Generación por hardware de mapas de normales

Carlos González
Dept. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12071 Castellón de la Plana
cgonzale@sg.uji.es

Jesús Gumbau
Dept. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12071 Castellón de la Plana
jesus.gumbau@anubis.uji.es

Miguel Chover
Dept. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12071 Castellón de la Plana
chover@.uji.es

Resumen

En este artículo se presenta un método para la generación de mapas de normales por *hardware*. Estos mapas de normales pueden ser aplicados a objetos de baja resolución para tener la apariencia de objetos más detallados.

El método usado para la generación del mapa de normales es novedoso, ya que hasta ahora este proceso se ha realizado mediante *software*. La generación por *hardware* proporciona una gran disminución de tiempo frente a las soluciones actuales. Además, permite la modificación dinámica de este mapa.

La única restricción es que el objeto esté bien mapeado y que las coordenadas de textura se encuentren distribuidas.

Para la generación y uso del mapa de normales se hace uso de *vertex* y *pixel shaders*.

1. Introducción

Un mapa de normales contiene por cada píxel tres canales de información (las coordenadas de la normal en el plano XYZ). Esta información se codifica en un mapa de colores RGB, correspondiéndose cada coordenada X,Y,Z con los colores R, G, B respectivamente.

En la técnica del mapa de normales participan dos versiones de un modelo poligonal: el de alta resolución, a partir del cual se calcula el mapa de normales, y el de baja, sobre el cual se aplica.

El método presentado propone una generación rápida de mapas de normales mediante *hardware*, haciendo uso de los *vertex* y *pixel shaders*. Esto implica una generación del mapa de normales del objeto en tiempo real. Y conlleva una mayor velocidad de procesado que los métodos existentes hasta el momento, ya que éstos hacen uso de la CPU para generarlo.

Los métodos realizados hasta la actualidad se generan por *software* (excepto [12]), de modo que se programan las instrucciones necesarias para la generación del mapa de normales para que la CPU las procese.

Hay que tener en cuenta que el método propuesto puede presentar un aspecto del objeto más fiel a la realidad. Esto se debe a que mientras los métodos realizados por *software* realizan operaciones para calcular la normal adecuada a aplicar en el modelo de baja resolución, el método propuesto aplica el valor real de la normal.

Existen dos requisitos que se deben cumplir para poder aplicar este método de manera correcta:

- El objeto ha de estar mapeado de modo que no haya *texels* usados por más de un triángulo, ya que sino se superpondrían los colores generados para el mapa de normales. Pero este requisito esta tratado en la literatura [6][10], destacando el método presentado en [8].
- Las coordenadas de textura han de estar distribuidas, de modo que una misma textura se aplique correctamente sobre ambos modelos.

Si el primer requisito no se cumple el mapa normales no será generado correctamente. Por otro lado, si no se cumple el segundo requisito el mapa de normales no se aplicará adecuadamente sobre el modelo de baja resolución.

Hay muchos tipos de modelos donde estas restricciones están totalmente cumplidas, como por ejemplo en terrenos.

2. Estado del arte

Existen aplicaciones de generación de mapas de normales ya implementadas y probadas, pero todas ellas generan el mapa de normales mediante *software*, con el correspondiente uso de la CPU. Destaca el método propuesto en [11], que utiliza un método similar al propuesto, pero implementado por *software*. La principal ventaja de nuestro método es que se ejecuta por completo en la GPU, liberando así la CPU para que pueda realizar otras tareas de forma paralela.

Sin embargo, el método [12] realiza la generación del mapa de normales mediante *hardware*, aunque nuestro método es más rápido si se cumplen las restricciones nombradas en el apartado 1.

Estos mapas de normales se pueden generar mediante programas de edición 3D, como por ejemplo *3DStudio MAX 7* o *Maya 6.0*. También existen aplicaciones dedicadas exclusivamente a la creación de mapas de normales, como el *NormalMapper* de ATI [1] o el *nVidia Melody* [7].

nVidia's Melody es un programa independiente que presenta una interfaz sencilla con distintas opciones para cargar y generar el mapa de normales.

Ati's Normal Mapper ofrece librerías y se maneja mediante una línea de comandos.

Tanto el *software* de nVidia como el de ATI hacen uso del objeto en dos niveles de detalle distintos, de modo que desde el modelo de baja resolución se lanzan las normales hacia el de alta y donde intersecte cada una se coge la normal del modelo de alta resolución en ese punto para aplicársela al modelo de baja resolución.

Sin embargo, con *3DStudio MAX* y *Maya* se puede hacer uso de una serie de *plugins* disponibles, cada uno de los cuales realiza la generación del mapa de normales de manera distinta.

3. Fundamentos

Un mapa de normales contiene información sobre el relieve del objeto, con lo que se puede modificar para que su apariencia sea distinta sin necesidad de alterar la geometría del objeto. De este modo, el mapa de normales puede ser usado para que un objeto de baja resolución (poco mallado) aparente ser uno de mayor resolución, con la ventaja del ahorro de creación de más triángulos y el coste tanto espacial como temporal que ello supone. Cabe destacar el artículo [2].

Un mapa de normales puede haber sido construido respecto a tres espacios: el espacio del mundo, el espacio del objeto y el espacio tangente. Este método funciona sobre cualquiera de ellos, ya que la normal se calcula en el espacio tangente.

Para la generación del mapa de normales por *hardware* se hace uso de los *vertex* y *pixel shaders*. Los *vertex* y *pixel shaders* son pequeños fragmentos de código programables, de modo que especifican el uso que la GPU hace de los vértices y píxeles de la imagen. Así, OpenGL envía la geometría del objeto a la *pipeline* gráfica y ésta trabaja con ella. Pero usando los *vertex* y *pixel shaders* podemos especificarle cómo ha de trabajar con ella. En la literatura hay varios artículos que tratan el tema, como por ejemplo [3][4][5][9].

4. Método

A diferencia de los métodos ya implementados para la generación de mapas de normales, el método que se presenta los genera por *hardware*, haciendo uso de los *vertex* y *pixel shaders*.

El método se basa en el manejo de un objeto 3D modelado con dos niveles de detalle distintos. La idea es generar el mapa de normales del objeto de mayor resolución con el fin de poder aplicar ese mapa de normales al objeto de baja resolución, para que aparente tener más detalle sin necesidad de aumentar la geometría.

A continuación se comentan los pasos seguidos para la generación del mapa de normales por *hardware*. Para la aplicación del mapa de normales sobre el modelo de baja resolución existen ya métodos realizados por *hardware*.

Para la generación del mapa de normales se utilizará un *vertex shader* y un *pixel shader*, que harán las siguientes funciones:

- El *vertex shader* se encarga de aplanar la imagen, de modo que transforma las coordenadas de cada vértice según sus coordenadas de textura, es decir, toma como coordenadas x=u, y=v, z=0, siendo (x,y,z) las coordenadas del objeto y (u,v) las coordenadas de la textura. Además, pasa la normal, la binormal y la tangente a la *pipeline*. En el Algoritmo 1 se muestra el pseudocódigo los pasos de este *vertex shader*.
- El *pixel shader* genera el mapa de normales, de modo que se pasan las coordenadas de las normales a las componentes RGB del color resultante tras ejecutar el *pixel shader*. Para ello hay que pasar los valores de la normal, transformados al espacio tangente, al rango de valores aceptado por el plano RGB, que es [0,1]. En el Algoritmo 2 se muestra el pseudocódigo de este *pixel shader*.

```
result.pos.x = in.texcoord.u;
result.pos.y = in.texcoord.v;
result.pos.z = 0;
result.normal = in.normal;
result.binormal = in.binormal;
result.tangent = in.tangent;
```

Algoritmo 1. *Vertex shader* para la generación de un mapa de normales

```
normal= in.normal.tangentSpace()
normal = normal.range(0,1);
result.color.r = normal.x;
result.color.g = normal.y;
result.color.b = normal.z;
```

Algoritmo 2. *Pixel shader* para la generación de un mapa de normales

El resultado se almacena directamente sobre una textura para poder ser aplicada como mapa de normales.

5. Resultados y conclusiones

El método ha sido probado con varios modelos 3D, observándose los resultados deseados, de modo que utilizando un modelo de bajo mallado se obtiene una imagen del objeto con una apariencia mucho más detallada.

Los tiempos obtenidos no son comparables con los métodos *software* realizados hasta la fecha, ya que el método que se presenta emplea escasos milisegundos para generar el mapa de normales correspondiente.

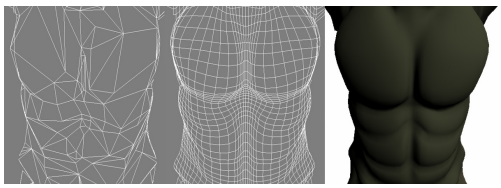

Figura 1. Mallado del modelo de baja resolución, del modelo de alta resolución y modelo de alta resolución iluminado

Se muestran en la Tabla 1 los tiempos empleados con las herramientas *ATI NormalMapper* [1], *nVidia Melody* [7] y el método propuesto sobre los modelos probados.

A continuación, se muestra una serie de imágenes con el fin de poder comparar la calidad del método con los métodos ya realizados por ATI y nVidia. Para ello se ha hecho uso del modelo *Tarrasque* usando dos niveles de detalle, una de 725 polígonos y otra de 6117 polígonos.

En la Figura 1 se muestran los mallados del modelo de baja y alta resolución, y el modelo de alta resolución iluminado.

En la Figura 2 se muestran los mapas de normales del modelo *Tarrasque* generado por el método de ATI, el de nVidia y el presentado. Así, como los modelos de baja resolución iluminados correspondientes.

Tris modelo baja resolución	*Tris modelo alta resolución*	*Tiempo Método*	*Tiempo ATI*	*Tiempo nVidia*
536	1696	<1	16850	16306
4274	7910	<1	24125	50589
23378	48048	31	97704	160975
20517	61644	31	129969	179569

Tabla 1. Tabla de tiempos en milisegundos de la generación del mapa de normales

Se ha realizado pues un método que mejora notablemente el tiempo de generación de mapas de normales respecto a los métodos actuales, siendo similar la calidad del método propuesto a las de ATI y nVidia, ya que las diferencias son casi despreciables.

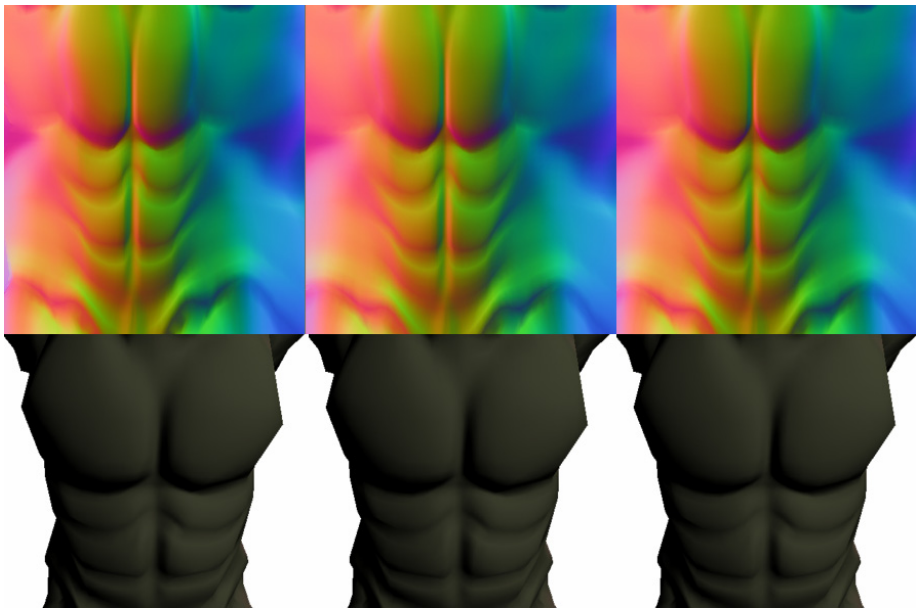

Figura 2. Mapas de normales y modelos iluminados de ATI, nVidia y nuestro, respectivamente

6. Agradecimientos

Este trabajo ha sido financiado por el gobierno español (TIN2004-07451-C03-03 y FIT-350101-2004-15) y la Unión Europea (IST-2-004363 y fondos FEDER).

Referencias

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

[2] Cohen et al. Appearance-preserving simplification of polygonals models. *SIGGRAPH '98*

[3] Ernst, I., Jackèl, D., Rüsseler, H., Wittig, O. Hardware-supported bump mapping. *Computers and Graphics 20*, 1996, núm. 4, pp. 515-521

[4] Heidrich and Seidel. Realistic, hardware-accelerated shading and lighting. *SIGGRAPH '99*

[5] Hirche, J., Ehlert, A., Guthe, S. Hardware accelerated per-pixel displacement mapping. *Graphics Interface 2004*

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

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

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

[9] Schröcker, G. Hardware Accelerated per pixel shading. *CESCG 2002*

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

[11] Tarini, M., Cignoni, P., Rocchini, C., Scopigno, R. Real time, Accurate, multi-featured rendering of bump mapped surfaces. *Eurographics 2000*. Volume 19, Number 3

[12] Yigang Wang, Fröhlich B., Göbel M. Fast normal map generation for simplified meshes, *ACM Journal of Graphics Tools*, Volume 7 , Issue 4 (December 2002)