Realistic Illumination of Vegetation in Real-time Environments

Malte CLASEN, Hans-Christian HEGE

Abstract

In this paper we show how to combine and extend state-of-the-art computer graphics algorithms for realistic illumination of vegetation. Our main target are realtime rendering environments, namely the Lenné3D visualization system. We focus on plants near the viewer where we separate the rendering equation to model the different lighting effects: Direct sun light is handled by advanced fragment shaders and shadow mapping whereas indirect light is calculated from environment maps and precomputed radiance transfer (PRT). This way we can handle both low and high frequency lighting. Soft shadows and diffuse indirect lighting are contributed by the PRT, hard shadows, diffuse, glossy and specular direct lighting are calculated using the common rendering pipeline including shaders. We assume static scenes (fixed terrain and plant positions, fixed lighting) to precompute the radiance transfer. Note that this does not introduce view dependance.

Our main contribution is the extension of these algorithms to large scenes with a very high number of objects. Previous work usually covered illumination of single objects. Scene complexity was hidden by environments maps. In large scenes, one environment map does not fit all objects due to neighbouring objects occluding parts of the outer scene environment. This also introduces a source of indirect lighting among different plants. We propose a new method of clustering and interpolating environment maps to handle the varying local environments of the plants.

1. Introduction

Landscape visualization in real-time environments on desktop computers has become possible over the last five years. The Lenné3D Player is capable of rendering scenes with more than 100.000 plants (Fig. 1), the commercial game Far Cry renders a densely covered island as a setting for an adventure game (Fig. 2). The sheer number of plants seems to promise convincing photo-realistic scenes.

Fig. 1: Lenné3D Player (www.lenne3d.de)   **Fig. 2:** Far Cry (www.farcry.de)

On the other hand, a new kind of physically correct illumination algorithms appeared. Whereas previous attempts focussed on realistic but non-interactive rendering, Sloan used consumer graphics hardware to display single objects of competitive realism in interactive frame rates (Fig. 3).

Fig. 3:   Comparison of diffuse, unshadowed illumination and diffuse, interreflected illumination (Sloan 2002)

Before we proceed with a description on how to combine these two areas of research, we give a short introduction to the different histories.

2. Previous Work

There are two goals when it comes to realistic image synthesis: In the first place you have to deal with a very large number of detailed objects that shall be projected onto the screen. Secondly these object all interact with the light in the scene - the light that finally results in

your image. These steps are often subject to different research projects although they are inherently connected.

2.1 Landscape Visualization

Landscape visualization usually concentrates on the handling of large scenes. These scenes consist of hundreds of thousands of plants, each of which is built from thousands of primitives. DEUSSEN ET.AL. (1998) developed a software system especially designed to handle outdoor scenes (Fig. 4). They replaced plants or groups of plants by representative objects to reduce the memory requirements and divided the scene into small sub-scenes before rendering.

Fig. 4: Stream Scene (DEUSSEN ET.AL. 1998)   **Fig. 5:** Sunflower Field (DEUSSEN ET.AL. 2002)

DEUSSEN ET.AL. (2002) rendered scenes with 70 million triangles at 3 frames per second by replacing some parts of the triangle geometry with point and line primitives. This level of detail technique (LOD) is based on the limited screen resolution: Objects far away from the viewer are rendered only a few pixels large on the screen. There's no need for thousands of primitives in this case.

But these approaches have in common that the limit the illumination to quite simple and physically incomplete or incorrect models. The common rendering interface OpenGL (SEGAL 2004) offers local illumination, composed of diffuse, specular and ambient terms. That means there are initially no shadows. It's as if your universe contained the sun and only one leaf at a time. Light does not reflect from one plant to the next (or itself), nor does it attenuate while passing though the crown of a tree. These effects can be approximated or passed over by artistic choice of the colors of the plants, but it's not realistic by default. Shadow mapping is a common technique to add at least hard shadows to real-time rendering. MARTIN ET.AL. (2004) describe an algorithm that results in very accurate shadows, but it's limited to fully opaque objects and light sources that are infinitely far away.

2.2 Realistic Illumination

KAJIYA (1986) proposed an equation to handle basically all light effects, the so called rendering equation. It can handle all common surface characteristics and results in physically absolutely correct images. But there's one major draw-back: It's not solvable analytically. It's main integral is usually approximated stochastically, resulting in an

algorithm called monte-carlo-path-tracing. The other interesting aspect is the modeling of the surface materials: OpenGL uses colors and color textures for each of it's light effect terms, but that's usually not sufficient to describe real-world materials. SCHLICK (1993) described a reflectance model that handles colored multi-layered surfaces that can be arbitrarily rough and anisotropic. It's main advantage is that it's not only physically corrent but intuitive. Artists can control it by adjusting just a few parameters to get a wide spectrum of common real-world materials.

These techniques alone allow the creation of very convincing images, but they are far from being usable in interactive environments. SLOAN ET.AL. (2002) found a way to use this power to precalculate the light effects that OpenGL doesn't handle. This so-called precomputed radiance transfer (PRT) stores how light interreflects on a single object. Combined with an environment map (a spherical map that records the incoming light at one point in the scene) this PRT data can be evaluated to the reflected light that is gathered by your virtual camera.

Although it is possible to render single objects in an astonishing quality (Fig. 3), neither Sloan nor the following papers dealt with scenes with large numbers of objects. This is were we try to combine the best of these two research areas. But before we continue on algorithms, we first describe the data we operate on.

3. Data

We target typical natural landscapes with a significant amount of vegetation. Our scenes remain static, neither objects and plants nor sky and sun move. However, we allow the viewpoint to change interactively.

3.1 Plants

Out plants are modelled with Xfrog, a system that evolved from LINTERMANN ET.AL. (2002). The resulting models are converted to a triangle mesh structure, organized by different materials and textures. The basic mesh properties such as vertex positions are used directly in our renderer whereas the materials have to be converted. We use the Schlick reflection model instead of the more common OpenGL lighting equations, so a few parameters can be reused, but most have to be guessed. This limits direct artistic control over the visualization, but the results look more realistic. We use a two layered material, the upper layer being glossy and white, the lower diffuse and colored. This allows typical glossy reflections of wax surfaces while it retains the distinct coloring of the lower leaf cells.

Another step that makes a significant difference in perceived realism is the generation of normal maps. The models usually come with color maps only, so we cannot use the real height structure of the surfaces. We approximate the height by the mean color intensity of the color map. This has no physical background but showed to be visually pleasing.

3.2 Terrain

The terrain is given as a combination of height map and color map. Both have a limited resolution that fit well into texture memory of current GPU, so no special geometry streaming algorithms have to be used.

3.3 Sky

The sky is our only source of illumination. We use images with extremely high dynamic range (1:600,000), kindly provided by Stumpfel et.al. (2004). We determine the sun position and extract it from the image, so we can handle it as a usual distant directional light source in the following rendering steps. This preserves the realism by using images of real skies while reducing the algorithmic effort to efficient and well known techniques.

4. Rendering

The system is optimized for the stroller's perspective. We assume that only a few medium or large plants (trees, shrubs) are in the near field, so most of the trees cover only a small area on the screen. Detailed illumination cannot be perceived for these plants, so we can use the common algorithms for most of the scene. Only a few plants need improvement.

4.1 LOD Extension to Higher Detail

The LOD approach by Deussen et.al. (2002) started with detailed plant geometry and reduced the number of primitives for improved rendering speed. We extend the quality scale in the other direction: We add physically correct illumination to the detailed model. Since only very few plants are rendered at the highest level of detail, the cost for the additional computations remains manageable. The rendering system can choose from a wider range of quality levels, so it's not only possible to improve speed by removing details but to improve quality on fast hardware platforms.

The basic idea is to add Sloan's PRT to the models. Since there are only a few hundred models instanced numerous times in the scene, the memory overhead is negligible. But there's still a problem: Due to restrictions of the graphics hardware and algorithmic shortcomings with PRT, we can only use it for low frequency lighting. This means that we can have soft shadows, smooth colored reflections and translucence, but there's no way to store the shadow of direct sun light.

4.2 Direct and Indirect Light

Sun light is basically the only light in outdoor scenes that's capable of creating hard shadows. So why not handle this case separately? Kajiya's rendering equation allows us to split the lighting into two parts: Light that comes directly from the light source, in our case the sun, and light that has been reflected at least once. This indirect light has the nice property of having hardly any high frequencies: Natural surfaces usually reflect light diffuse or glossy. Specular reflections happen on water surfaces, but these are only perceivable as such in absence of wind and flow, so we can safely omit them. Diffuse and glossy reflection distribute the light over a wide angle, so there's no way in practice reflected light can throw hard shadows.

So we use two different lighting terms when rendering the scene. On the one hand there's direct light. This closely resembles the default OpenGL rendering pipeline, but we use custom shaders instead of the diffuse/ambient light model. This allows us to integrate Schlick's shader model and Martin's shadow mapping. On the other hand there's the PRT,

also integrated into our shaders. But we still need the environment maps to evaluate the PRT. The PRT itself is stored with the model, but the environment maps are required for each instance. Since storing the positions of each plant is already problematic for large scenes, storing complete environment maps is next to impossible. We only need a few maps while rendering a single frame, but since the viewer can be anywhere in the scene, we don't know which ones are unnecessary, we have to store them all.

Let's have a closer look at these environment maps. They represent the incoming light, similar to a 360° panorama photo. Low resolution maps suffice, so they appear quite blurry when displayed directly. This property can be used for a simple but helpful observation: The environment maps of two adjacent plants are usually similar. So you can build clusters of environment maps where each plant uses a map that is reconstructed from the nearest cluster maps. This clustering and interpolation scheme has to adapt to the local situation. You cannot aggregate the environment maps of border plants where one plant is illuminated from the side whereas the other is located among equally high individuals.

PAULY ET.AL. (2002) compare different simplification strategies for point based geometry. The clustering by region growing approach fits our needs. You start with one sample and add its neighbours to the cluster until a certain error threshold is reached. Then you proceed with the nearest unclustered sample and build the next group.

5. Results

Since we are currently bug fixing and evaluating, we cannot show final results. These are expected to be available in february, so the final paper will have a proper result presentation. For now, we can show you a two images that were produced during research and development:
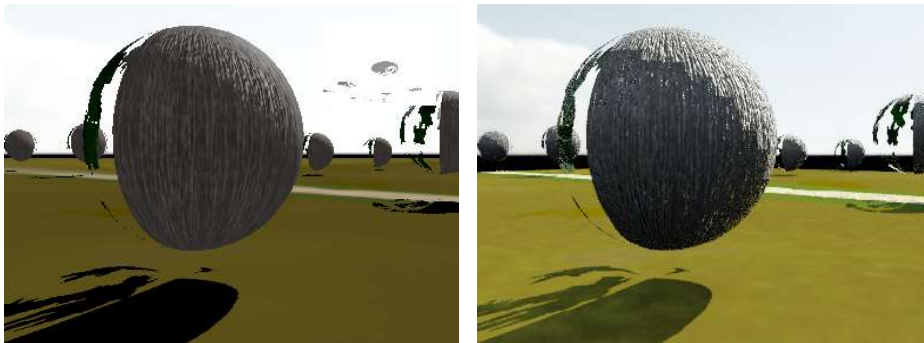

Fig. A: Comparison of placeholder geometry with different lighting terms: Direct lighting (left), direct and indirect lighting (right)

6. Conclusion

Since we still don't have our final results, it's hard to conclude. Intermediate results look promising, however it's still unclear how well these methods work on slightly flawed plant models. New visualization routines frequently show previously hidden errors in models that simply do not matter in conventional rendering methods. We expect that some more research in plant model preprocessing is required to fully leverage the strength of our proposed visualization system.

7. Acknowledgements

All plant models have been built using Xfrog by Greenworks. They are kindly provided by the Lenné3D project.

8. Bibliography

Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prusinkiewicz, P. (1998): *Realistic modeling and rendering of plant ecosystems, SIGGRAPH 1998*

Deussen, O., Colditz, C., Stamminger, M., Drettakis, G. (2002): *Interactive Visualization of Complex Plant Ecosystems*. IEEE Visualization 2002

Kajiya, J. (1986): *The rendering equation.* SIGGRAPH 1986

Lintermann, B., Deussen, O. (1998): *A Modelling Method and User Interface for Creating Plants.* Eurographics Computer Graphics Forum Volume 17, Issue 1 (March 1998)

Martin, T., Tan, T. (2004): *Anti-aliasing and Continuity with Trapezoidal Shadow Maps.* Eurographics Symposium on Rendering 2004

Pauly, M., Gross, M., Kobbelt, L. (2002): *Efficient Simplification of Point Sampled Surfaces.* IEEE Visualization 2002

Schlick, C. (1993): *A Customizable Reflectance Model for Everyday Rendering.* Eurographics Workshop on Rendering Proceedings 1993

Segal, M., Akeley, K. (2004): *The OpenGL Graphics System: A Specification.* http://www.opengl.org/

Sloan, P., Kautz, J., Snyder, J. (2002): *Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments.* SIGGRAPH 2002

Stumpfel, J., Jones, A., Wenger, A., Tchou, C., Hawkins, T., Devebec, P. (2004): *Direct HDR Capture of the Sun and Sky.* AFRIGRAPH 2004