MSc Computer Games and Entertainment

Maths & Graphics Unit 2011/12

Lecturer: Gareth Edwards

GPU Based Programmable Shaders

An Overview

Acknowledgments

Artwork and source code has been taken from various academic and commercial sources ,including:

- **Ogre3d - http://www.ogre3d.org/**
- **Synapse - http://www.synapsegaming.com/**
- **Unity3d - http://unity3d.com/**
- **Quest3d - http://quest3d.com/**
- **XNA – Dylan Helperin - http://forums.xna.com/forums/t/27849.aspx**
- **Frank Luna - 3D Game Programming with DirectX 9.0c: A Shader Approach**
- **Wikipedia**

Lecture Overview

Part 1 - Shaders

- What is a Shader?

Part 2 - Shaders in Action

- We are NOT going to look at specific Computer Games:

- We ARE going to look at a mixture of:
  - Game Engines – Ogre3d, Synapse, Unity
  - Visualisation Engines – Quest3d
  - Worked examples

Part 3 - Programming Shaders

- Techniques and Technologies
  - Light, Shadow, Environment Reflection and Displacement mapping
  - XNA
  - DirectX
  - HLSL

Part 1: Shaders

What is a GPU Based Shader?

- A set of software instructions, which is used primarily to calculate rendering effects on graphics hardware with a high degree of flexibility.

- Used to program the graphics processing unit (GPU) programmable rendering pipeline, which has mostly superseded the fixed-function pipeline that allowed only common geometry transformation and pixel shading functions.

- With GPU Based shaders, customized effects can be used.

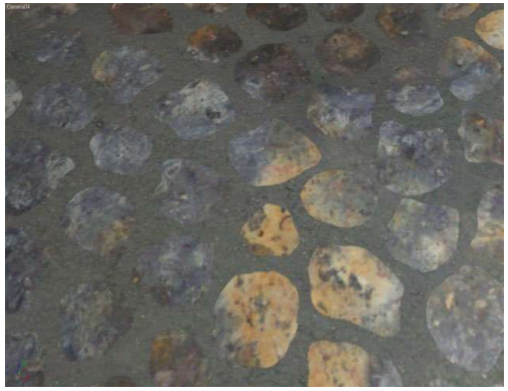
Why Bother?

- Because programmed shaders are visually far superior to the results that can be achieved using the standard rendering pipeline light response (illumination) model.

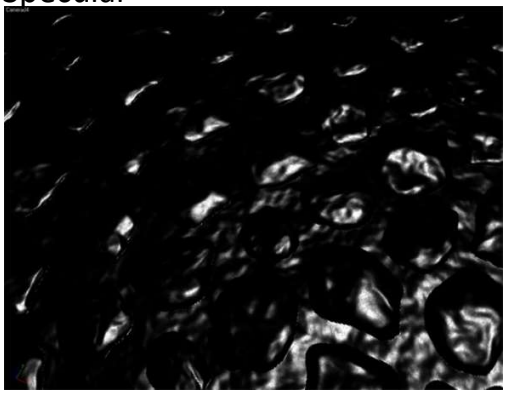
- Example: a pixel “Offset Mapping Shader”
  http://www.bencloward.com/shaders_offset.shtml

Complete Shader

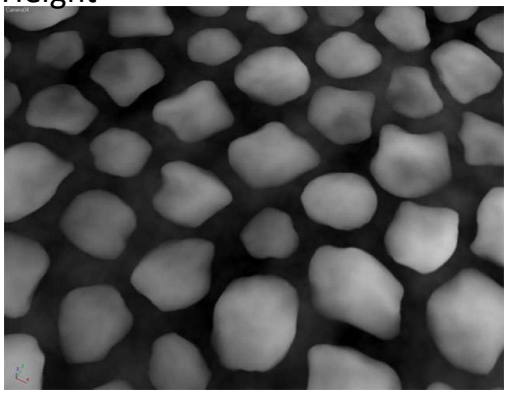
Diffuse

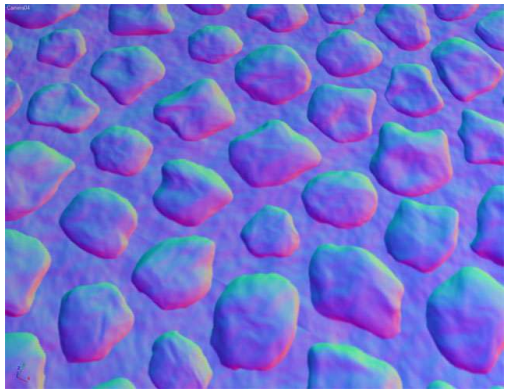


Specular

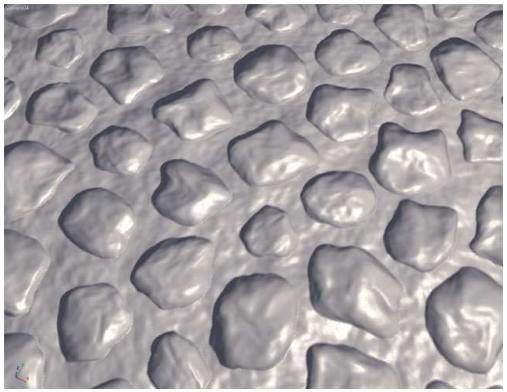


Height

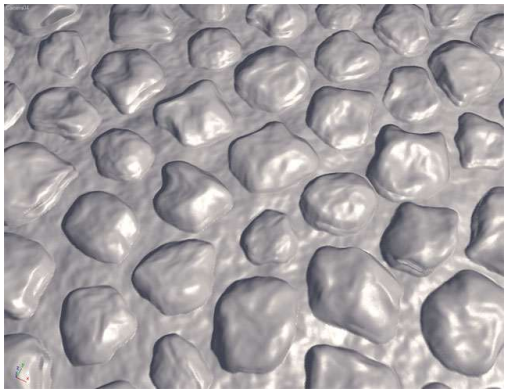




Normal



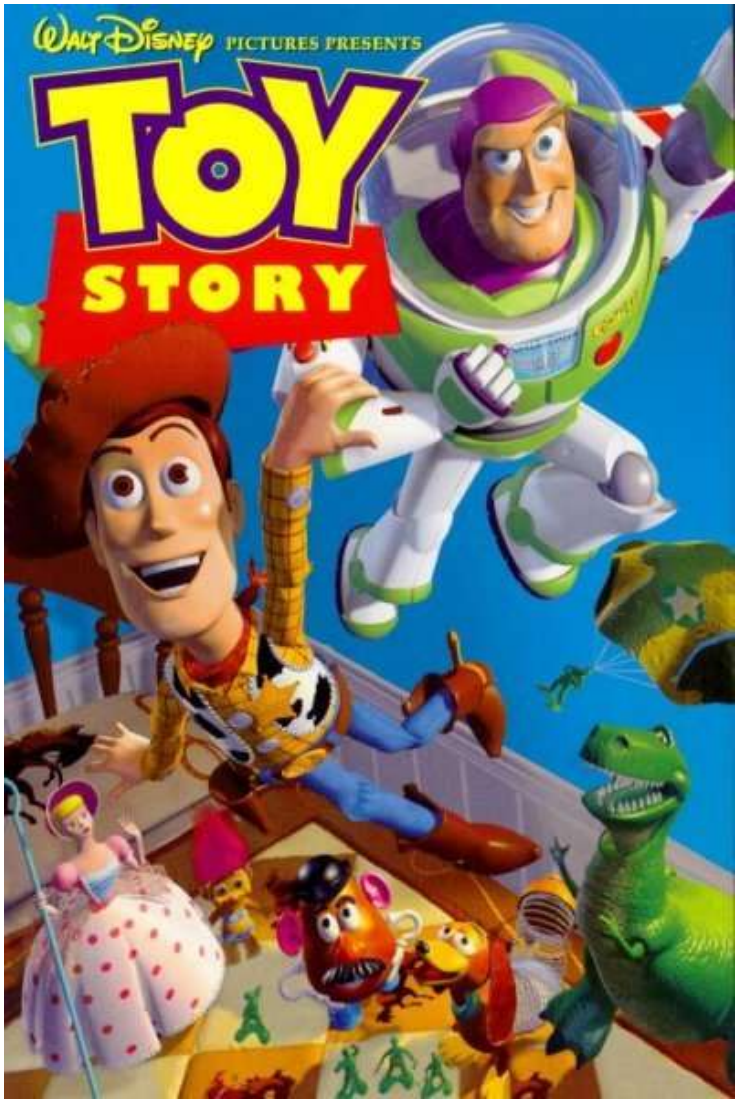
Diffuse & Specular



Texture Coords Offset

History

- The term "Shader" originated with Pixar's RenderMan - a program that takes an entire description of a scene including camera positions, object geometry and renders the final output. RenderMan was introduced in 1989, but it wasn't until the 1995 release of Pixar's movie "Toy Story" that the general public was introduced to the power of RenderMan. Such computer-generated imagery (CGI) became more and more prolific in movies and television.



The Moors
RenderMan
Free Jimmy
Gareth Edwards

The Moors
RenderMan
Free Jimmy
Gareth Edwards

History

- Initially, shaders were used to perform pixel shading only.
- The term stuck and is now used for other pipeline stages.
- As Graphics Processing Units evolved, major graphics software libraries such as OpenGL and Direct3D began to exhibit enhanced ability to program these new GPUs by defining special shading functions in their API.

Unity3d

Shader Overview

- Shaders are simple programs that describe the traits of either a vertex or a pixel. Vertex shaders describe the traits (position, texture coordinates, colours, etc.) of a vertex, while pixel shaders describe the traits (colour, z depth and alpha value) of a pixel.

- A vertex shader is called for each vertex in a primitive (possibly after tessellation) - thus one vertex in - one (updated) vertex out. Each vertex is then rendered as a series of pixels onto a surface (block of memory) that will eventually be sent to the screen.

- Shaders replace a section of video hardware that's typically called the Fixed Function Pipeline (FFP). This is because it performs lighting and texture mapping in a hard-coded manner, while shaders let you replace this hard-coded approach with a programmable one.

- Over the past two decades shaders and shader programming have become more and more sophisticated, providing for increasingly advanced image synthesis.


DirectX 7


DirectX 8.0


DirectX 9


DirectX 9
with HDR

Direct3D and OpenGL graphic libraries use three types of shaders

1) **Vertex** shaders are run once for each vertex given to the graphics processor.

   The purpose is to transform each vertex's 3D position in virtual space to the 2D coordinate at which it appears on the screen (as well as a depth value for the Z-buffer).

   Vertex shaders can manipulate properties such as position, colour, and texture coordinate, but cannot create new vertices.

   The output of the vertex shader goes to the next stage in the pipeline, which is either a geometry shader if present or the rasterizer.

Direct3D and OpenGL graphic libraries use three types of shaders

2) **Geometry** shaders can add and remove vertices from a mesh.

Geometry shaders can be used to generate geometry procedurally or to add volumetric detail to existing meshes that would be too costly to process on the CPU.

If geometry shaders are being used, the output is then sent to the rasterizer.

Direct3D and OpenGL graphic libraries use three types of shaders

3) **Pixel** shaders, also known as fragment shaders, calculate the colour of individual pixels.

   The input to this stage comes from the rasterizer, which fills in the polygons being sent through the graphics pipeline.

   Pixel shaders are typically used for scene lighting and related effects such as bump mapping and colour toning.

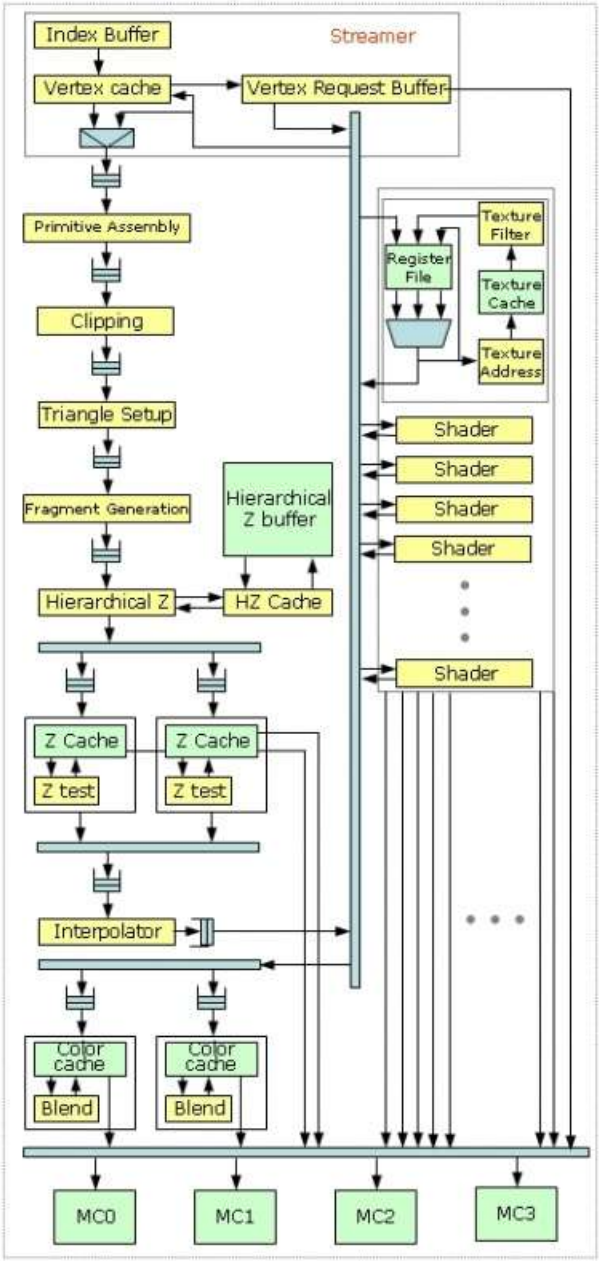
   Direct3D uses the term "pixel shader," while OpenGL uses the term "fragment shader.“

   The latter is arguably more correct, as there is not a one-to-one relationship between calls to the pixel shader and pixels on the screen.

   The most common reason for this is that pixel shaders are often called many times per pixel for every object that is in the corresponding space, even if it is occluded; the Z-buffer sorts this out later.

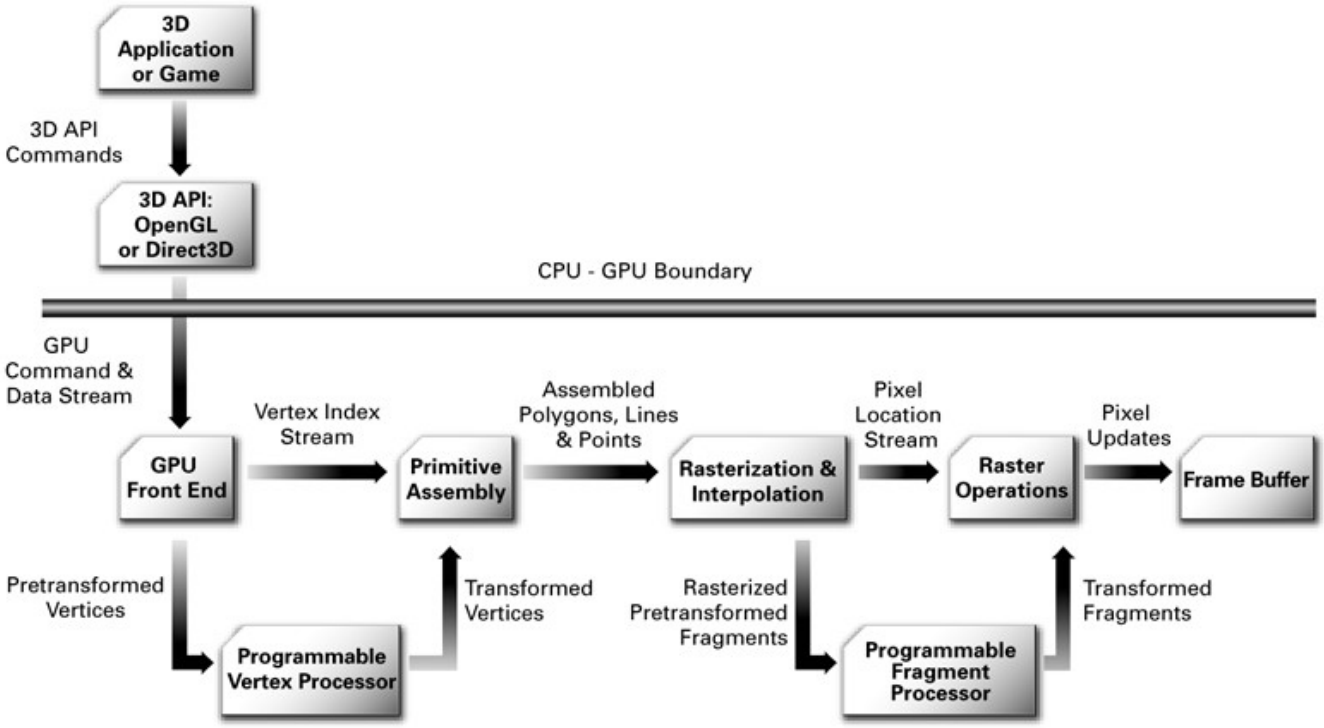
Unified Shader Model

- The unified shader model unifies the three aforementioned shaders in OpenGL and Direct3D 10.
- As these shader types are processed within the GPU pipeline.
- Known in Direct3D 10 as Shader Model 4.0, uses a consistent instruction set across all shader types. All shader types have almost the same capabilities - they can read from textures, data buffers and perform the same set of arithmetic instructions.
- However, the instruction set is not completely the same between different shader types!



Simplified graphic processing unit pipeline

- The CPU sends instructions (compiled shading language programs) and geometry data to the graphics processing unit, located on the graphics card.
- Within the vertex shader, the geometry is transformed and lighting calculations are performed.
- If a geometry shader is in the graphic processing unit, some changes of the geometries in the scene are performed.
- The calculated geometry is triangulated (subdivided into triangles).
- Triangles are transformed into pixel quads (one pixel quad is a 2 × 2 pixel primitive).



Shader Parallel Processing

- Shaders are written to apply transformations to a large set of elements at a time, for example, to each pixel in an area of the screen, or for every vertex of a model.

- This is well suited to parallel processing, and most modern GPUs have multiple shader pipelines to facilitate this, vastly improving computation throughput.

Shader Programming

- OpenGL (version 1.5 and newer) provides a C-like Shader language called OpenGL Shading Language, or GLSL.

- In the Microsoft Direct3D API (Direct3D 9 and newer), shaders are programmed with High Level Shader Language, or HLSL.

- Cg or C for Graphics is a high-level shading language developed by Nvidia in close collaboration with Microsoft for programming vertex and pixel shaders. It is very similar to Microsoft's HLSL.

Part 2: Shaders in Action

Shaders in Game Engines

- Ogre3d
  - Features
  - Graphics Engine Demo Video
  - Sun Effect Video

- Synapse
  - SunBurn Framework
  - SunBurn Lighting and Rendering
  - Real-time Fill Lighting Video

- Unity3d
  - Features
  - Shaders
  - Getting Started
  - Vertex and Fragment Programming
  - Shader Tutorial Video
  - Bumped Specular Light Video
  - Light Mapping Video

Shaders in Visualisation Engines

- Quest3d
  - Therme Vals Features
  - Therme Vals 3D
  - AquaJelly 3D
  - AquaJelly Video
  - Azure Temple

Part 3: Programming Shaders

XNA

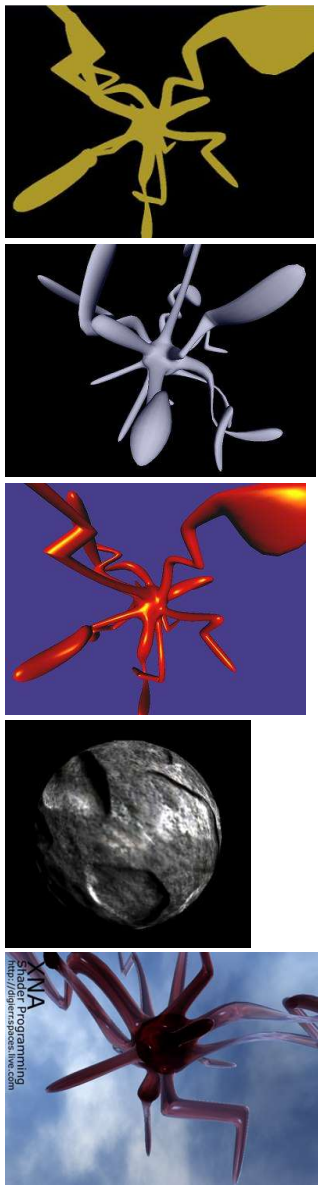
- **Microsoft XNA** ('XNA's Not Acronymed') is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management. XNA attempts to free game developers from writing "repetitive boilerplate code“ and to bring different aspects of game production into a single system.

- **XNA Build** is a set of game asset pipeline management tools, which help by defining, maintaining, debugging, and optimizing the *game asset pipeline* of individual game development efforts.

- **XNA Game Studio** is an integrated development environment (IDE) for development of games. Five revisions have been released so far.

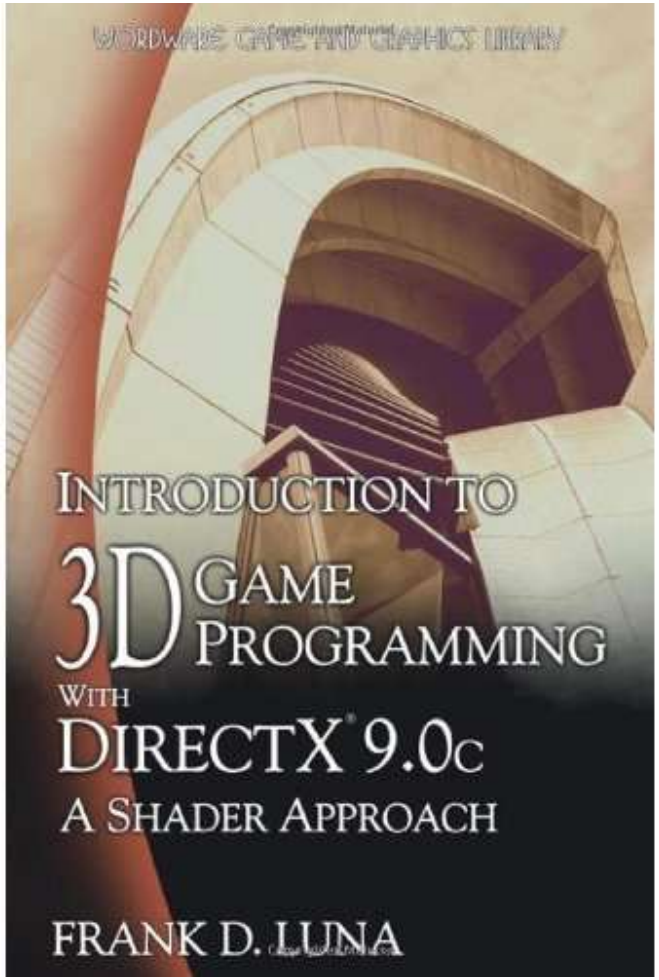
XNA Shader Examples

- Ambient
  - Text
  - Video
- Diffuse
  - Text
  - Video
- Specular
  - Text
  - Video
- Normal
  - Text
  - Video
- Refraction
  - Text
  - Video



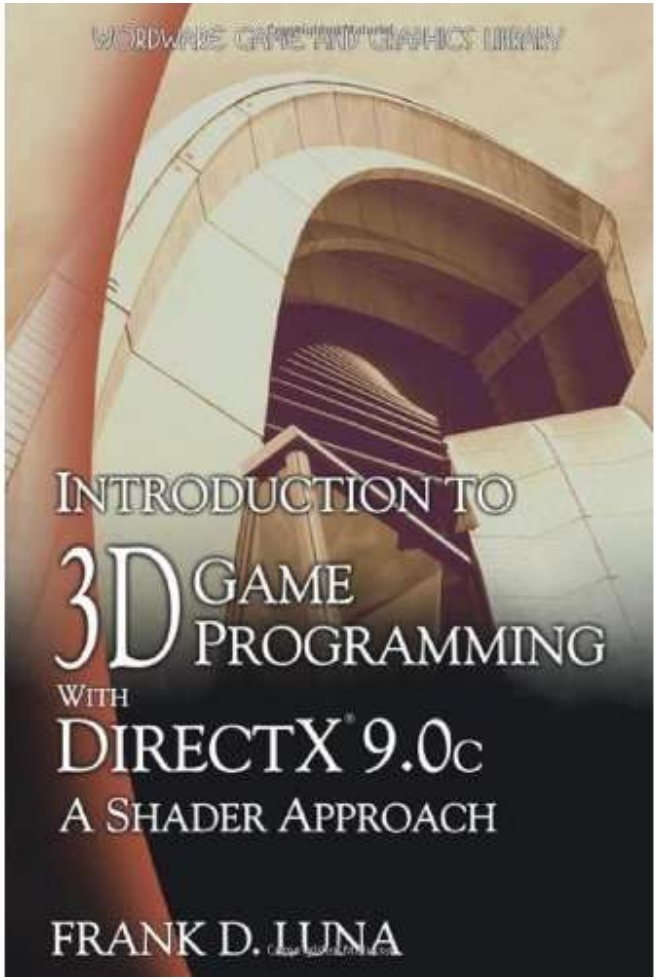
Frank Luna - 3D Game Programming with DirectX 9.0c: A Shader Approach

- Emphasis on game development, using real-time shaders with DirectX 9.0
- Three parts that explain basic mathematical and 3D concepts, show how to describe 3D worlds and implement fundamental 3D rendering techniques, and demonstrate the application of Direct3D to create a variety of special effects
- Not too difficult introduction
- Download demos from the web at http://www.d3dcoder.net/d3d9c.aspx#summary



Frank Luna - 3D Game Programming with DirectX 9.0c: A Shader Approach

http://www.d3dcoder.net/d3d9c.aspx#summary



- Supplement Files:
  - Project Setup in Visual Studio 2005
  - Detailed Table of Contents
  - Source Code Part I
  - Source Code Part II
  - Source Code Part III
  - Source Code Appendix A
  - Part I Solutions
  - FAQ
  - Errata

Frank Luna - 3D Game Programming with DirectX 9.0c: A Shader Approach

http://www.d3dcoder.net/d3d9c.aspx#summary

END