Introduction

- Two ways of 3D computer graphics
  - 1. Pre-rendering
    - Ray tracing, ray casting, radiosity
    - Precise, even physically correct
    - Slow
  
  - 2. Real-time rendering
    - DirectX, OpenGL
    - To fake or not to fake
    - Hardware accelerated
    - Fast
Real-time Rendering
Yet another way to render 3D content
We can create graphics and special effects that will appear nearly identical on any operating system and any hardware that supports OpenGL.

Graphics library
Specify a set of commands
Each command executes a drawing action or creates a special effect
Texture mapping, transparency, antialiasing, fog, and lighting effects
Infinite amount of possibilities
Development

- Long way – 20 years
- **OpenGL 1.X** – Fixed Function
  - **OpenGL 2.X** – Vertex and Fragment Shaders
  - **OpenGL 3.X** – Geometry Shaders
  - **OpenGL 4.X** – Tessellation and Compute

- Your version?
OpenGL support in C#

Support – not a new library!
Same functions
Same principles
Under development

Advanced projects? – ship to C++ if necessary
Console vs. WinForms

- Console
  - Game.cs
  - Faster but without GUI

- WinForms
  - GUI
  - Slower, but fast enough for simple applications

- More?
  - OpenTK documentation
OpenGL vs. OpenTK

- Same parameters
- Proof

**OpenGL**
- glLoadIdentity
- glMatrixMode
- glClear
- glViewport
- gluLookAt
- gluPerspective
- etc.

**OpenTK**
- GLLoadIdentity
- GLMatrixMode
- GL.Clear
- GL.Viewport
- Matrix4.LookAt
- Matrix4.
- CreatePerspectiveFieldOfView
- etc.
Exercise Plan

- Not strict
- Basic OpenGL
  - Console vs. WinForms
  - Where, what and when

- Working environment
  - Viewports, perspective and orthogonal projection
  - Text rendering
  - Camera

- Rendering content
  - Shaders
Graphics Pipeline

- Based on architecture of graphics cards
- Input = geometry and its properties
- Output = pixels
- Pipeline constantly evolves
- Simple case

Diagram:

1. Application
2. Geometry
3. Rasterizer
Graphics Pipeline

- Uniform state
- Vertex array
- Element array
- Vertex shader
- Triangle assembly
- Rasterization
- Fragment shader
- Testing and blending
- Framebuffer
OpenGL 4.X
Shaders

- Simple programs which provide a programmable alternative in graphics pipeline
- Replace a section of video hardware typically called the Fixed Function Pipeline (FFP)
Vertex Shaders

- Transform each vertex's 3D position in virtual space to the 2D coordinate
- Properties such as position, color, and texture coordinate
- Input
  - Data (Fields of vectors etc.)
- Output
  - To geometry shader if present
  - Otherwise to rasterizer
Geometry Shaders

- Generate new graphics primitives, such as points, lines, and triangles
- Geometry tessellation, shadow volume extrusion

Input
- Whole primitives (triangle etc.)

Output
- For each triangle emit zero or more primitives
Fragment Shaders

- Compute color and other attributes of each pixel
- Bump mapping, shadows, specular highlights, translucency
- Input
  - Pixels after rasterization
- Output
  - Pixels
All in All

- We could divide OpenGL program
  - Basic structure
    - Init, Reshap, On render frame, On update frame

- Environment
  - Projection, Camera, Viewports

- Rendering
  - Phong shader, etc.
Now you know how 3D graphics program works
- Blender, 3Ds Max, Maya, Zbrush

Blender
- Modeling environment and game engine in real-time graphics
- But if you hit the render button
  - Pre-render
  - Compute geometry and start ray tracing
All in All
Do not try to remember each GL function
- Remember its **purpose instead**

Do not try to remember each setup of OpenGL
- OpenGL in C#, console, WinForms, C++ ...
- “for any of these, the setup is kind of different and kind of same – so don’t worry, if you can learn one use of OpenGL, any other use of OpenGL you can learn very, very quickly”
OpenGL – Read This

❖ Base
  ❖ http://www.opengl.org/

❖ Books – older editions
  ❖ http://www.starstonesoftware.com/OpenGL/

❖ Tutorials
  ❖ http://www.arcsynthesis.org/gltut/

❖ Google, GLSL, Shading, Shaders ...
OpenTK – Read This

- [http://www.opentk.com/doc](http://www.opentk.com/doc)
- “must read” – ~120 pages
- From your first triangle to shaders in C#

- Documentation in your installed OpenTK
  - Examples

- OpenTK projects
  - [http://www.opentk.com/project/all](http://www.opentk.com/project/all)
Better Pages

- http://www.cg.tuwien.ac.at/courses/Realtime/slides.html