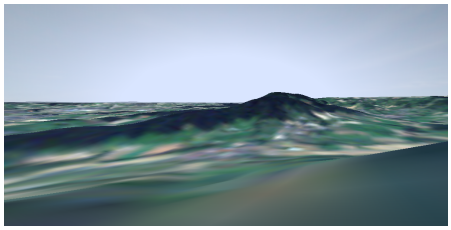
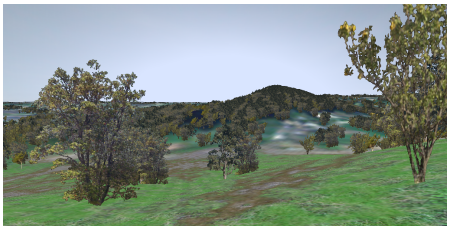
NDVI-BASED VEGETATION RENDERING

Stefan Roettger
Computer Graphics Group, University of Erlangen
www.stereofx.org, stefan@stereofx.org



plain terrain



with NDVI-based vegetation

ABSTRACT
The area of terrain rendering has seen great improvements both in rendering performance and image quality. The latest terrain rendering algorithms efficiently utilize the capabilities of actual programmable graphics hardware in order to achieve real-time visualizations of large terrain models. Despite these advances, a remotely sensed dataset will not look realistic unless the terrain model also accounts for the vegetation. We present an extension of the C-LOD technique that is able to enrich large GIS scenes with forestal detail by rendering trees and bushes in a view-dependent way. The placement of the trees is derived automatically from real NDVI data. We also present a NDVI based texture splatting method in order to render the underlying grass layer in a realistic way.

KEYWORDS: Terrain Rendering, Continuous Level of Detail, Vegetation Rendering, Texture Splatting.

1 Introduction and Previous Work

Terrain rendering has a long tradition now. Ten years back the publication of the so called continuous level of detail (C-LOD) method [12, 8] enabled the display of very large landscapes. These approaches use a very fine grained triangulation that is driven by the screen space error of the coarsened mesh. This means that objects that are far away from the viewer are represented at a lower level of detail than objects that are nearby.

C-LOD methods are typically limited by the speed of the CPU, so that recent advances [4, 2, 13] try to utilize the performance of programmable computer graphics hardware. This led to algorithms that nowadays easily achieve triangle counts of several tens of millions triangles per second.

Nevertheless, in order to achieve a high degree of realism plain terrain rendering is not sufficient. Man-made objects and the various types of vegetation such as grass, bushes and trees have to be considered as well (compare title figure, which shows the "Hetzlas" in Mid Frankonia). A wide variety of methods exists that are able to model virtual landscapes [9] and plant ecosystems [7, 3] in a very realistic way. Due to the huge amount of detail of an ecosystem it is very difficult to apply these techniques to the large scenes that are visible in a typical GIS scenario. Several approaches have been made to increase the rendering performance: point based [6], volume rendering based [5] and billboard based [1].

2 NDVI Measurement

While the described approaches try to model vegetation in a procedural or stochastical way few approaches are known that can display the vegetation of a real GIS scene located somewhere on earth. Partly this was due to the restricted availability of data that describes the distribution of vegetation in a specific area (compare [19]). With the availability of GLCF data [10] Landsat satellite imagery with a worldwide resolution of up to 30 meters is available for free.

The latest Landsat ETM+ satellites (Enhanced Thematic Mapper) provide spectral bands 1-8 ranging from visible to infrared wave lengths. The Landsat channels 3 (red) and 4 (near infrared) can be used to compute the so called NDVI (the Normalized Difference Vegetation Index). Living vegetation absorbs light in the frequency range of band 3 but shows almost no absorption in the range of band 4. So from the intensity difference of the two bands we can easily measure the activity of vegetation A_{NDVI} using the following standardized formula:

$$A_{NDVI} = \frac{I_4 - I_3}{I_4 + I_3}$$

Vegetation usually has NDVI values in the range from 0.1 to 0.7. Higher index values are associated with higher levels of healthy vegetation cover, whereas clouds and snow will cause index values near zero, making it appear that the vegetation is less green (see Figure 1).

Figure 1. Left: Landsat channels 1-3. Center: NDVI (channel 4/3). Right: Corine Land Cover Classification.

Commercial sources also provide land cover maps and vegetation indices, but these are usually hand-processed NDVI or related data. Due to the intensive manual work, on the one hand the correspondence with the real vegetation is usually good but on the other hand the resolution is fairly low. Therefore, important details like highways cannot be captured which is unacceptable for interactive walk-throughs (compare Corine Land Cover Classification with 100m base resolution on the right side of Figure 1).

In the following we describe a novel method that uses the original hi-res NDVI data to enrich real scenes with the corresponding real vegetation.

3 Large Scale Forest Rendering

The NDVI data is available from the GLCF as tiled grey scale images that cover a geo-referenced area. Together with the DEMs (digital elevation maps) from the SRTM (Space Shuttle Radar Mission) and the visible channels 1-3 of the Landsat data we effectively have a high resolution landscape and vegetation description of any particular area in the world.

Since the data is overlapping each other, the first step to visualize the scene is to resample the data. We use the libMini [11] terrain rendering library which is based on the C-LOD algorithm described in [15]. It supports paging of large scenes in an efficient way. For this to work, the scene is broken up into regular tiles. For each tile a LOD pyramid is constructed, so that the library can page in the visible LODs and page out those which have become invisible.

In each frame the library processes a subset of all visible tiles and updates the triangulation for these tiles depending on the actual point of view. The updated tiles are stored in one of two render caches. After a complete update of all tiles one cache contains the mesh of the entire scene and is used display the scene in subsequent frames while the other cache is silently updated.

By switching forth and back between the two render caches the library does not need to perform a complete update in every frame which reduces the CPU load dramatically. Also, the usage of a double buffered render cache enables us to store the geometry in a vertex buffer so that the graphics hardware is able to process the triangles very efficiently. Additionaly, the render cache has the advantage

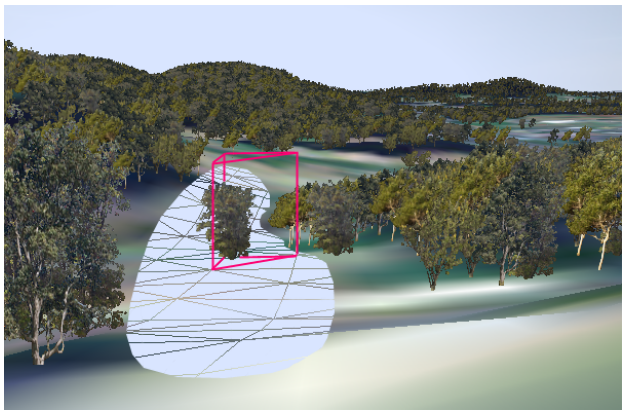


Figure 2. Prism stacked on a base triangle with one bush placed randomly inside the prism volume.

that it can be easily extended for the purpose of large scale forest rendering as described in the following:

The C-LOD algorithm uses a height field for the compact storage of terrain data. From this data the mesh is generated in a view-dependent way and stored tile by tile in the render cache. We now introduce a second height field which represents the height of the associated vegetation. From this vegetation height field we generate a volumetric description of the space occupied by the vegetation in the following way: For each triangle that is output by the terrain renderer a prism is stacked on top of these base triangles (see Figure 2). The height of the 3 vertical edges of each prism is taken from the vegetation height field, so that the set of prisms is a view-dependent volumetric description of the vegetation on the landscape. Since the base triangles are coupled with the stacked prisms, the screen space error which drives the triangulation must be modified to be the maximum of the screen space error for both the height field and the vegetation field (compare [16]).

We assume that there is a direct relationship of the plants height and the intensity of the vegetation, that is the NDVI values. This is reasonable, because a higher index indicates a higher bio mass and a higher mass indicates larger plants on the average. Thus, the vegetation height field can be derived directly from NDVI data by applying a monotone scalar mapping. The mapping could be measured manually by gathering field samples. In our case we assume the mapping to be linear with a maximum user-definable plant height H_{plant} (see also Figure 3).

As the next step we take each prism and place trees and bushes in it in a pseudo random fashion. The main two parameters for driving the seeding process are the prism height and the target density of the plants. Every triangle that is output by the terrain renderer is recursively subdivided into 4 sub-triangles until the triangle size is below the target density. Then each sub-triangle receives one plant where the type and height of the plants is determined stochastically by the prism height. The higher the prism, the more likely it is that a tree is placed in it. The lower the

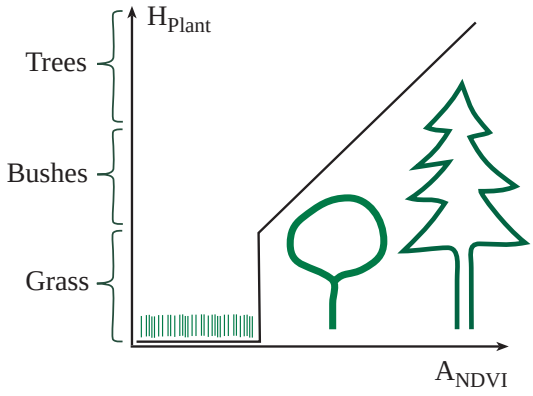


Figure 3. Mapping NDVI values to plant height.

Figure 4. Tree and bush billboards.

Figure 5. Grass textures.

prism, the more likely it is that a bush is placed in it in order to reflect the lower bio mass given by the corresponding NDVI value. The plants are positioned in a deterministic but random way. We use a pseudo random number generator which is seeded by the horizontal position of the base triangles in order to stochastically vary the actual position of each plant. In our case the placement also depends on a particular height threshold. Below this threshold no plants are generated and the prism is simply culled. Additionally, the placement density decreases with the viewing distance in order to keep the total number of generated plants within a manageable limit.

All the generated plants are stored in a tree render cache which is also double buffered. Currently the plants are rendered as billboards in each frame. Due to the view-dependency of the tree generating process the number of visible trees in a scene is reduced dramatically. At the same time the forest is still stretching out very far, since the render cache allows to render a large number of trees in real-time. Since the cache is updated permanently, the placement of the plants is also repeatedly updated. As a result, the height of the plants corresponds with the elevation of the terrain even though the elevation may change over time due to a changing view point.

The title figure shows the benefit of enriching a scene with vegetation in the described way (see Figure 4 for the applies set of trees and bushes). The first advantage is that the scene and especially the horizon looks much more like what one would expect in reality. The second advantage is that the viewing experience is much more realistic, because the skirts of a forest completely obscure all details behind it. A practical example for this case is a tour on the highway through a forest, where basically only the highway and the trees on its side are visible.

4 Rendering the Grass Layer

High to medium values of the NDVI usually correspond to trees or bushes, respectively. These can be displayed efficiently using our described billboard approach. Low NDVI values correspond to either very sparse vegetation or grass. For the display of meadows with a large extent billboards are not suitable, so we use an approach which is commonly known as *texture splatting*. In general, texture splatting means that from a set of appearance parameters a selection of tilable patterns is blended together and then splat onto the surface (see also [17]).

As mentioned previously, below a certain height or NDVI threshold we do not render plants. Instead, this lower range of NDVI values is mapped to a grass classification parameter g (see also Figure 3). Low values of g correspond to stony areas whereas high values correspond to a healthy meadow.

For this range of different grass types we use a set of geotypical tilable texture patterns (see Figure 5) that are stored in a 3D texture stack. When processing the prisms we check whether or not plants need to be placed. In the case that no plants are generated we render an additional grass layer by performing a 3D texture lookup with the first two texture coordinates being calculated from the vertices' position on the plane and g being the third texture coordinate along the texture stack (so the logical term for this texturing method is *3D texture splatting*). In order to make the textures appear more natural we also perturb g with a 2D Perlin Noise texture [14].

Without the use of mip-mapping for the 3D grass texture stack the described approach is straight-forward. However, since our scene is large mipmapping is mandatory. But we cannot use the standard mipmapping approach that is built into OpenGL. This is due to the fact that standard 3D mipmapping requires the mip map levels to be halved along *each* edge of the texture volume. This assumption is not fulfilled because the number of stacked textures needs to be kept constant and cannot be halved from one mip-map level to the next one.

As a consequence we implemented a pixel shader which performs mipmapping by hand. For the n possible mip-map levels of the 3D texture we blow up all levels to the size of level 0 and store them stacked above each other in a single 3D texture. Then the pixel shader only has to

determine the actual level of detail and perform a lookup in the corresponding 3D texture subset.

It would be a waste of texture memory to store all n levels of detail. At a certain distance threshold the original texture ortho-photos of each tile are detailed enough to replace the 3D textured surface. The LOD number that corresponds to this threshold usually lies in the range from 3-5, so that we only need to store the first 4 LODs for example. So, in comparison to the application of plain 3D textures the usage of 3D-mipmapping for the texture splats requires four times the amount of texture memory. The texture stack shown in Figure 5, for example, consumes 24MB.

5 Results and Future Work

The scene displayed in Figure 6 contains a total of approximately 10 million trees. The described C-LOD method reduces this to an amount of roughly 100.000 visible trees depending on the point of view. On an AMD Athlon 1.8 GHz PC with an NVIDIA GeForce 5800 graphics accelerator the frame rate for the entire scene is about 30 frames per second with an effective number of 10 million vertices per second on the average and about 15 million vertices peak performance. These results show the efficiency of our caching system.

Another possible option is to accelerate the display of the cached geometry by using impostors [18] and/or billboard clouds [1] for distant trees. Additionally, a geometrical plant representation [6] can be used to improve the appearance of nearby trees.

In the future we plan to improve the plant placement scheme by not only specifying one additional vegetation height field but in fact several of them. For example one vegetation field for coniferous and one for deciduous tree height would enable us to model the forest type exactly. The proposed model is also open for a wide variety of other GIS parameters that are used in practice such as average rainfall or soil classifications.

6 Conclusion

We described an approach that is able to use satellite NDVI data for the display of real-world forest scenes. We rendered the large-scale vegetation data at real-time by applying a view-dependent C-LOD algorithm. Besides the use case demonstared in the paper, we would like to emphasize that our approach is able to serve as a general frame work for a wide variety of other GIS exploration tasks.

References

[1] S. Behrendt, C. Colditz, O. Franzke, J. Kopf, and O. Deussen. Realistic real-time rendering of landscapes using billboard clouds. In *EUROGRAPHICS 2005*, pages 507–516, 2005.

[2] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, and F. Ponchio. Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM). In *Proc. Visualization '03*, pages 147–155. IEEE, 2003.

[3] M. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for texture and image generation. In *SIGGRAPH 2003*, pages 287–294, 2003.

[4] Willem H. de Boer. Fast Terrain Rendering Using Geometrical Mipmapping. *E-mersion Project*, 2000.

[5] P. Decaudin and F. Neyret. Rendering forest scenes in real-time. In *Eurographics Symposium on Rendering*, pages 93–102, 2004.

[6] O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis. Interactive visualization of complex plant exosystems. In *IEEE Visualization 2002*, pages 219–226, 2002.

[7] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modelling and rendering of plant ecosystems. In *SIGGRAPH 1998*, pages 275–286, 1998.

[8] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, Ch. Aldrich, and M. B. Mineev-Weinstein. ROAMing Terrain: Real-Time Optimally Adapting Meshes. In *Proc. Visualization '97*, pages 81–88. IEEE, 1997.

[9] D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling, A Procedural Approach*. AP Professional, second edition, isbn 0-12-228730-4 edition, 1998.

[10] GLCF. The Global Landcover Facility. *http://glcf.umiacs.umd.edu*, 2005.

[11] libMini. A Real-Time Terrain Rendering Engine. *http://stereofx.org/#terrain*, 2005.

[12] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. In *Proc. SIGGRAPH '96*, pages 109–118. ACM, 1996.

[13] F. Losasso and H. Hoppe. Geometry clipmaps: Terrain rendering using nested regular grids. In *SIGGRAPH 2004*, pages 769–776, 2004.

[14] Ken Perlin. An Image Synthesizer. *Computer Graphics (Proc. SIGGRAPH '85)*, 19(3):287–296, 1985.

[15] S. Roettger, W. Heidrich, Ph. Slusallek, and H.-P. Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. In *Proc. WSCG '98*, pages 315–322. EG/IFIP, 1998.

[16] Stefan Roettger and Thomas Ertl. Cell Projection of Convex Polyhedra. In *Proc. Volume Graphics '03*, pages 103–107, 2003.

[17] Stefan Roettger and Ingo Frick. The Terrain Rendering Pipeline. In *Proc. EWV '02*, pages 195–199. OCG Schriftenreihe, R. Oldenburg, Vienna, 2002.

[18] G. Schaufler. Per-Object Image Warping with Layered Impostors. In *Proc. 9th Workshop on Rendering '98*, pages 145–156. Eurographics, 1998.

[19] M. Suter and D. Nüesch. Automated generation of visual simulation databases using remote sensing and GIS. In *Proc. Visualization '95*, pages 135–142. IEEE Computer Society Press, 1995.

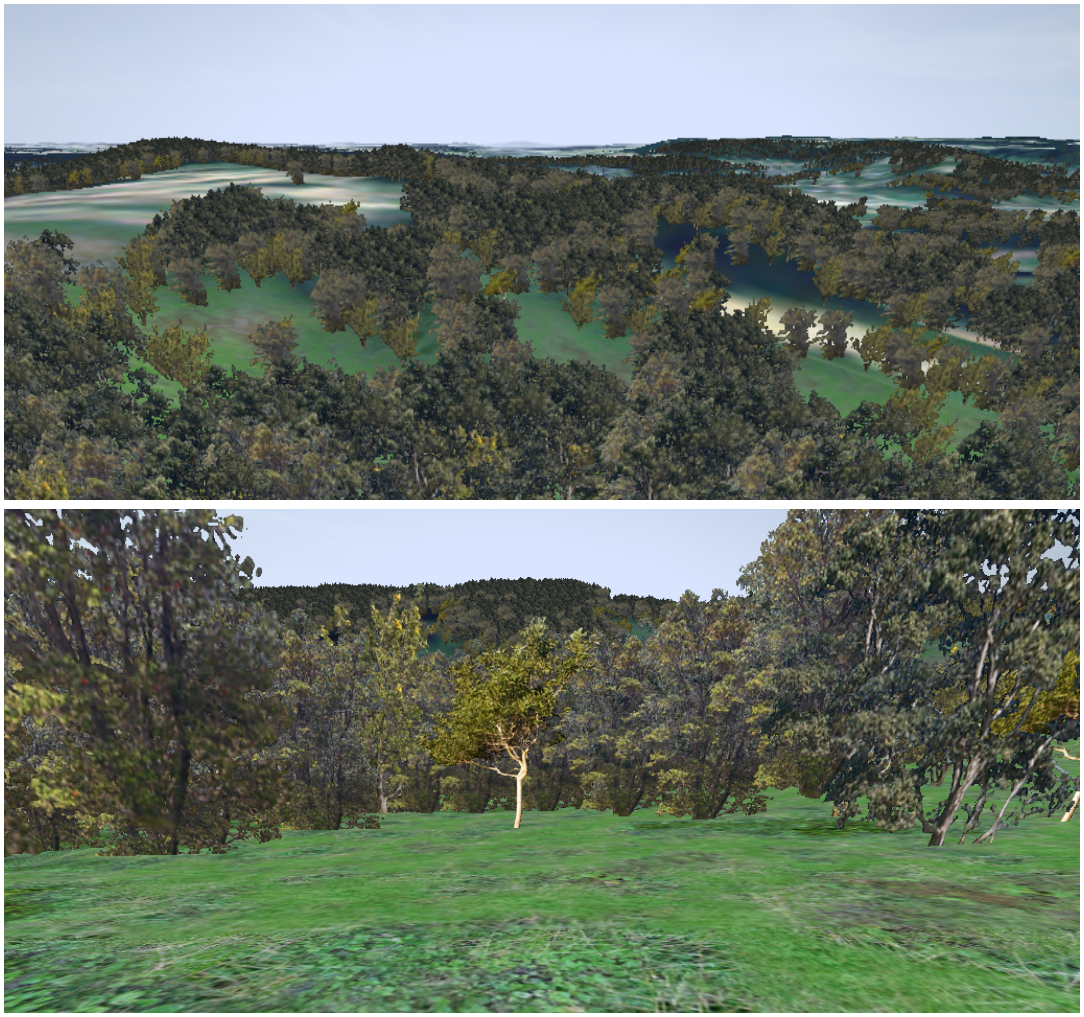


Figure 6. Display of forest scarps at the ”Hetzlas” and a closeup of the grass layer.