**GLSL**

http://nehe.gamedev.net/article/glsl\_an\_introduction/25007/

a short term for the official OpenGL Shading Langauge.

Similar to C/C++ level programming language for several parts of the graphics card

With GLSL I can code short programs, called shaders, which are executed on the GPU

**What's the difference between Fixed Function Pipeline and GLSL**

vertex and fragment shaders

a vertex shader operates on every vertex (glVertex or glDrawArrays)

**Geometry Shader**

**Vertex Processor**

http://www.lighthouse3d.com/tutorials/glsl-tutorial/vertex-processor/

**Vertex Shader (rasterization: changing 3D into 2D images)**

Vertex Transformation

Normal Transformation, Normalization and Rescaling

Lighting

Texture Coordinate Generation and Transformation

**Fragment Shader**

Texture access and application (Texture environments)

Fog

**Bump\_mapping**

http://www.ozone3d.net/tutorials/bump\_mapping.php

http://www.blacksmith-studios.dk/projects/downloads/tangent\_matrix\_derivation.php/

**oZone3D.Net tutorials**

http://www.ozone3d.net/tutorials/

http://ogldev.atspace.co.uk/index.html

http://research.ncl.ac.uk/game/mastersdegree/graphicsforgames/

**Frame buffer explained**

**Depth Testing**

when a pixel is drawn, it's assigned a value (called the z value) that denotes its distance from the viewer's perspective. Later when another pixel needs to be drawn to the screen location, the new pixel's z value is compared to that of the pixel that is already stored there. This maneuver is accomplished internally by a depth buffer with storage for a depth value for every pixel on the screen. Almost all the samples in the book use depth testing.