GPU-Based Lighting and Shadowing of Complex Natural Scenes
Florent Cohen, Philippe Decaudin, Fabrice Neyret

To cite this version:

HAL Id: inria-00537461
https://inria.hal.science/inria-00537461
Submitted on 18 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
GPU-Based Lighting and Shadowing of Complex Natural Scenes

Florent Cohen* Philippe Decaudin* Fabrice Neyret*
GRAVIR/IMAG-INRIA www-imagis.imag.fr/Publications/2004/CDN04/

Purpose and motivations
Rendering realistic natural scenes in real-time is a challenging topic in the field of computer graphics. More specifically, we want to render scenes that contain complex shapes bounded in a layer above a surface. Landscapes (e.g., terrain, vegetation) are examples of such scenes. Lightings and shadows are essential elements toward realism because they help understanding shapes. However:

- Using polygonal meshes and classical shadowing algorithms are too costly in terms of computation for complex scenes;
- Adapted algorithms for dedicated representations such as heightfields are efficient, but these representations are too limited for our needs (e.g. forest);
- Although more powerful representations like surfels or texels exist, algorithms for quality and efficient lighting and shadowing are not available yet.

Thus, our goal is to propose lighting and shadowing techniques adapted to such representations as well as the GPU.

Lighting
In order to highlight complex scenes, per-pixel lighting is required. In order to compute it accurately, we should rely on the normals distribution of the elements which project on each pixel. This data can be obtained from the shape representation (e.g., see [Cohen et al. 1998] for meshes). Note that bump mapping techniques correspond to the brutal undersampling of this distribution (1 normal per pixel). Since it is not properly filtered strong aliasing can appear. [Fourrier 1992] gives some ideas for representing and filtering the normals distribution. We propose to rely on [Lafortune et al. 1997] shading model to represent and filter the normal distributions.

The filtering will be pre-computed and stored in textures, allowing the GPU to process the computation quickly.

Shadowing
Processing the shadows on an object basis is often impractical since the amount of objects is very large. Instead, we extend the horizon mapping techniques [Max 1988; Sloan and Cohen 2000] in such a way that it is adapted to non-heightfield scenes: we pre-compute several horizon maps at different heights above the surface (see below). These maps are then interpolated for each given fragment within the volumetric layer. Thus, shadows can be rendered at low cost compared to the visual complexity.

Application: Forest scenes
In order to test and validate our approach, we implemented our techniques based on volume encoded forest scenes [Decaudin and Neyret 2004], in which precomputed lighting and shadows were used. We pre-computed a 3D "texcell" containing the normals of each forest scene element.

For lighting, we currently calculate a simple dot product between normals and light vector. The next step will be to encode the Lafortune model and to evaluate it in a fragment program.

For shadowing, we pre-computed two horizon maps at the top and bottom of the forest, and let the ordinary texture management interpolate them for each fragment.

Results are visually promising, but some problems still remained: some major optimizations are still pending, and we are currently working on implementing Lafortune normals filtering. We had achieved 5 frames per second on each pictured scene on a Pentium IV 2 GHz with a GeForce FX5800.

References


