EUROGRAPHICS 2025 / M. Christie, N. Pietroni, and Y.-S. Wang
(Guest Editors)

Volume 44 (2025), Number 7

Procedural and learning-based generation of coral reef islands

Submission1265

Figure 1: *From top-view and profile-view sketches, our system procedurally generates coral reef islands and trains a cGAN to produce diverse and realistic terrains.*

Abstract

We propose a procedural method for generating single volcanic islands with coral reefs using user sketching from two projections: a top view, which defines the island's shape, and a profile view, which outlines its elevation. These projections, commonly used in geological and remote sensing domains, are complemented by a user-defined wind field, applied as a distortion field to deform the island's shape, mimicking the effects of wind and waves on the long term and enabling finer user control. We then model the growth of coral on the island and its surronding to construct the reef following biological observations. Based on these inputs, our method generates a height field of the island. Our method is capable of creating a large variety of island models composing a dataset used for training a conditional Generative Adversarial Network (cGAN). By applying data augmentation, the cGAN allows for even greater variety in the generated islands, providing users with higher freedom and intuitive controls over the shape and structure of the final output.

Keywords: Procedural modeling, Terrain generation, cGAN, Coral reef, Sketch-based interface

1. Introduction

Simulating the formation of coral reef islands presents significant challenges due to the complex interplay of geological, environmental, and biological factors (Hopley, 2014). One major difficulty lies in capturing the long-term subsidence of volcanic islands, which occurs over millions of years, while simultaneously modeling the upward growth of coral reefs that rely on environmental conditions such as water depth, temperature, and sunlight. This combination of slow geological processes and dynamic biological growth is challenging to replicate in a computational model.

Additionally, the biological aspects of coral growth are inherently tied to environmental factors. Coral reefs grow only within a specific range of water depth and sunlight, and their growth patterns are affected by the health of the reef ecosystem and the availability of resources. Accurately modeling these biological dependencies in a procedural system is challenging, as these factors are numerous and difficult to generalize. Moreover, the scarcity of data available obstructs the global understanding of these ecosystems. In a recent high-resolution mapping of shallow coral reefs (Lyons et al., 2024), researchers estimated the total surface area of this biome to cover less than 0.7% of Earth's area, and more specifically that coral habitat represents less than 0.2%.

Moreover, the need to design plausible reef structures in digital environments for applications such as scientific visualization, simulation, or digital entertainment, calls for models that balance realism with controllability. Existing terrain generation methods, such as Perlin noise-based algorithms or uplift-erosion models, are often ill-suited for these processes. While they can generate natural-looking landscapes (such as alpine landscape, representing about a quarter of land area (Körner et al., 2014)), they do not account for the unique geological and biological interactions that govern coral reef island formation, thus missing coherency. Capturing these dynamics, while also providing user control during the modeling of a terrain, requires a balance between realism and flexibility, allowing for both accurate computationally expensive simulation of natural processes and intuitive user control in interactive time.

Despite advances in terrain generation, existing methods struggle with user-controlled design of specific island shapes and achieving realism without real data. Coral reef islands exemplify this gap: we lack datasets to directly train deep models, and purely procedural methods require expert tuning to mimic their features.

To address these issues, we propose to use curve-based modeling techniques as an initial step in our approach as a means to efficiently create a large and diverse set of training examples for a learning-based model. Each synthetic example is represented by a terrain height field and a corresponding semantic label map that marks different regions, providing structured input-output pairs for the learning stage.

We trained a conditional Generative Adversarial Network (cGAN) as the core of our learning-based approach (Mirza and Osindero, 2014; Isola et al., 2017). A cGAN is a type of deep learning model that learns to generate realistic data based on an input condition or context. In our case, the cGAN takes as input the semantic label map of an island generated by the procedural step and learns to produce a realistic island height field that matches this layout. By training on a large variety of examples from our curve-based algorithm, the cGAN captures the subtle terrain features and variations characteristic of coral reef islands, this approach surpasses the capabilities of traditional procedural rules, aided by extensive data augmentation. The cGAN model can be used on its own to generate new island terrains with simplified and more intuitive user inputs through digital drawing, and the model will generate a realistic island terrain accordingly.

The key contributions are as follows: 1) a novel curve-based procedural algorithm for shaping island terrains from top and profile views, 2) The training of a deep learning model on synthetic data derived from procedural rules, serving as an abstraction layer that hides underlying complexity, 3) A demonstration that the cGAN approach tolerates imprecise, low-detail user input sketches, broadening usability, without the need for cutting-edge network architectures, and 4) An insight that procedural generation remains essential to produce training data in data-sparse domains such as coral reef islands. These contributions collectively show a pathway to blend user-driven design with learning-based generation in terrain modeling.

2. Related work

Procedural terrain generation spans a spectrum from purely noise-driven algorithms to physics-based simulations and, more recently, data-driven methods. In the context of coral reef islands, where both long-term geological subsidence and biogenic reef accretion interplay, existing approaches fall short in one of two ways: either they lack ecological or geological grounding, or they offer insufficient authoring control.

Procedural and simulation-based methods Noise-based techniques such as Perlin noise (Perlin, 1985), Simplex noise (Perlin, 2001), and the Diamond-Square algorithm (Fournier et al., 1982) (often extended via fBm or multifractal noise (Musgrave et al., 1989; Ebert et al., 2003)) remain popular for their simplicity and speed. Island shapes are generated by modulating noise with radial falloff masks (Olsen, 2004), but these methods cannot reproduce reef rings, lagoons, or atoll structures in a geologically coherent manner. They treat terrain purely as a signal-processing problem, separated from processes like volcanic subsidence or coral growth (Smelik et al., 2009; Galin et al., 2019).

Simulation-based approaches introduce surface deformations by modeling erosion (Beneš et al., 2006; Neidhold et al., 2005; Mei et al., 2007), tectonic uplift (Cordonnier, Braun, et al., 2016; Cordonnier, Cani, et al., 2017; Schott et al., 2023), or vegetation-terrain feedback (Ecormier-Nocca et al., 2021; Cordonnier, Galin, et al., 2017). Hydraulic and thermal erosion capture fluvial networks and slope-driven mass wasting, but they omit underwater sedimentation and biogenic carbonate accretion. Tectonic and isostatic models excel at orogeny but ignore coral reef dynamics, while vegetation-based methods do not generalize to marine ecosystems. Consequently, none of these simulations jointly capture the slow subsidence of volcanic islands and the compensatory growth of surrounding reefs on the timescales required for atoll formation.

Sketch-based terrain modeling Sketch-driven interfaces bridge user intent and procedural detail. Curve-based systems let users draw ridges, valleys, or coastlines that guide surface deformation and noise propagation (Gain et al., 2009; Hnaidi et al., 2010). Constraint-based methods extend this by enforcing absolute elevation or slope values at control curves or points, solved via diffusion or fractal interpolation (Gasch et al., 2020; Talgorn and Belhadj, 2018), and even gradient-domain editing for slope control (Guérin, Peytavie, et al., 2022). Semantic approaches encode high-level "terrain atoms" from a dictionary of primitives (Génevaux et al., 2015) or interpret geological schematics into 3D models (Natali et al., 2012). While these techniques grant artists fine-grained control, they typically lack ecological constraints and have not been tailored to marine biogeomorphology.

Learning-based terrain synthesis Deep generative models offer a way to learn complex patterns without explicit procedural rules. Unconditional GANs have been applied to digital elevation maps of mountains (Wulff-Jensen et al., 2018) and joint height-texture synthesis (Spick and Walker, 2019), but their reliance on latent noise prevents precise layout control. Two-stage pipelines use an initial GAN for heightmaps and a conditional GAN for textures (Beckham and Pal, 2017) or reverse the order (imagery-to-DEM) (Panagiotou and Charou, 2020), yet still lack the ability to guide authoring.

Conditional GANs (cGANs) extend image-to-image translation methods such as pix2pix (Isola et al., 2017) to terrain, enabling sketch- or label-map-conditioned generation. Prior work includes sketch-to-DEM translation for generic landforms (Guérin, Digne, Galin, Peytavie, et al., 2017) and sparse "altitude dot" conditioning (Voulgaris et al., 2021), as well as cGANs that invert satellite imagery into elevation (Sisodia, 2022). However, these models require extensive paired real-world datasets which are scarce for coral reef islands.

3. Overview

Our method for procedural generation of coral reef islands is composed of two independent modeling techniques shown in Figure 2. A first curve-based algorithm (top-left blue block, presented in Section 4), parametrize the surface of islands by a top-view sketch in-

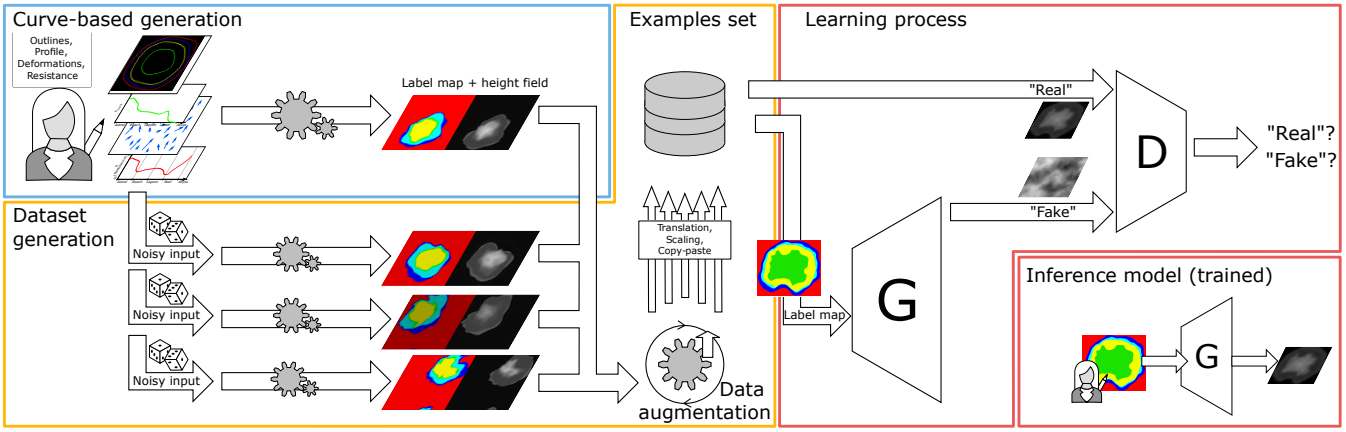

Figure 2: *Our pipeline first prompt our user to create single pairs of coral reef islands' height fields and label maps using our proposed curve-based modeling algorithm (Section 4). The same algorithm is applied multiple times on randomly altered versions of the user input to create a dataset, which we further enhance with data augmentation techniques Section 5. Finally, we train a conditional Generative Adversarial Network which can then be used standalone to create height fields from labeled sketches (Section 6).*

terface, describing the outlines of the regions composing it and a stroke-based wind field deforming them, as well as a profile-view sketch describing for each region its altitude and resistance to deformations. User inputs are given as 1D functions (altitude and resistance), a 1D polar function (island outlines) and a 2D vector field (wind field) as shown in Figure 3, which are convient to randomize through the use of noise, but hindering users' modeling freedom.

We propose a second learning-based generation algorithm (right red blocks Figure 2, developed in Section 6), which trains a neural generator G to transform a labeled image into a height field. Parallely, a discriminator D is train to distinguish height fields provided from the training set against height fields generated by G. This adversarial training strengthen the outputs of the generator, up to the point where we discard the discriminator and training data, only keeping the generation step (bottom-right). Users are then able to provide unseen label maps and obtain height fields as close as possible to the training dataset. This method for terrain modeling is poorly constraint, however requires a large amount of prior data in its training set, which are not available in our case.

We connect these two algorithms through a process of dataset generation (central orange block Figure 2, described in Section 5) in order to enforce the benefits of each while reducing their limitations. Given the initial curve-based user inputs, we create a dataset composed of similar samples processed by our first algorithm, rasterized into a 2D image of the parametrized regions, and paired with the resulting height field. We then enhance greatly the size of the dataset by employing various data augmentation techniques such as translations, scaling and copy-pasting of multiple islands in a single sample. We then obtain a dataset large enough for training our cGAN and enable the modeler to create procedurally coral reef islands with a high level of control.

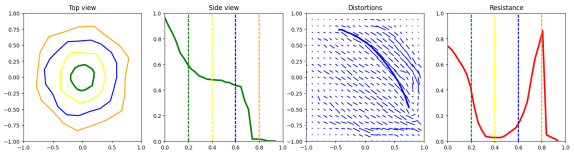

Figure 3: *The user can interact directly on the island by editing the different canvases in no specific order. This UI shows, from left to right, the top-view sketch with the different outlines of each regions, the profile-view sketch with the outlines represented in dotted lines, the wind velocity sketch drawn with strokes (last stroke is visible), and the resistance function showing here a high resistance at the top of the island and on the front reef.*

4. Curve-based island generation

The generation of coral reef island terrains involves a structured process that takes the user's sketches and produces a complete 3D terrain model. This process begins with the creation of the initial height field based on the user's input, followed by the application of wind deformation to introduce natural variations, and concludes with the integration of coral reef features through subsidence and coral growth modeling.

The generation of coral reef islands in our system begins with two intuitive curve-based inputs from the user: a top-view sketch and a profile-view sketch, which define the islands horizontal layout and vertical elevation profile. In addition to these sketches, the user can further refine the terrain by applying wind deformation strokes, which simulate the effects of wind and waves on the islands shape. This combination of sketches and wind inputs gives users precise control over both the islands structure and its natural variations, such as irregular coastlines or concave features. We will present the usefulness of these sketches in this section, and describe the technical details in the next section.

4.1. Initial height field generation

The top-view sketch defines the island's outline as seen from above. Using a simple drawing interface, the user delineates concentric boundaries for key regions such as the island itself, the beaches, the lagoon, and the surrounding abyss, around the canvas center. Each boundary is represented in polar coordinates, where $r_{\mathbf{p}}$ is the radial distance from the island's center and $\theta_{\mathbf{p}}$ is the angular position. Allowing r to vary with θ introduces irregular, natural shapes rather than perfect circles (Figure 4).

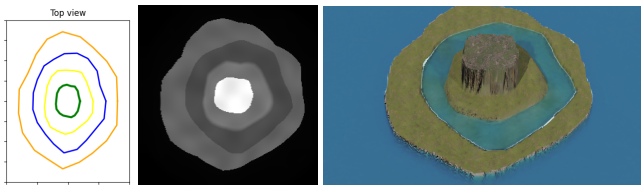

Figure 4: *Using only the outlines from the top-view sketch, each point in the field is assigned a region (island, beach, lagoon, abyss), which later guides its height assignment.*

The profile-view sketch defines the island's vertical elevation along any radial direction. Here, the user draws a curve that specifies height at key terrain milestones, such as the central peak, island border, beach, lagoon, and abyss, creating a continuous 1D height function $h_{\text{profile}}(\tilde{x})$ where $\tilde{x}$ is a parametric distance measuring position along the sequence of regions. This continuous profile ensures smooth elevation transitions across all terrain features.

By combining the top-view and profile-view sketches via revolution modeling, the system generates a full 3D terrain model that matches the user's design. The process begins by transforming those two sketches into a coherent height field (Figure 6).

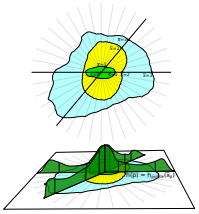

Figure 5: *Parametric distance $\tilde{x}$ depending on angle θ.*

For any point $\mathbf{p}$ on the terrain, the system computes polar coordinates $(r_{\mathbf{p}}, \theta_{\mathbf{p}})$. The radial distance $r_{\mathbf{p}}$ determines which region the point belongs to (island, beach, lagoon, reef, or abyss) based on the user-defined radial limits. Those outlines from the top-view sketch provide the exact boundaries between regions.

Each point's height is computed using the profile function $h_{\text{profile}}(\tilde{x})$. Instead of using the raw radial distance $r_{\mathbf{p}}$, we define a parametric region distance $\tilde{x}_{\mathbf{p}}$ that maps each point to a normalized position along the concentric regions (see Figure 5). The radial space is divided by user-defined boundaries $R_0, R_1, \ldots, R_n$, corresponding to the island center, border, beach, lagoon, and abyss.

When a point $\mathbf{p}$ lies between two boundaries, say R_i and R_{i+1}, its parametric distance is

$$\tilde{x}_{\mathbf{p}} = i + \frac{r_{\mathbf{p}} - R_i}{R_{i+1} - R_i}, \tag{1}$$

where i is the index of the region containing $\mathbf{p}$ (i.e., $R_i \leq r_{\mathbf{p}} < R_{i+1}$). This linear mapping stretches each region's radial span to the interval $[i, i+1]$, ensuring smooth interpolation across region boundaries. The final height at point $\mathbf{p}$ is then $h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$.

4.1.1. Wind deformation

To break the radial symmetry inherent to our curve-based terrain generation and introduce more organic island shapes, we allow the user to define a wind velocity field via freehand strokes on a 2D canvas. Each stroke is represented as a parametric curve C, interpreted as a local wind flow with direction C', strength S, and influence width σ. These strokes simulate wind and wave erosion effects on the terrain.

The deformation vector at any terrain point $\mathbf{p}$ is computed as a sum over all strokes:

$$\Phi(\mathbf{p}) = \sum_{C \in \text{curves}} S \frac{C'(\mathbf{q})}{\|C'(\mathbf{q})\|} \cdot G_\sigma\left(\|\mathbf{p} - \mathbf{p}_C^*\|\right) \tag{2}$$

weighted by a Gaussian falloff centered on the closest point $\mathbf{p}_C^*$ along each curve (Figure 7, top):

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}. \tag{3}$$

To preserve semantic structure across terrain regions, a resistance function $\rho(\tilde{x})$ modulates the deformation based on terrain zones such as beach, lagoon, or abyss (Figure 8). The final deformation vector becomes:

$$\tilde{\Phi}(\mathbf{p}) = (1 - \rho(\tilde{x}_{\mathbf{p}})) \cdot \Phi(\mathbf{p}). \tag{4}$$

This warp is applied to both the height field and the label map, ensuring consistent semantic deformation. For example, applying strokes to one side of a circular island creates concave coastlines

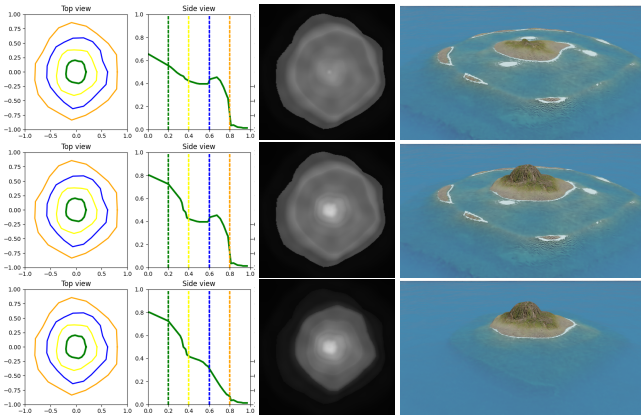

Figure 6: *Providing a smooth function between each region results in islands with plausible reliefs. We fixed the outlines while editing only the height function in order to produce, from top to bottom, a low island, a coral reef island, and finally an identical island without the reef.*

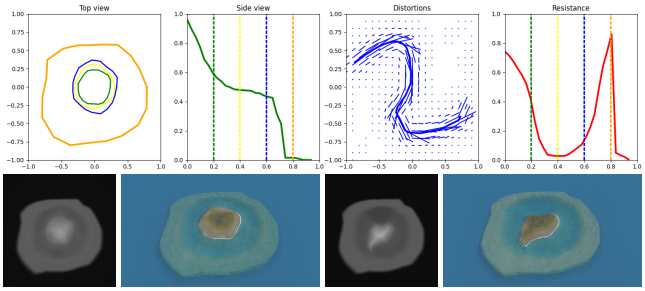

Figure 7: *Defining a top-view wind vector field from user-provided strokes (top, blue) in association with a resistance function (top, red), a height field is deformed accordingly. Left: original height field and render; right: altered results. The beach and lagoon regions are defined with low resistance, which is visible by having only these regions deformed in bottom results.*

while leaving high-resistance regions (e.g., the abyss) unaffected, simulating the localized impact of natural erosion (Figure 7, bottom).

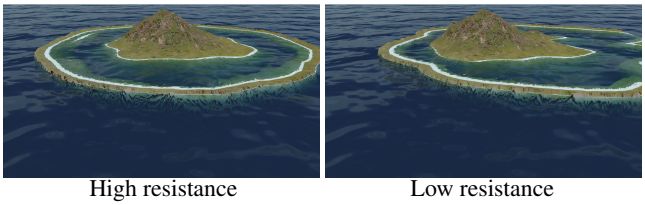

Figure 8: *(Left) An island defined with a high resistance, under a uniform lateral wind velocity field, and (right) the same island with lower resistance on the reef borders, simulating the effect of coastal erosion.*

4.2. Coral reef modeling

Once the terrain has been generated and deformed by the wind, we simulate the long-term geological evolution of coral reef islands through two parallel processes: the subsidence of the volcanic island and the upward growth of coral reefs. As observed in nature, the volcanic base sinks over time while coral formations grow vertically to remain close to the water surface, following the "keep-up" strategy of reef development.

4.2.1. Subsidence

Subsidence is modeled by uniformly scaling the original terrain height downward, simulating the gradual sinking of the volcanic landmass due to tectonic processes. The user specifies a subsidence rate $\lambda \in [0,1]$, which controls how much the island has sunk. The subsided terrain is computed as:

$$h_{\text{subsid}}(\mathbf{p}) = (1-\lambda)\cdot h_0(\mathbf{p}). \tag{5}$$

This factor is applied uniformly across the island, offering a geologically plausible and computationally efficient approximation of large-scale subsidence.

4.2.2. Coral reef growth

Coral reef growth is modeled independently from the subsiding terrain. The system generates a coral-specific height field $h_{\text{coral}}(\mathbf{p})$ that remains near the sea surface regardless of the island's vertical shift, reflecting coral growth in biologically viable depth ranges (typically 0-30 meters below sea level).

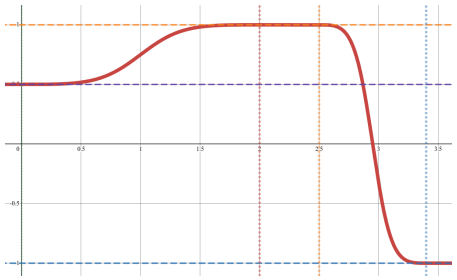

Figure 9: *The modeling of the reef growth in our model is described by a piecewise function h_{coral} which is flat in the lagoon, the crest and abyss, and follows a smoothstep function as transitions for the backreef and fore reef regions. Zones' anchor heights are represented by horizontal dashed lines; zones' limits are dotted vertical lines.*

We define distinct reef zones anchored at specific depths:

- Reef crest near sea level: $h_{\text{crest}} = -2\,\text{m}$,
- Back reef and lagoon: $h_{\text{back}} = -20\,\text{m}$,
- Fore reef sloping to abyss: $h_{\text{abyss}} = -100\,\text{m}$.

Each reef subregion is defined over a parametric domain $x \in [0,1]$, with $x = 0$ the begining of the reef region and $x = 1$ its end, directly inputing the parametric distance $x = \tilde{x} - i_{\text{reef region}}$. For instance:

- Back reef: $x_{\text{back,start}} = 0$, $x_{\text{back,end}} = 0.5$,
- Reef crest: $x_{\text{crest,start}} = 0.75$, $x_{\text{crest,end}} = 0.8$,
- Abyss begins at $x_{\text{abyss,start}} = 1$.

We model transition zones between these regions using a smoothstep operator:

$$\text{smoothstep}(x) = 3x^2 - 2x^3. \tag{6}$$

We denote the interpolating function as:

$$S(a,b,x_0,x_1,x) = a + (b-a)\,\text{smoothstep}\left(\frac{x-x_0}{x_1-x_0}\right). \tag{7}$$

The complete coral height field, as displayed in Figure 9, is built

as a piecewise function:

$$h_{\text{coral}}(x) = \sum_{r \in \text{subregions}} \begin{cases} h_r & \text{if } x_{\text{r,start}} \le x \le x_{\text{r,end}} \\ 0 & \text{otherwise} \end{cases} + \sum_{t \in \text{transitions}} \begin{cases} S(h_t, h_{t+1}, x_{\text{t,end}}, x_{\text{t+1,start}}, x) & \text{if } x_{\text{t,end}} < x < x_{\text{t+1,start}} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

4.2.3. Output

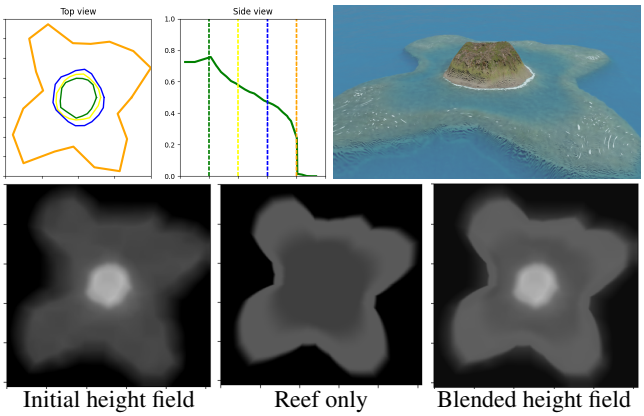

Figure 10: *Volcano with single vent. (Bottom left) The initial height field is computed directly from the user input, (bottom center) the reef height field is outputed from Equation* (8)*, and finally, (bottom right) we blend the two results with Equation* (9)*.*

Finally, we merge the island base height field and the coral reef height field by our *ad-hoc* smooth maximum operator smax defined as:

$$\text{smax}(a,b) = \begin{cases} a + \dfrac{\delta}{2k}\left(\dfrac{1}{1+e^{-k\delta}} + \dfrac{1}{1-e^{-k\delta}}\right) & \text{for } a \neq b \\ a + \dfrac{1}{2k} & \text{for } a = b \end{cases} \tag{9}$$

with $\delta = b - a$ for conciseness, and k a sharpness parameter approximating the max operator as k grows. At $k = 5$, the operator max is already well approximated while conserving continuousness in the resulting height field $\mathcal{H}(p) = \text{smax}(h_{\text{subsid}}(\mathbf{p}), h_{\text{coral}}(\mathbf{p}))$.

The resulting terrain represents a plausible coral reef island, where the volcanic island has subsided, and coral reefs have grown upward to keep pace with the water level (Figure 10). The smooth blending between the subsided terrain and the coral features ensures a natural transition between regions like the island, lagoon, and coral reefs.

5. Data generation

The creation of the dataset is done through the use of the procedural curve-based modeling algorithm for which we alter the input parameters. For each generation, the top-view and profile-view sketches use an initial layout. Each outline of the top-view sketch is defined as a centered circle of random radius $r_{\min} \le r^* \le r_{\max}$. We add another deformation based on fBm noise η such that the final contour, profile, and resistances, are defined as

$$r(\theta) = r^* + \eta_{\text{contours}}(\theta)$$
$$h_{\text{profile}}(\tilde{x}) = h^*_{\text{profile}}(\tilde{x}) \cdot \eta_{\text{profile}}(\tilde{x})$$
$$\rho(\tilde{x}) = \rho^*(\tilde{x}) \cdot \eta_{\text{resistance}}(\tilde{x})$$

Finally, a wind velocity field is randomly generated as a final input for the curve-based modeling algorithm, introducing deformations on the resulting islands. The realistic nature of wind is ignored for the generation of the wind strokes in order to provide complexity and variety in the results. We generate a random number n of strokes and their path by a uniformly sampling a random number m of points. The spread and intensity of each stroke is also random.

Once all inputs are set, we generate an example for multiple level of subsidence $\lambda \in [0,1]$ to obtain a height field incorporating the coral reef modeling and the associated label map.

We use of the Hue component to encode the labels directly from the parametric distance $H = \lfloor \tilde{x} \rfloor$ while the subsidence rate into the Value component $V = \lambda$. Moreover, we purposefully left the Saturation component unchanged at this stage, reserving space for potentially including another parameter in the future.

Data augmentation

To enhance the variety of the dataset and improve the model's ability to generalize, we apply several data augmentation techniques in addition of the usual affine transformations (rotation, scaling, and flipping):

Translation: Since the original algorithm always centers the island, we translate the islands within the image to remove this constraint (Figure 11, leftmost). This ensures that the cGAN can generate islands in any position within the frame. By wrapping the island on the borders, the model learns that data can appear on the edges on the image.

Copy-paste: In some cases, we combine multiple islands into a single sample, with only a maximum intersection over union (IoU) of 10%, allowing the abysses to overlap but without obtaining other region types to merge. Height fields blending is done through the smooth maximum function from Equation (9), and the label blending uses the usual max operator. The regions not covered by any island are assigned the abyss ID, which is very close to the minimal height, meaning that in this case specifically, the subsidence factor (Value channel) is close to irrelevant (Figure 11, rightmost).

All augmentation techniques are applied both to the height field

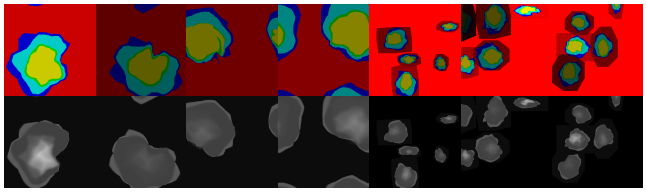

Figure 11: *Examples from a dataset with increasing data augmentation.*

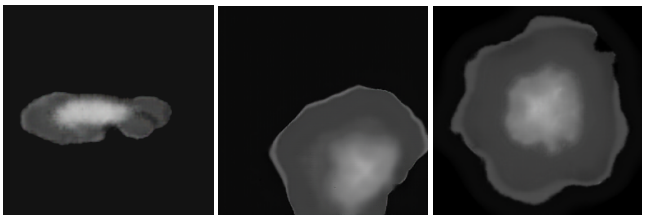

Figure 12: *After the first epoch (left), grid artifacts similar to low resolution image are visible, but are being corrected during a second epoch (middle) where erosion patterns appear in the height fields (right).*

and the label map simultaneously to ensure consistency between the input (the label map) and the output (the height field).

6. Learning-based generation

In this section, we introduce the use of a conditional Generative Adversarial Network (cGAN), specifically the pix2pix model proposed by Isola et al., 2017, to enhance the island generation process by increasing the variety and flexibility of terrains. While the curve-based modeling algorithm can create numerous island examples, cGAN provides additional flexibility in generating more complex terrain without the rigid constraints of the curve-based algorithm that stem from our initial assumptions based on coral reef formation theory.

The training was performed using a batch size of 1, and optimized using the Adam optimizer with a learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The loss function combined a conditional adversarial loss and an L1 loss, where the L1 loss was weighted by a factor $\lambda = 100$. The generator follows a U-Net architecture with encoder-decoder layers and skip connections, while the discriminator was based on a PatchGAN, classifying local image patches. These parameters are directly used from the original paper.

The inference of our model, with a dataset of approximatively 10 000 examples representing about 30 minutes of training, still outputs grid artifacts, visible in Figure 12. After the second epoch, visual artifacts are already rare. This training time seems long but, at the best of our knowledge, is the shortest training time for terrain generation with learning-based approach in the litterature, mostly due to the synthetic nature of our dataset.

Once the model is trained, the user colors a 256×256 RGB image to sketch the different regions. Each region is given a specific color based on the HSV encoding used for the dataset generation. The examples presented in this paper uses red for abysses, blue for reefs, cyan for lagoons, green for beaches and yellow for mountains. The subsidence factor presented in Section 4.2.1 is controllable through the luminosity of the input image.

The Python script for the initial island dataset generation is poorly optimized and takes about 2.5s per island of size 256×256 as we do not perform parallelization here. Implementing an optimized C++ version of the initial generation process reduces this execution time to 50ms per generation.

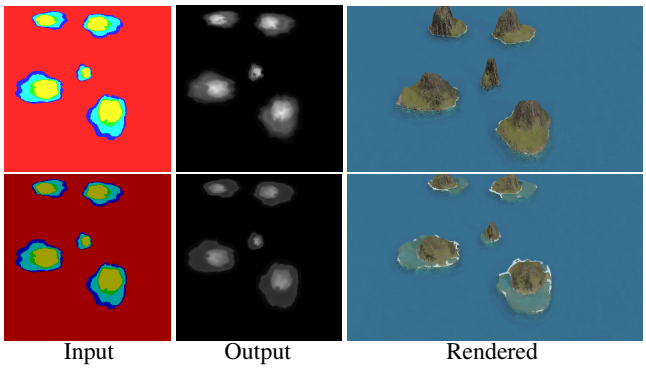

Figure 13: *An identical label map yield similar height fields over multiple inferences from the model, even after modifying the subsidence factor (visible in the luminosity of the input image).*

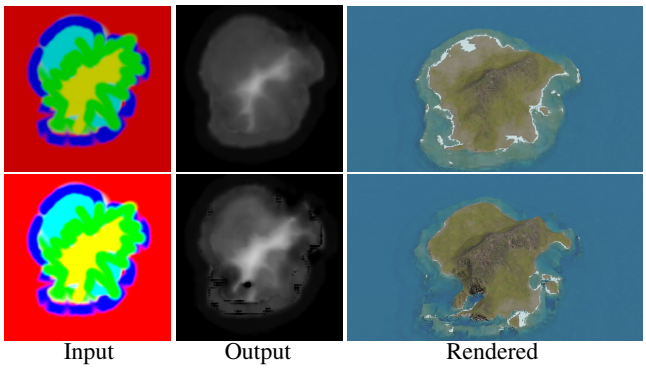

Figure 14: *User applied fuzzy brushes to draw the label map, resulting in some pixels that are inconsistent with the dataset and unlogical island layouts: abyss regions (red) are found between beach (green), lagoon (cyan) and reef regions (blue). The neural model ignores the layout inconsistencies, and partially over-brightness as long as the pixel isn't completely white.*

On the other hand, the inference time of our deep learning model for a single input image of dimension 256×256 is constant whatever the complexity of the scene. Using the NVIDIA GeForce GTX 1650 Ti GPU with Python 3.10 and PyTorch version 2.5.1+cu121, the inference time measured is 5ms (std 1.1ms).

7. Results

The resulting model for coral island generation enables a high control-level from a user perspective as the unconstraint painting allows for complex scenarios while producing in real-time the resulting height fields. In this paper we used the software Blender to provide renders directly from the outputed height fields. As our pix2pix model is trained to output 256×256 images, the resolution of the 3D models is limited by this architecture.

Using deep-learning-based models, most constraints from our initial assumptions are lifted (radial layout, isolated islands, ...).

The control over the overall shapes of the islands regions are given through digital painting, here using the GIMP software. Each pixel of the image are encoded in HSV, with the region identifier encoded in the Hue channel. The user may increase or decrease the subsidence level of the island by modifying the Value channel over the whole image (see Figure 13).

Since the model is based on statistics over the pixel values instead of hard values, users are not limited to a finite number of region identifiers, meaning that the output is more or less robust to noise (due to image compression, for example) and to the fuzzy values resulting from anti-aliasing of brushes often set by default, or resizing algorithm, by image editors, or even due to compression algorithm. The example displayed in Figure 14 presents a sketch for which the outlines of the regions are at the same time blurry and with layouts that are not expected (such as the small red regions inside the southern lagoon region or the adjacency of beach regions directly with the abyssal region) on the top figure and showing highlight clipping on the bottom figure. The learned model does not include inconsistancies and results in plausible 3D models. We can note that the input label map doesn't require to be hand drawn, as a simple Perlin noise can produce it randomly, as shown in the examples of Figure 17.

The tolerence over the input values may be used to provide even more control about the transitions between two regions. Figure 15 shows an example of input map with regions that are leaking over neighboring regions, and the introduction of new hue values non-existant in the dataset (light green and dark green) but are the interpolated hue value of mountain regions and beach regions.

Since the curve-based procedural phase included low randomness, the output of the cGAN is limiting its inpredictibility and the results to a slight change on the input create only slight changes on the output, preventing unexpected results. Figure 13 shows the result of an input map with only a variation on the subsidence level, the resulting height fields are very similar. Adding the real-time computation of outputs, it becomes possible to construct progressively a landscape and correct small mistakes to intuitively design islands inspired by real-world regions. A creation of an island similar to Mayotte, presented in Figure 16 was hand-made in few minutes.

Limitations While this approach brings significant advantages, there are also some limitations to consider. The reliance on a synthetic dataset means that the cGAN inherits some biases and limitations of our or initial curve-based algorithm. This could limit the true diversity of the terrains that the model can generate, as the output is confined by the patterns present in the training data. Additionally, the cGAN model's internal logic lacks transparency, offering limited user control over the generation process once the model has been trained. Moreover the training time is far from real-time.

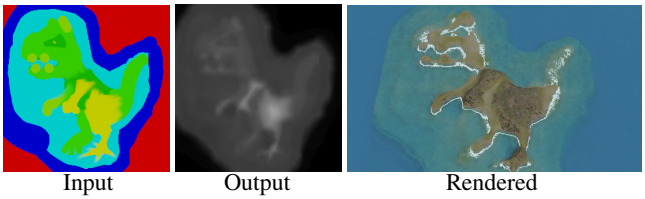

Figure 15: *Without constraints on the generation, the user may use unrealistic layout and the neural network will however output a plausible result. Here new colors has been added (pistachio), but the network naturally consider it as a transition from mountain to beach.*

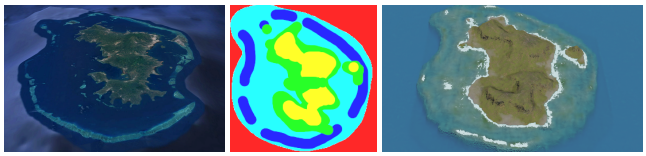

Figure 16: *Comparison between of real (left, from Google Earth) and synthetic (right) islands of Mayotte. The label map (center) is hand drawn to match approximatively its real-world counterpart.*

Future work Further improvements could be made to the synthetic dataset. Incorporating more complex geological processes, such as wave erosion or tidal influences, could lead to even more realistic terrains. Additionally, refining the way islands are blended in multi-island samples, or adding more diverse input conditions (e.g., different geological settings), could help the model generalize better and produce more varied and dynamic landscapes. While the current model allows for rapid terrain generation, adding more options for users to interact with the cGAN, such as tweaking parameters like wind strength or island size, could enhance the flexibility of the system. Many other neural networks models could be exploited to increase the possibilities, such as newer variants of cGANs (Park et al., 2019), or models with style transfer functionalities (Gatys et al., 2015; Zhu et al., 2020) in order to change the overall aspect of a terrain (Perche et al., 2023b, 2023a), use text-to-images models (Rombach et al., 2021; Radford et al., 2021) to generate height fields from a verbal prompt, or super-resolution models (Dong et al., 2014) to increase the definition of details in the final output (Guérin, Digne, Galin, and Peytavie, 2016).

8. Conclusion and future work

This work has presented a novel approach to generating coral reef island terrains by combining traditional procedural methods with deep learning techniques. We first developed a procedural generation algorithm capable of creating a wide variety of island terrains through a combination of top-view and profile-view sketches, wind deformation, and subsidence and coral reef growth simulation. By applying these methods, we were able to produce realistic terrains based on geological processes, capturing key features of coral reef islands such as beaches, lagoons, and coral reefs.

To further enhance flexibility and realism in the generation process, we incorporated a conditional Generative Adversarial Network (cGAN), using the pix2pix model to generate height maps from label maps of island features. The cGAN model allowed us to overcome some of the constraints inherent in the procedural algorithm, such as radial symmetry and fixed island positioning. With data augmentation techniques, we were able to train the cGAN on a synthetic dataset, generating varied and realistic island terrains.

References

Beckham, C., & Pal, C. (2017). A step towards procedural terrain generation with gans. http://arxiv.org/abs/1707.03383 (cit. on p. 2).

Beneš, B., Těšínský, V., Hornyš, J., & Bhatia, S. K. (2006). Hydraulic erosion. *Computer Animation and Virtual Worlds*, *17*, 99–108. https://doi.org/10.1002/cav.77 (cit. on p. 2).

Cordonnier, G., Braun, J., Cani, M.-P., Benes, B., Galin, É., Peytavie, A., & Guérin, É. (2016). Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum*, *35*, 165–175. https://doi.org/10.1111/cgf.12820 (cit. on p. 2).

Cordonnier, G., Cani, M.-P., Beneš, B., Braun, J., & Galin, É. (2017). Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE Transactions on Visualization and Computer Graphics*, *24*. https://doi.org/10.1109/TVCG.2017.2689022 (cit. on p. 2).

Cordonnier, G., Galin, É., Gain, J., Beneš, B., Guérin, É., Peytavie, A., & Cani, M.-P. (2017). Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3072959.3073667 (cit. on p. 2).

Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Image super-resolution using deep convolutional networks. http://arxiv.org/abs/1501.00092 (cit. on p. 8).

Ebert, D. S., Peachey, D., Worley, S., Hart, J. C., Musgrave, K., Perlin, K., & Mark, W. R. (2003). *Texturing and modeling, a procedural approach* (Vol. Third edition). Morgan Kaufmann. (Cit. on p. 2).

Ecormier-Nocca, P., Cordonnier, G., Carrez, P., Moigne, A. M., Memari, P., Benes, B., & Cani, M. P. (2021). Authoring consistent landscapes with flora and fauna. *ACM Transactions on Graphics*, *40*. https://doi.org/10.1145/3450626.3459952 (cit. on p. 2).

Fournier, A., Fussell, D., & Carpenter, L. (1982). Computer rendering of stochastic models. *Communications of the ACM*, *25*, 371–384. https://doi.org/10.1145/358523.358553 (cit. on p. 2).

Gain, J., Marais, P., & Straßer, W. (2009). Terrain sketching. *Proceedings of I3D 2009: The 2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, *1*, 31–38. https://doi.org/10.1145/1507149.1507155 (cit. on p. 2).

Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.-P., Benes, B., & Gain, J. (2019). A review of digital terrain modeling. *Computer Graphics Forum*, *38*, 553–577. https://doi.org/10.1111/cgf.13657 (cit. on p. 2).

Gasch, C., Chover, M., Remolar, I., & Rebollo, C. (2020). Procedural modelling of terrains with constraints. *Multimedia Tools and Applications*, *79*, 31125–31146. https://doi.org/10.1007/s11042-020-09476-3 (cit. on p. 2).

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. http://arxiv.org/abs/1508.06576 (cit. on p. 8).

Génevaux, J.-D., Galin, É., Peytavie, A., Guérin, É., Briquet, C., Grosbellet, F., & Beneš, B. (2015). Terrain modelling from feature primitives. https://doi.org/https://doi.org/10.1111/cgf.12530 (cit. on p. 2).

Guérin, E., Peytavie, A., Masnou, S., Digne, J., Sauvage, B., Gain, J., & Galin, E. (2022). Gradient terrain authoring. *Computer Graphics Forum*, *41*, 85–95. https://doi.org/10.1111/cgf.14460 (cit. on p. 2).

Guérin, É., Digne, J., Galin, É., & Peytavie, A. (2016). Sparse representation of terrains for procedural modeling. *Computer Graphics Forum*, *35*, 177–187. https://doi.org/10.1111/cgf.12821 (cit. on p. 8).

Guérin, É., Digne, J., Galin, É., Peytavie, A., Wolf, C., Beneš, B., & Martinez, B. (2017). Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3130800.3130804 (cit. on p. 2).

Hnaidi, H., Guérin, E., Akkouche, S., Peytavie, A., & Galin, E. (2010). Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, *29*, 2179–2186. https://doi.org/10.1111/j.1467-8659.2010.01806.x (cit. on p. 2).

Hopley, D. (2014). *Encyclopedia of modern coral reefs : Structure, form and process*. Springer, Credo Reference. (Cit. on p. 1).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967–5976. https://doi.org/10.1109/CVPR.2017.632 (cit. on pp. 2, 7).

Körner, C., Spehn, E. M., Bugmann, H., & Zurich, E. (2014). *Mountain systems* (tech. rep.). https://www.researchgate.net/publication/238663492 (cit. on p. 1).

Lyons, M. B., Murray, N. J., Kennedy, E. V., Kovacs, E. M., Castro-Sanguino, C., Phinn, S. R., Acevedo, R. B., Alvarez, A. O., Say, C., Tudman, P., Markey, K., Roe, M., Canto, R. F., Fox, H. E., Bambic, B., Lieb, Z., Asner, G. P., Martin, P. M., Knapp, D. E., … Roelfsema, C. M. (2024). New global area estimates for coral reefs from high-resolution mapping. *Cell Reports Sustainability*, *1*, 100015. https://doi.org/10.1016/j.crsus.2024.100015 (cit. on p. 1).

Mei, X., Decaudin, P., & Hu, B. G. (2007). Fast hydraulic erosion simulation and visualization on gpu. *Proceedings - Pacific Conference on Computer Graphics and Applications*, 47–56. https://doi.org/10.1109/PG.2007.27 (cit. on p. 2).

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. http://arxiv.org/abs/1411.1784 (cit. on p. 2).

Musgrave, F. K., Kolb, C. E., & Mace, R. S. (1989). The synthesis and rendering of eroded fractal terrains. *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989*, 41–50. https://doi.org/10.1145/74333.74337 (cit. on p. 2).

Natali, M., Viola, I., & Patel, D. (2012). Rapid visualization of geological concepts. *Brazilian Symposium of Computer Graphic and Image Processing*, 150–157. https://doi.org/10.1109/SIBGRAPI.2012.29 (cit. on p. 2).

Neidhold, B., Wacker, M., & Deussen, O. (2005). Interactive physically based fluid and erosion simulation. *Natural Phe-*

nomena, 25–32. http : / / graphics . uni - konstanz . de / publikationen/Neidhold2005InteractivePhysicallyBased/ Neidhold2005InteractivePhysicallyBased . pdf (cit. on p. 2).

Olsen, J. (2004). Realtime procedural terrain generation. *Department of Mathematics And Computer Science*, 20. https : / / pdfs . semanticscholar . org / 5961 / c577478f21707dad53905362e0ec4e6ec644 . pdf % 5Cnhttp://www.tsi.enst.fr/~bloch/P6/PRREC/terrain_ generation.pdf%5Cnhttp://web.mit.edu/cesium/Public/ terrain.pdf (cit. on p. 2).

Panagiotou, E., & Charou, E. (2020). Procedural 3d terrain generation using generative adversarial networks. http://arxiv.org/abs/2010.06411 (cit. on p. 2).

Park, J., Redwine, J., Hill, T. D., & Kotun, K. (2019). Water resource and ecotone transformation in coastal ecosystems. *Ecological Modelling*, *405*, 69–85. https://doi.org/10.1016/j.ecolmodel.2019.04.015 (cit. on p. 8).

Perche, S., Peytavie, A., Benes, B., Galin, E., & Guérin, E. (2023a). Styledem: A versatile model for authoring terrains. http://arxiv.org/abs/2304.09626 (cit. on p. 8).

Perche, S., Peytavie, A., Benes, B., Galin, E., & Guérin, E. (2023b). Authoring terrains with spatialised style. *Computer Graphics Forum*, *42*. https://doi.org/10.1111/cgf.14936 (cit. on p. 8).

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics*, *19*, 287–296. https://doi.org/10.1145/325165.325247 (cit. on p. 2).

Perlin, K. (2001). Noise hardware. *Real-Time Shading SIGGRAPH Course Notes* (cit. on p. 2).

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. http://arxiv.org/abs/2103.00020 (cit. on p. 8).

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. http://arxiv.org/abs/2112.10752 (cit. on p. 8).

Schott, H., Paris, A., Fournier, L., Guérin, E., & Galin, E. (2023). Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, *42*, 1–15. https://doi.org/10.1145/3592787 (cit. on p. 2).

Sisodia, Y. (2022). Gan-generated terrain for game assets. *Indian Journal of Artificial Intelligence and Neural Networking*, *2*, 1–3. https://doi.org/10.54105/ijainn.F1060.102622 (cit. on p. 2).

Smelik, R. M., Kraker, K. J. D., Groenewegen, S. A., Tutenel, T., & Bidarra, R. (2009). A survey of procedural methods for terrain modelling. *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)* (cit. on p. 2).

Spick, R., & Walker, J. A. (2019). Realistic and textured terrain generation using gans. *Proceedings - CVMP 2019: 16th ACM SIGGRAPH European Conference on Visual Media Production*. https://doi.org/10.1145/3359998.3369407 (cit. on p. 2).

Talgorn, F. X., & Belhadj, F. (2018). Real-time sketch-based terrain generation. *ACM International Conference Proceeding Series*, 13–18. https://doi.org/10.1145/3208159.3208184 (cit. on p. 2).

Voulgaris, G., Mademlis, I., & Pitas, I. (2021). Procedural terrain generation using generative adversarial networks. *European Signal Processing Conference*, *2021-August*, 686–690. https://doi.org/10.23919/EUSIPCO54536.2021.9616151 (cit. on p. 2).

Wulff-Jensen, A., Rant, N. N., Møller, T. N., & Billeskov, J. A. (2018, January). Deep convolutional generative adversarial network for procedural 3d landscape generation based on dem. In *Interactivity, game creation, design, learning, and innovation* (pp. 85–94). Springer. https://doi.org/10.1007/978-3-319-76908-0_9 (cit. on p. 2).

Zhu, D., Cheng, X., Zhang, F., Yao, X., Gao, Y., & Liu, Y. (2020). Spatial interpolation using conditional generative adversarial neural networks. *International Journal of Geographical Information Science*, *34*, 735–758. https://doi.org/10.1080/13658816.2019.1599122 (cit. on p. 8).

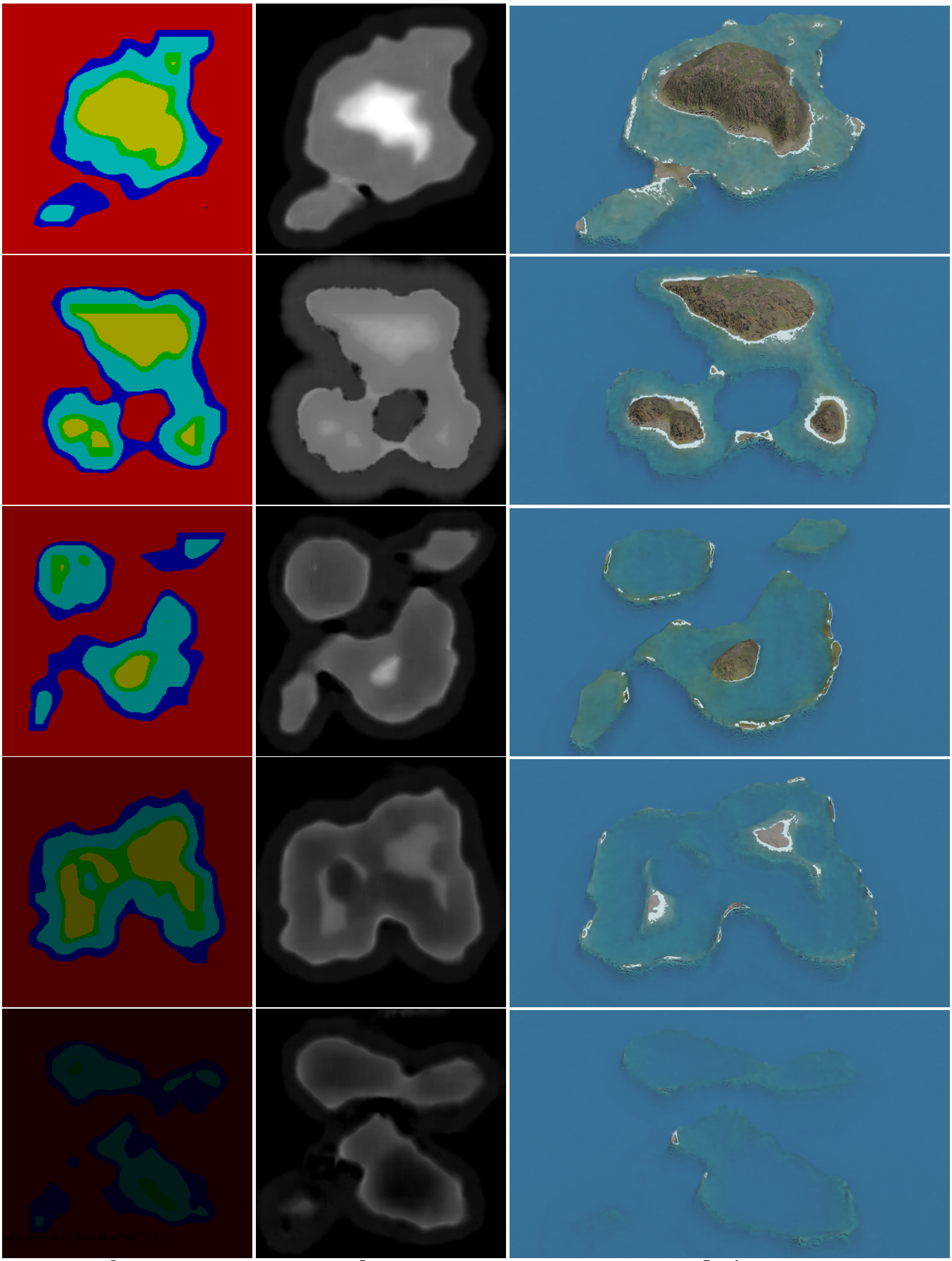

Figure 17: *Starting from random Perlin noise, transformed into label maps with increasing subsidence, we can generate a large variety of results.*

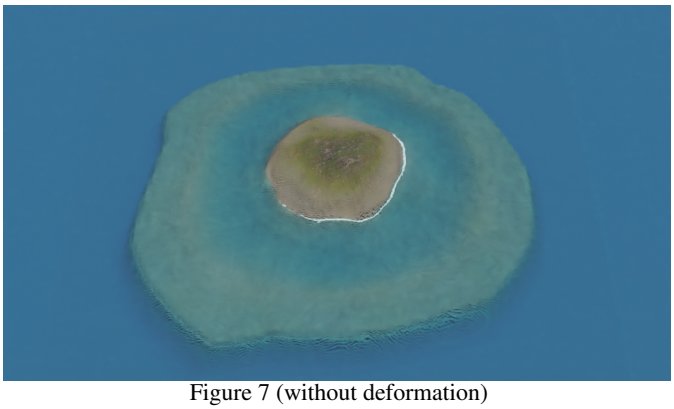

Figure 7 (without deformation)

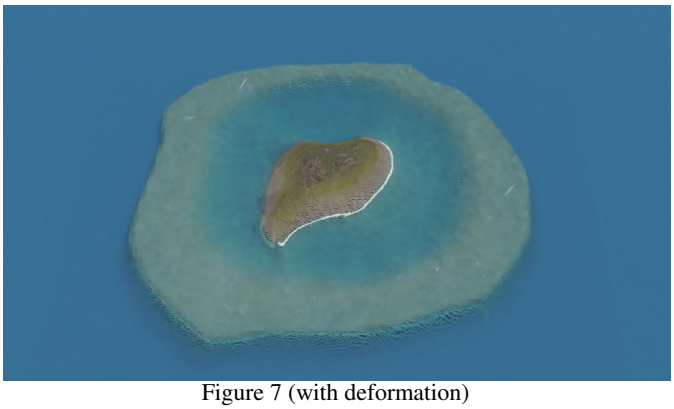

Figure 7 (with deformation)

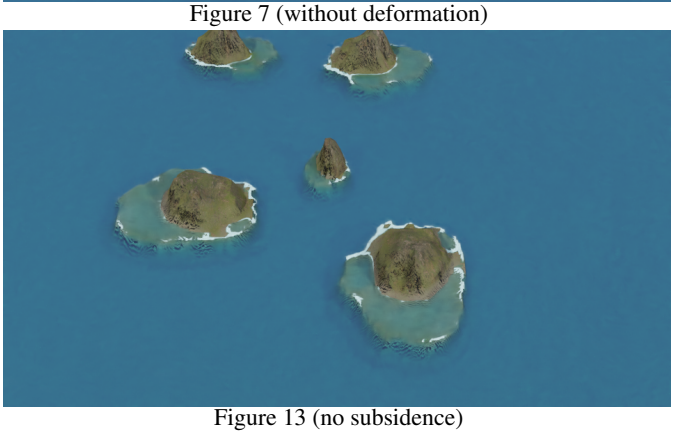

Figure 13 (no subsidence)

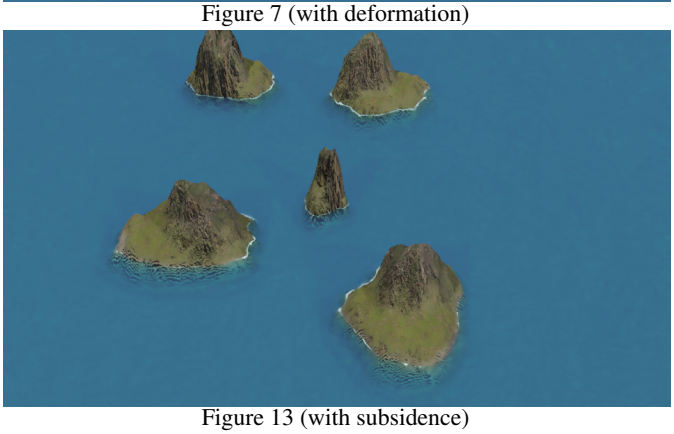

Figure 13 (with subsidence)

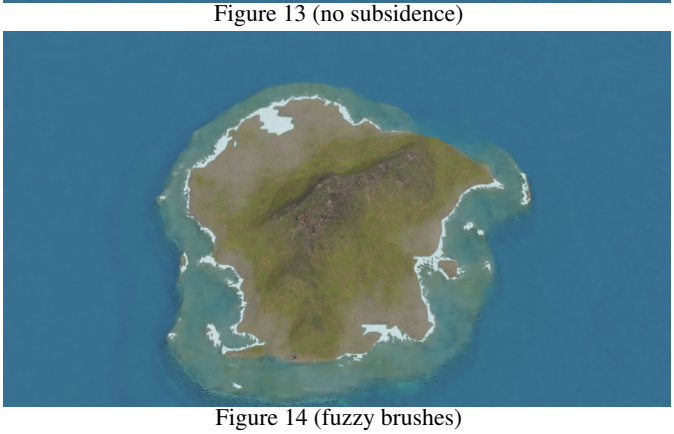

Figure 14 (fuzzy brushes)

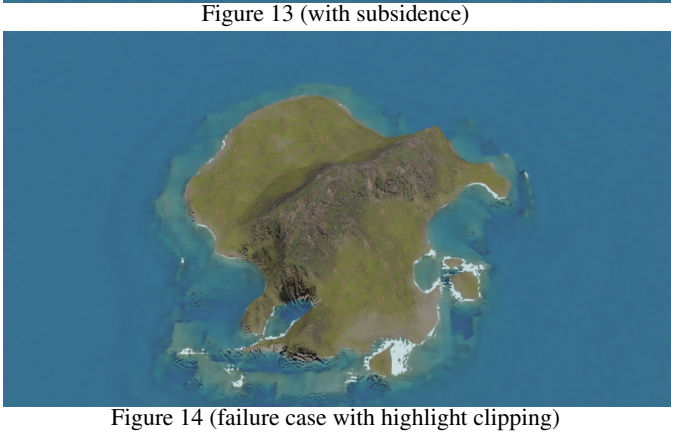

Figure 14 (failure case with highlight clipping)

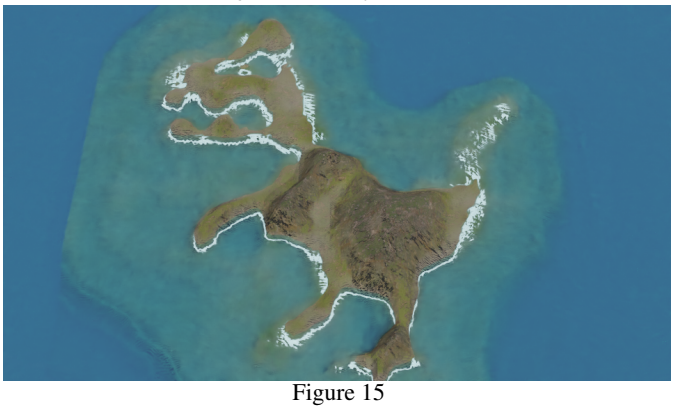

Figure 15

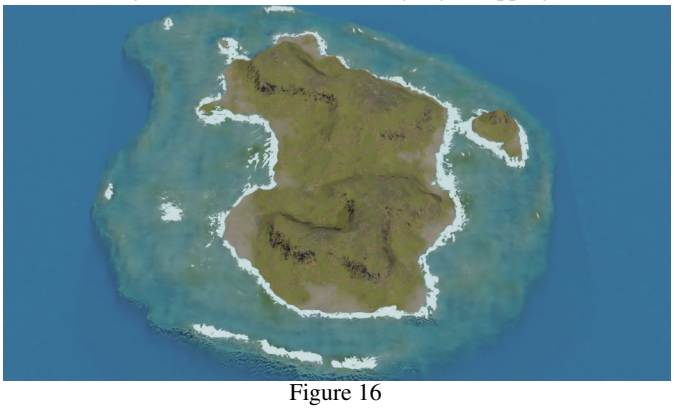

Figure 16

Figure 18: *High-resolution version of figures found in this paper.*