Enseñanza Avanzada de Computación Gráfica basada en Hardware Gráfico

Claudio Delrieux[1,2], Federico Lois, Gustavo Ramoscelli[1]

[1]Departamento de Ingeniería Eléctrica y de Computadoras
Universidad Nacional del Sur - Av. Alem 1253, (8000) Bahía Blanca
e-mail: claudio@acm.org
[2]Departamento de Computación
FCEyN - Universidad de Buenos Aires - Pabellón I, Ciudad Universitaria, Buenos Aires

1. Contexto Tecnológico

En los últimos 4 años, el hardware gráfico (GPU) experimentó una transición definitiva, de poseer la capacidad para ejecutar una parte sustanciál del *pipeline* gráfico *scan-line* con algoritmos rígidamente implementados, a poder implementar una estructura de renderizado programable, con la capacidad de vincularse dinámicamente con la aplicación, y –además– la capacidad de ser programada interactivamente desde la aplicación por medio de compiladores incorporados a bibliotecas *runtime*.

Estas facilidades, más allá de la potencia inherente al hardware gráfico altamente paralelo, implican una forma completamente diferente y más efectiva de programar aplicaciones gráficas, liberando al desarrollador de la complejidad de implementar complejos modelos de iluminanción, los cuales se programan en la GPU con *shaders* con acceso directo a los valores relevantes del modelo. Estos shaders normalmente requieren muy pocas líneas de código, generalmente modular, paramétrico y autodocumentado.

Pero aún esas ventajas no son todo. La verdadera revolución está en que, con mayor o menor esfuerzo de adaptación, muchos de los algoritmos de rendering con modelos de iluminación no locales (ray tracing, radiosidad, image based rendering, simulación montecarlo, photon mapping) pueden ser implementados en maneras más o menos generales gracias a las posibilidades de programación de las GPU, con tiempos de ejecución muy cercanos al tiempo real.

Aprovechar estas ventajas implica abandonar las bibliotecas mayormente utilizadas para desarrollar aplicaciones gráficas (OpenGL y DirectX), o, al menos, utilizarlas solo como mecanismo de comunicación entre la aplicación y la GPU y solo en caso de estar seguros de que no existen conflictos de versiones entre la GPU utilizada y la versión de la biblioteca empleada. Una vez más, y en muy pocos años, los cambios tecnológicos han producido otra revolución en la Computación Gráfica, la cual vuelve obsoleta a la manera de enseñar basada en bibliotecas.

Una de las motivaciones de este trabajo, por lo tanto, consiste en desarrollar el contenido curricular de una materia avanzada de Computación Gráfica, con el objetivo de explorar las posibilidades teóricas y aplicadas de esta nueva tecnología, y al mismo tiempo evaluar la manera de introducirla en los contenidos de la materia introductoria. En este trabajo se describen los contenidos de dicha asignatura, Computación Gráfica II, y se reseñan las experiencias de su dictado en el Departamento de Computación de la FCEyN-UBA, durante 2004.

2. Computación Gráfica II (UBA 2004)

El objetivo del curso consiste en el estudio e implementación de modelos y técnicas de iluminación de alta calidad cuyo rendering pueda realizarse en tiempo real. Los modelos de iluminación se basan en simplificaciones de los ya estudiados en la materia introductoria (Computación Gráfica I), las cuales permiten su implementación utilizando la programación de las GPU. Por lo tanto, el fundamento del curso está en un adecuado manejo de dicha programación, para lo cual la primera parte del curso se dedica a los fundamentos de las arquitecturas para computación gráfica, y el aprendizaje de lenguajes de programación específicos para GPU, en particular el lenguaje Cg, basándonos en el libro de Kilgard y Fernando [3]. El objetivo de esta parte de la materia no es el estudio de modelos avanzados de iluminación, sino más bien el repaso de los conceptos básicos del pipeline gráfico, su implementación en GPU, y la posibilidad de su programación con Cg.

Si bien el Cg fue desarrollado por *n*VIDIA, el mismo genera código compatible también para las tarjetas ATI, tanto para aplicaciones construidas en base a DirectX (HLSL) como con OpenGL (ARB). Es verdad que la elección de un lenguaje propietario para fines didácticos puede ser riesgosa en un momento de grandes cambios tecnológicos, pero el hecho es que Cg es gratuito, se puede descargar del sitio de *n*VIDIA junto con una gran cantidad de ejemplos y material didáctico, y provee la suficiente compatibilidad hacia atrás (es posible correr los programas aún sin GPU, por medio de un simulador por CPU). También es destacable que Cg permite el binding dinámico con variables de la aplicación para el pasaje de parámetros entre la misma y la GPU, así como una biblioteca runtime que permite compilar y cargar shaders en la GPU durante la ejecución de la aplicación.

La experiencia adquirida como para poder implementar los modelos de iluminación y shading *scan-line* tradicionales (incluyendo el shading de Phong), permite comprender los puntos esenciales de los demás tópicos cubiertos en el curso, comenzando por las mejoras a los modelos de iluminación locales (por ejemplo, mapeo de entornos, reflexiones, texturas y sombras), y modelos de iluminación no isotrópicos. En todos estos casos se describe de qué manera las ecuaciones de los modelos planteados pueden implementarse en tiempo real con shaders específicos.

Luego se presentan los resultados más recientes en implementación de modelos de iluminación no local implementados en GPU, basados en técnicas como la factorización homomórfica de BRDF, la radiancia precomputada factorizada en componentes principales, la descomposición espectral del modelo de iluminación y su representación precomputada como producto triple de wavelets, los armónicos esféricos, la elaboración de iluminación

atmosférica, etc. Una idea más completa del contenido de la materia puede obtenerse con la lectura de la bibliografía citada más abajo.

Un tema al cual se dedicó también parte de la atención, es al uso de las GPU como mecanismo de cómputo sofisticado. No solo pueden utilizarse hábilmente sus funciones para usos diferentes a aquellos para las que fueron diseñadas las placas gráficas (como por ejemplo procesamiento de imágenes, visión, visualización científica, simulación numérica), sino que también pueden pensarse muchas de las funciones de rendering como primitivas numéricas, con lo cual la GPU se convierte en un coprocesador numérico de características sobresalientes. Por ejemplo, para invertir matrices de varios millones de coeficientes, la GPU requiere casi un orden de magnitud menos de tiempo que la CPU.

Otro de los temas que mayor interés despierta en función del perfil profesional y personal de los alumnos es el de los modelos de iluminación y shaders utilizados en juegos y películas. Es importante entonces destinar algunos momentos a describir el modelo de iluminación y los shaders empleados en dichos productos (en esta ocasión discutimos la implementación del *Half Life 2*).

Como es el estilo usual en este tipo de cursos, los trabajos prácticos incluyen la elección de papers para implementar y presentar en comisiones de dos alumnos. Entre los temas elegidos por las comisiones se pueden contar implementación de ray tracing en GPU, uso de la GPU para simular en tiempo real los dispositivos de visión nocturna, visualización en tiempo real de flujos y espacios vectoriales 3D, simulación de Navier-Stokes para fluídos incompresibles, simulación de humo, y rendering basado en imagenes.

Tratándose de una materia optativa y con grandes exigencias de correlatividades, el hecho de haber tenido una matrícula importante, y con muy buenas opiniones en las encuestas correspondientes, indican que la elección del perfil de la materia se corresponde con las expectativas. Además, se cumple con el propósito de acercar a los alumnos a las nuevas tecnologías, tanto de hardware como de software, que en el área de la graficación por computadora y disciplinas asociadas está produciendo una revolución sin precedentes. Por último, la experiencia realizada permite tener una base para poder introducir estos conceptos en las materias introductorias de computación gráfica.

Referencias

[1] W. Chen, R. Grzeszczuk, and J. Bouguet. Light Field Mapping: efficient representation and hardware rendering of surface light fields. *ACM SIGGRAPH*, 32(4):447–456, 2002.

[2] R. Fernando, S. Fernandez, K. Bala, and D. Greenberg. Adaptive shadow maps. *ACM SIGGRAPH*, 31(4):387–390, 2001.

[3] Randima Fernando and Mark Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison–Wesley, 2004.

[4] P. Kipfer J. Krueger, T. Schiwietz and R. Westermann. Numerical simulations on pc graphics hardware. In *ParSim 2004 (Special Session of EuroPVM/MPI 2004)*, 2004.

[5] Jens Krueger and Ruediger Westermann. Linear algebra operators for gpu implementation of numerical algorithms. *ACM Transactions on Graphics (TOG)*, 22(3):908–916, 2003.

[6] L. Latta and A. Kolb. Homomorphic Factorization of BRDF-based Lighting Computation. *ACM SIGGRAPH*, 32(4):509–516, 2002.

[7] X. Liu, P. Sloan, H. Shum, and J. Snyder. All-Frequency Precomputed Radiance Transfer for Glossy Objects. In *Proceedings Symposium on Rendering (EGSR04)*, pages 111–118. Eurographics, 2004.

[8] M. Mccool, J. Ang, and A. Ahmad. Homomorphic factorization of BRDFs for high-performance rendering. *ACM SIGGRAPH*, 31(4):171–178, 2001.

[9] A. Moller and E. Haines. *Real-Time Rendering (2nd ed)*. Chapman-Hall, New York, 2002.

[10] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM SIGGRAPH*, 33(4):376–381, 2003.

[11] R. Ng, R. Ramamoorthi, and P. Hanrahan. Wavelet Triple Product Integrals for All-Frequency Relighting. *ACM SIGGRAPH*, 34(4):226–235, 2004.

[12] T. Purcell, I. Buck, W. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. *ACM SIGGRAPH*, 32(4):703–712, 2002.

[13] R. Ramamoorthi and P. Hanrahan. Frequency Space Environment Map Rendering . *ACM SIGGRAPH*, 32(4):517–526, 2002.

[14] P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered Principal Components for Precomputed Radiance Transfer. *ACM SIGGRAPH*, 33(4):382–391, 2003.

[15] R. Wang, J. Tran, and D. Luebke. All-Frequency Relighting of Non-Diffuse Objects using Separable BRDF Approximation. In *Proceedings Symposium on Rendering (EGSR04)*, pages 57–68. Eurographics, 2004.