion and are able to transform landscapes eroded to just sand back into earthen environments. This in turn affects vegetation growth, allowing for different species to grow as the terrain is transformed. The process is very flexible, as the user can freely design the different vegetation species. While there are a total of 26 parameters that define a species, the parameters are intuitive since most of them directly relate to the conditions they can survive in. To achieve the scene in Figure 1, a total of 4 species were created. Palm trees were configured to be resilient to being covered by terrain due to their leaves being placed very high up, were set to require low amounts of water and moisture, as well as a preference for sand. This results in palm trees spawning in a desert environment next to water sources, forming an Oasis. Seaweed was set up as an underwater plant and begins to fill the scene as the terrain is flooded, transforming sand to soil in the process. After reducing the amount of rain significantly, water bodies evaporate again, causing exposed seaweed to wither. The low amount of rain seeps into the ground as moisture instead of accumulating as surface water, enabling the growth of bushes across the entire scene which require more water and moisture. As the bushes transform sand into soil, trees that require soil can eventually spawn and displace bushes due to competition, forming forests as they spread in the wind direction. Disabling rain causes another mass extinction event and slowly returns the scene to a desert as soil dries out and is abraded by aeolian erosion.

The volumetric illumination model proposed by us is very efficient to calculate and enables us to further diversify the possible plant species. Combined with the competition model, we can create plants that only grow in the shade of larger plants or plants that only grow in the shaded areas of the terrain, but not in the presence of another species. Together with the other environmental parameters, it is possible to have a wide variety of species in the same scene that each inhabit their own ecological niche (see Figure 5). We additionally make use of our illumination model while visualizing the simulation, which enhances depth perception and replaces traditional shadow mapping and ambient occlusion techniques for free.

We evaluate the performance of our simulation on a RTX 4080 GPU at a terrain resolution of 2048^2 with varying number of plants (see Table 2). For each measurement, we imposed an upper limit on the number of plants, waited until the number of plants reached this maximum and averaged the next 10,000.00 simulation steps. The scene contains large trees combined with small bushes that are set to grow in the shade of larger trees with no competition. This mixture of densely placed, overlapping plants with high size difference is particularly demanding, which is why we chose it to test performance.

The performance of aeolian and hydraulic erosion is unaffected by the number of plants, with aeolian erosion requiring about $11.7ms$ per simulation step. Hydraulic erosion is very fast at only $2.4ms$. Aeolian erosion is more demanding as it involves multiple fourier transformations, but the main reason is sand avalanching as in previous work (Nilles et al., 2024a). We used 50 sand avalanching iterations per simulation step, requiring around $7.5ms$. This many iterations are only necessary in parts of the scene where dunes are forming, which was not the case in our scene. An adaptive approach would thus be very beneficial.

Vegetation computation time grows slower relative to the number of plants. At two million plants, it requires $9.3ms$ per simulation step, bringing the total method to $23.4ms$ which is still real-time. The shadow map creation is independent of the number of plants and only takes $0.3ms$. Vegetation growth is the most expensive and the only part that grows faster relative to the number of plants at vegetation count above one million, requiring $4.6ms$ for two million plants, followed by data structure creation with $2.6ms$ and vegetation rasterization with $1.8ms$. This is expected as the data structure was optimized for rasterization. Increasing the number of plants much further would quickly become limited by vegetation growth calculation with our current data structure, indicating that even more plants are possible in real-time if this is improved.

5 CONCLUSION AND FUTURE WORK

In conclusion, we successfully integrated aeolian erosion, hydraulic erosion, and vegetation simulation into a single method capable of real-time performance at resolutions of 2048^2 with two million plants, made possible by our acceleration data structure. The different components were tied together by adding a ground moisture level and our split-Gaussian vegetation density rasterization. Our vegetation model supports multiple plant species, including underwater vegetation, which can be configured alongside other parameters like rain and water sources. Each species is defined by simple parameters based on environmental compatibility, paired with a basic volumetric illumination calculation. It is possible to interactively transform between entirely different environments, such as deserts, underwater landscapes and lush forests, while observing changes in real-time.

For future work, we would like to further improve the vegetation data structure to allow for even larger numbers of plants in real-time. Additionally, the sand avalanching step from aeolian erosion still has a high cost, as in the previous work. Performance could be increased by adaptively reducing the number of avalanching steps used, as only dry deserts require many iterations. Alternatively, it would be worth investigating a machine learning solution to avalanching. A neural network could be trained by using the current avalanching implementation to produce a ground truth.

We have not yet incorporated temperature into our simulation, which is another important factor for vegetation growth. In order to support the full range of real-world temperatures, we would like to combine this with snow and ice simulations, potentially accounting for thermal erosion. In particular, the aeolian erosion framework seems well-suited to be adapted for snow dune simulation. Using real-world elevation and weather data or alternatively, a full weather simulation are other avenues worth exploring. In a similar manner, further weather effects such as lightning strikes, as well as forest fires as implemented in (Cordonnier et al., 2017) are not yet considered.

Another limitation of our model is the high number of parameters, which make interaction more complex for an artist in the current state. As pointed out by a reviewer, we think that further work should identify meaningful presets and organize parameters into main parameters as well as less important ones for fine-tuning. Another possibility would be developing a set of more intuitive meta-parameters that control the current parameters behind the scenes.

Lastly, there is a wide array of research available with more realistic vegetation models. As our expertise is in erosion simulations and our goal was to combine multiple different simulations into a single real-time implementation, we chose to leave this additional complexity out and developed our own simple method, allowing us to freely design the vegetation model to suit the needs of the erosion simulation. Incorporating the state of the art in vegetation simulations is thus left for future work.

ACKNOWLEDGEMENTS

The textures and meshes used for trees, bushes and seaweed are from Sketchfab users (evan4129, 2024; OwenCalingasan, 2024) and licensed as CC BY 4.0 (Creative Commons, 2024).

REFERENCES

Alsweis, M. and Deussen, O. (2006). Wang-tiles for the simulation and visualization of plant competition. In Nishita, T., Peng, Q., and Seidel, H.-P., editors, *Advances in Computer Graphics*, pages 1–11, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benes, B. (2007). Real-Time Erosion Using Shallow Water Simulation. In Dingliana, J. and Ganovelli, F., editors, *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2007)*. The Eurographics Association.

Berger, U., Hildenbrandt, H., and Grimm, V. (2002). Towards a standard for the individual-based modeling of plant populations: self-thinning and the field-of-neighborhood approach. *Natural Resource Modeling*, 15(1):39–54.

Ch'ng, E. (2011). Realistic placement of plants for virtual environments. *IEEE Computer Graphics and Applications*, 31(4):66–77.

Cordonnier, G., Galin, E., Gain, J., Benes, B., Guérin, E., Peytavie, A., and Cani, M.-P. (2017). Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. Graph.*, 36(4).

Creative Commons (2024). CC BY 4.0 Attribution 4.0 International. https://creativecommons.org/licenses/by/4.0/.

do Nascimento, B. T., Franzin, F. P., and Pozzer, C. T. (2018). Gpu-based real-time procedural distribution of vegetation on large-scale virtual terrains. In *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 157–15709.

evan4129 (2024). LOD/Billboard Summer Trees Pack, Trees and bush Pack LOWPOLY, Palm Tree Pack LOWPOLY. https://sketchfab.com/evan4129.

Gain, J., Long, H., Cordonnier, G., and Cani, M.-P. (2017). Ecobrush: Interactive control of visually consistent large-scale ecosystems. *Computer Graphics Forum*, 36(2):63–73.

Hammes, J. (2001). Modeling of ecosystems as a data source for real-time terrain rendering. In Westort, C. Y., editor, *Digital Earth Moving*, pages 98–111, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hartley, M., Mellado, N., Fiorio, C., and Faraj, N. (2024). Flexible terrain erosion. *The Visual Computer*.

Hawkins, B. and Ricks, B. (2023). Improving virtual pipes model of hydraulic and thermal erosion with vegetation considerations. *The Visual Computer*, 39(7):2835–2846.

Krištof, P., Beneš, B., Křivánek, J., and Šťava, O. (2009). Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2):219–228.

Lü, P., Dong, Z., and Rozier, O. (2018). The Combined Effect of Sediment Availability and Wind Regime on the Morphology of Aeolian Sand Dunes. *Journal of Geophysical Research: Earth Surface*, 123(11):2878–2886.

Mei, X., Decaudin, P., and Hu, B.-G. (2007). Fast Hydraulic Erosion Simulation and Visualization on GPU. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 47–56.

Nilles, A. M. and Günther, L. (2024). CUDA Dune Simulation. https://github.com/Clocktown/CUDA-Dune-Simulation.

Nilles, A. M. and Günther, L. (2025). Oasis. https://github.com/Clocktown/Oasis/tree/GRAPP2025.

Nilles, A. M., Günther, L., and Müller, S. (2024a). Real-Time Desertscapes Simulation with CUDA. In *Proceedings of the 19th International Joint Conference* *on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*, pages 34–45. INSTICC, SciTePress.

Nilles, A. M., Günther, L., Wagner, T., and Müller, S. (2024b). 3D Real-Time Hydraulic Erosion Simulation using Multi-Layered Heightmaps. In Linsen, L. and Thies, J., editors, *Vision, Modeling, and Visualization*. The Eurographics Association.

O'Brien, J. and Hodgins, J. (1995). Dynamic simulation of splashing fluids. In *Proceedings Computer Animation'95*, pages 198–205.

OwenCalingasan (2024). Seaweed. https://sketchfab.com/OwenCalingasan.

Pałubicki, W., Makowski, M., Gajda, W., Hädrich, T., Michels, D. L., and Pirk, S. (2022). Ecoclimates: climate-response modeling of vegetation. *ACM Trans. Graph.*, 41(4).

Paris, A., Peytavie, A., Guérin, E., Argudo, O., and Galin, E. (2019). Desertscape Simulation. *Computer Graphics Forum*, 38(7):47–55.

Taylor, B. and Keyser, J. (2023). Real-Time Sand Dune Simulation. *Proc. ACM Comput. Graph. Interact. Tech.*, 6(1).

Tsoar, H. (1983). Wind Tunnel Modeling of Echo and Climbing Dunes. In Brookfield, M. and Ahlbrandt, T., editors, *Eolian Sediments and Processes*, volume 38 of *Developments in Sedimentology*, pages 247–259. Elsevier.

Šťava, O., Beneš, B., Brisbin, M., and Křivánek, J. (2008). Interactive terrain modeling using hydraulic erosion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, page 201–210, Goslar, DEU. Eurographics Association.

Weier, M., Hinkenjann, A., Demme, G., and Slusallek, P. (2013). Generating and rendering large scale tiled plant populations. 10(1).