The Terrain Rendering Pipeline

Stefan Roettger†, Ingo Frick‡

†VIS Group, University of Stuttgart

wwwvis.informatik.uni-stuttgart.de

‡Massive Development, Mannheim

www.massive.de

Abstract:
From a game developers point of view terrain rendering is much more than just the real-time display of a landscape. More accurately, the latter is only one part of what we call the terrain rendering pipeline. The most important stages of this pipeline are the design of the terrain geometry, the real-time display of the terrain, natural texturing, dynamic lighting, dynamic shadowing, and the enrichment of the terrain with organic features. We intend to survey each stage of this terrain rendering pipeline by giving an example from the actual multi-award winning DX8 game AquaNox. In this way we illustrate the data flow through each stage of the pipeline, effectivley giving a report on the current state of the art in terrain rendering.

1 Introduction

With the upcoming of advanced terrain rendering algorithms the complexity of the outdoor scenes displayed in interactive entertainment has increased significantly over the past years. By today's standards the most efficient method to display a terrain is the so called continuous level of detail technique (C-LOD). Despite its advantages these techniques are nowadays just beginning to migrate into the design of modern 3D graphics engines. With the demand for more and more complex outdoor scenes this situation will clearly change in the future. With the C-LOD algorithm being the key component for the real-time display of large outdoor scenes, there exist a variety of other aspects that have to be considered to achieve the desired look and feel of an organic landscape. From a game developers point of view, terrain rendering is a data driven process, which not only involves the real-time display of a given terrain but also the design of the artificial landscapes and the realistic texturing and lighting thereof. The entire story can be described as what we call the terrain rendering pipeline. Following the data flow from the beginning to the end of the pipeline, we describe a complete framework which utilizes current state of the art techniques at each stage. For the modelling of the surface properties we introduce the three functional groups illumination, material, and global effects. On the one hand, this separation offers a high flexibility with respect to the visual appearance of the surface. On the other hand, each functional group results in one or more

surface textures which can be rendererd efficiently using multi-texturing. In addition, the rendering process can be easily divided into several distinct passes, which enables us to customize the entire pipeline for different graphics hardware. This approach seems to be a reasonable concept, which has been demonstrated successfully in the multi-award winning DX8 game AquaNox [1].

The terrain rendering pipeline consists of 6 main stages which are described in the following (see also Figure 1):

2 Landscape Data Generation

The generation of terrain data is a complex task, which is often underestimated. The terrain data has to satisfy diverse requirements. Obviously, the visual appearance is of prime importance. Furthermore, the topology of the terrain is one of the key elements for the subsequent mission design process. Last but not least, the data generation process should be cheap in terms of time and money. We have come up with a feasible solution, which fulfills the described requirements. As a first step, real world terrain data is collected. This data source guarantees the natural appearance and authenticity. For the purpose of mission design, the level designer uses common image editing tools. The tools are utilized to manually generate displacment maps, containing the features which are relevant for the game play. In a final step, the height field is filtered in various ways (noise, edge enhancement, etc.). To avoid quantization artifacts the entire process should be carried out with at least 16 bits of accuracy.

3 Real-Time Display of the Terrain

Once the terrain is defined by a two-dimensional cartesian height field, sophisticated algorithms are needed to display the landscape in real-time. The size of a height field easily exceeds 1024x1024 grid points, which in turn corresponds to more than 2 million triangles that have to be rendered for each frame. Since the exact display is not feasable with respect to interactive rendering, one seeks to approximate the terrain in the following fashion: Far distant details which map to only a few pixels on the screen can be represented with less triangles than large objects nearby. This observation leads to a view-dependent triangulation of the height field which tries to minimize the screen space error of the approximation. The current state of the art in this area are the so called continuous level of detail techniques (C-LOD [4, 2, 5]), which easily achieve a screen space error of less than one pixel. As a consequence, the approximated terrain is indistinctable from the original terrain.

In interactive entertainment, however, terrain rendering is only one task among many others that have to be carried out for each frame. Therefore the achievable screen space error is usually well above one pixel. Larger screen space errors manifest themselves in the so called popping effect. When approaching a specific detail from the distance this detail will suddenly pop up if the local screen space error is exceeded. As a solution to this problem, the geomorphing technique smoothly inter-

polates between the different levels of detail effectively rendering the popping effect invisible [5]. This allows smooth immersive terrain visualizations even on low end graphics hardware. Since the interpolation needs to be updated at least every 100ms to keep the illusion of a static triangulation, frame to frame coherence is difficult to exploit. In order to speed up terrain rendering, we predict the view frustum for the next 100ms, compute the triangulation for the predicted and enlarged visible area and cache the generated triangles until the next update of the triangulation. In this way smooth visualizations of a terrain are generated at real-time and with a low average CPU load. We call this concept ”semi dynamic” geometry generation.

4 Terrain Material

Each material is assembled from three textures, a coarse color texture, a finer material texture, and an even finer detail texture. We found out that it is important to differentiate between color and structure. The coarse color map is used to generate the large scale coloring of the entire terrain. The material map represents the structure of a material at a mid-freqency level, whereas the detail map contains only intensities at a high-frequency level. The latter map is used to represent small details close to the viewer.

5 Terrain Illumination

The illumination can be constructed by summing up the emission of all static and dynamic light sources. The static light map covers the entire terrain and is generated in a preprocessing step, which gathers the ambient and diffuse contributions of all static light sources. Since we use ray casting to calculate the incoming intensity for each texel of the light map, static shadows are already included at this point.

As the terrain geometry is generated by a C-LOD system, it only makes sense to perform the dynamic lighting calculation on a per-pixel basis. For this purpose, we generate a dynamic light map, which only covers the view frustum in order to ensure a sufficient texture resolution. Using the light map as a render target, the incoming light is accumulated for each dynamic light source. The light intensity is calculated by applying a dot product between the normal map of the terrain and a radial light field which is specific for each light source. At this point, dynamic shadows are also taken into account. We project a bounding sphere approximation of each dynamic object onto the terrain to maintain a shadow buffer in the alpha channel of the render target. The alpha channel represents the height of the object. When rendering each light source, we have to determine whether each texel is lighted or not. This is achieved by comparing the height of the light source with the height coded in the alpha channel. This concept is applied to all dynamic lights including caustics and the sun light. A total of more than 500 light sources can be treated in real-time using this texture based lighting approach.

6 Organic Features

In the next step organic features are added to the scene. We distinguish between local and global phenonema. As an example for local phenomena we present a large scale plant rendering system. The plants are categorized into several groups. For each group a density distribution is painted by the level designer. According to this distribution, the plants are placed in a pseudo-random fashion. The plant seeding is performed on the fly for the visible part of the terrain only. In order to maximize the geometry throughput, we cache the plants based on a tiling scheme. As an example for a global phenomenon we use a particle system for the display of floating plankton. This particle system has been designed to run entirely in the vertex shader of current graphics hardware.

7 Global Volumetric Effects

The last stage of the pipeline handles volumetric effects such as fog or turbidity. In general, the visualization of volumetric effects requires the solution of the light transport equation. We use a volumetric fog which is defined to occur below a specific height. In this restricted case, the light transport equation simplifies to a two dimensional integral for a constant height of the viewer. The solution of this integral is precalculated and stored in a standard 2D texture which is mapped over the entire scene [3]. This method avoids the appearance of popping artifacts, since the fog attenution and emission is calculated on a per-pixel basis. Interestingly enough this feature is not only a visual effect but also offers game play relevant elements such as hiding.

8 Conclusion

In this paper we have given an overview of the terrain rendering pipeline. The abstraction into 6 main stages has lead to a clear separation of rendering tasks which simplifies development. For each stage current state of the art techniques were illustrated by describing the corresponding part of the AquaNox game engine.

References

[1] Massive Development. AquaNox Home Page, `www.aquanox.de`, 2001.

[2] 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.

[3] Justin Legakis. Fast Multi-Layer Fog. In *ACM SIGGRAPH '98 Conference Abstracts and Applications*, page 266, 1998.

[4] 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.

[5] 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.

Figure 1: The main stages of the terrain rendering pipeline in the AquaNox game engine.