Interactive Landscape Visualization Using GPU Ray Casting

Stephan Mantler* and Stefan Jeschke†

* VRVis Research Center †Vienna University of Technology

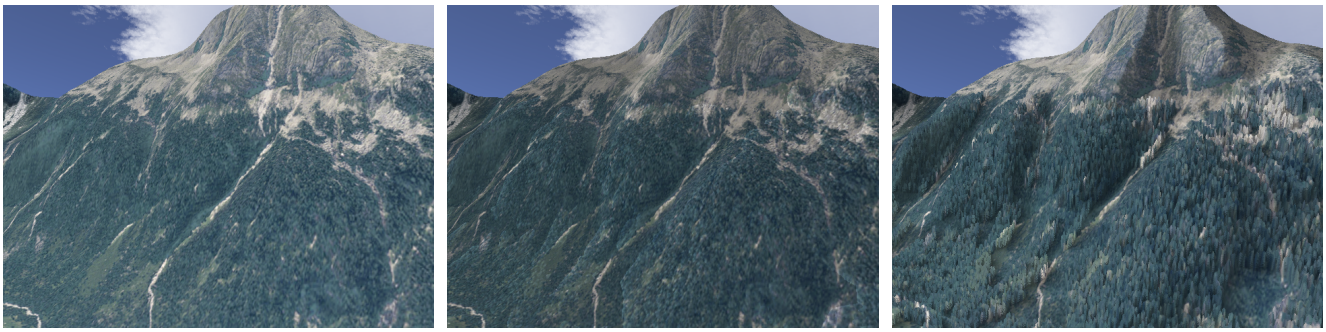


Figure 1: Rendering a landscape with orthophotographic data. Left: textured and unlit. Center: textured and lighted. Right: The GPU ray casting approach presented in this paper.

Abstract

This paper demonstrates the simple yet effective usage of height fields for interactive landscape visualizations using a ray casting approach implemented in the pixel shader of modern graphics cards. The rendering performance is output sensitive, i.e., it scales with the number of pixels rather than the complexity of the landscape. Given a height field of a terrain and a topographic map or similar data as input, the vegetation cover is extracted and stored on top of the height field in a preprocess, enhancing the terrain with forest canopies or other mesostructure. In addition, enhanced illumination models like shadowing and ambient occlusion can be calculated at runtime with reasonable computational cost, which greatly enhances the scene realism. Finally, including the presented technique into existing rendering systems is relatively simple, mainly consisting of data preparation and pixel shader programming.

CR Categories: I.3.3 [Computer Graphics]: Display Algorithms—Real-time rendering; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; I.3.8 [Computer Graphics]: Applications—Landscape rendering

Keywords: Real-time rendering, Landscape Visualisation

1 Introduction

Interactively exploring large landscapes has recently become quite popular. Examples for publicly available and widely used tools are "Google Earth" (`http://earth.google.com/`), "NASA World Wind" (`http://worldwind.arc.nasa.gov/`), "Autodesk Map 3D" (`www.autodesk.de/map`), "3DGeo" (`http://www.3dgeo.de/`) and many more. These systems use images and height data obtained from laser range scanners, GPS data, and satellites to generate and render a textured terrain mesh. Much work has been spent to also represent vertical structures like building facades in the representation, because such structures are typically not immediately available in the acquired data. On the other hand, to the author's best knowledge, vegetation has not been faithfully represented in such systems so far, although this obviously would contribute to a more realistic look.

This paper presents a method for increasing the realism of vegetation rendering in landscape visualization systems. Since in most of the above systems the terrain is internally stored as height map, we directly render the landscape from this height map data using a GPU ray casting approach. The focus of this work is to show the advantages of such a strategy compared to traditional mesh-based representations:

- The rendering time is output sensitive, i.e., it depends on the number of rendered pixels rather than on the complexity of the landscape.
- It is possible to easily "refine" the landscape for a more realistic representation. This means that special surfaces in the landscape can be modeled with higher realism without the need for complex operations like remeshing the scene. An example for this is shown for enhanced vegetation realism: given an input heightfield and an according color map, the technique extracts a special vegetation heightfield from the color data. During the terrain exploration this heightfield is rendered "on top" of the terrain thus providing parallax and occlusions like a real forest canopy. However, please note that height field data inherently becomes visible as such when viewed nearby, and the presented technique is merely thought for distant rendering only, as is typically the case for landscape explorations.
- Global illumination calculations like shadow mapping and ambient occlusion can be easily and quickly performed at runtime and changed dynamically. Furthermore, by exploiting natural implications about the data such as the appearance of trees, stones, glaciers etc., shadow behavior, and light interac-

tions within a landscape, a fairly good image quality can be achieved.

After reviewing related work in the next section, Section 3 describes how the realism of the landscapes can be increased in a preprocess with emphasis on the generation of forest canopy. Afterwards, Section 4 describes the rendering algorithm, including enhanced illumination calculations. In Section 5, results obtained with the described methods are shown and Section 6 concludes the paper and shows roads for further research.

2 Related Work

The work presented in this paper is related to terrain rendering and here specifically height field rendering. Furthermore, several publications about vegetation rendering are also reviewed here.

2.1 Height Field Rendering

Techniques such as bump and normal mapping have been in use to simulate the mesostructure of surfaces for many years [Blinn 1978]. However, they lack parallax and self-occlusions within the surface. Consequently, enhanced techniques like parallax mapping [Kaneko et al.] (optionally with offset limiting [Welsh 2004]) and parallax occlusion mapping [Tatarchuk 2006] were developed that simulate the appearance of a heightfield by shifting the texture coordinates within a texture, depending on the view angle. Unfortunately, these techniques still cannot display correct object silhouettes in all cases.

Kajiya and Kay [Kajiya and Kay 1989] (and later Neyret [Neyret 1998]) first used ray casting of 3D texture maps in order to display fur and other complex surface structures. Recent advances in programmable shaders and increasing GPU performance have allowed implementations with various optimizations that run directly on graphics hardware. As an example, *relief mapping* proposed by Policarpo et.al. [Policarpo et al. 2005] provides believable parallax effects within a surface, but does not render correct silhouettes. This is often not a problem for rendering object surface mesostructure. However, for landscape rendering, the undulating nature of the terrain would result in visual artifacts without correct silhouettes. To overcome this problem, Oliveira and Policarpo [Oliveira and Policarpo 2005] proposed a method that uses quadrics to better approximate height fields on curved surfaces. This method relies on a preprocessing step that computes a quadric approximation of the surface for each vertex and stores its coefficients as additional vertex parameters. Unfortunately, the proposed approximation only works for smooth surfaces, i.e., it fails at sharp edges, and also requires the storage of additional per-vertex data, making it difficult to incorporate in existing systems.

Existing methods that do produce correct silhouettes are usually based on rendering tetrahedra or prisms instead of the original surface to avoid visual artifacts. Examples for tetrahedra-based algorithms are [Hirche et al. 2004; Porumbescu et al. 2005; Dufort et al. 2005]. Methods based on prisms have been proposed by Wang et al. [Wang et al. 2004]. Unfortunately, rendering prisms or tetrahedra significantly increases the polygon count and (even worse) overdraw. A GPU implementation of a ray casting renderer for (terrain) height fields has also been proposed by Qu et.al. [Qu et al. 2003]. Although it lacks the acceleration methods proposed in other works, this approach is the most similar to our method.

2.2 Vegetation Rendering

Given the huge number of publications for rendering vegetation, presenting an exhaustive description of all the work in this field is not possible here. In this section we will present a selection of various techniques that are similar to our approach. For a more complete reference we refer to [Deussen 2003; Mantler et al. 2003].

One of the key publications in this field is the work by Deussen et.al. [Deussen et al. 1998], which was later extended to run interactively [Deussen et al. 2002]. In their system individual plants are placed through a growth simulation, and interactive rendering of the individual plants is performed at multiple levels of detail via a hybrid point and polygon representation.

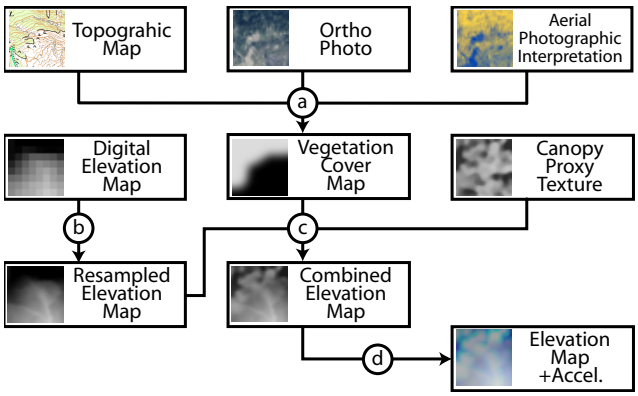
A similar approach has been recently proposed by Gilet et.al. [Gilet et al. 2005], in which the vegetation geometry (individual trees or tree groups) is hierarchically clustered and preprocessed into a triangle+point representation. At runtime, the hierarchy is traversed until the current level can be safely approximated by a point or individual polygons are rendered.

These approaches provide a highly detailed model suitable for close up rendering, but suffer problems when very large areas (ie. forests) need to be rendered: the population of several square kilometers requires an extensive preprocessing step, and the runtime cost of calculating the appropriate level of detail also grows with the number of plant primitives within the scene (although this is less of an issue for Gilet's method, where trees can also be clustered into groups). In both cases, some sort of instancing is required to keep the total amount of vegetation geometry within manageable bounds, and special care must be taken to avoid artifacts, such as trees hovering above or sinking into the ground, if levels of detail are used for terrain rendering. Furthermore, neither approach handles shading and shadowing, which must therefore be handled by other means.

Besides such point based approaches, much work has been done on image-based rendering of vegetation, typically by reusing precomputed textures [Max 1996]. Meyer and Neyret have presented an algorithm that models trees through their bidirectional shading properties [Meyer et al. 2001]. A hierarchy of *bidirectional texture functions* is precomputed and used to render many trees with shading and shadows at interactive frame rates. Decaudin and Neyret [Decaudin and Neyret 2004] have proposed an algorithm that is based on rendering multiple texture layers. Instead of placing individual trees, aperiodic tiling is performed to cover large areas. A similar offline rendering method had been previously demonstrated by Neyret [Neyret 1996]. Volumetric information can be acquired from existing models or through methods such as [Reche et al. 2004].

While producing very good visual results including dynamic illumination, the method by Meyer and Neyret again requires an extensive preprocess (the authors state 75 minutes for 1000 trees on an Onyx^2 InfiniteReality). Runtime performance is also relatively low, although there is no performance data available for current GPU hardware. The rendering system by Decaudin and Neyret uses aperiodic tiling to cover large areas, which would require a number of additional tiles to account for forest boundaries that match a given vegetation cover map. Also, their approach as described does not handle dynamic illumination, although the authors described this feature as future work. It is also unclear how these approaches could be integrated with a terrain level of detail rendering system to cover very large areas.

In contrast to the above mentioned algorithms, our approach requires little preprocessing, is mostly independent of the terrain rendering system, provides shading and shadowing, and achieves interactive frame rates even for large scenes.



a: Identification of Arboreous Regions and Vegetation Cover Map generation.
b: Elevation Data Resampling.
c: Height Map Texture Creation.
d: Calculation of Ray Casting Acceleration Data.

Figure 2: Overview of the preprocessing steps.See text for details.

3 Preprocess: Enhancing Landscape Detail

The goal of the preprocess component is to enhance the realism of a landscape model by adding surface detail in dependence on the represented surface.

There are several possible forms of input data. A DEM (digital elevation map) which contains the height values of the surface obtained, for instance, from SRTM (*Shuttle Radar Topography Mission*) or LIDAR (*Laser Imaging Detection And Ranging*, a time-of-flight based range laser scanning method) data. Along with the DEM there may either exist aerial photographic maps, topographic maps, and/or aerial photographic interpretation data. Note that we assume that these maps are already registered to the DEM.

The idea is to use the additional information to modify the elevation and/or photographic map in order to increase the realism in the scene. In this paper, the focus is put on arboreous regions which constitute a special challenge due to their canopy structure. The goal of the preprocess is to create a *combined elevation map* from the various existing data sources (refer to Figure 2): by exploiting the available information, a *vegetation cover map* is generated that identifies arboreous regions. This map is then used to augment the original DEM with a generic displacement map (called *canopy proxy texture*) in order to geometrically enhance the forest canopy. The resulting *combined elevation map* represents both the terrain and its vegetation cover.

Please note that the basic idea to augment the original scene with a proxy appearance is also applicable to other surfaces like water, rocks, grass, glaciers and so on. Depending on the level of detail within the classification, this can be done generically (*water*, *grass*, *rock* or *trees*), or at a highly specific level (*mixed forest, dominant species larch, 60% closed*). In the latter case the proxy geometry can be varied at a much finer scale than if only a rough distinction is available.

The following subsections describe the individual steps for generating the combined elevation map, including the identification of arboreous regions (Section 3.1), resampling terrain elevation data to an appropriate level and combining terrain elevation with a suitable canopy proxy model to the combined elevation map (Section 3.2). A schematic overview of these preprocessing steps is also depicted in Figure 2.

3.1 Identification of Arboreous Regions

Arboreous regions must be identified in order to generate the vegetation cover map. This can be done through a number of means. If aerial photographic interpretation or other GIS data is available, this typically already contains the required information. Each region within the information database is assigned a classification ID, through which properties such as primary and secondary plant species, height and density can be queried.

If no such data is available, the information can be extracted manually or (semi-) automatically from a given photographic and/or topographic map. For example, there are a number of methods for automatic deriving trees and other features [Straub and Heipke 2001; Gougeon and Leckie 2003]. If the input data is a topographic map instead of photographic material, the forested areas designated in the map can of course also be used in a similar fashion. Figure 5 shows an example of vegetation data derived purely from a 1:50,000 scale topographic map. In this example, the arboreous regions were identified manually and smoothed through a morphological *close* (dilation followed by erosion) operation to combine regions separated eg. by height lines. However, note that algorithms exist to perform this process automatically and to also identify additional features from the map [Arrighi and Soille 1999; Miyoshi et al. 2004]. The resulting *vegetation cover map* contains values between 0 for non arboreous regions and 1 for regions identified as forests.

3.2 Canopy Proxy Models and Height Map Texture Creation

Since the vegetation cover map only describes the presence of vegetation on the map but does not capture the undulating canopy surface typically seen in forests, we use a separate canopy proxy model to achieve this effect. This canopy proxy is essentially a generic, tileable patch of canopy height data based on a LIDAR scan of a deciduous forest. For large areas, we propose a tileable, non-repeating solution such as Wang tiles [Cohen et al. 2003]. Although we only use one canopy proxy because our vegetation cover map does not identify individual species, the identification preprocess could be extended, allowing the use of several distinct canopy proxies for different surfaces, e.g., for deciduous or coniferous forests, or bushes.

In the ray casting step (see Section 4), the combined local height of the terrain and the canopy is required to calculate exact ray-canopy intersection points. To provide this information, the tileable canopy proxy model is multiplied with the vegetation cover map, and the resulting vegetation height added to the local terrain height and stored in the final texture.

An important issue in this context is the (vertical) map resolution. The DEM typically stores height values between 0 and 2000 meters (for high mountains even more), resulting in a vertical resolution of approximately 8 meters for an 8 bit map. Because this is clearly too coarse for modeling a forest canopy (for an example see Figure 3), the map is extended to 16 bit resolution before it is modulated with the proxy map.

Furthermore, since digital elevation data is typically available in resolutions of approximately 10-25 meters at best, it needs to be resampled at a higher (horizontal) resolution to capture individual trees in a canopy with sufficient detail. We have found 0.75 to 1.5 meters to be sufficient for this task. Note that bilinear filtering should be used for this step in order to preserve a smooth surface.

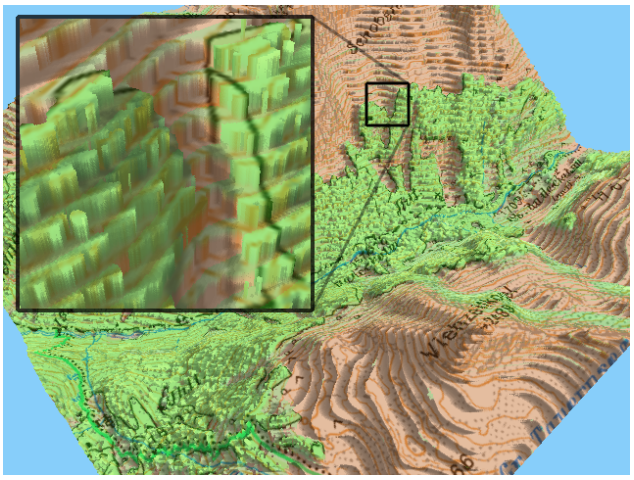


Figure 3: Visual artifacts as a result of insufficient vertical resolution.

One might argue that we could have used two separate textures for the DEM and canopy proxy map and combine them in the fragment shader at runtime in order to save memory. In fact our first prototype system implemented this approach but the resulting shader was significantly (about 4 times) slower than using a single 16-bit texture. We attribute this to having to access two 8-bit textures per ray casting step. Although the total data per access is the same for two 8-bit textures as for one 16-bit texture, accessing a second texture apparently results in much worse cache coherence. Furthermore, the inner loop of the shader becomes more complex (2 texture + 12 arithmetic operations compared to 1 texture + 9 arithmetic operations, a 40% increase). Although this approach significantly increases the required memory, we do not expect this to be an issue for large scale terrain rendering, because such applications typically already support dynamic texture loading and unloading.

4 Runtime: Interactive Landscape Rendering

Given the DEM, enhanced with the canopy geometry as described in Section 3, the goal is to interactively display the landscape with realistic illumination. The rendering algorithm is based on per pixel ray casting using the GPU.

Due to its high resolution (at 0.5m resolution, a $1km^2$ area amounts to 2000x2000 samples, or roughly 8 million triangles) it is not practical to directly convert it to a triangular mesh for rendering. Of course any number of geometric reduction algorithms could be applied to create a less detailed representation. However, retaining full detail is typically desirable, eg., to identify outliers and systematic errors from the data acquisition process. We therefore propose to render a very coarse approximation as a convex hull, and to 'fill in' details as needed through an exact ray casting algorithm. Furthermore, the presented pixel shader approach allows an easy adaptation to different rendering styles and illumination methods (just by slightly changing the shader program). Note that this would be much more tedious with a fully geometric representation.

Section 4.1 discusses the choice of the geometry to be rendered in order to generate the fragments necessary for the ray casting. Section 4.2 describes the basic ray casting step in more detail and Section 4.3 discusses the avoidance of rendering artifacts that are often noticeable in previous approaches. Section 4.4 discusses possibilities how to accelerate the ray casting step and Section 4.5 describes how advanced illumination techniques like shadow mapping can be implemented.

4.1 Choice of Rendering Primitives

At runtime a low-polygon approximation of the terrain is sent to the graphics card. The generated pixel fragments are used as starting points for ray casting within the pixel shader (see Section 4.2). The ray caster then uses the canopy map to derive exact visibility and optionally compute surface color and illumination.

Since the geometric representation of the terrain is only used to create the fragments for the pixel shader ray caster, it can be relatively coarse. In fact, even rendering a simple plane in front of the camera would be sufficient for this task. On the other hand, the geometry defines the starting point for the ray caster in the map (see Section 4.2). It is desirable to generate a starting point close to the landscape surface so that an intersection of the viewing ray with the surface can be found quickly. Many landscape exploration systems already use geometric levels of detail for the terrain, and fortunately these approximations always results in starting points close to the surface.

Since the geometry must encompass the terrain as well as the vegetation on top of it, an appropriate vertical offset is necessary. This can be easily performed in the vertex shader, which also allows a smooth transition to other rendering methods that are not based on displacement mapping.

4.2 Basic Algorithm: Per Pixel Ray Casting

We use a ray casting step that intersects the viewing ray against the combined elevation map, similar to the one described in previous work [Qu et al. 2003; Hirche et al. 2004; Policarpo et al. 2005]. Every vertex is assigned a 2D texture coordinate for the combined elevation map. The per fragment 2D texture coordinate (generated during the hardware rasterization) is the starting point for the ray casting algorithm in texture space. For simplicity, we define the landscape in world space over a (x,y) plane and z points up. The 2D direction in texture space is then simply defined by the (x,y) coordinates of the viewing ray in world space. Beginning with the starting point, the ray caster compares the value (the *height*) in the combined elevation map with the respective z value of the viewing ray at that position. If the ray is *lower* than stored height, an intersection with the heightfield occurred and the ray caster stops. Otherwise, the next position along the ray is evaluated using a user-defined distance from the current position. This process is repeated for a user-defined number of steps. If the surface has not been hit, the fragment is clipped in order to provide correct landscape silhouettes.

4.3 Rendering Quality Considerations

Per-pixel displacement mapping algorithms are often associated with rendering artifacts such as texture distortion and/or incorrect silhouettes. For correct ray casting of a displacement map on arbitrary objects, the ray would need to be distorted appropriately to capture the curvature of the object. This problem is typically solved by using a piecewise linear (or, in the case of [Oliveira and

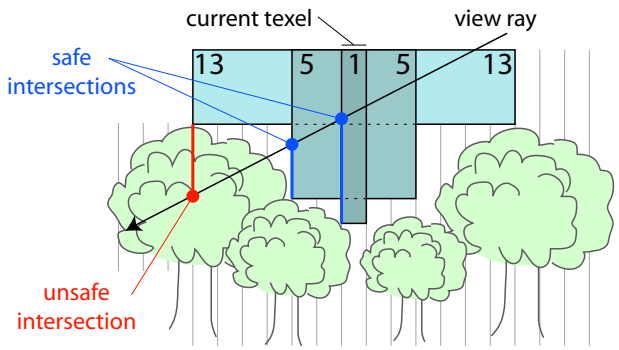


Figure 4: Ray casting acceleration. In addition to the local height (=1 texel safety zone), two safety zones (brighter green boxes) are stored for each texel. The maximum step size is determined by the largest box that is not intersected in its floor.

Policarpo 2005], quadric) approximation for each individual triangle of the object. Triangles of the original mesh are rendered as prisms or tetrahedra to guarantee continuity and to avoid artifacts due to missed intersections. The use of these primitives also allows the computation of the exit point of a ray from the current prism or tetrahedron. By knowing both the start and exit point, the number of ray casting steps required for full coverage can be easily computed.

In our case, object curvature issues are obviated, because the displacement map stores absolute height information relative to a ground plane, i.e. the original mesh can be regarded as a plane with zero curvature. Therefore, no conversion to prisms or tetrahedra is required. However, given a very large terrain the number of ray casting steps required for full coverage of the terrain may be excessively large. To avoid this problem, we subdivide the terrain into tiles that are rendered individually. Each tile has a vertical boundary 'skirt' to avoid missed intersections, which can be directly compared to using prisms on a per-triangle level. Fortunately, similar skirts are also used in many tile based terrain rendering techniques to hide T vertices and other tiling artifacts [**?**; NASA], and such geometry could be re-used directly for our approach.

Also note that the distance between two ray casting steps as well as the overall number of steps provide simple means for balancing rendering speed and image quality.

4.4 Ray Casting Acceleration

Our per-pixel ray caster is based on the acceleration scheme proposed by Kolb et.al. [Kolb and Rezk-Salama 2005]. For every texel, a so-called *safety zone* is defined, being the largest height value within a user-defined radius around that texel (see Figure 4). The safety zones for all texels are computed and stored in a preprocess. At runtime, the minimum height of a ray within the radius of the current texel is computed using some simple (and fast) math. If this minimum height of the ray within the radius is above the maximum height, it is guaranteed that the ray does not intersect the height field within the current safety zone and the next sampling point can be placed just outside the zone instead using a constant distance as was described in Section 4.2. It is also possible to precompute *multiple* safety zones per pixel and to evaluate them in parallel at runtime by exploiting the vector arithmetic of graphics hardware. As an example, in Figure 4 the ray intersects the two inner safety zones (the height of the current texel as well as the first (5 texel) safety zone) at their sides (blue dots, respectively) whereas the second (13 texel) safety zone is intersected through the floor (red dot). Therefore, the largest safe step size is determined by the medium box (left blue dot). Although Kolb et al. use three parallel safety zones, experiments showed that the performance gain of two safety zones is hardly lower compared to three zones, so we only used 4 and 32 texel safety zones. The last component was then used for the illumination calculation (see Section 4.5).

4.5 Illumination Calculation

In some situations it might be desirable to relight the scene, for instance, when the sun should be simulated over a whole day. In order to get decent lighting, we first experimented with recovering local normal vector information and calculating diffuse illumination in addition to ambient occlusion and self shadowing. Unfortunately, this resulted in a quite unrealistic, 'plastic' look of the forest. We attribute this to the fact that the reflectance properties of forest canopies are hard to capture with a diffuse reflection model, and just using an ambient solution seems to be the best compromise if short of using a BRDF from measured data [Martens et al. 1991]. Consequently, the color read from the orthophoto map (or, respectively, topographic map) at the intersection between view ray and elevation map is defined as base color.

In order to shadow the scene, the ray casting approach allows for a simple and efficient shadow computation which avoids implementing shadow mapping or shadow volumes [Akenine-Moeller et al. 2004]. Beginning from the intersection between view ray and elevation, a ray is constructed to the light source (several light sources are of course also possible, requiring an individual ray to every light). The ray casting process is then basically performed as was described in Section 4.2. If the ray intersects the scene, the current pixel is in shadow. Otherwise, it is lit. Unfortunately, we have found that in the case of forests and trees, this produces very hard shadows that look not quite natural. Instead, we accumulate shadowing over several steps similar to ray casting of volumetric data [Kajiya and Herzen 1984] and in the spirit of the *deep shadow map* technique by Lokovic and Veach [Lokovic and Veach 2000]. To also capture the contribution of distant scene parts (such as distant mountains), the step size is increased linearly, such that at close range, occluders are finely sampled and cast darker shadows whereas more distant occluders have a smaller contribution to the final shadowing. This results in visually pleasing shadowing, and the diminishing contribution can be likened to a pseudo ambient occlusion solution (where more distant occluders are outweighed by a dominating ambient illumination). See also Figure 6.

Also note that typical orthophoto maps already contain shadows, which must be removed beforehand. We redirect the reader here to Premoze et al. [Premoze et al. 1999] whose discuss efficient methods for this task. Figure 7 shows an example for this.

5 Results

We have implemented our method as a C# / Managed DirectX application using HLSL shader model 3.0 shaders. Screen shots and performance figures were taken on a 3.2GHz Pentium D PC with 1 GB RAM and an ATI Radeon X1900 XTX GPU with 512MB DDR3 memory.

For evaluation, we acquired sample data of a 3.5km x 3.5km section of the Stillwald area situated in the Nationalpark Hohe Tauern in

Austria. A digital elevation model, topographic map data and an aerial photography interpretation was available for the entire area, whereas the orthophotographic data at hand only covered the inner 2.5km x 2.5km subset. The original DEM data had a resolution of 10m (350x350 texels) and the orthophotographic data was provided at a resolution of 1.25m (2000x2000 texels).

Our canopy proxy texture was derived from a LIDAR scan of a 500m x 500m coniferous forest. The scan data was converted to an elevation map with 0.5m resolution (1000x1000 texels) and afterwards manually converted into to a large, tileable texture. The output resolution of our canopy proxy was 2048x2048 texels for the entire area (1.7m resolution), which was also the resolution of the combined elevation map.

The aerial photography interpretation included a highly detailed classification including vegetation density, dominant and subdominant species and underlying terrain structure. In theory this could be used to synthesize a highly detailed combined elevation map that also differentiates between various tree species, coppice, rocks and grassland. However since we only had a single canopy proxy texture we simplified the classification to only discern between arboreous and other areas.

Although aerial photography interpretation is based on orthophotographic data, the identified areas did not correspond *exactly* to the provided orthophoto. We attribute this to the interpretation being a partially manual process, such that small border details may be missed or consciously omitted. Similarly, the topographic map is an even stronger abstraction of reality where the correspondence of identified forests to 'reality' is an even coarser approximation.

5.1 Visual Quality

The main benefit of our rendering method is the visual detail provided by the forest canopy. Effects such as correct parallax are hard to capture in screen shots, however for example Figure 5 shows how a terrain textured with a topographic map can be visually enriched.

Figure 1 displays (from left to right) the same view rendered as a plain textured landscape, with textured with illumination, and with our approach. The benefits of adding vegetation to the rendering are clearly visible. In Figures 1 and 9, slight misregistration artifacts between the orthophoto and the forest canopy can be observed. This is a result of the approximate nature of the (manual) aerial photographic interpretation used for these models, which did not capture the exact forest boundaries.

In Figure 10, the orthophotographic information was embedded in a larger area and additional topographic data was used to create a canopy model that covers both the detailed orthophoto and the surrounding area. This is a versatile means of providing a meaningful setting for detailed landscape data.

Examples for dynamic illumination are also presented in Figures 6 and 7. For Figure 7, the original orthophoto contained illumination information which was removed manually. Of course this is not practical for larger areas, however automatic algorithms exist for this task [Premoze et al. 1999].

The performance / quality tradeoff of our approach is illustrated in Figure 8. The left image is rendered with 30 ray casting steps and exhibits some artifacts near the horizon as depicted in the magnified inset. On the right, a higher step count (70) is used, resulting in a much higher visual quality.

	2048x2048			4096x4096		
	10	20	30	10	20	30
30	35.4	29.0	24.1	25.3	20.7	19.1
50	32.2	27.0	22.6	23.1	19.1	16.2
70	30.1	26.0	21.9	22.1	18.4	15.5
90	29.8	25.1	21.7	21.5	18.0	15.4

(down : intersection steps. across : shadow steps)

Table 1: Average frames per second for 2048x2048 texels and 4096x4096 texels combined elevation maps with various values for ray casting intersection and shadow accumulation steps.

5.2 Performance Evaluation

To evaluate the rendering performance of the proposed method, we defined a camera path through the terrain and calculated average frame rates for various parameter values. Note that frame rate variance was low because the performance mostly depends on the number of rendered pixels which was constantly high. All frame rates were measured with the application running full screen at 1280x1024 pixels. Given the output sensitivity of our approach, smaller screen sizes (e.g. in a windowed application) have correspondingly higher performance. For example, we achieved an average frame rate of 15.4 frames per second (fps) in full screen mode for 90 view ray intersection steps + 30 shadow ray intersection steps, and in windowed mode (640x480 pixels) with the same parameters the application ran at 24fps.

The lowest setting that produced visually acceptable results in full screen mode used 30 steps for intersection calculation, and 10 steps for shadowing accumulation (see Table 1). With a 2048x2048 texels combined elevation map, we achieved an average frame rate of 35fps. In contrast, a very high setting (70 intersection steps) still ran at 30fps. Only after adding significantly more shadowing accumulation steps (70 + 30), the frame rate dropped to approximately 22fps.

We did also experiment with higher resolution canopy proxy maps and combined elevation maps (0.85m, resulting in 4096x4096 texels). However, this did hardly improve the visual quality while causing a significantly reduced rendering performance (see Table 1) due to an increased number of necessary ray casting steps in order to calculate the view ray/elevation map intersection point. In addition, higher resolution elevation maps increase the time required for preprocessing (between 80 and 180 seconds). However, the required time can still be called fairly short, and our preprocessing code leaves much room for further optimizations if preprocessing time becomes an issue.

6 Conclusions and Future Work

We have demonstrated an interactive technique for realistic rendering of landscape data. The work focussed on enhanced representations for vegetation without the need to model each plant individually. Instead, the approach combines a generic tileable canopy proxy model with landscape specific information to create a height map that can be rendered directly on the GPU. Although in theory it may be possible to achieve a comparable approximation of vegetation by creating a purely geometric approximation (ie. a suitably fine, adaptively tesselated mesh), we believe that our solution has several advantages over such an approach. Its integration into existing terrain rendering frameworks is quite straightforward, only requiring one additional texture and the vertex and pixel shader.

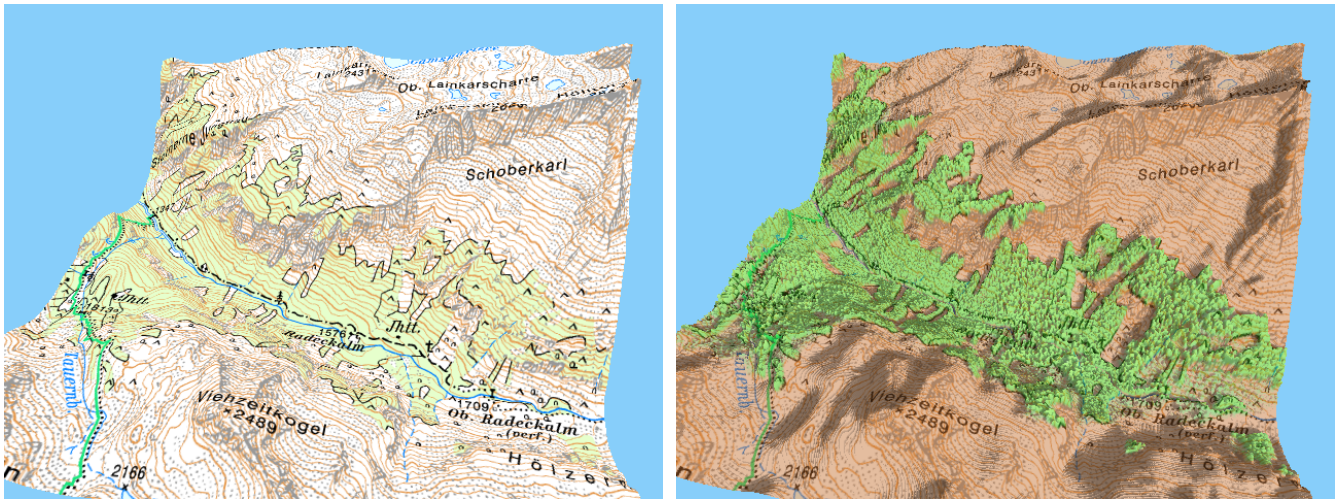


Figure 5: Comparison of topographic map and our visualization derived from the same map

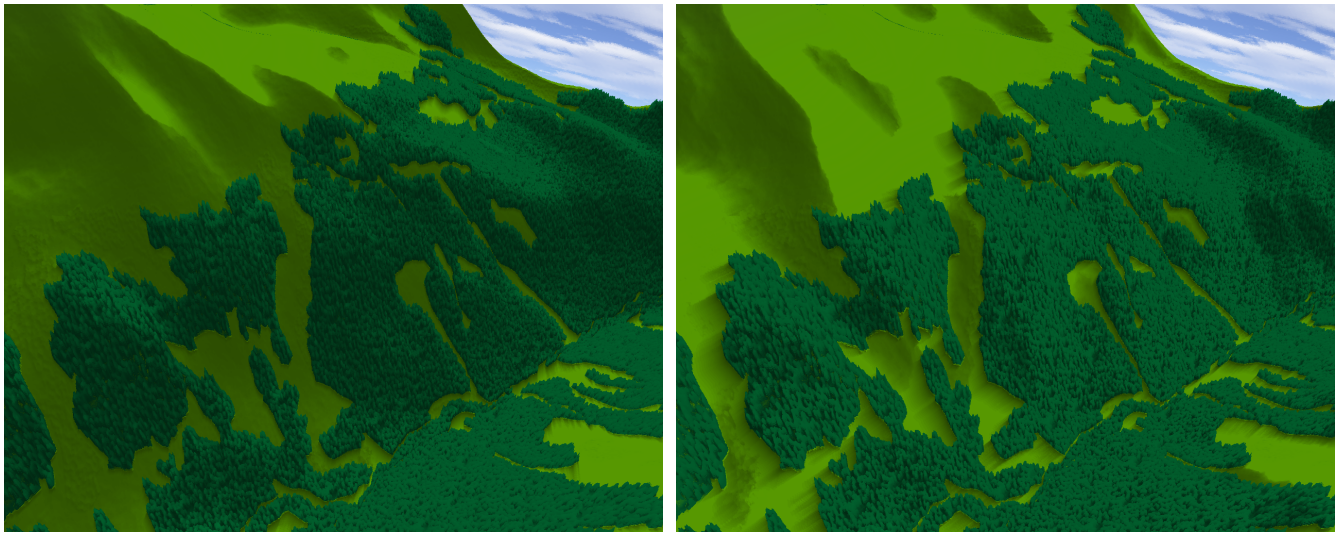


Figure 6: Example for dynamic coloring and illumination of a scene derived from a topographic map.

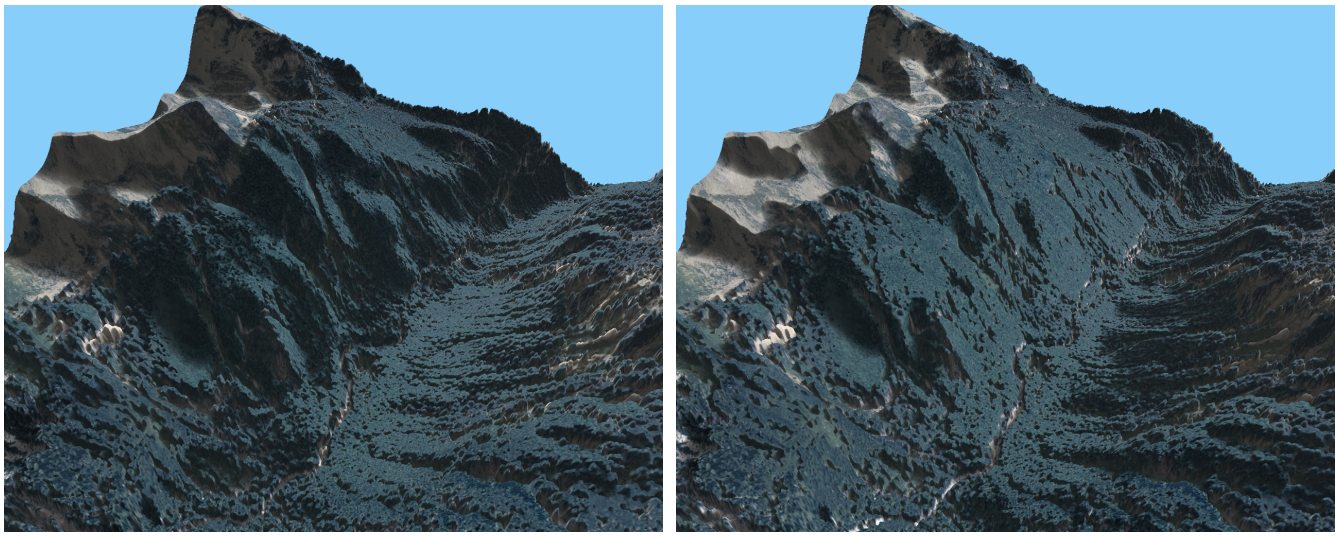


Figure 7: Dynamic illumination of a landscape with orthophotographic data

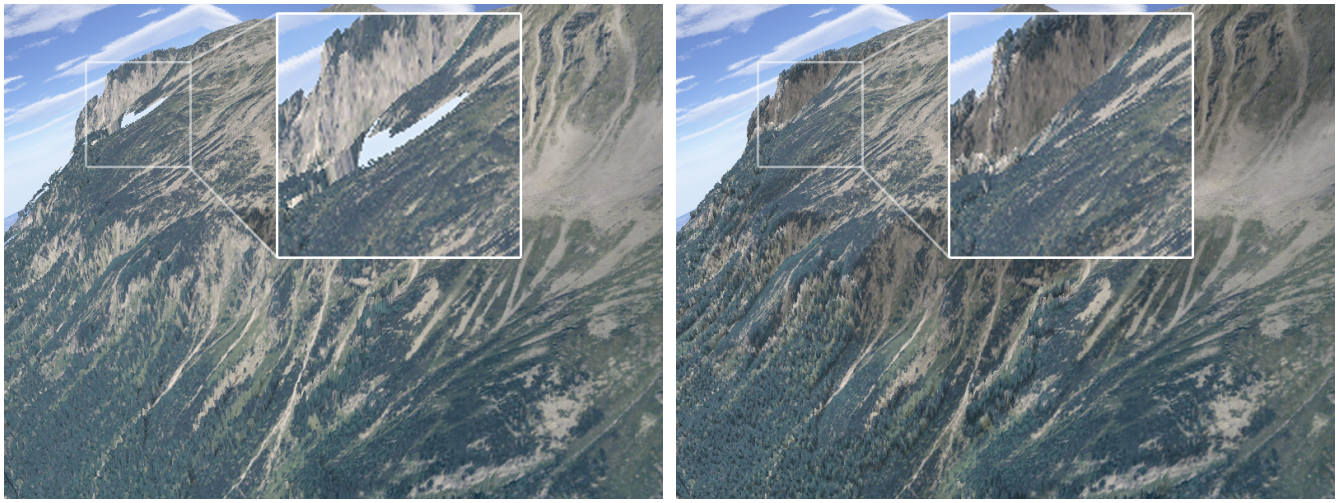


Figure 8: Artefacts caused by too few ray casting steps (left) and the same scene rendered with a higher number of steps (right)

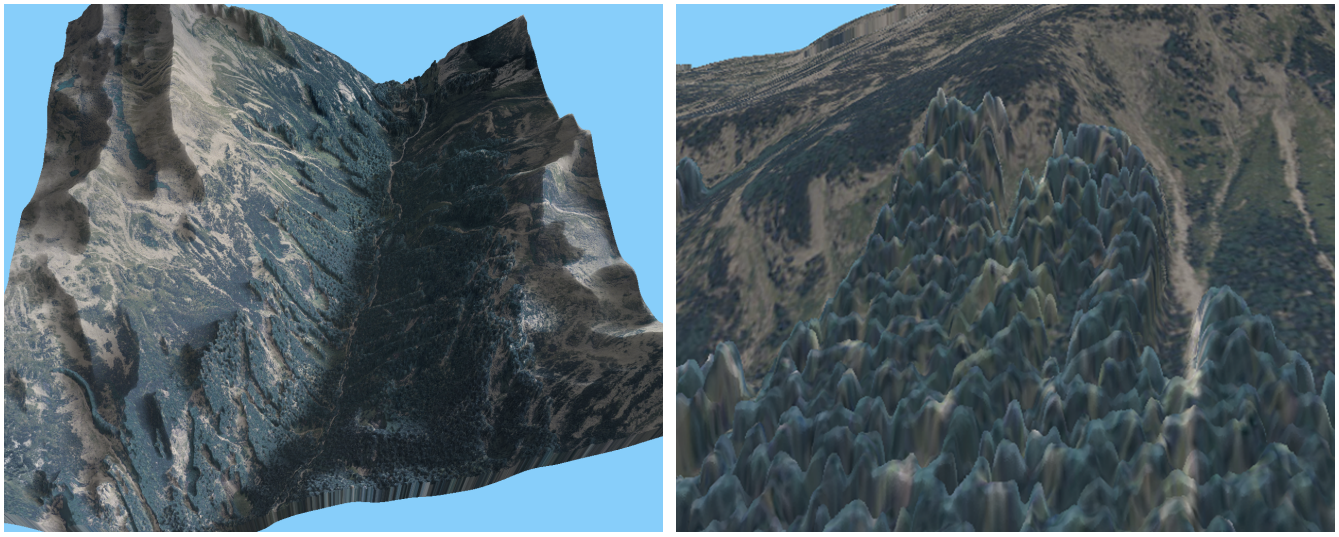


Figure 9: (left) Visualization derived from orthophotos, aerial image interpretation and digital elevation model. (right) close-up view.

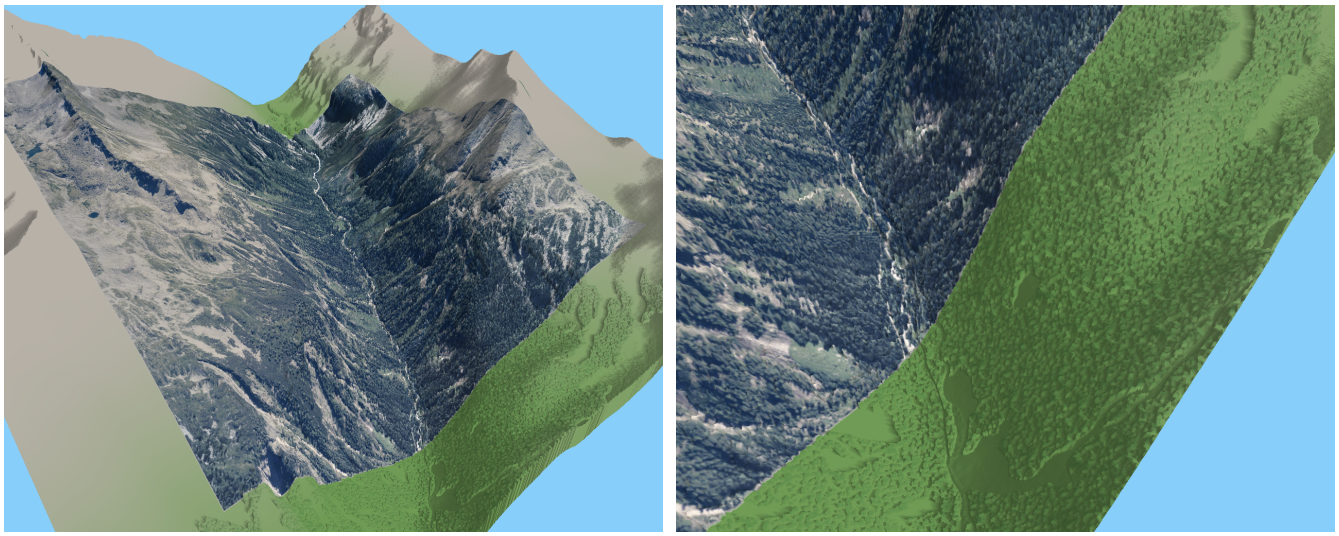


Figure 10: examples of an orthophotographic image embedded in a larger model, with continuous canopy information.

Changing the appearance of the rendition is very simple by adapting the shader program. Also note that the presented approach requires a fairly short preprocessing time and the ray casting shader implementation is relatively simple.

In terms of future work we are currently working on integrating the algorithm into NASA's open source World Wind application [NASA]. This will also demonstrate the applicability of our approach to large scale terrain data and LOD rendering methods. In large scale terrain rendering, distant areas are typically rendered with coarser geometry and lower resolution textures. Using smaller textures instead of mipmapping significantly reduces the amount of GPU memory required for distant areas, and the textures are dynamically replaced by full resolution data as needed. We believe that our method could be easily adapted to such a system, and lower resolution textures would require fewer ray casting steps for full coverage and thus result in higher render performance for the far field.

Other ongoing research includes ways to enhance the technique for close-up views, where the limitations of our approach become obvious. One possible way would be the capability of selectively 'hiding' very close areas of vegetation, which would then be represented with detailed geometric models. In order to avoid visual artifacts, this would require good correspondence between the vegetation map and the detailed models.

Our approach could also be extended for visualizing other terrain features such as grass and shrubs, rocks, snow cover and glaciers similar to the work by Premoze et.al. [Premoze et al. 1999]. It would also be interesting to investigate how this could be adapted for dynamic surfaces such as water bodies.

Non-photorealistic rendering methods could also be implemented to distinguish regions with special properties, for example, forest areas to be cut or reforestation areas or to display additional data (vegetation health, soil/air quality, ...). This could also be generalized to rendering data in a time dependent fashion. It might be possible to use a 3D texture for this task and exploit hardware interpolation for smooth blending over time.

Finally, we are investigating other ray casting acceleration schemes that may be more efficient for our kind of data, and extending our data to have multiple canopy proxy maps to be able to distinguish between tree species and also integrate other structures such as bodies of water, rocks, glacier and buildings.

7 Acknowledgements

Terrain and vegetation data courtesy of the Nationalpark Hohe Tauern administration. LIDAR scan data courtesy of Sorin Popescu, Texas A&M University Spatial Sciences Laboratory. This publication was supported by the Austrian Science Fund (FWF) contract no. P17260 and P17261-N04.

References

AKENINE-MOELLER, T., CHAN, E., HEIDRICH, W., KAUTZ, J., KILGARD, M., AND STAMMINGER, M. 2004. Real-time shadowing techniques. In *SIGGRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes*, ACM Press, New York, NY, USA, 27.

ARRIGHI, P., AND SOILLE, P. 1999. From scanned topographic maps to digital elevation models. In *Proc. of Geovision'99: International Symposium on Imaging Applications in Geology*, Universit de Lige, Belgium, D. Jongmans, E. Pirard, and P. Trefois, Eds., 1–4.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 286–292.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3, 287–294.

DECAUDIN, P., AND NEYRET, F. 2004. Rendering forest scenes in real-time. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, 93–102.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MěCH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 275–286.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. In *IEEE Visualization '02*.

DEUSSEN, O. 2003. *Computergenerierte Pflanzen*. Springer Verlag.

DUFORT, J.-F., LEBLANC, L., AND POULIN, P. 2005. Interactive rendering of meso-structure surface details using semi-transparent 3d textures. In *Proc. Vision, Modeling, and Visualization 2005*, 399–406.

GILET, G., MEYER, A., AND NEYRET, F. 2005. Point-based rendering of trees. In *Eurographics Workshop on Natural Phenomena*, P. P. E. Galin, Ed.

GOUGEON, F., AND LECKIE, D. 2003. Forest information extraction from high spatial resolution images using an individual tree crown approach. PFC Information Report BC-X-396, Canadian Forest Service.

HIRCHE, J., EHLERT, A., GUTHE, S., AND DOGGETT, M. 2004. Hardware accelerated per-pixel displacement mapping. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 153–158.

KAJIYA, J. T., AND HERZEN, B. P. V. 1984. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 165–174.

KAJIYA, J. T., AND KAY, T. L. 1989. Rendering fur with three dimensional textures. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 271–280.

KANEKO, T., INAMI, M., KAWAKAMI, N., YANAGIDA, Y., MAEDA, T., TAKAHEI, T., AND TACHI, S. Detailed shape representation with parallax mapping.

KOLB, A., AND REZK-SALAMA, C. 2005. Efficient empty space skipping for per-pixel displacement maps. In *Proceedings of the VMV 2005 Conference*.

LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. In *SIGGRAPH '00: Proceedings of the 27th annual confer-*

ence on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385–392.

MANTLER, S., TOBLER, R. F., AND FUHRMANN, A. L. 2003. The state of the art in realtime rendering of vegetation. Tech. rep., VRVis Zentrum fr Virtual Reality und Visualisierung Forschungs GmbH.

MARTENS, S., USTIN, S., AND NORMAN, J. 1991. Measurement of tree canopy architecture. *Journal of Remote Sensing 12*, 1525–1545.

MAX, N. 1996. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *Rendering Techniques '96 (Proceedings of the Eurographics Workshop on Rendering 96)*, Springer-Verlag Wien New York, X. Pueyo and P. Schröder, Eds., Eurographics, 165–174. ISBN 3-211-82883-4.

MEYER, A., NEYRET, F., AND POULIN, P. 2001. Interactive rendering of trees with shading and shadowing. In *Workshop on Rendering*, Springer-Verlag Wien New York, Eurographics.

MIYOSHI, T., LI, W., KANEDA, K., YAMASHITA, H., AND NAKAMAE, E. 2004. Automatic extraction of buildings utilizing geometric features of a scanned topographic map. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3*, IEEE Computer Society, Washington, DC, USA, 626–629.

NASA. World wind homepage. http://worldwind.arc.nasa.gov/.

NEYRET, F. 1996. Synthesizing verdant landscapes using volumetric textures. In *Eurographics Rendering Workshop 1996*, Springer Wein, New York City, NY, X. Pueyo and P. Schröder, Eds., Eurographics, 215–224. ISBN 3-211-82883-4.

NEYRET, F. 1998. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 55–70.

OLIVEIRA, M., AND POLICARPO, F. 2005. An efficient representation for surface details. Tech. Rep. RP-351, Instituto de Informática UFRGS.

POLICARPO, F., OLIVEIRA, M. M., AND COMBA, J. L. D. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 155–162.

PORUMBESCU, S. D., BUDGE, B. C., FENG, Z. L., AND JOY, K. I. 2005. Shell maps. *ACM SIGGRAPH 2005, ACM Transactions on Graphics 24*, 3, 626–633.

PREMOZE, S., THOMPSON, W. B., AND SHIRLEY, P. 1999. Geospecific rendering of alpine terrain. In *Rendering Techniques '99, Proceedings of the Eurographics Workshop in Granada, Spain, June 21-23, 1999*, Springer, D. Lischinski and G. W. Larson, Eds., 107–118.

QU, H., QIU, F., ZHANG, N., KAUFMAN, A., AND WAN, M. 2003. Ray tracing height fields. *cgi 00*, 202.

RECHE, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings) 23*, 3 (July).

STRAUB, B.-M., AND HEIPKE, C. 2001. Automatic extraction of trees for 3d-city models from images and height data. *Automatic Extraction of Man-Made Objects from Aerial and Space Images III*, 267–277.

TATARCHUK, N. 2006. Dynamic parallax occlusion mapping with approximate soft shadows. In *Proceedings of Symposium on Interactive 3D Graphics and Games*, 63–69.

WANG, X., TONG, X., LIN, S., HU, S.-M., GUO, B., AND SHUM, H.-Y. 2004. Generalized displacement maps. In *Rendering Techniques*, Eurographics Association, A. Keller and H. W. Jensen, Eds., 227–234.

WELSH, T. 2004. Parallax mapping with offset limiting: A per pixel approximation of uneven surfaces. Tech. rep., Infiscape Corporation.