Hogeschool van Amsterdam

Advanced Graphics Programming

How To

2015-2016

Alexander Bonnee,
Richard de Koning,
Juriaan Mulder,
Reggie Schildmeijer

CREATING TOMORROW

Course Overview

- Relevance
- Goals
- Literature
- Schedule
- Assignments & Grading
- Rubric

Relevance

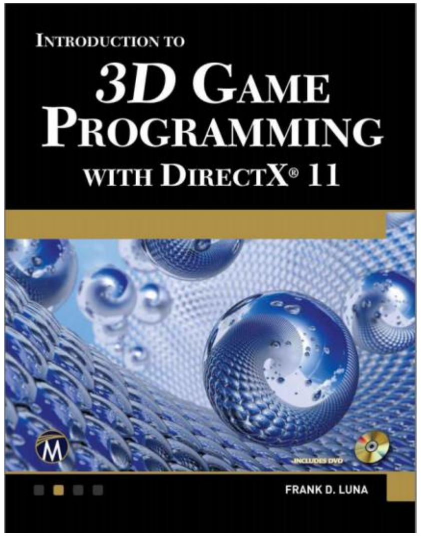
Visual appearance is paramount and all games use a graphics renderer. Graphics Programming is crucial to get the most out of the hardware.

- Graphics programming will focus on different rendering techniques to generate an image from a 2D or 3D model
- The history and relevance of the programmable graphics pipeline are discussed and the best of breed tools and libraries in graphics developments are used.

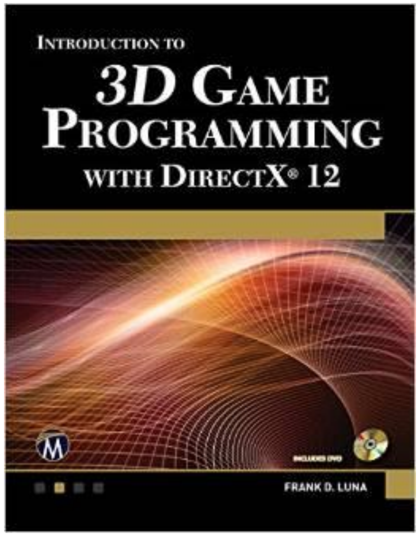
Goals

- Be familiar with the history of Graphics Programming. Understand the broad strokes of the DirectX and OpenGL programmable graphics pipeline and know commonly used graphics terminology
- To evaluate performance of the rendering pipeline, and understanding the limits of draw calls, vector- and fill rate cost.
- Be familiar with and use different lighting in a scene. Understanding the different shading techniques to render smooth surfaces
- Be familiar with common used mapping techniques (specular mapping, normal mapping, parallax mapping, opacity mapping, etc.)
- To understand texture related principles like pixel sampling and mipmaps; To understand its impact on performance and memory and be familiar with compression techniques

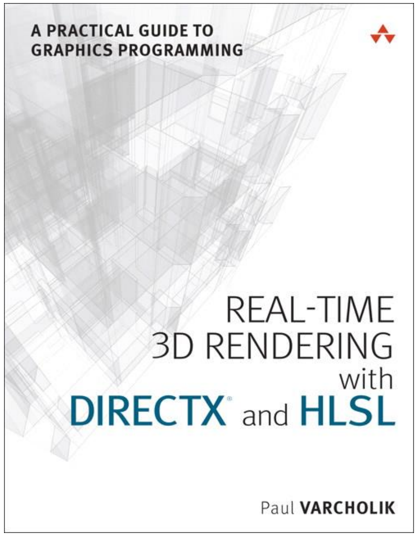
Literature



Now



March 2016



Recommended

- C++ (Frank Luna): http://www.d3dcoder.net/d3d11.htm
- C# version: http://www.richardssoftware.net/Home/DirectX11Tutorials

Schedule

Week	1st lecture	2nd lecture	Luna Chapters	
1	How To 3D Basics Luna Overview	Rendering Pipeline & Direct3D initialization	1-3	4, 5
2	Drawing primitives	Models, Meshes & Camera	6	14, 23
3	Pipeline Performance	Instancing, Blending & Stenciling		9, 10, 15
4	Lighting & Smoothing	Texturing	7	8
5	[progress meeting]	Texture Performance		17
6	Shading	Virtual Reality	18	
7	Shadow Mapping	Post Render Effects	21	

Remarks

The following chapters will be skipped, but are recommended for achieving excellent grades!

- Chapter 1-3, vectors/matrices (pre-requisite)
- Chapter 11-13, Geometry, Compute, Tessellation Shader
- Chapter 16, Picking
- Chapter 19-20, Terrain Rendering, Particle systems
- Chapter 24 Quaternions

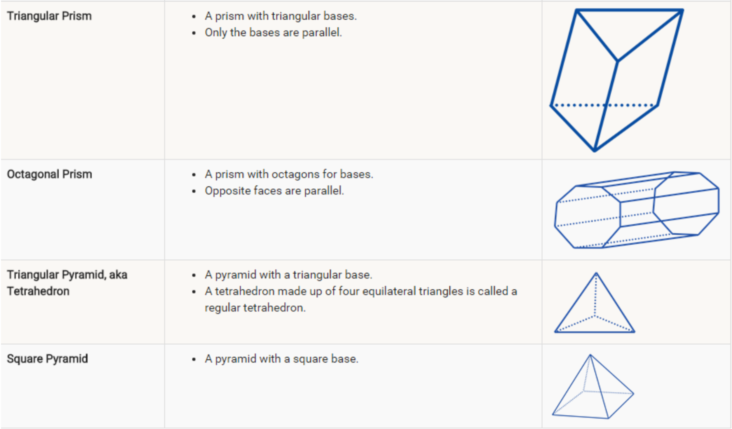
Assignments & Grading

- Git repo and report about assignments on

1. Primitives
2. Models
3. Blending
4. Lighting
5. Texturing
6. Shading

1 Assignment primitives

- Modify project **BoxDemo** to create a square pyramid; can you show it in wireframe? Add another 3D object to the scene from the examples given below.
- CPU: add a n-prism to the scene: for example a 8-prism called an octagonal prism.
- GPU: use a (tesselation) shader to ‘calculate’ the geometry on the GPU.



Triangular Prism	• A prism with triangular bases. • Only the bases are parallel.	
Octagonal Prism	• A prism with octagons for bases. • Opposite faces are parallel.	
Triangular Pyramid, aka Tetrahedron	• A pyramid with a triangular base. • A tetrahedron made up of four equilateral triangles is called a regular tetrahedron.	
Square Pyramid	• A pyramid with a square base.	

2 Assignment models

- Modify project **MeshView** so a mesh can be stored in binary format and name this format .M3B
- Study the .obj file format and Modify project **MeshView** so .obj files can be rendered.
- Study the Open Asset Library and write a viewer for one of its formats (.blend for example).

3 Assignment blending & stenciling

- Create a (very simple) demo scene with 2 triangles and demonstrate: different (color) blend modes and alpha blend modes
- Add an additional (transparent) triangle and draw them in different orders; explain what happens.
- Limit blending effects to a viewport or layer using a stencil.

4 Assignment Lighting

(book, ch 7.14 & 7.16)

Use the lighting demo of chapter 7(Luna)

- Modify the directional light, point light and spotlight; let them emit different colors
- Change the specular power component of the used material
- Create a toon shader effect by implementing a discrete ‘step’ function for the pixel shader (see 7.16)

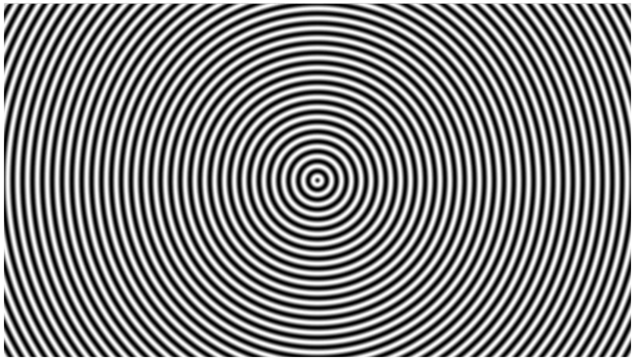
5 Assignment texturing

- Part A
  - Alter the crate demo of Chapter 8 to create a dice instead of a crate. Do this in two ways:
  - Look up the lay-out of a dice
  - Using your favorite paint program, create a complete texture for the dice (use the unfolding technique)
  - Create a single texture just for a dot and render the dice using only this texture to display all the dots on the dice.
- Part B
  - Find front and back images of a mobile phone.
  - Render a rectangular box and apply the texture to make it look like a mobile phone
  - Take a picture with your mobile phone and apply this picture to the rendered phone (as if you view the picture on that phone)
  - Render the scene from the viewpoint of the phone and apply this view as a texture on the phone (i.e. “the phone’s camera is switched on”).

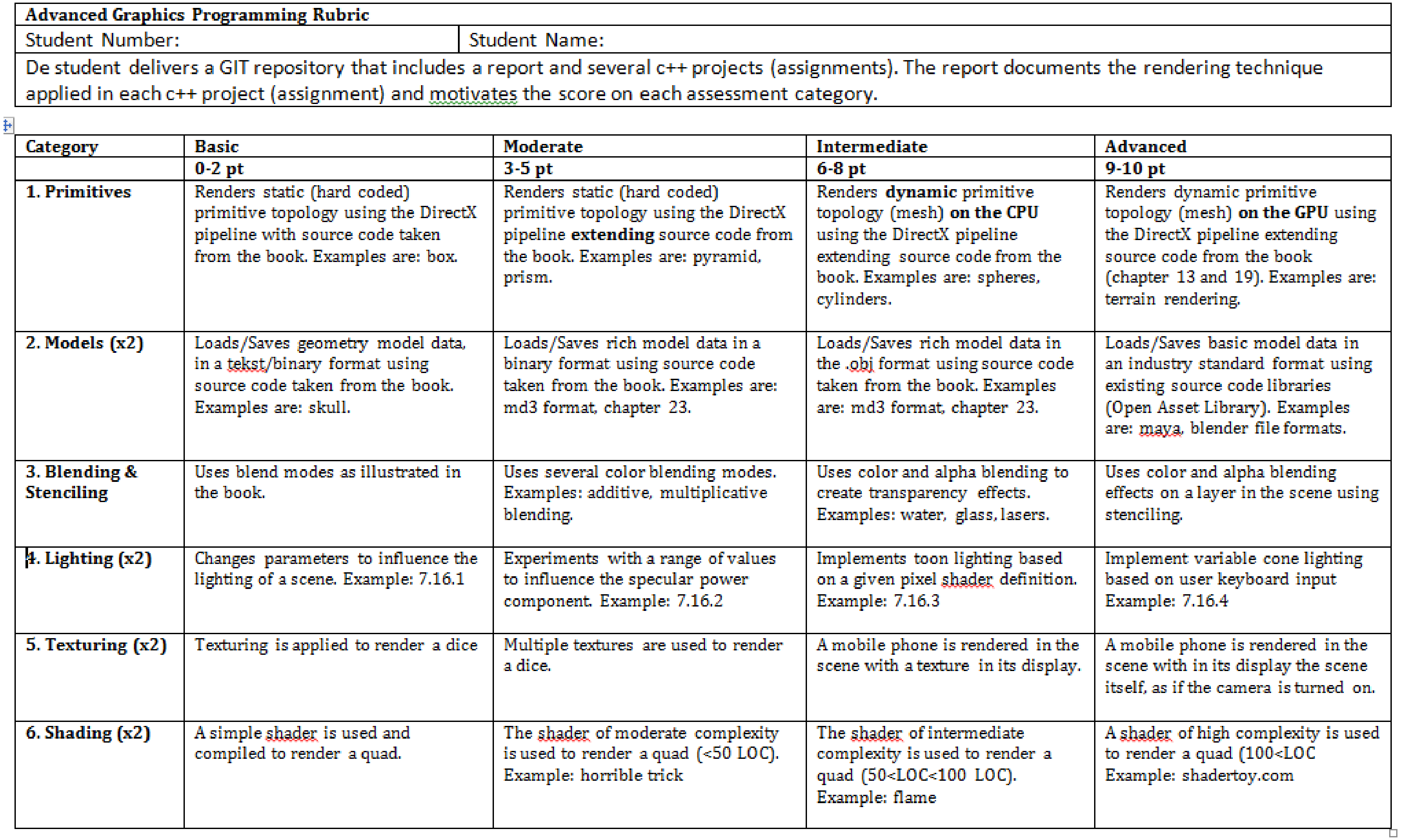
HBO-ICT
GAME
DEVELOPMENT

6 Assignment shading

- Explore shadertoy and study shader ‘horrible trick’ (https://www.shadertoy.com/view/lt2XWc); create a quad (2 triangles) and port this shader
- Study and port flame (https://www.shadertoy.com/view/MdX3zr)
- Chose a shader of your choise and port



Rubric



Advanced Graphics Programming Rubric	
Student Number:	Student Name:
De student delivers a GIT repository that includes a report and several c++ projects (assignments). The report documents the rendering technique applied in each c++ project (assignment) and motivates the score on each assessment category.	

Category	**Basic**	**Moderate**	**Intermediate**	**Advanced**
	0-2 pt	**3-5 pt**	**6-8 pt**	**9-10 pt**
1. Primitives	Renders static (hard coded) primitive topology using the DirectX pipeline with source code taken from the book. Examples are: box.	Renders static (hard coded) primitive topology using the DirectX pipeline **extending** source code from the book. Examples are: pyramid, prism.	Renders **dynamic** primitive topology (mesh) **on the CPU** using the DirectX pipeline extending source code from the book. Examples are: spheres, cylinders.	Renders dynamic primitive topology (mesh) **on the GPU** using the DirectX pipeline extending source code from the book (chapter 13 and 19). Examples are: terrain rendering.
2. Models (x2)	Loads/Saves geometry model data, in a tekst/binary format using source code taken from the book. Examples are: skull.	Loads/Saves rich model data in a binary format using source code taken from the book. Examples are: md3 format, chapter 23.	Loads/Saves rich model data in the .obj format using source code taken from the book. Examples are: md3 format, chapter 23.	Loads/Saves basic model data in an industry standard format using existing source code libraries (Open Asset Library). Examples are: maya, blender file formats.
3. Blending & Stenciling	Uses blend modes as illustrated in the book.	Uses several color blending modes. Examples: additive, multiplicative blending.	Uses color and alpha blending to create transparency effects. Examples: water, glass, lasers.	Uses color and alpha blending effects on a layer in the scene using stenciling.
4. Lighting (x2)	Changes parameters to influence the lighting of a scene. Example: 7.16.1	Experiments with a range of values to influence the specular power component. Example: 7.16.2	Implements toon lighting based on a given pixel shader definition. Example: 7.16.3	Implement variable cone lighting based on user keyboard input Example: 7.16.4
5. Texturing (x2)	Texturing is applied to render a dice	Multiple textures are used to render a dice.	A mobile phone is rendered in the scene with a texture in its display.	A mobile phone is rendered in the scene with in its display the scene itself, as if the camera is turned on.
6. Shading (x2)	A simple shader is used and compiled to render a quad.	The shader of moderate complexity is used to render a quad (<50 LOC). Example: horrible trick	The shader of intermediate complexity is used to render a quad (50<LOC<100 LOC). Example: flame	A shader of high complexity is used to render a quad (100<LOC Example: shadertoy.com