Eurographics Symposium on Rendering (2004)
A. Keller, H. W. Jensen (Editors)

Rendering Procedural Terrain by Geometry Image Warping

Carsten Dachsbacher, Marc Stamminger

Computer Graphics Group, University of Erlangen-Nuremberg
{dachsbacher,stamminger}@cs.fau.de

Abstract

We describe an approach for rendering large terrains in real-time. A digital elevation map defines the rough shape of the terrain. During rendering, procedural geometric and texture detail is added by the graphics hardware. We show, how quad meshes can be generated quickly that have a locally varying resolution that is optimized for the inclusion of procedural detail. We obtain these distorted meshes by importance based warping of geometry images. The resulting quad mesh can then be rendered very efficiently by graphics hardware, which also adds all visible procedural detail using vertex and fragment programs.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image GenerationDisplay algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismVirtual Reality;

1. Introduction

Interactive terrain rendering is one of the classical challenges in computer graphics. A lot of research has been spent on how to procedurally generate large terrains and how to render such large landscapes. Typical applications are outdoor computer games or geographic information systems. In many of these applications, a terrain renderer should ideally get the global terrain structure from a precomputed or loaded height field that only sketches the landscape. This low detail terrain should then be augmented procedurally with geometric and color detail, ideally directly at run time.

Previous approaches mostly ignore this scenario and focus on offline terrain generation or on the interactive rendering of precomputed, static data. In this paper we present a novel approach for realtime view-dependent level-of-detail (LOD) rendering of terrains, where the global shape is defined by a static heightmap and procedural detail is added at rendering time. Our method takes into account properties of modern graphics hardware and is designed to optimally exploit their computation power.

The approach is based on the idea of geometry images [GGH02]. Geometry images are pictures, in which each color triple corresponds to a 3D surface point. Topology is defined implicitly, the resulting surface automatically has the topology of a square. Imaging operations on a geometry image transfer to its geometry, e.g., downfiltering the geometry image also results in downfiltered geometry. We exploit this property to achieve a very efficient terrain LOD by warping the geometry image of the terrain such that mesh resolution is increased where needed and removed where it does not contribute to the current view.

As noted before [GGH02], geometry images are perfectly suited for graphics hardware. A geometry image is a texture that can be directly interpreted as a quad mesh. Recent graphics boards and very recent drivers offer this reinterpretation as OpenGL extension, that allows to bind a texture as a vertex or attribute array.

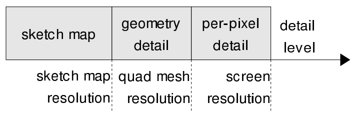

Figure 1: *Handling of detail levels*

Our method handles geometric detail of the terrain at different scales, as shown in Fig. 1. Coarse detail is taken from the input height field (sketch map). It contains detail up to a frequency of $f_0 = 1/2d_0$, where d_0 is the grid distance of the sketch map. This height field is upsampled to a finer grid with grid distance d_g. We add procedural detail up to frequency $1/2d_g$ to the upsampled mesh vertices as geometry detail. Finally, procedural detail up to screen frequency $1/2d_s$ is added by per-pixel lighting.

The pipeline is depicted in Fig. 2. We start with the height map of our world. This map is stored in main memory and can thus be large, e.g. cover the entire Alps. It only sketches the rough shape of our world and cannot contain fine detail, thus we call it *sketch atlas*. For every frame, the square region enclosing the current view frustum is copied out of the atlas and stored as *sketch map*.

In regions close to the camera, the sketch map has very low screen resolution. On these large triangles, per-pixel lighting alone is not sufficient to represent procedural detail. We thus adapt mesh resolution, such that all visible mesh cells cover similar screen area. Depending on surface characteristics and viewing parameters, we compute the required sampling resolution of a mesh (generated from the sketch map) in world space. The view dependent parameters include orientation of the surface, distance to the viewer and whether the surface is located inside the view frustum. These criteria are combined to a single value (see Section 3.1). As in [SWB98], we call the reciprocal of this value the *importance*, which is computed for each texel of the sketch map and stored in an *importance map*.

In the next step, we convert the sketch map to a geometry image by writing the world x and y coordinates to the red and green channel and the height values of the sketch map to blue. We then warp this geometry image according to the importance map, such that regions with high importance are enlarged and low importance regions shrink. In Section 3, we will present a simple image operation that allows us to do this step efficiently on the CPU with sufficient quality. We call the resulting distorted, view-dependent geometry image the *sketch geometry image*. Note that the sketch geometry image has higher resolution than the original sketch map in order to preserve all visible detail.

The resulting quadmesh contains the (resampled) original height field, but with generally smaller cells that have roughly similar screen size. Due to its increased resolution, we can add procedural detail resulting in the *detailed geometry image*. At this stage, to every vertex v we can only add low-frequency detail that can be represented by the mesh, i.e. detail up to frequency $1/2d_g(v)$, where $d_g(v)$ is the average grid distance around v.

The resulting detailed geometry image represents a mesh of quads that have an image size of only a few pixels in the current view. This quad mesh is then rendered, where procedural sub-triangle detail is added on the fragment level using bump mapping. Here, we only consider procedural detail from frequency $1/2d_g(v)$ up to screen resolution.

With our approach we are always either dealing with geometry images or quadrilateral meshes which are both very suitable for rendering and processing by the GPU. Our mesh always has the same topology and the same number of primitives. The warping of the sketch map according to the importance map results in a dynamic, view dependent level-of-detail method for terrain rendering, that is tailored for the inclusion of procedural detail at run time by the graphics processor.

After the description of previous and related work, we present the warping procedure in Section 3. In Section 4, we describe the band-limited evaluation of our procedural model. Finally, we show results and conclude.

2. Previous and Related Work

Terrain rendering is one of the classical challenges in computer graphics. Terrain data sets are usually very large–a moderate height field of size $4k^2$ already corresponds to 32 million triangles and thus cannot be directly rendered in realtime. However, the observer is usually relatively low above the ground, so view frustum culling, level-of-detail methods, and occlusion culling can reduce the geometry to be rendered enormously. Furthermore, the topology of terrains is simple, which simplifies the application of these methods. An excellent overview of terrain rendering methods can be found at `www.vterrain.org`.

Triangulated Irregular Networks represent the terrain by a reduced triangle mesh [GH95]. *View Dependent Progressive Meshes* reduce the mesh resolution according to the current view point and provide a particularly smooth transition [Hop98]. Other level-of-detail methods are usually more discrete. They work on regular grids or quadtrees and reduce the polygon count by replacing distant or invisible cells of the terrain by coarser meshes, e.g. [LKR*96, DWS*97, RHSS98, Ulr, CGG*03]. The methods differ in how the varying resolutions are generated and how the transition is smoothed. Our method is in between all these approaches. We generate a continuous level-of-detail, but the mesh also has fixed topology, so it can easily be handled by the GPU.

Interesting terrains are generally very large. Real-world heightfield data and corresponding aerial textures are usually only available in low resolution (10m to 1000m) or have to be bought at high costs. For many applications, procedural terrains that can be evaluated at arbitrary detail levels are better suited. Procedural terrains are usually generated by adding noise functions of increasing frequency, an excellent overview is given in [EMP*98].

Geometry images are a simple way to represent objects with disk topology by a texture [GGH02]. Image operations can be used to manipulate geometry. We will apply such an operation on a geometry image of the terrain to obtain an optimized level-of-detail. In [AMD02], a remeshing approach is presented that also generates meshes with adaptive density. In [SWB98] a method is presented to distort textures such that regions with much detail receive more texture space. The required sampling density is stored in an importance map, which then controls the texture deformation. In [SGSH02], a similar approach is presented, however here a signal-stretch metric is used to steer the reparametrization.

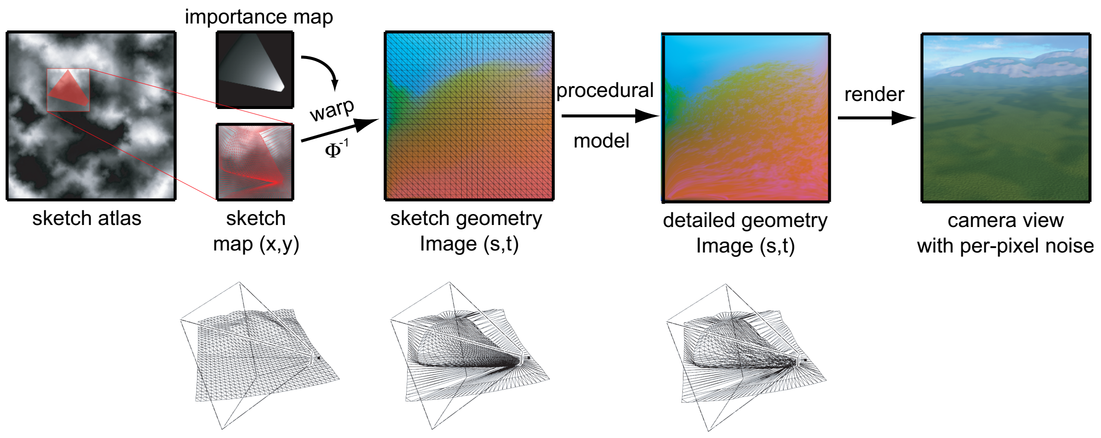
importance map
warp
Φ^{-1}
procedural
model
render
sketch atlas
sketch map (x,y)
sketch geometry Image (s,t)
detailed geometry Image (s,t)
camera view with per-pixel noise

Figure 2: *The pipeline: the rough shape of the world is defined in a* sketch atlas *(left). For the region covered by the current view frustum (*sketch map*) an* importance map *is computed that measures the required mesh density. A non-uniform quad mesh is generated based on this importance as a geometry image (*sketch geometry image*). This quad mesh is augmented with procedural detail (*detailed geometry image*) and rendered with bump-mapping.*

We will show another importance based warping algorithm in this paper, which delivers sufficient results and is fast enough to be applied for realtime rendering.

In parallel to this work, Losasso et al. developed a different terrain rendering approach with the same objective as ours [LH04]. The approach is also optimized for GPU rendering and can also be used to add procedural detail during rendering.

3. Geometry Image Warping

In this section, we describe the generation of the warped geometry image. The goal is to generate a finer resolution quad mesh of the sketch atlas. The mesh cell size is spatially varying. In screen space, the projected size of a cell should be between one and a few pixels.

We represent the quad mesh as a geometry image. First, the sketch map is converted to a geometry image by storing x, y, and altitude in the red, green, and blue channel, where the geometry image has floating point precision. Then, this geometry image is warped adaptively. The warping does not modify the geometry, but it locally changes the resolution of the represented quad mesh (the consequences of resampling will be discussed below). We exploit this to adapt the quad mesh resolution according to the requirements of the current view and landscape detail.

3.1. Importance Map

We control the warping using an importance map, where the importance $I(x,y)$ at a position (x,y) on the height field is the desired density of grid points around that point, i.e. a cell around that point should roughly have the extent $1/I(x,y)$ in x- and y-direction. Note that our importance measure is isotropic and cannot differentiate between directions.

First, the importance at a surface point $p = (x,y,z)$ is determined by the view distance. So we compute how densely the region around p is sampled by the pixel grid in the current camera view, which corresponds to the desired local sampling rate. This measure is inversely proportional to the viewing distance. Note that the orientation of the surface has to be regarded carefully, because in typical situations the sampling of the surface (regarding the scan conversion) is anisotropic. We account for the orientation at a later step and define $I_v(x,y) := C/d(p)$, where the constant C is the desired maximum image space size of the quads (ignoring their orientation) and $d(p)$ is the view distance of p.

Importance can be reduced for surface parts outside the view frustum or on backfacing mountain sides. To account for this, we define two functions S_f and S_b. $S_f(x,y)$ is zero for points outside the view frustum and one otherwise, with a smoothed safety transition zone. $S_b(x,y)$ accounts for backfacing and for silhouette regions. It is one for silhouette and frontfacing regions and is smoothly decreased for backfacing regions.

Furthermore, we take surface characteristics into account. For example, for smooth, large features like dunes a coarse resolution is sufficient, even for close ups. Thus, we store an upper bound $I_s(x,y)$ on the importance for each pixel of the heightmap, which depends on the surface material. This enables us to represent smooth terrain regions with few, large triangles.

To obtain the final importance I, we combine the above mea-

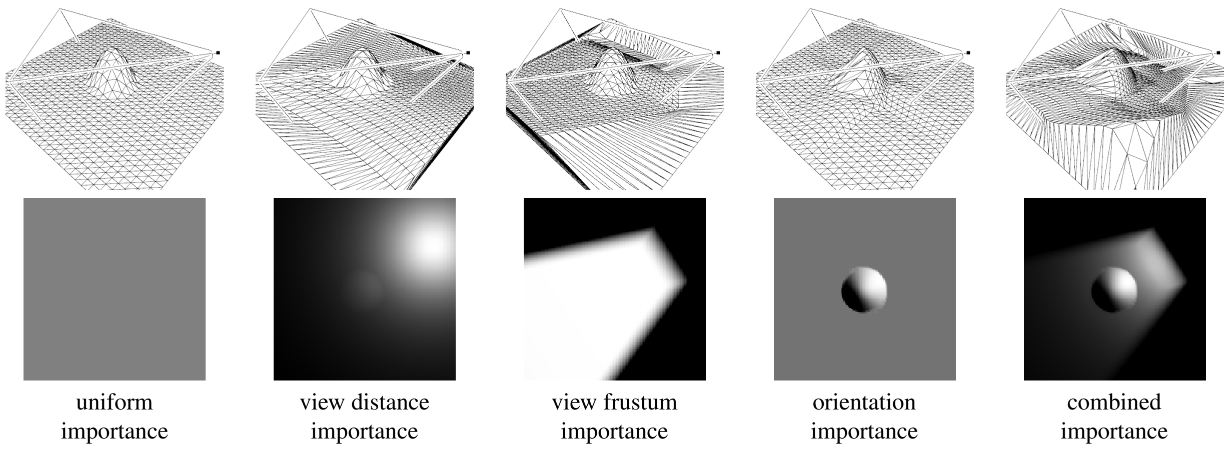

Figure 3: *The influence of importance (bottom row) on the resulting quad mesh (top row).*

sures as follows:

$$I(x,y) = \min\{I_v(x,y)S_f(x,y)S_b(x,y), I_s(x,y)\}$$

Analogously to frustum and backface culling, we could account for occlusion culling information if available.

The effect is shown in Fig. 3. The left column shows a uniform mesh covering a simple landscape with a gaussian mountain (top). The uniform sampling arises from a uniform importance distribution (bottom). The importance according to view distance is shown at the bottom of the second column, the mesh resulting from the importance-driven warping process described in the next section is shown above. An importance function which is zero outside the camera frustum distorts the mesh as shown in the third column. The fourth column shows the importance according to the surface orientation. The final importance (last column) combines the previous three importance values.

3.2. Mapping Function

The warped sketch geometry image is a three channel image with height *and* (x,y) values as colors. The warping did not change geometry, it only modified the resolution of the geometry image quad mesh.

The red and green channels of the geometry image represent the warping function Φ, that maps geometry image coordinates (s,t) to world space coordinates (x,y): $\Phi(s,t) = (x,y)$. Φ maps a warping target position to the input position, i.e. it is opposite to the warping direction. We use this definition to avoid ambiguity problems.

The sampling rate of the sketch geometry image in world space are the derivatives of Φ, i.e. $||\Phi_s(s,t)||$ in s- and $||\Phi_t(s,t)||$ in t-direction. Accounting for importance means that we have to choose Φ such that this sampling rate is the reciprocal of the importance at the target point:

$$||\Phi_s(s,t)|| \approx I(\Phi(s,t))^{-1} \quad \text{and} \quad ||\Phi_t(s,t)|| \approx I(\Phi(s,t))^{-1}$$

Because the resolution of the sketch geometry image is fixed, we cannot guarantee the above equation, but we can aim for a sampling rate that is proportional to $I(\Phi(s,t))^{-1}$. In the following we describe how we can compute such a mapping $\Phi(s,t)$.

3.3. Warping

In [SWB98], the importance driven warping is computed using a spring-mass system, where the spring lengths encode the desired importance. This process generates good results, however the relaxation process is iterative and too time consuming for our purpose. Temporal coherence or incremental changes could be used for speed-up in such systems.

Instead, we apply a two-pass approach, where each step distorts along one axis, first row- then column-wise. Our approach is not iterative and thus fast enough to be executed on the CPU once per frame. We explain the row-wise distortion, the column-wise distortion is analogous. The pixel value of the i-th pixel is p_i, its importance is I_i, with $i \in [0; n-1]$.

We consider the pixel row as a piece-wise linear function as in Fig. 4 (top). The warping moves the control points such that the interval around control point p_i gets relative size I_i (second row). The vertical axis in this graph stands for the entire color triple of a point, i.e. its (x,y,z) coordinates, and not for a single height value. So moving a point horizontally does not change its position in world space, it only modifies the available geometry image resolution.

Finally, we resample the warped function uniformly to ob-

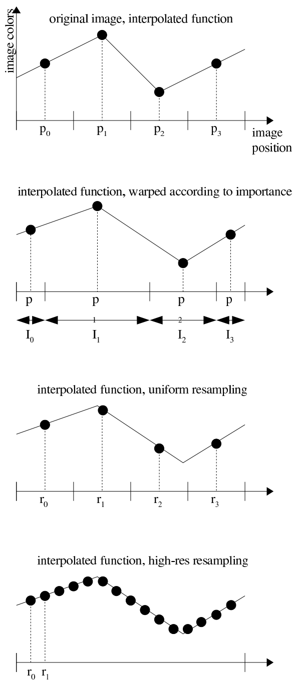
original image, interpolated function
image colors
p0 p1 p2 p3
image position
interpolated function, warped according to importance
p p p p
I0 I1 I2 I3
interpolated function, uniform resampling
r0 r1 r2 r3
interpolated function, high-res resampling
r0 r1

Figure 4: *Importance-driven warping*

```
$a = 0$;  $I_{read} = I_a$;  $p_{prev} = I_0$;

for every pixel $i \in [0; n-1]$:
  $I_{cur} = 0$;  $p_{cur} = (0,0,0)^T$;
  while ($I_{read} < I_{cur}$) {
    $p_{cur} += (p_{prev} + p_a) \cdot \frac{I_{read}}{2}$
    $I_{cur} -= I_{read}$;  $p_{prev} = p_a$;  $a++$;  $I_{read} = I_a$;
  }
  $p_{clip} = p_{prev} + (p_a - p_{prev}) \cdot \frac{I_{cur}}{I_{read}}$;
  $p_{cur} += (p_{prev} + p_{clip}) \cdot \frac{I_{read}}{2}$;
  $I_{read} -= I_{cur}$;  $p_{prev} = p_{clip}$;

  $r_i = \frac{p_{cur}}{I_{avg}}$;
```

Figure 5: *Pseudo code for the row-wise importance driven warping*

tain the warped geometry image with values r_i (third row). In the horizontal distortion step, the importance map is resampled in the same way, in order to have the distorted importance values available for the vertical distortion. Pseudo code to do this warping efficiently is given in Fig. 5.

Of course this simple warping procedure does not result in an optimal warping function. An iterative process of further row- and columnwise warping operations could improve the result. One must consider that a suboptimal grid sampling in some regions only results in reduced quality, because geometric detail must be represented at fragment level. In practice, the effect is not visible, and only appears in very rare cases.

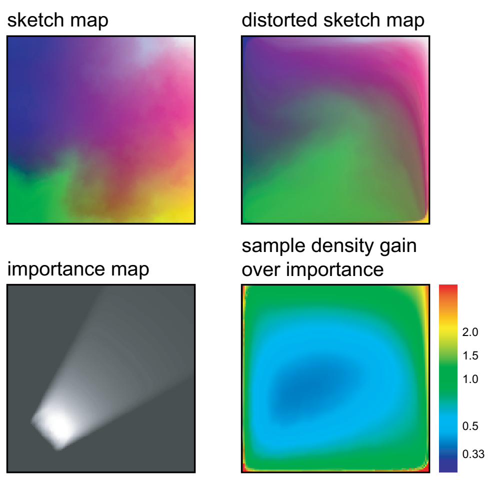

Figure 6: *A geometry image (upper left) is warped according to an importance map (bottom left). In the warped geometry image the ratio between demanded and obtained sample density is between 0.5 and 1 (bottom right).*

In our examples the results were satisfactory for our purpose. Fig. 6 shows the color coded ratio between obtained and demanded resolution for an example terrain. One can see that we obtain the demanded resolution quite uniformly–mostly, the ratio is between 0.5 and 1 (green and cyan).

3.4. Warp and Zoom

In the warping procedure described above, the resulting warped image has the same resolution as the input sketch map, in our implementation this is 128^2 pixels. In this resolution we can do the warping on the CPU in a few milliseconds, so it is fast enough to do it in the rendering loop. However, the sketch geometry image should have higher resolution, e.g. 512^2 to provide mesh cells that only cover a few image pixels.

Furthermore, the resampling during warping smoothes the terrain data, which is also visible in the third row of Fig. 4. Therefore, the increase of the resolution from the sketch map to the sketch geometry image should happen during the warping, as shown in the bottom row of Fig. 4. The warping can be easily modified to generate larger result images, however, the CPU implementation becomes too slow then.

Instead, we compute the warping in low resolution, and repeat the warping by the GPU, this time with increased output resolution. This can be achieved by rendering a uniform quad mesh with the same resolution as the sketch map and the red and green channel of the low-res warped image as texture coordinates.

4. Applying the Procedural Model

In this section, we describe how the procedural features are added. In fact, we apply a procedural model to add displacement and also to attribute the terrain with color. We begin with the description of the geometric procedural model, the simpler color model is then described after that.

4.1. Procedural Displacement

Procedural displacement is partly represented in the geometry of the rendered quad mesh and partly accounted for during per-pixel lighting (bump-mapping). Since the resolution of the rendered quad mesh is locally varying, we need to determine for each quad mesh vertex, which frequency domain of the procedural detail can be represented by a vertex offset and which domain has to be accounted for in the lighting model. This means that we must be able to restrict the evaluation to frequency bands.

As model for procedural detail we chose spectral synthesis of band-limited noise functions (like turbulence or perlin noise). The detail is the sum of noise functions $n_k(x,y)$ ($k \geq 1$) of increasing frequency and decreasing amplitude: $n(x,y) = \sum_{k=1}^{\infty} w_k n_k(x,y)$ In our implementation, the frequency of $n_k(x,y)$ is $2^k f_0$. Because of this frequency doubling (lacunarity of 2), the noise functions are called octaves [EMP*98].

The procedural detail is supposed to generate features that cannot be represented in the original sketch atlas. We thus use as base frequency f_1 the first noise octave that cannot be represented in the sketch map. Thus, if d_0 is the grid distance of the sketch map, we select $f_1 = 1/d_0$.

Low frequencies of the above sum can be represented in the warped quad mesh. The upper frequency bound varies with the quad mesh resolution, so we compute for every quad mesh vertex (i,j) the maximum distance $d(i,j)$ to its four direct neighbors. According to the sampling theorem, we can represent signals up to the Nyquist frequency of $\frac{1}{2d_{i,j}}$ around that vertex. So, for every vertex, we calculate the *split octave* $o_{i,j}$ which describes up to which octave k the procedural detail can be represented:

$$o_{i,j} = \mathrm{ld}\frac{d_0}{d_{i,j}} \tag{1}$$

All signals contained within noise octaves up to $\lfloor o_{i,j} \rfloor$ can be reconstructed with the sampling rate and should thus be added as geometric offset. Detail with higher frequency (and up to screen resolution) must be accounted for in lighting on a per-pixel basis.

In order to achieve a smooth transition between procedural detail accounted for by geometry and by bump maps, we share the contribution of $n_{\lfloor o_{i,j} \rfloor}$ between these two. This method is comparable to the way of anti-aliasing procedural models, also called clamping and fading. It reduces aliasing to a neglible amount, provided that the frequency of the noise octaves is also bounded below. Otherwise, higher octaves may also contain low-frequency information which will only be considered during lighting computation but will no longer affect the geometry.

In all our previous discussions, we assumed that with a sample distance of d, detail up to a frequency of $1/2d$ can be represented. However, this is an upper bound only, and a faithful reconstruction is only possible for significantly smaller sample distances. If this effect is not considered, the procedural geometric detail is "swimming" over the mesh. The effect can be corrected by using more conservative estimates of the split octave in Equ. 1 and shifting more detail to the bump mapping. By this, we cannot avoid swimming artifacts, but reduce them to a tolerable amount.

4.2. Procedural Texturing

The procedural texturing is handled differently. First, for every pixel we determine the surface type like snow, grass, or rock. This can be computed according to the underlying surface (altitude, slope, orientation) or be read from a surface type map. In order to wash out the boundaries, we perturbate the surface position by a precomputed and periodic turbulence function before we evaluate its type. According to the surface type, different color and material parameters are chosen.

Because we use precomputed, periodic turbulence textures in this stage, the frequency clamping is done by the mipmapping hardware. This comes at the price of periodicity in the texture, which is sometimes noticeable. The costly alternative is to compute the turbulence at rendering time.

5. Implementation and Results

We implemented our approach using OpenGL and programmable graphics hardware supporting the render-to-vertex-array functionality. Since these extensions are not yet revised by the OpenGL ARB board we used a preliminary ATI version of this extension.

As sketch atlas we use the GTOPO30 data set of Europe (`http://edcdaac.usgs.gov/gtopo30/gtopo30.html`). This data set of 55 MB exhibits a resolution of 30 arc seconds, which corresponds to about 700m. Out of this sketch atlas resolution, we copy a 128^2 subimage (sketch map) covered by the view frustum. This allows us to cover a visible range of about 90 kilometers.

Next, we calculate the view dependent importance map and compute the distorted geometry image with the warping function in the red and green channel as described above. This step is done on the CPU, and requires only a few microseconds. The resulting warping function texture and the sketch map are transferred to the GPU, where the warping is repeated with higher target image size. Additionally, in this step the split octaves are computed and procedural geometry up to this level is added. The resulting image is the detailed geometry image. The split octave value is stored in the alpha channel.

In another pass, we compute the normals of this detailed geometry image and store the vertex coordinates, normals, and split octaves in so called super-buffers. Super-buffers are generalized memory blocks in video memory which can be used as textures, index or vertex arrays. Thus it is possible to use the calculated coordinates and normals as geometry information for rendering without further overhead.

Finally, the detailed geometry image is rendered as a quad mesh from the viewer's perspective. The split octave is added as vertex attribute and interpolated over the triangle. Procedural detail from the interpolated split octave up to screen resolution is added using bump-mapping in a fragment program.

The main memory consumption only depends on the resolution of the sketch atlas plus temporary space in dimension of a few hundred kilobytes, namely for visible part of the map, the importance map and the warped visible low-res map (all 128^2).

The video memory consumption directly depends on the size of the super buffers used for rendering. We used two 512^2 resolution super buffers, each $512 \cdot 512 \cdot 4 \cdot 4 = 4$ megabytes, and 3 p-buffers, each $512 \cdot 512 \cdot 4 \cdot 4 = 4$ megabytes. Apart from that, the terrain surface types are stored in a simple color texture, whose resolution depends on the input data. The memory requirements for the noise textures are small, we used a 512^2 color texture (8 bits per channels) for storing the noise function.

In our implementation, we use a fairly low resolution sketch atlas as input, so that a 128^2 resolution sketch map is sufficient. This is small enough to do the importance and warping computation on the CPU and to transfer the result to the GPU once per frame. For a higher resolution sketch atlas, a down sampled version of the sketch atlas would be required for fast importance and warping computation on the CPU. This resulting warping function can be used to generate sketch geometry images in arbitrary resolution. For this, the visible part of the sketch atlas must be kept in video memory, which can be achieved by simple memory management.

Our terrain rendering approach can be hardly compared with previous approaches. Our method is focused on procedural detail generation at runtime without precomputations. For displaying a given height field without the addition of further detail, well-known previous approaches are more efficient.

In Fig. 7 one can see snapshots of a flight towards the Alps. On a PC with a 2.4 GHz Pentium and an ATI Radeon 9700 pro, we achieve very constant frame rates of about 35 frames per second for an image resolution of 512^2. Fig. 8 shows how our procedural model augments the original height field.

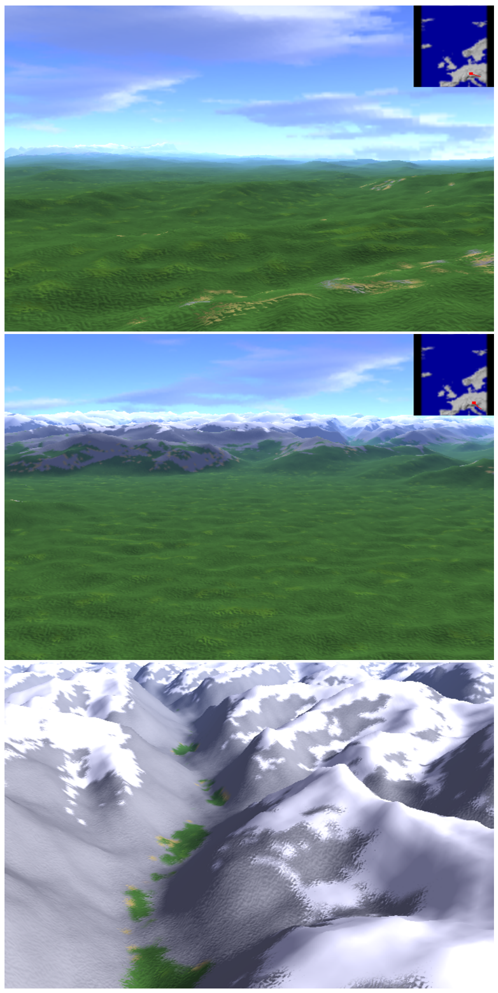

Figure 7: *Snapshots from a flight over the Alps at about 35 frames per second*

6. Conclusions and Future Work

We presented a novel approach for view dependent level-of-detail terrain rendering which requires almost no precalculation and very small storage cost as a consequence of the application of procedural models. The rendering using quadrilateral meshes enables high performance rendering without problems implicated by other level-of-detail methods, like chunk-wise rendering and related connection constraints or t-vertices.

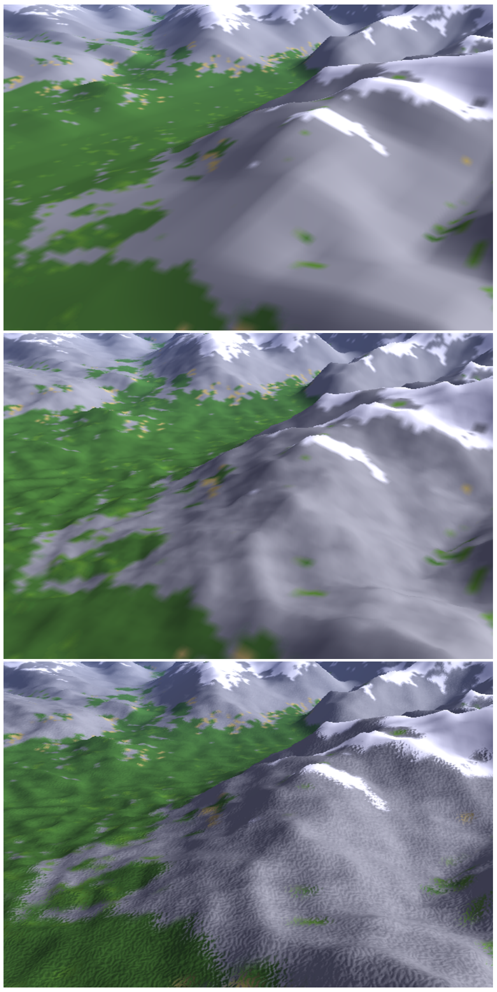

Figure 8: *The procedural model. Top: original height field. Center: procedural detail represented in geometry. Bottom: procedural detail in geometry and lighting.*

The procedural model allows rendering of very large scale terrain, as the high geometric detail is generated on the fly and does not need to be stored explicitly. We achieve load partitioning between CPU and GPU with only moderate data transferred over the system bus. On the downside, since the procedural model is evaluated on the GPU, collision detection requires the re-evaluation of the model on the CPU.

Our method does a resampling of the original data during rendering. This smoothing of the original data can result in swimming artifacts. However, since by the four times upsampling of the sketch atlas these annoying are reduced to a tolerable amount in our tests.

We cannot guarantee any error bounds. Since we always use a quad mesh with fixed topology, cases can be found, where this quad mesh cannot provide sufficient resolution. Our warping process is very simplistic. The separate horizontal and vertical warping delivers surprisingly good results, but it cannot guarantee a close to optimal result.

As future work we want to implement the entire process using graphics hardware. The computation of the importance map can be done very efficiently using fragment programs. The distortion computation requires flexible memory access patterns or a larger number of rendering passes using contemporary graphics hardware. Implementing the warping calculation on the GPU would avoid costly data bus transfers.

The dynamic sampling of a heightmap terrain is also very suitable for point based rendering approaches, where view dependent sampling point densities and sample point sizes are required for rendering.

References

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. *ACM Transactions on Graphics, Proc. SIGGRAPH 2002 21*, 3 (2002), 347–354.

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum 22*, 3 (2003), 505–514.

[DWS*97] DUCHAINEAU M. A., WOLINSKY M., SIGETI D. E., MILLER M. C., ALDRICH C., MINEEV-WEINSTEIN M. B.: Roaming terrain: Real-time optimally adapting meshes. In *IEEE Visualization '97* (1997), pp. 81–88.

[EMP*98] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modelling*. AP Professional, 1998.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. *ACM Transactions on Graphics, Proc. SIGGRAPH 2002 21*, 3 (July 2002), 355–361.

[GH95] GARLAND M., HECKBERT P.: *Fast Polygonal Approximation of Terrains and Height Fields*. Tech. Rep. CMU-CS-95-181, Carnegie Mellon University, 1995.

[Hop98] HOPPE H. H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *IEEE Visualization '98* (Oct. 1998), pp. 35–42.

[LH04] LOSASSO F., HOPPE H.: Geometry clipmaps: Terrain rendering using nested regular grids. In *ACM Transactions on Graphics, Proc. SIGGRAPH 2004* (2004). to appear.

[LKR*96] LINDSTROM P., KOLLER D., RIBARSKY W., HUGHES L. F., FAUST N., TURNER G.: Real-time, continuous level of detail rendering of height fields. In *Proc. SIGGRAPH 96* (Aug. 1996), pp. 109–118.

[RHSS98] RÖTTGER S., HEIDRICH V., SLUSALLEK P., SEIDEL H.-P.: Real time generation of continuous levels of details for height fields. In *Sixth International Conference in Central Europe on Computer Graphics and Visualization* (Feb. 1998).

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Rendering Techniques 2002: Proc. EG Workshop on Rendering* (2002), pp. 87–98.

[SWB98] SLOAN P.-P. J., WEINSTEIN D. M., BREDERSON J. D.: Importance driven texture coordinate optimization. *Computer Graphics Forum 17*, 3 (1998), 97–104.

[Ulr] ULRICH T.:. "Super-size it! Scaling up to Massive Virtual Worlds" course at SIGGRAPH 02". see also `http://tulrich.com/geekstuff/chunklod.html`.