GRAPHIC SHADERS FOR SCIENTIFIC VISUALIZATION

DAVID FERNÁNDEZ ALCOBA

DOBLE GRADO EN INGENIERÍA INFORMÁTICA - MATEMÁTICAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTESNE DE MADRID

Trabajo Fin Grado en Ingeniería Informática - Matemáticas

19 de julio de 2019

Director:
Ana Gil Luezas

Autorización de difusión

David Fernández Alcoba

19 de julio de 2019

El/la abajo firmante, matriculado/a en el Doble Grado en Ingeniería Informática y Matemáticas de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado "GRAPHIC SHADERS FOR SCIENTIFIC VISUALIZATION", realizado durante el curso académico 2018-2019 bajo la dirección de Ana Gil Luezas en el Departamento de Sistemas Informáticos y Computación, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

David Fernández Alcoba

Resumen

En el mundo actual, las investigaciones y estudios científicos generan gran cantidad de datos que han de ser interpretados de una manera eficaz, con el fin de sacar las mejores conclusiones y obtener resultados fiables que no den lugar a la duda. En este contexto, una de las disciplinas más importantes es la de la visualización, ya que puede ayudar a entender, ilustrar y obtener información relevante acerca del fenómeno que se está estudiando.

De igual forma, en los últimos años se ha dado una expansión considerable de las capacidades de las GPUs, ofreciendo nuevas posibilidades dentro de la informática gráfica e incrementando el rendimiento tanto en computación paralela como en aplicaciones de visualización.

En este trabajo se exploran estas ideas, haciendo hincapié en las posibilidades que nos ofrecen los distintos tipos de shaders gráficos dentro de la especificación OpenGL, y la manera en la que nos pueden ser útiles a la hora de interpretar datos y obtener representaciones para problemas típicos de visualización científica, como puede ser la visualización de datos en tres dimensiones, visualización de volúmenes, renderizado de curvas y superficies, etc.

Palabras clave

Visualización científica, GPU, Shader, OpenGL, Bézier, Superficies.

Abstract

Nowadays, scientific studies and investigations generate a great amount of data that has to be well interpreted, so as to extract the best possible conclusions and obtain reliable results. Within this context, one of the most important disciplines is that of visualization, since it can help understand, illustrate and obtain relevant information about the phenomenon being studied.

Similarly, in the last few years, a considerable expansion of the capabilities of GPUs has been taking place, offering new possibilities within graphics computing and increasing performance both in parallel computing and in visualization applications.

In this text those ideas are explored, emphasizing the possibilities that the different kinds of graphic shaders have to offer within the OpenGL specification, and the way these can help interpret data and obtain representations for common scientific visualization topics, such as three dimensions data Visualization, volume visualization, surface rendering, etc.

Keywords

Scientific Visualization, Graphic Shaders, GPU, OpenGL.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Tal y como se cuenta en Defanti and Brown [1], los científicos computacionales basan su trabajo en fuentes de datos de gran volumen. Sin embargo, estos datos tienen tal magnitud que los científicos se ven, a menudo, superados. Entre las fuentes de datos de gran volumen se encuentran:

- Supercomputadores
- Inteligencia militar, satélites, datos astronómicos y de tiempo atmosférico
- Sondas enviando datos desde otros planetas
- Radio telescopios terrestres
- Instrumentos capturando temperaturas oceánicas, movimientos tectónicos y actividad volcánica y sísmica
- Escáneres médicos empleando distintas técnicas de imagen como tomografía, resonancias magnéticas, etc

Simplemente con un formato numérico, el cerebro humano es incapaz de interpretar gigabytes de datos cada día, resultando en mucha información desperdiciada. De aquí surge la necesidad de una alternativa a los números. La posibilidad de los científicos para visualizar cómputos complejos y simulaciones es absolutamente esencial para asegurar la integridad de análisis y predicciones, así como presentar esta información al resto.

Esta capacidad de visualización se hace especialmente importante en el ser humano, puesto que, de todas nuestras funciones cerebrales, nuestro sistema de visión es el que mayor capacidad de procesamiento de información tiene. Según expertos en conocimiento, el procesamiento de información en humanos tiene dos formas: preconsciente y consciente. El procesamiento de información preconsciente es involuntario, similar a la respiración. Este es el tipo de procesamiento que se da en información gráfica. Rohrer [2]

Teniendo esto en cuenta y el hecho de que cada persona tiene una capacidad de vision espacial diferente, la informática gráfica puede ayudar a aquellos que tienen una mayor dificultad y que, de otro modo, serían incapaces de visualizar conceptos complejos.

Estos hechos muestran una necesidad ha resultado en el surgimiento, en la última década, de una disciplina totalmente independiente, la visualización científica.

1.1. Motivación

La importancia de lo expuesto anteriormente sirve como suficiente motivación, aunque a esto se ha de añadir el reto personal de, con este trabajo, aprender y entender un área de la informática que no forma parte del itinerario en mi formación, como es la informática gráfica, y que engloba muchas de las materias vistas hasta ahora tanto en ingeniería informática como en matemáticas.

Además, esta rama dentro de la investigación científica es relativamente reciente, asociándose su nacimiento en 1987 al artículo McCormick et al. [3], por lo que aún hay muchos retos y problemas por resolver, haciendo su estudio muy interesante.

1.2. Objetivos

El objetivo principal de este trabajo es el de aprender el funcionamiento básico de los gráficos y la aplicación de éstos a la investigación científica. Este objetivo se puede desglosar en otros subobjetivos más concretos y que marcan la línea de trabajo:

1. Comprender el pipeline de gráficos y la utilidad y funcionamiento de los shaders, así como aprender el lenguaje GLSL para su escritura.

2. Aprender las técnicas más conocidas de visualización científica y cómo desarrollar shaders que las implementen.

3. Desarrollar una aplicación que ponga de manifiesto lo aprendido, desarrollando shaders que ilustren algunas de las técnicas vistas.

1.3. Plan de trabajo

Con estos objetivos en mente, se desarrolló el siguiente plan de trabajo, acordado en reuniones iniciales entre tutora y autor del trabajo.

- **Toma de contacto con OpenGL** Durante esta fase se leyeron tutoriales sobre OpenGL y se experimentó con diversos shaders y librerías para familiarizarse con la tecnología, a la vez que se aprendía el lenguaje GLSL.

- **Documentación** Durante la duración completa del proyecto se llevó a cabo una documentación acerca de las distintas fuentes de información, con el objetivo de no olvidar incluir partes importantes en la memoria.
- **Comunicación con el tutor** Se concretaron diversas reuniones con la tutora durante las partes intensivas del proyecto con el fin de mostrar avances y acordar los siguientes pasos. Asimismo, se mantuvo una comunicación mediante correo electrónico para aquellas dudas menores que surgieron durante la realización del trabajo.
- **Preparación del entorno de desarrollo** Durante esta fase se preparó el equipo, instalando las librerías y programas necesarios para el correcto funcionamiento de la aplicación.
- **Desarrollo de la aplicación** Una vez preparado el entorno, se continuó durante toda la duración del trabajo con el desarrollo de la aplicación, incluyendo cada vez nuevas capacidades.
- **Redacción de la memoria** Se inició la redacción de la memoria una vez se tenían conocimientos suficientes, a mitad de la elaboración del trabajo. Una vez comenzada la redacción, se fue reeditando y mejorando en un proceso iterativo.

1.4. Estructura de la memoria

El siguiente capítulo, **OpenGL y DirectX** 2, presenta las dos grandes especificaciones dentro de la informática gráfica, centrándose en OpenGL y analizando sus características, capacidades y debilidades, así como las diferencias entre ambas.

Posteriormente, el capítulo **Shaders y Visualización Científica** 3 explora las técnicas más comunes dentro del campo de visualización científica y qué tipos de shaders son útiles para cada una de ellas, introduciendo algunos de los que más adelante se presentarán junto a la aplicación.

En el capítulo **Aplicación** 4 se presenta la aplicación desarrollada, explicando el diseño, capacidades, experimentos realizados...

Por último el capítulo **Conclusiones y Trabajo Futuro** 5 incluye un análisis del trabajo realizado, el nivel de cumplimiento de los objetivos propuestos y posibles líneas de trabajo futuro.

El código de la aplicación desarrollada puede encontrarse en el siguiente enlace:
http://github.com/davidfdezalcoba/TFG

Capítulo 2

OpenGL y DirectX

El objetivo de este capítulo es explicar en qué consiste OpenGl, así como su pipeline de gráficos, los tipos de shaders que incluye y las diferencias que presenta con DirectX, su principal competidor.

2.1. ¿Qué es?

OpenGL se define como una API *(application programming interface)*, que es simplemente una librería de software para acceder a capacidades del hardware de gráficos (ver Shreiner et al. [4]).

OpenGL está diseñado como una interfaz independiente del hardware que puede ser implementada en muchos sistemas hardware de gráficos diferentes, o completamente como software, en el caso de que el sistema no posea hardware de gráficos. OpenGL no proporciona ninguna funcionalidad para describir modelos en tres dimensiones ni operaciones para leer ficheros (como imágenes JPEG, por ejemplo). En su lugar, se deben construir los objetos tridimensionales a partir de un pequeño conjunto de primitivas geométricas—puntos, líneas, triángulos y parches.

2.2. Breve historia de OpenGL

OpenGL nace a principios de los años 90, desarrollada por Silicon Graphics (SGI). En los años 80, Silicon Graphics poseía una API privada denominada IRIS GL, utilizada para producir gráficos en sus estaciones de trabajo IRIS. Posteriormente, debido a la pérdida de cuota de mercado, decidió hacer su API pública. Sin embargo, a causa de problemas con patentes y el hecho de tener características poco relevantes para los gráficos 3D como la funcionalidad de ventanas, se decidió reescribir algunas de las partes y se lanzó lo que ahora se conoce como OpenGL.

Esta nueva especificación consiguió logros importantes para la informática gráfica, como estandarizar el acceso al hardware gráfico, trasladar a los fabricantes la responsabilidad del desarrollo de las interfaces con el hardware y delegar la funcionalidad de ventanas al sistema operativo. Todo esto supuso un gran impacto en la industria, al ofrecer a los desarrolladores una plataforma de alto nivel sobre la que trabajar.

En 1992, Silicon Graphics lideró la creación del OpenGL Architecture Review Board (OpenGL ARB) [5], un grupo de empresas del sector que sería la encargada de mantener y extender la especificación en los años siguientes. El OpenGL ARB estaba formado por 3Dlabs, Apple, ATI, Dell, IBM, Intel, Nvidia, SGI and Sun Microsystems.

En otoño de 2006, el OpenGL ARB y los directores de Khronos votaron para transferir el control de OpenGL a Khronos Group. El secretario de la ARB Jon Leech observó: *"Hemos decidido mover OpenGL a Khronos para asegurar la salud futura de OpenGL en todas sus formas."*[5]

2.3. Diseño

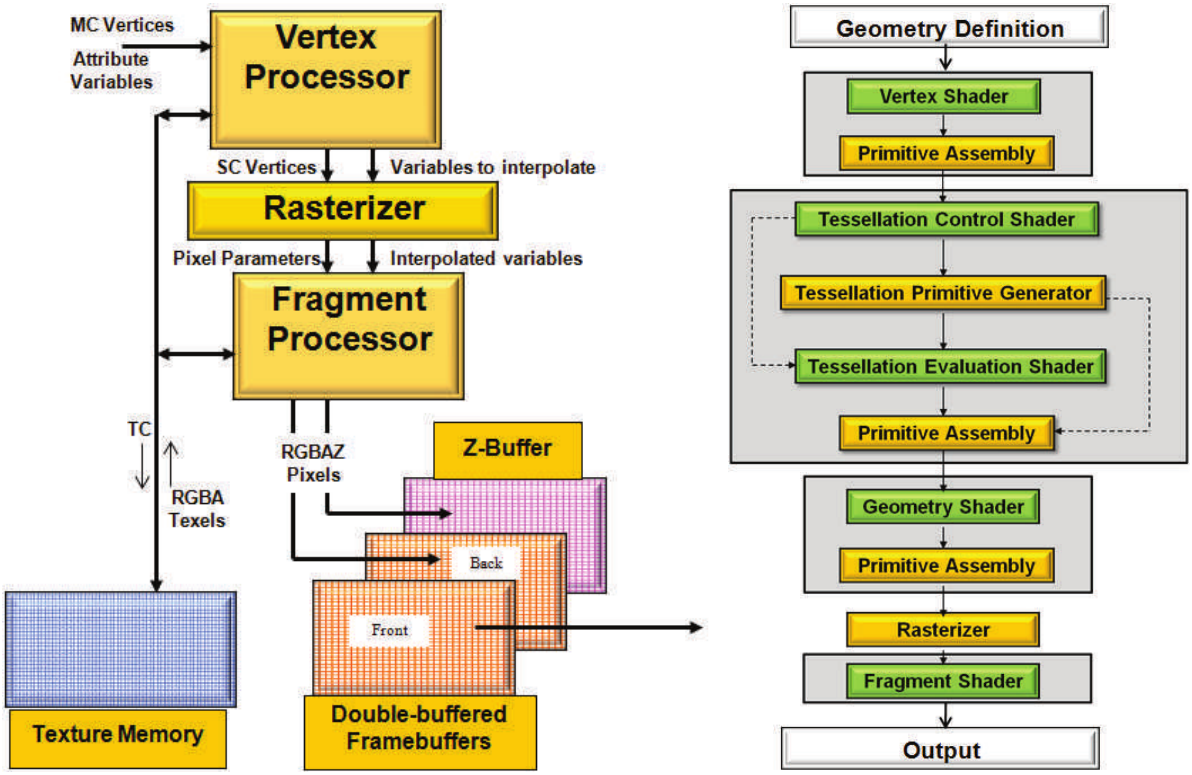

Figura 2.1: *Pipeline de OpenGl en el hardware gráfico.*

Capítulo 3

Shaders y Visulización Científica

Capítulo 4

Aplicación

Capítulo 5

Conclusiones y Trabajo Futuro

Bibliografía

[1] Thomas A. Defanti and Maxine D. Brown. Visualization in scientific computing. volume 33 of *Advances in Computers*, pages 247 – 307. Elsevier, 1991. doi: https://doi.org/10.1016/S0065-2458(08)60168-0. URL http://www.sciencedirect.com/science/article/pii/S0065245808601680.

[2] Matthew W. Rohrer. Seeing is believing: The importance of visualization in manufacturing simulation. In *Proceedings of the 32Nd Conference on Winter Simulation*, WSC '00, pages 1211–1216, San Diego, CA, USA, 2000. Society for Computer Simulation International. ISBN 0-7803-6582-8. URL http://dl.acm.org/citation.cfm?id=510378.510552.

[3] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 20(6), 1987.

[4] Shreiner, Dave, and The Khronos OpenGL ARB Working Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. Addison-Wesley Professional, 7th edition, 2009. ISBN 0321552628, 9780321552624.

[5] The Khronos Group Inc. About the OpenGL ARB. https://www.opengl.org/archives/about/arb/. Accessed: 19/07/2019.