Fast GPU-based normal map generation for simplified models

Jesús Gumbau
Universidad Jaume I,
Spain
jgumbau@sg.uji.es

Carlos González
Universidad Jaume I,
Spain
cgonzale@sg.uji.es

Miguel Chover
Universidad Jaume I,
Spain
chover@uji.es

ABSTRACT

This paper presents a method for normal map generation in the GPU. These normal maps are generated from a high resolution mesh and can be applied to any simplification of this mesh. This method takes advantage of the fact that there must be a correspondence between the texture coordinates of the low resolution mesh and the ones of the high resolution mesh. The proposed method for normal map generation is a brand-new method, since nowadays this process is being performed through software techniques. Hardware generation greatly reduces time in comparison with present-day solutions. Moreover, it allows for a dynamic modification of the map. There are some restrictions in relation to how texture coordinates must be distributed. However, this approach works perfectly with simplified models where these restrictions are fulfilled. This method makes use of vertex and pixel shaders for the normal map generation.

Keywords: normal map, GPU, shaders, hardware, simplified mesh.

1 INTRODUCTION

The presented method proposes a fast hardware generation of normal maps that uses *vertex* and *pixel shaders*. This idea involves a real-time normal map generation from the high-level object.

The final quality of the resulting normal map is comparable to those obtained from other software-based solutions [1] [3] when applied to multirresolution and simplified models.

Two restrictions have to be accomplished:

- Let $\vec{t_i} = (u_i, v_i)\ \ \forall i \in \{1,2,3\}$ be the texture coordinates for each vertex on a triangle T_j, where T_j is a given triangle of the high resolution mesh. So, the condition $\bigcap T_j = \emptyset\ \forall T_j \in triangles(Mesh)$ must be fullfiled. However, this requirement is studied in the literature [2][4][5].
- Texture coordinates have to be distributed so that the texture should be correctly applied to both models.

This method works perfectly for terrains and walls, because these objects usually meet the requirements.

2 METHOD

Unlike the present-day normal map generation methods, the presented method generates the maps by *hardware*, by making use of *vertex* and *pixel shaders*.

Posters proceedings ISBN 80-86943-04-6
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

Object Space Normal Maps The tasks performed by the *shaders* for normal map generation are:

- The *vertex shader* flattens the image by doing: (x=u, y=v, z=0), where (x,y,z) are the coordinates of the vertex, and (u,v) are its texture coordinates.
- The *pixel shader* generates the normal map by outputting the normal coefficients as RGB components per pixel. For this purpose, it is necessary to convert the normal values into the accepted range by the RGB plane, that is, [0,1].

Tangent Space Normal Maps Moreover, for the normal map generation in tangent space the following extra tasks must be performed:

- The *vertex shader* will receive the tangent space vectors (normal, tangent and binormal) as vertex attributes and will output them so that each fragment will receive these values linearly interpolated.
- The *pixel shader* must transform each normal per pixel into tangent space. The inverse tangent space matrix is already calculated in the *vertex shader* and passed to the *pixel shader*, so transforming each normal to tangent space is as easy as multiply the object space normal by this matrix.

Once calculated, an extra pass must be performed in order to expand texture borders. This is needed to avoid artifacts when filtering is enabled at application time.

3 RESULTS

The presented method has been tested with some 3D models. The obtained times are not comparable with

Polygons of the original model	Time of our method	Time of ATI	Time of nVidia
1696	0.08	16850	16306
7910	0.53	24125	50589
48048	4.74	97704	160975
61644	6.02	129969	179569

Table 1: Table of times in milliseconds of normal map generation

those of present-day *software* methods, since a few milliseconds are taken by the presented method to generate the corresponding normal map.

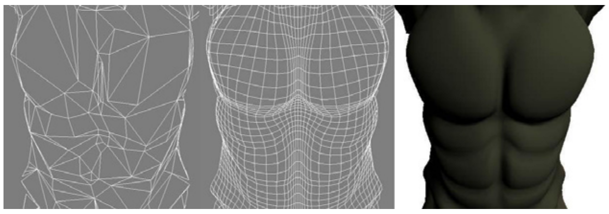
Figure 1: Low and high resolution model meshes of *Tarrasque* (725 and 6117 polygons) with the rendered model

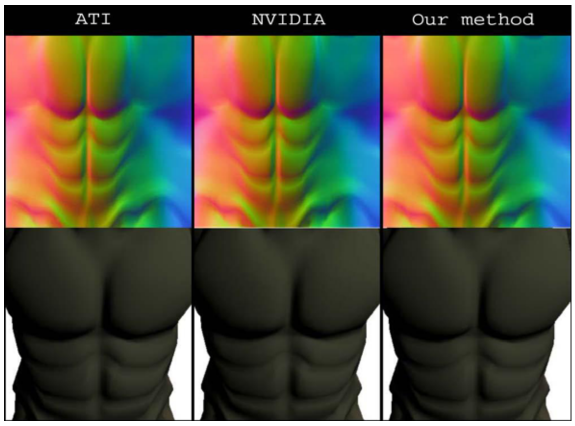

Figure 2: The normal maps in object space of the high resolution model generated and the simplified version with this normal map applied

Measured times using an AMD Athlon64 3500+ with an nVIDIA 6800 can be observed in Table 1.

Figure 1 shows the meshes of both a simplified model and the high resolution model of *Tarrasque*, and the high resolution model rendered.

Figure 2 compares the quality of this method with ray-tracing based methods. The figure shows the generated normal maps and the low resolution model of *Tarrasque* with these normal maps applied, using the methods proposed by ATI, nVidia and our approach, respectively.

The quality of our method is similar to that of the ATI's and nVidia's methods, as seen in the images.

Moreover, the method also works for terrain and wall objects because they usually meet the requirements of this method. Figure 3 and Figure 4 show an example with the *Crater* model.

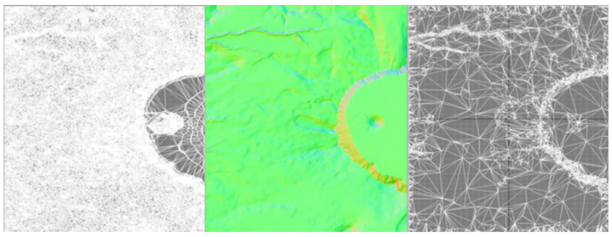
Figure 3: Low and high resolution model meshes of *Crater* and the normal map in object space

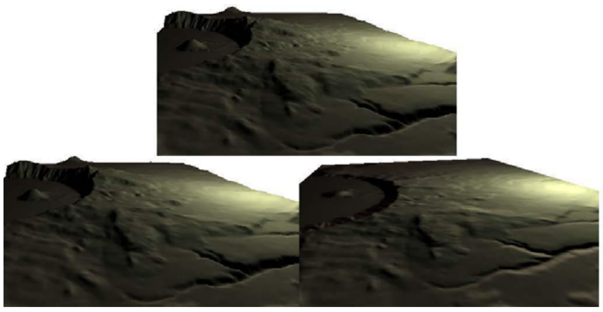
Figure 4: Renders of the high resolution terrain (above), low resolution one with the normal map applied (left bottom) and a plane with the normal map applied (right bottom)

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Science and Technology (TIN2004-07451-C03-03), the European Union (IST-2-004363) and FEDER funds.

REFERENCES

[1] ATI. Normal Mapper Tool, 2002. *http://www.ati.com/developer/tools.htmlAkeley*

[2] Igarashi, T., Cosgrove, D. Adaptative unwrapping for interactive texture painting. *Symposium on Interactive 3D Graphics 2001*, pp. 209-216

[3] nVidia. nVidia Melody User Guide, 2004. *http://developer.nvidia.com/object/melody_home.html*

[4] Sander, P. V., Snyder, J., Gortler, S. J., Hoppe, H. Texture mapping progressive meshes. *ACM SIGGRAPH 2001*, pp. 409-416

[5] Sloan, P.-P., Weinsteun, D. Brederson, J. Importance driven texture coordinate optimization. *Computer Graphics Forum (Proceedings if Eurographics '98)* 17(3), pp. 97-104