Real-Time Hyper-Amplification of Planets

Yann Cortial, Eric Guérin, Adrien Peytavie, Eric Galin

Real-Time Hyper-Amplification of Planets

Yann Cortial · Adrien Peytavie · Eric Galin · Eric Guérin

Abstract We propose an original method for generating planets with a high level of detail in real-time. Our approach relies on a procedural *hyper-amplification* algorithm: a controlled subdivision process that faithfully reproduces landforms and hydrosphere features at different scales. Starting from low-resolution user-defined control maps that provide information about the elevation, presence of large-scale water bodies and landforms types, we apply subdivision rules to obtain a high resolution hydrologically consistent planet model. We first generate a large-scale river network connected to inner seas and oceans, then synthesize the detailed hydrographic landscapes, including river tributaries and lakes, mountain ranges, valleys, plateaus, deserts and hills systems. Our GPU implementation allows to interactively explore planets that are produced by tectonic simulations, generated procedurally or authored by artists.

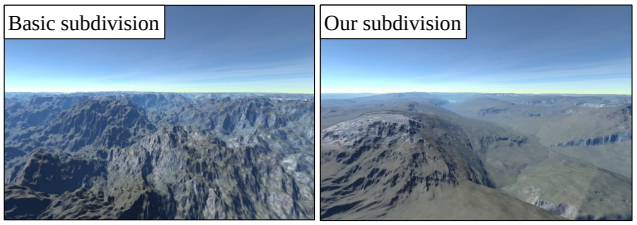


Fig. 1 Compared to standard subdivision techniques, our method exhibits natural landform features such as valleys, rivers and varying landscape types.

Y. Cortial, E. Guérin
LIRIS-CNRS INSA de Lyon

A. Peytavie, Eric Galin
LIRIS-CNRS Université de Lyon

Keywords Planets, Terrain modeling, Amplification, Adaptive parallel subdivision, Procedural generation

1 Introduction

The surface of planets represents a vast domain, typically several hundreds of millions of square kilometers, featuring a vast variety of landforms, and possibly a hydrosphere and an atmosphere. Modeling and generating huge scale environments with a high level of detail remains a complex problem in computer graphics. This challenge stems not only from the range of scales and their corresponding landforms and patterns that need to be synthesized, but also from the global geomorphological consistency that has to be preserved.

But for a few notable exceptions [7,6,4], this challenge has seldom been investigated in computer graphics research. In contrast, the entertainment industry is investigating in the direction of large-scale procedural generation methods, a trend exemplified by the still-in-development game *Star Citizen*©. Pipelines for interactive or real-time procedural planets generation and exploration have been proposed by professional studios. Those methods focus on control and mostly rely on blending heightfields authored by artists. Therefore, the generated terrains lack geomorphological coherence and consistency, for the most part due to the absence of hydrographic features. At the time of this writing, publicly available material allows to infer that the generated planets have a reduced 1:15 scale, *i.e.* a corresponding radius of $\approx$ 400km . A viable option for planet modeling would be to amplify a low resolution model by introducing details. Several terrain amplification methods such as sparse-modeling [11] or deep learning [14] have been proposed for terrains, unfortunately those

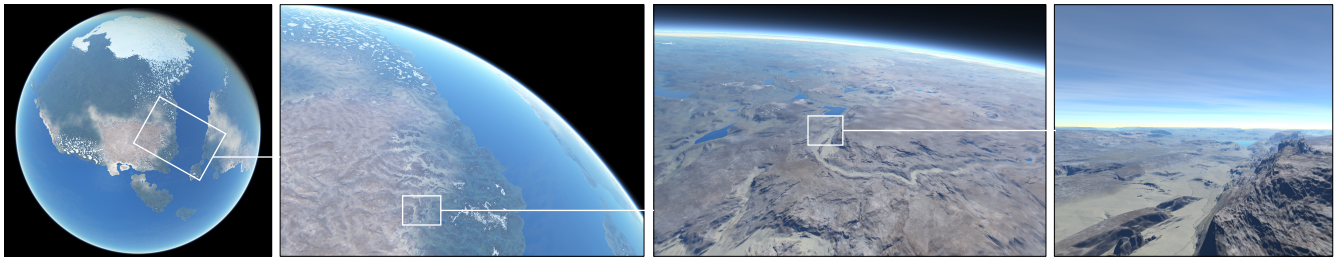


Fig. 2 Given a low-resolution representation of a planet (at ≈ 50km precision), our method performs a hierarchical procedural *hyper-amplification* process based on subdivision rules to generate a detailed representation of the relief (at ≈ 50cm precision) featuring a vast variety of coherent landforms at different range of scales, such as large rivers and mountain ranges, smaller valleys and hills, lake shores and river banks.

approaches are limited both in terms of amplification magnitude and size of the terrain that may be processed.

In this paper, we address the problem of generating detailed planets while allowing interactive exploration. Our method proceeds in two major steps. Starting with a CPU pre-processing phase, we generate a low-resolution model of the planet that describes the landforms and the river network at a large scale of about 50 km. This low-resolution model is then refined in a real-time, multi-resolution, *hyper-amplification* method based on hierarchical rule-based subdivisions on the GPU. Amplification allows the generation of new river branches, augmenting the initial network and producing realistic basins with drainage patterns, lakes, valleys with cross sections and landforms, driven by low-resolution user control maps (Figure 1). We implemented our method on graphics hardware which enabled real-time exploration of the synthesized planets at any-scale.

Our contributions are as follows: 1) We propose a *real-time hyper-amplification* algorithm for generating detailed planet relief with coherent water bodies. Our approach based on hierarchical subdivision rules generates *high resolution* elevation at a ≈ 50cm precision (Figure 2) from control maps at a ≈ 50km precision yielding a $\approx 10^5 \times 10^5$ amplification factor. 2) Our method generates river networks and synthesizes geomorphologically and hydrologically *coherent* planets with consistent valleys and landforms. 3) Our framework provides a high level control over the distribution of landforms and patterns; moreover additional specific landforms can be prescribed by using overriding global control maps and locally changing the final model. The generation of the detailed relief of the planet relies on a constrained subdivision process that lends itself for a parallel graphics hardware implementation (for reproducibility reasons, the source code will be released upon acceptance).

2 Related Work

Our work relates to procedural planet modeling, terrain amplification, and subdivision techniques. Here we briefly review those categories and refer the reader to a recent terrain modeling survey [9] for general terrain generation techniques.

Virtual Planets are in general procedurally generated, for computational efficiency reasons, by combining coherent noise functions at different scales and frequencies to approximate fractional Brownian motion and generate fractal relief [7]. Mid-point displacement techniques were proposed to generate river networks over an entire planet [6]. Unfortunately, the underlying homogeneous and isotropic fractal model fails at generating realistic reliefs at all scales, from continents to local landforms such as mountains. A procedural approximation of plate tectonics was recently proposed in [4] to generate large-scale features such as continents, island arcs and mountain ranges. Relief details are produced by a real-time amplification of the planet using heightfield blending and combining noise functions, which cannot guarantee geomorphological coherence. The method does not generate river networks, and the resolution of the synthesized terrains is limited to ≈ 100m. In contrast, our approach generates a consistent global river network and relies on hierarchical subdivision rules to produce detailed river patterns.

Amplification techniques incrementally add details onto an initial low-resolution terrain. A commonly used technique consists in adding high-frequency noise scaled according to the characteristics of the input terrain [7]. Sparse modeling [11] replaces low-resolution local features by detailed landforms selected in a dictionary of atoms and generates more geomorphologically consistent details. The major drawback of these techniques is that they cannot guarantee the coherence of the drainage network. Recently, Generative Adversarial Networks [14] were proposed for learning styles from exemplars and amplifying terrains according to the analyzed details.

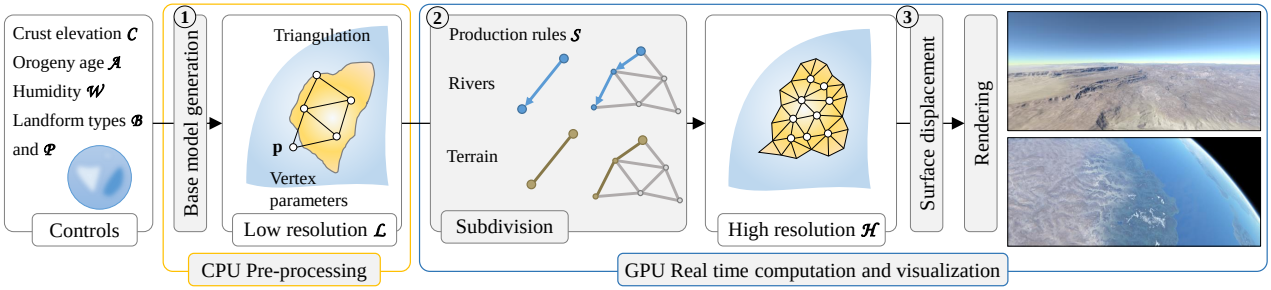


Fig. 3 Given several control maps $\mathcal{M}$ defining the crust elevation $\mathcal{C}$, the orogeny age $\mathcal{A}$, the wetness $\mathcal{W}$ and the landform type $\mathcal{P}$ or $\mathcal{B}$, we synthesize an initial low-resolution model $\mathcal{L}$ which is recursively amplified in real time by using subdivision rules to generate details.

While this method better preserves the overall hydrological consistency, it is limited to small amplification factors, in general about an order of magnitude. In contrast, we achieve a $\approx 10^5 \times 10^5$ amplification factor.

Subdivision techniques were used not only to generate reliefs but also create river networks that play an important part in the realism of large scale terrains. *Diamond-square* and *square-square* algorithms [8] synthesize fractal terrains with self-similar elevations using a unique random displacement rule applied after mid-point subdivision. Several enhancements were proposed to constrain the subdivision process with a view to providing better control. Prusinkiewicz *et al.* [13] introduced subdivision rules that account for the river network generation, and [2] constrained subdivision to prescribed crests and rivers altitudes. Our method takes its inspiration from those methods: instead of relying on a single stochastic subdivision rule, we use a set of production rules at different levels of detail to generate a vast variety of landforms. [10] specifically directly addressed the problem of river network generation using a grammar-based network expansion, and then generated the elevation of the terrain according to the elevation of the river network. A similar strategy was employed in [1] for synthesizing landscapes from graphs representing the ridges lines connecting peaks and the river network. Our method relies on consistent large-scale river networks generated at a low-resolution, and subdivision rules that automatically generate details.

3 Overview

Our algorithm (Figure 3) may be decomposed in three main steps: low-resolution planet synthesis from control maps, *hyper-amplification* using subdivision rules at different scales to generate details, and finally surface displacement to prescribe specific landform patterns.

Starting from a set of user-defined control-maps $\mathcal{M}$, we first generate a low-resolution planet $\mathcal{L}$ as a spherical triangulation whose vertices $\{\mathbf{p}_i\}$ store control information such as mean crust elevation, age, wetness and landform type. The distance between two vertices of a triangle is ≈ 50km. This input can be either generated procedurally using fractal noise [7], authored by artists, or even results from tectonic planet simulation methods [4]. The data structure of $\mathcal{L}$ embeds all the control parameters required to perform the real-time amplification of the planet surface, including large-scale main river networks, coastlines and mountain ranges.

The low resolution model is then recursively subdivided to generate a high-resolution representation of the planet $\mathcal{H}$, according to landforms production rules $\mathcal{S}$. The subdivision rules are controlled by the type of the landforms stored at the vertices and edges, the values of the corresponding control parameters, and the level at which subdivision rules are applied.

More precisely, the production rules rely on the following set of control parameters. *Crust elevation* $\mathcal{C} \in [-10^4, 10^4]$ provides a hint about the relief of the planet at low-resolution, defining continents and oceans and the large-scale mountain ranges, massifs, and plains. As subdivision takes place, true elevations are produced and eventually characterize the exact relief of the surface of the planet. *Orogeny age* $\mathcal{A} \in [0, 10^8]$ allows to distinguish between young and ancient mountain ranges, and plays an important part in the choice of subdivision rules for producing smooth rounded mountains or high spiky mountains. *Humidity* $\mathcal{W} \in [0, 1]$ can be viewed as a precipitation map, defines arid and wet areas, and controls the distribution of lakes as well as specific landforms like mesas and terraces in rocky deserts. Plateau and hill presence, denoted as $\mathcal{P} \in [0, 1]$ and $\mathcal{B} \in [0, 1]$ respectively, control the presence of the corresponding landforms.

4 Low resolution model generation

Whenever the low resolution planet is not directly provided as a mesh, which may be the case for function-

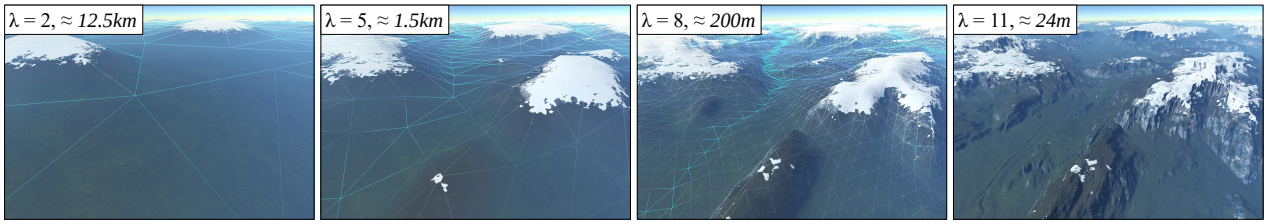


Fig. 4 Four different levels of detail λ from the same viewpoint.

based representations [7], we generate it as illustrated in Figure 5. We first sample the surface of the planet provided by the user using a Poisson Disk distribution [3] at a 50km precision. From the samples, we construct a spherical Delaunay triangulation, using incremental algorithms adapted to the spherical domain, *i.e.* by relying on geodesic distances instead of Euclidean distances. The goal is to produce an initial *irregular* triangulation, in order to avoid rectilinear and otherwise unrealistic large-scale rivers. Our experiments show that an acceptable candidate is the triangulation of Poisson samples. Starting with an initial octahedron or icosahedron and using dyadic subdivision does not produce realistic rivers, even with strong jittering, since geometric patterns are still clearly detected by the human eye. The Delaunay property also garantees well formed triangles, which is important since triangles will undergo massive subdivision in our process.

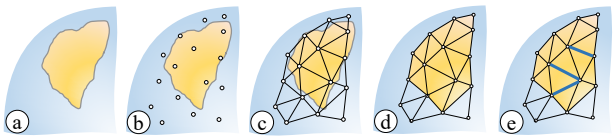


Fig. 5 The pre-processing stage: (a) control maps $\mathcal{M}$, (b) Poisson disk sampling yielding vertices $\mathbf{p}_i$, (c) spherical Delaunay triangulation, (d) coast processing, (e) river network growth and computation of the hydrographic characteristics such as Horton-Strahler index, flow ϕ_i and water elevation.

This triangulation is then regularized in order to produce well-defined coastal triangles, capable of producing estuaries (Figure 5d). This enables the generation of the river networks, starting from the coasts towards the centers of the continents and islands in a similar way to [10]. We finally compute the length of the rivers, their Horton-Strahler index [12] which allows to evaluate the water flow.

The triangulation of low-resolution planets involves two computationally intensive steps: the Poisson disc sampling and the Delaunay triangulation. They are performed once and for all during a pre processing step (on the CPU, generating $230k$ vertices and $460k$ triangles with an average edge length of 52km takes 9.2 seconds). The computation of the parameters at the vertices $\mathbf{p}_i$ and the parameters of the river network is performed in ≈ 0.5 second.

5 Adaptive subdivision

Our planet amplification scheme is based on a hierarchical rule-based subdivision process, which uses sets of rules operating at different scales.

We use a two stage approach for generating the surface details. First, we refine the triangles of the model using a dyadic scheme, *i.e.* inserting a new vertex at the center of every edge. It is a rule-based process, controlled by the various local configurations found in the triangulation. Both vertices and edges have types associated to them, which serve to identify these configurations. This first step produces an adaptively refined triangulation according to the desired level of detail (Figure 4).

The second stage involves the post-processing of the resulting triangulation, by displacing vertices vertically in order to shape riverbeds and valleys, as described in Section 6.

5.1 Subdivision

The real-time subdivision of the triangulation relies on a set of rules, which are triggered depending on the configuration of the vertices and edges. Let $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{U}$ denote the edge types associated to river, terrain, and unknown respectively. Let r, t, l, s, g and u denote the vertices types associated to river, terrain, lake, sea, gully and unknown respectively. In our implementation, aside from the necessary topology information, edges are characterized by their type. Vertices store their type and the parameters detailed in Table 1 and Section 5.2.

According to the desired level of detail, the triangulation is subdivided using a rule-based algorithm in order to obtain the desired landforms. The subdivision algorithm is sequentially divided in two steps. First, we subdivide each triangle edge with semantic production rules (Figure 6), and compute the associated parameters of each new vertex using a compute graph. Then,

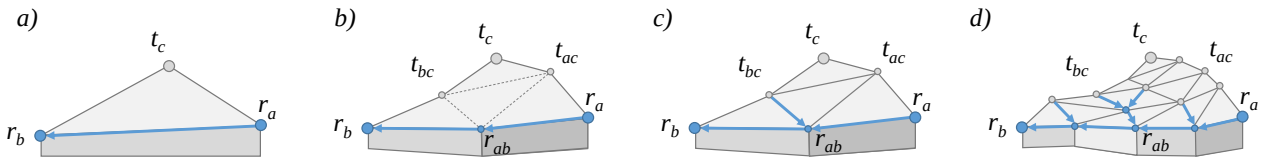


Fig. 6 Illustration of two subdivision steps: a) starting triangle, b) dyadic subdivision and insertion of 3 new vertices, c) internal edges creation and instantiation of a river tributary, d) the process recursively repeats at increasing λ.

Symbol	Description
x	Type $\in$ {*River, Terrain, Lake, Gully, Sea*}
h	Elevation
h_w	Water elevation
h_r, d_r	Nearest riverbed elevation and distance
ϕ	River flow
e_r	River cross-section reference
s_h, s_d	Nearest gully elevation and distance

Table 1 Parametrization of the terrain vertices

we create the three internal edges inside a split triangle and assign edges types using a second specific set of rules (Figure 11).

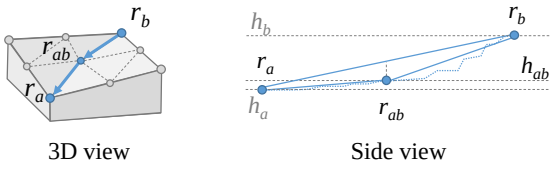


Fig. 7 Illustration of **R1** where a river edge is subdivided and creates a new river vertex. The computation of this new vertex elevation has to respect the flowing of the river.

Edge subdivision This process splits an edge into two edges and adds a new vertex. An input configuration composed of a triplet $(u_a, \mathcal{U}_{ab}, u_b)$ triggers a specific rule depending on the configuration. Rules associated with *River* and *Terrain* vertex types are:

R1 $(r_a, \mathcal{R}_{ab}, r_b) \rightarrow (r_a, \mathcal{R}_{ac}, r_c, \mathcal{R}_{cb}, r_b)$
R2 $(r_a, \mathcal{T}_{ab}, r_b) \rightarrow (r_a, \mathcal{T}_{ac}, t_c, \mathcal{T}_{cb}, r_b)$
R3 $(r_a, \mathcal{T}_{ab}, t_b) \rightarrow (r_a, \mathcal{T}_{ac}, t_c, \mathcal{T}_{cb}, t_b)$
R4a $(r_a, \mathcal{R}_{ab}, t_b) \rightarrow (r_a, \mathcal{R}_{ac}, r_c, \mathcal{R}_{cb}, t_b)$
R4b $(r_a, \mathcal{R}_{ab}, t_b) \rightarrow (r_a, \mathcal{R}_{ac}, r_c, \mathcal{T}_{cb}, t_b)$
R5 $(t_a, \mathcal{T}_{ab}, t_b) \rightarrow (t_a, \mathcal{T}_{ac}, t_c, \mathcal{T}_{cb}, t_b)$

R1 transforms a river into two river sections (Figure 7), **R2** is used to create crests between two river vertices (Figure 8), and **R3** creates a terrain vertex along the slope (Figure 9). Rules **R4a** and **R4b** are used to increase the density of the river network and generate

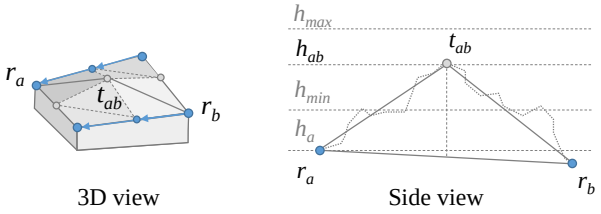


Fig. 8 Illustration of **R2** where a crest is created between two valleys. The new elevation shall be higher than the two river altitudes.

new river branches by adding a new tributary, it is only possible after the second subdivision, *i.e.* when internal edges have been created. **R4b** is applied when the level of detail has reached a threshold, this prevents rivers to be too close to mountains peaks.

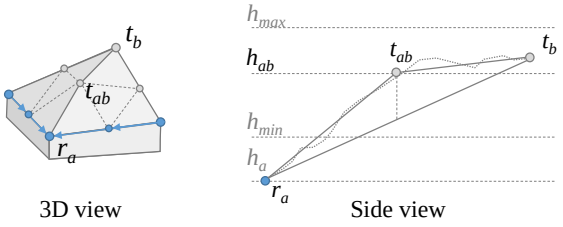


Fig. 9 Illustration of **R3** where a terrain vertex is created along the slope.

Internal edge During the subdivision process, three new edges are inserted inside the triangle face (Figure 11). The new edges connect the three new vertices obtained from the previous step. Specific rules fix the current undefined edge type $\mathcal{U}_{xy}$ as follows:

Ia $(t_a, \mathcal{U}_{ab}, t_b) \rightarrow (t_a, \mathcal{T}_{ab}, t_b)$
Ib1 $(r_a, \mathcal{U}_{ab}, t_b) \rightarrow (r_a, \mathcal{R}_{ab}, t_b)$ if β is satisfied
Ib2 $(r_a, \mathcal{U}_{ab}, t_b) \rightarrow (r_a, \mathcal{T}_{ab}, t_b)$ otherwise
Ic $(r_a, \mathcal{U}_{ab}, r_b) \rightarrow (r_a, \mathcal{T}_{ab}, r_b)$

Rules **Ib1** and **Ib2** are variants, the production of **Ib1** allows the triggering of rule **I4a** in return, which makes the river network denser (Figure 11 right). This rule is applied only if β is satisfied. The condition β combines several tests. A first test is checked on the angle between the existing river edge and the new one,

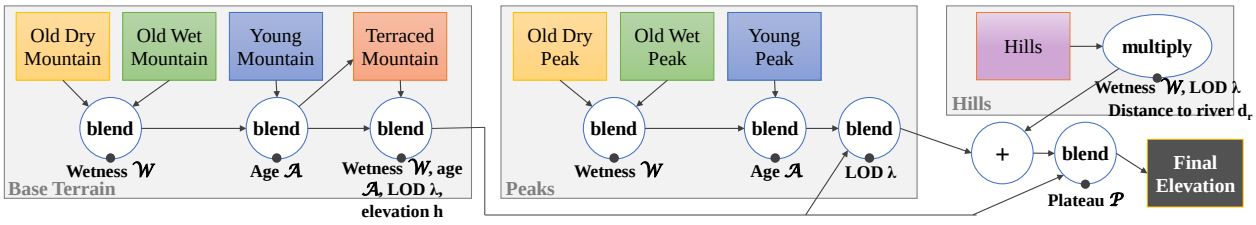


Fig. 10 Elevation compute graph for the case where terrain vertices are connected by a Terrain edge $(t, \mathcal{T}, t)$.

which should be less than 90° to avoid a counter intuitive visual aspect of the confluence point. Another test consists in being on the right level of detail, in practice a tributary appears on internal edges of length ranging in $\approx [3, 25]$ km. Additionally, this internal river edge should always appear in the same triangle incident to the main river edge, this is guaranteed by enforcing determinism as described in Section 7. If the condition β passes then the new river edge appears, otherwise we add a new gully instead.

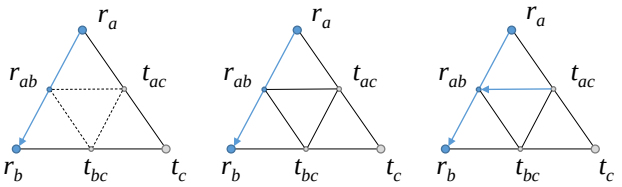


Fig. 11 Internal edge type computation. We subdivide the triangle face by adding 3 edges linking the 3 new vertices of the previous step. The edge type is obtained by using specific rules for the internal edges. Blue indicates river vertices and edges, whereas gray indicates terrain.

5.2 Vertices parameters

New created vertices are assigned a set of parameters. The global control parameters ($\mathcal{A}$, $\mathcal{W}$, $\mathcal{P}$, $\mathcal{B}$) are simply propagated as the average of the two input vertices parameters, at every level of detail. For the other parameters default averaging also applies but specific rules override if needed with specific compute graphs in order to produce specific landform features.

Compute graph In order to control the terrain types and ensure continuity in our model, we combine a set of procedural nodes in a graph operating on continuous parameters, that we call a *compute graph*. Each procedural generation node is parameterized by the two input vertices parameters of the edge being split and defines specific features like mountains, hills, or plateaus. Depending on the configuration, and therefore the triggered rule, specific graphs are chosen. Figure 10 depicts the graph used for the $(t, \mathcal{T}, t)$ configuration, Figure 13 for the $(r, \mathcal{T}, r)$ configuration, and Figure 12 the $(r, \mathcal{T}, t)$ configuration. These are the most frequent configurations. Other configurations are similar or simpler.

Procedural nodes The purpose of these nodes is to compute vertices parameters. Among these parameters, the output elevation h_i is derived from two input elevations h_a and h_b of the two parent vertices t_a and t_b as illustrated in Figure 7, 8 and 9. Thanks to the tagging of vertices with types (mainly river and terrain), these procedural nodes can guarantee natural landforms such as crests or valleys. Appendix A gives the detailed expressions we use for terrain nodes.

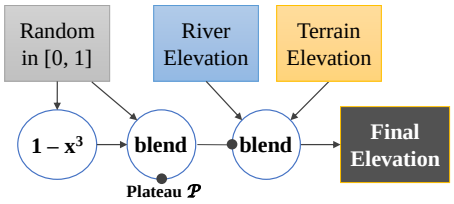
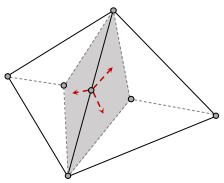


Fig. 12 Elevation compute graph for the case $(r, \mathcal{T}, t)$.

Horizontal displacement In order to break the regularity and linearity of landforms, we apply a deterministic random horizontal displacement to the vertices. This translation is made with barycentric coordinates inside a randomly chosen triangle incident to the edge. To prevent self-intersections the new position is chosen inside a safe zone determined by the barycenters of the neighboring triangles (see Figure above). This step is especially important for rivers, lakes and dry rocky landscapes.



5.3 Specific landforms

Gullies are defined by vertices g, similar to river ones and have the same production rules **R1** to **R5**. They

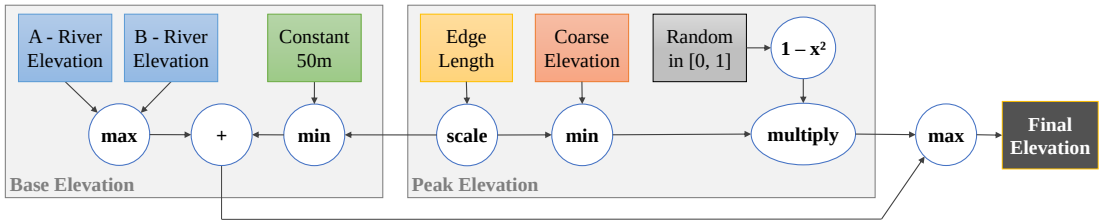


Fig. 13 Elevation compute graph for the case where river vertices are connected by a Terrain edge $(r, \mathcal{T}, r)$.

have different generation parameters: they may have steeper slopes, and do not drain water, *i.e.* they act as dry drainage patterns.

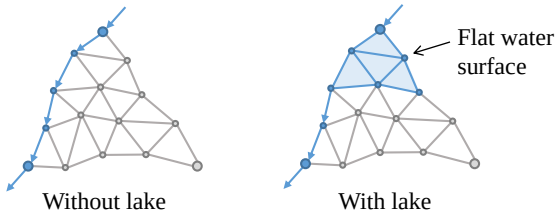


Fig. 14 A lake is created as an extension of the river. Specific water height computation is introduced to guarantee its flatness.

Lakes are not endorheic, *i.e.* they are connected to the local river network (Figure 15). Our model handles lakes as an extension of rivers by taking advantage of the river transverse cross section prescription (Section 6). Generation rule **R3** is modified to generate a lake and replace the terrain vertex into a lake. This is randomly triggered at a specific level of detail, based on a probability depending on the local wetness $\mathcal{W}(\mathbf{p})$. Lake vertices serve as an extension of the riverbed. Specific water height computations are triggered for lakes in order to guarantee that locally, the river and its neighboring lake will become planar and not flow as usual (Figure 14).

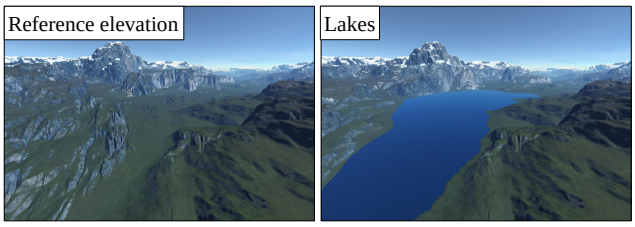


Fig. 15 Our method allows for the generation and control of lakes: left image shows a reference elevation map produced by our *hyper-amplification* algorithm, whereas right image show the same region with lakes.

6 Specific landform carving

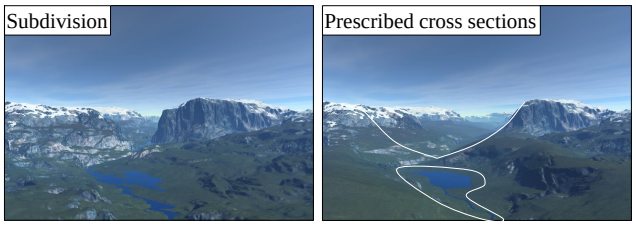


Fig. 16 Comparison between *hyper-amplification* using subdivision only (left) and with additional landform carving with riverbed and valley transverse cross section prescription (right). Both steps are performed in real-time on the GPU.

Once the subdivision is complete, we perform an important subsequent and final amplification step to produce realistic landforms conforming to the archetypes described in geomorphology (Figure 16).

We modify the elevation of the subdivided terrain, on a per-vertex basis, according to the distance to the nearest river d_r and its elevation h_r to carve the riverbed and shape the valleys (Figure 17). The cross-section of the valley, both in steepness and width, is determined by the river flow estimate $||\phi||$, which is also a per-vertex parameter carried through the subdivision. We use a power law [1] for the relationship between the flow value and the valley shape, higher drainage induce less slope and broader valleys. A cubic Hermite interpolation is used to blend the resulting profile with the existing raw subdivided terrain, in order to avoid unnatural slope discontinuities at boundaries.

The cross-section of the valley is defined by a reference e_r to an archetype one-dimensional curve representing the normalized valley elevation. Transverse cross sections archetypes have been proposed for different types of valley in geomorphology and have been used successfully for carving valleys as in [1]. Our framework relies on parametrized template models that lend themselves for on the fly evaluation and graphics hardware implementation. Real cross-sections from earth data-sets could be used instead, resulting in more re-

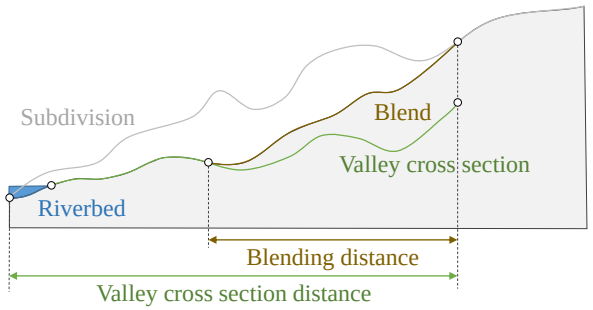


Fig. 17 Illustration of the transverse valley and riverbed generation, given the per-vertex parameters d_r, h_r and ϕ.

alistic valley shapes, at the expense of a more memory intensive data-driven terrain representation.

7 Implementation details

A key feature of our method is that it lends itself for parallel implementation on graphics hardware, which allows for real-time exploration of the planet (Figure 18).

Architecture The *hyper-amplification* subdivision and its post-process are performed using *GPGPU compute shaders* through a number of sequenced stages. The triangulation stays in video memory during this process, thus avoiding expensive transfers. The whole process is run once every $n \approx 10$ frames, and the output is directly streamed to the graphics pipeline for rendering.

During each modeling run, the target level of detail is first reached by iteratively calling four sequenced jobs: edge splitting, ghost-vertex marking, ghost-edge splitting and face splitting. Ghost operations guarantee a crack-free terrain geometry: ghost vertices are located at the middle of unsplit edges that belong to faces that need to be split. These four jobs represent one level of subdivision: they are repeated until no more edges remain to be split according to the desired level of detail.

We then displace the vertices of the triangulation according to the cross sections of the rivers and valleys. This process is followed by the bulk parallel creation of the water vertices, wherever visible, *i.e.* when the current vertex parameters obey $h_w > h$. The water vertices are connected using the same topology as the underlying terrain mesh. The modeling run ends with the repositioning of the terrain ghost vertices affected by the cross sections, and the computation of terrain normals. Finally, the rendering stages build the indexed graphics primitives (*i.e.* terrain and water triangles) from the obtained modeling data for the current frustum-culled view, and performs the final deferred shading.

Determinism The output of the pipeline is deterministic and guarantees the coherent production of the terrain at different scales. All compute operations of stochastic nature need to be reproducible. Compute nodes produce elevations that need to be deterministic, *i.e.* invariant across subdivision runs. This is ensured by using a unique seed tied to each vertex, which is created as the sum of the seeds of the two vertices of each edge, as in [6]. Parallelism is also an issue, the computation of the new internal edges in a triangle needs to be handled with care, as different neighboring triangles could access the same vertex concurrently. To ensure that a branching river is always created in the same incident triangle to the main river edge, we rely on choosing among the two incident faces by drawing a random number from the seed tied to the junction river vertex.

Precision Issues arising from the large scale range are handled by making use of log-z depth buffering [5] and of double-precision floating point representation wherever possible.

While double precision poses no problem in compute shaders, at least when using core *OpenGL* features, it remains challenging for the graphics stages. Note that the use of double precision is mandatory for virtual true-to-scale planets, as we need more than 7 significant digits to fully represent a given position at centimeter precision - for planet Earth (radius 6370km) indeed 9 digits are required. We use double floats projection matrices and vertex positions, however hardware interpolation for fragments is performed using single-precision only. To overcome this limitation, we interpolate and store in a 32 bits float the distance of the fragment to the camera; later in the fragment shader, we cast it to double precision and reconstruct the world position using the inverted double precision projection-view matrix. This allows for tri-planar texturing using shader-coded bi-linear interpolation performed on the double precision world coordinates. In our experiments, this approach provides stable, flicker-free texturing of the terrain even for textured details down to the order of the centimeter.

8 Results

We tested our prototype on an Intel i7-6700K clocked at 4.0 GHz with an NVIDIA GTX 1080Ti graphics card. Planets had approximately the size of the earth, (*i.e.*, radius of 6370 km) and the maximum ground resolution was 50 cm at the highest level of detail (see accompanying video). We achieved a rendering frame rate of approximately 25 Hz for worst case low-altitude

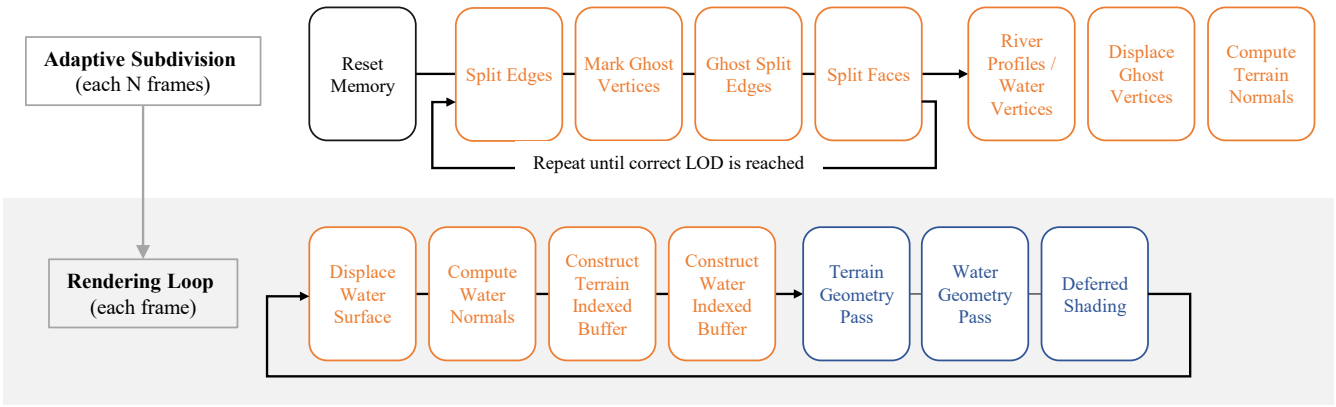


Fig. 18 Overview of the GPU pipeline: (orange) compute shaders, (blue) graphics pass.

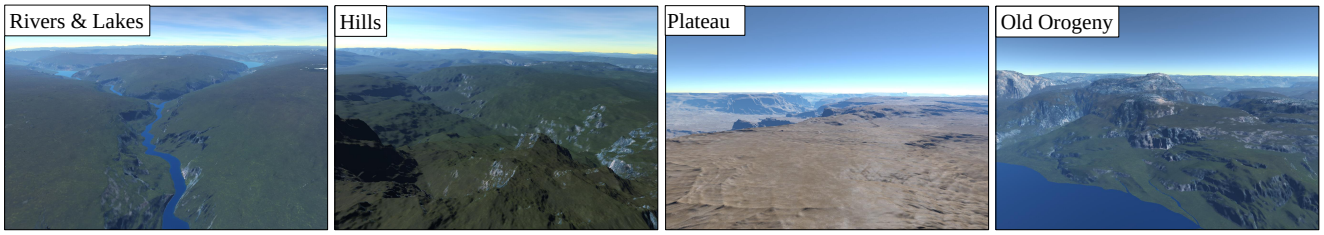


Fig. 19 Our method allows for the generation and control of a wide range of landforms features, like rivers and lakes (using $\mathcal{W}$), hills (using $\mathcal{B}$), young vs. old mountains (using $\mathcal{A}$) and plateaus (using $\mathcal{P}$).

views with computationally intensive shading for water animation, atmospheric effects and double-precision terrain texturing. Relief generation, performed every 10 frames, including subdivision, cross section specification and post-processing, took on average 80 ms, but this result is highly varying, depending on how close to the ground the camera gets, with worst-case peaks above 100 ms for close-to-ground views at centimeter precision where the total number of terrain-only triangles reaches 4 millions. The break-down of an average modeling run is as follows, the subdivision takes $\approx$ 50 ms (of which 44% is devoted to splitting edges, 15% on ghost operations and 41% to splitting faces) while riverbeds and valleys carving represents a few 3 ms, lastly the post-processing of the mesh takes roughly 30 ms. The cost of the adaptive surface generation is amortized over the 10 frames, and the GPU is free to schedule the compute jobs alongside graphics passes - this in turn allows for real-time exploration as demonstrated in the accompanying video.

Control Our method offers control capabilities to the user with both high-level and localized expression. High-level control is achieved through global planet-scale specification in the form of input maps. Figure 20 showcases a reproduction of the earth obtained by providing the relief of continents and wetness maps at 50 km resolution into the system. Local control is possible, by

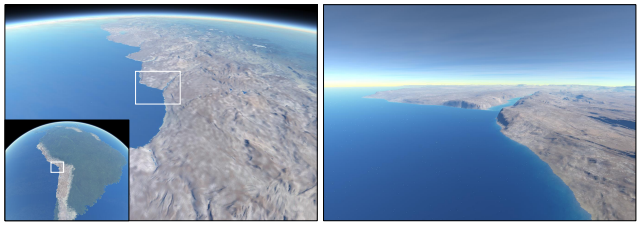


Fig. 20 Example of *hyper-amplification* performed on a low-resolution representation of the earth, showcasing a progressive zoom on the South American continent.

overriding amplification during the per-vertex elevation displacement post-process, through the prescription of specific height fields at user-specified locations.

Figure 19 shows a variety of features produced by our model. Lakes were produced in regions where the wetness $\mathcal{W}$ was high. Plateaus and hills appeared where the types $\mathcal{P}$ and $\mathcal{B}$ had high values respectively. The continuity of the generated terrains is guaranteed by the interpolation of the parameters in the spatial domain and the smooth combination of the elevations in the compute graph. The shape of mountains was influenced by the orogeny age $\mathcal{A}$, which produced more gentle, eroded, slopes for old mountains. The wetness parameter also influenced the shape and shading of the terrain as illustrated in Figure 21. Gullies and drainage patterns (Figure 23) also play an important part in the overall visual aspect of the detailed planetary surface.

We did not endeavor to test the authoring and editing potential of our system. However, it can be seen from the good performances we demonstrate, that such an approach would make sense. One can imagine altering the control maps by directly painting over the planet, both on a local or a global scale. In our current implementation half a second is needed to regenerate the whole low-resolution mesh including the base river network, whereas the subdivision and rendering are instantaneous. This is sufficient for interactive feedback and would therefore enable fluid authoring sessions.

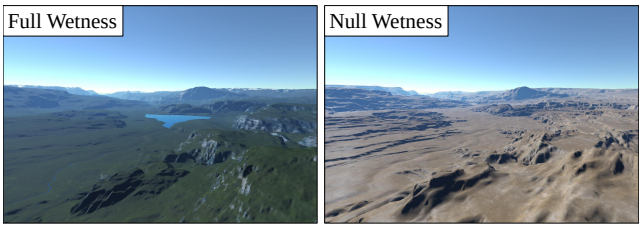


Fig. 21 Influence of the wetness parameter $\mathcal{W}$ resulting in slightly different landscapes: (left) presence of water, smooth low-altitude terrain slopes, (right) stronger horizontal displacement hinting towards eolian erosion, dry riverbeds and terrace effects on rocky slopes.

Discussion Our method compares favorably to standard subdivision techniques [8] as can be seen in Figure 1 and 22. This is the result of an adaptive subdivision process that accounts for the level of detail and adjusts to local features. Compared to other planet synthesis methods [7,6,4], our approach provides a unified framework to synthesize and combine a variety of landforms such as river networks, mountain ranges, valleys and plains. Particularly, [6] is able to generate the surface of a planet along with a river network, but suffers from a very simple subdivision scheme for the terrain that does not exhibit special features such as our method. On the other hand, [4] shows large-scale features that are the consequence of tectonic phenomena, but cannot produce a river network. This method also relies heavily on a data driven amplification method, which is expensive in terms of storage. Contrary to most of the previous methods, our method provides a direct high-level control by allowing the user to define the low-resolution elevation $\mathcal{L}$ that influences the *hyper-amplification*.

Limitations Our method is memory demanding and requires more than 2GB on the graphics card. The subdivision algorithm requires many parameters plus all the necessary topology information and flags stored in the triangulation. 32 bytes are used to store an edge,

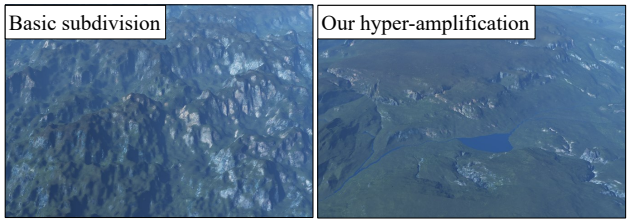


Fig. 22 Our method integrates landform features such as rivers, valleys, lakes, plateaus that look more natural than standard subdivision techniques.

32 bytes for a triangle, 48 bytes to represent the topology of vertex and 112 bytes to store the parameters. The use of double-precision for positions has a important impact. Water vertices buffers and other specific graphics buffers significantly increase the amount of required memory. Our prototype implementation could be optimized by packing some generation parameters such as $\mathcal{W}$ and $\mathcal{P}$.

Although our method is efficient and capable of synthesizing realistic models, it yields undesirable artifacts when it comes to the shading of the geometry. This can be seen in the accompanying video most noticeably on snow-covered peaks, where the snow effect depends on vertex slope which varies from one subdivision run to another every ≈ 10 frames. This could be mitigated by applying a temporal anti-aliasing technique, or more thoroughly by introducing progressive morphing between the modeling runs. Note that the last option would be memory demanding, and not a trivial extension to our method. With this in mind, we leave such an extension as future work.

In terms of realism, the final river network is not dense enough compared to real data, which comes from the geometric constraints in the subdivision scheme. More realistic peaks and crests could be obtained by re-using orometry concepts recently introduced in [1], we left this implementation as a future work and focused on the hierarchical subdivision algorithm.

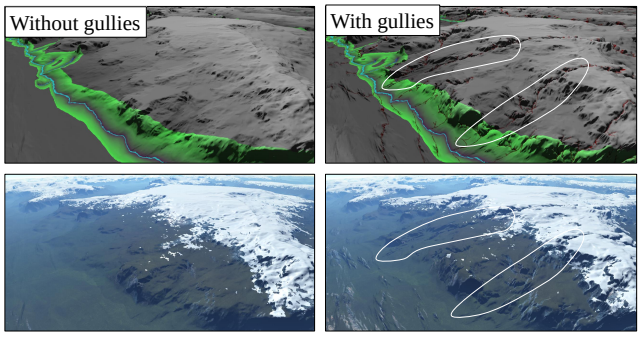


Fig. 23 Comparison of a terrain generated from a river edge with the corresponding influence region highlighted in green without (left) and with (right) drainage patterns.

9 Conclusion

We have introduced a *hyper-amplification* algorithm that generates the detailed relief of planet according to a low-resolution representation. The method relies on a multi-scale and landforms-aware subdivision scheme that generates a coherent river network. Our parallel implementation on graphics hardware shows real-time performance as for procedural noise-based terrains, but with a better variety and control over the generated landscapes.

In the quest for realism, an interesting direction consists in bridging the gap between our subdivision algorithm and data-driven approaches. In particular, the concept of exemplars of cross sections of real terrains could be generalized to shape more realistic valleys, enhance the distribution and the silhouette of peaks.

A Appendix - Procedural nodes

Here we detail the formulas used by the procedural nodes for the $(t_a, \mathcal{T}_{ab}, t_b)$ configuration represented in Figure 10. These nodes act as generators in the compute graph. We refer to the source code for the detailed calculation of the blending applied between nodes. Let $\xi \in [0,1]$ denote a pseudo-randomly generated number that depends on a seed attached to the new vertex, without loss of generality we consider the parent vertices t_a and t_b with minimum and maximum elevations $h_a < h_b$ respectively. Let Δ be the dependency on the current scale, λ being the Level of Detail, an integer starting with $\lambda = 0$ for the low-resolution model $\mathcal{L}$:

$$\Delta = 1 - \frac{2k}{5}, \qquad k = \begin{cases} \frac{\lambda}{10}, & \text{if } \lambda < 10 \\ 1, & \text{otherwise} \end{cases}$$

Classic mountain The elevation h is obtained by a fractal interpolation of the 2 heights h_a and h_b.

$$h = (1-\alpha)h_a + \alpha h_b$$

The blending coefficient α depends on the mountain type (young, old, wet or dry):

Old Dry	$\alpha = \frac{1}{2} + \frac{2\Delta}{5}(\xi - 1)$
Old Wet	$\alpha = 1 + \frac{3\Delta}{5}(\xi - 1)$
Young	$\alpha = \frac{1}{2} + \frac{\Delta}{2}(\xi^\gamma - 1)$

Peaks Peak elevations are defined according to the Crust elevation $\mathcal{C}$:

Old Dry	$h = \xi\mathcal{C}$
Old Wet	$h = (1-\xi^2)\mathcal{C}$
Young	$h = \xi^2\mathcal{C}$

Terraced Mountain This node modifies an input elevation h_i to obtain terrace features on dry bedrock areas:

$$h = T + \beta^2(3-2\beta)\tau \qquad with \qquad T = \lfloor \frac{h_i}{\tau} \rfloor \tau, \qquad \beta = \frac{h_i - T}{\tau}$$

τ represents the elevation step used for the terraces, which can be itself modulated by procedural nodes, although in our implementation we used a simple constant of 700 meters. Note that the Hermite interpolation factor β is clamped to $[0,1]$.

Hills offset This procedural node produces an elevation offset applied to the terrain surface:

$$H_i = \min(H_{max}, \frac{(1-\xi^2)\epsilon}{2})\mathcal{B}, \; \epsilon = \begin{cases} \frac{-||t_a - t_b||}{3}, & \text{if } \chi < 0.5 \\ ||t_a - t_b||, & \text{otherwise} \end{cases}$$

where $||t_a - t_b||$ is the current edge length, H_{max} is a constant in our implementation indicating the maximum hills elevation (we used 800 m), and χ is a uniform random number in $[0,1]$.

References

1. Argudo, O., Galin, E., Peytavie, A., Paris, A., Gain, J., Guérin, E.: Orometry-based terrain analysis and synthesis. ACM Transactions on Graphics **38**(6), 199 (2019)
2. Belhadj, F., Audibert, P.: Modeling landscapes with ridges and rivers. In: Proceedings of the Symposium on Virtual Reality Software and Technology, pp. 151–154. ACM, Monterey, USA (2005)
3. Bridson, R.: Fast poisson disk sampling in arbitrary dimensions. In: SIGGRAPH sketches, p. 22. ACM, New York, NY, USA (2007)
4. Cortial, Y., Peytavie, A., Galin, E., Guérin, E.: Procedural tectonic planets. Computer Graphics Forum **38**(2), 1–11 (2019)
5. Cozzi, P., Ring, K.: 3D engine design for virtual globes. AK Peters/CRC Press (2011)
6. Derzapf, E., Ganster, B., Guthe, M., Klein, R.: River networks for instant procedural planets. Computer Graphics Forum **30**(7), 2031–2040 (2011)
7. Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
8. Fournier, A., Fussell, D., Carpenter, L.: Computer rendering of stochastic models. Communications of the ACM **25**(6), 371–384 (1982)
9. Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.P., Benes, B., Gain, J.: A review of digital terrain modeling. Computer Graphics Forum **38**(2), 553–577 (2019)
10. Génevaux, J.D., Galin, É., Guérin, É., Peytavie, A., Benes, B.: Terrain generation using procedural models based on hydrology. Transaction on Graphics **32**(4), 143:1–143:13 (2013)
11. Guérin, E., Digne, J., Galin, E., Peytavie, A.: Sparse representation of terrains for procedural modeling. Computer Graphics Forum **35**(2), 177–187 (2016)
12. Horton, R.E.: Erosional development of streams and their drainage basins; hydrophysical approach to quantitative morphology. Geological society of America bulletin **56**(3), 275–370 (1945)
13. Prusinkiewicz, P., Hammel, M.: A fractal model of mountains and rivers. Graphics Interface **93**, 174–180 (1993)
14. Zhao, Y., Liu, H., Borovikov, I., Beirami, A., Sanjabi, M., Zaman, K.: Multi-theme generative adversarial terrain amplification. ACM Transactions on Graphics **38**(6), 200:1–200:14 (2019)