Introduction to Graphics Programming and its Applications

繪圖程式設計與應用

OpenGL Overview

Instructor: Hung-Kuo Chu

Department of Computer Science

National Tsing Hua University

CS4595

What is OpenGL ?

Yes or No?

- ✗ OpenGL = **Open G**ame **L**ibrary
- ✗ OpenGL is a programming language
- ✗ OpenGL is for windows programming
- ✓ OpenGL is OS independent
- ✓ OpenGL is windows system independent

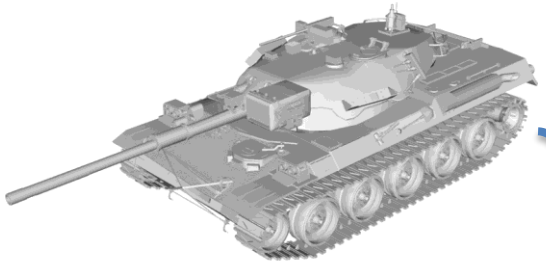
About OpenGL

- Abbreviated from **Open G**raphics **L**ibrary.
- OpenGL is a ***graphics rendering API***.
- Accessing features in graphics hardware.
- Over 500 commands to specify image, texture, object and shader programs.
- Producing interactive 3-dimensional computer-graphics applications.

About OpenGL (cont'd)

- OpenGL is designed as:
  - A streamlined, hardware-independent API.
  - A primitive-based rendering pipeline.
  - A client-server system.
- OpenGL is implemented by:
  - Video card manufacturer for each hardware/OS
    - NVidia, AMD.
  - OS provider (software rendering).
  - Yourself! (***Computer Graphics CS 5500***)

About OpenGL (cont'd)



3D Model

Texture

Light =(1.5, 0.6, 0.2)
Material=……

OpenGL®

Your OpenGL application

Rendered image

SGI and GL

- Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware (1982).
- Application programmers built a library called **GL** to access the system.
- The GL library largely simplifies programming 3D interactive applications.

The Birth of OpenGL

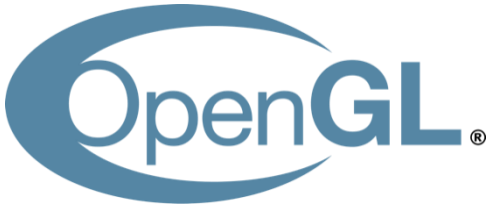
- The success of GL leads to OpenGL (1992), a platform-independent rendering API that is:
  - Easy to use.
  - Focusing on rendering.
  - Close enough to the hardware to achieve excellent performance.
  - Omitting windowing and input to avoid window system dependencies.

OpenGL Evolution

- Originally maintained by an Architectural Review Board (ARB)
  - Members include SGI, Microsoft, NVidia, HP, 3DLabs, IBM, etc.
- Current stable version is 4.5
  - Evolution reflects new hardware capabilities
    - 3D texture mapping and texture objects
    - Vertex programs
- Allowing platform specific features through extensions.
- ARB is substituted by Kronos group.

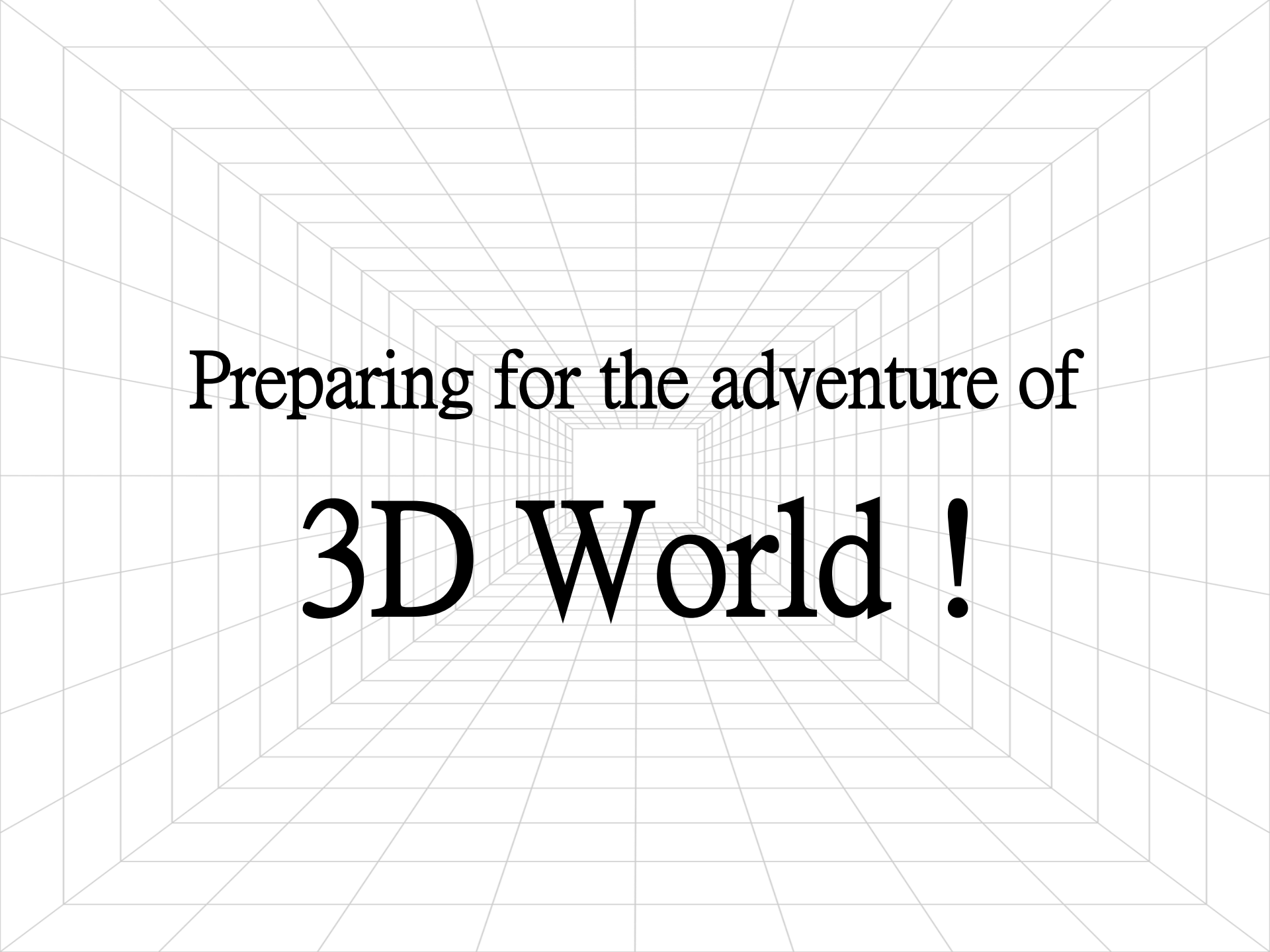
OpenGL for Different Platforms

- OpenGL
  - The industry's foundation for high performance graphics.
- OpenGL ES
  - The standard for embedded accelerated 3D graphics.
- WebGL
  - OpenGL ES 2.0 for the web.



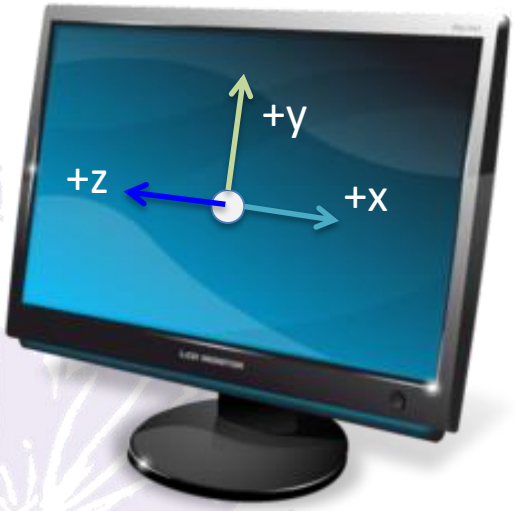
Vulkan

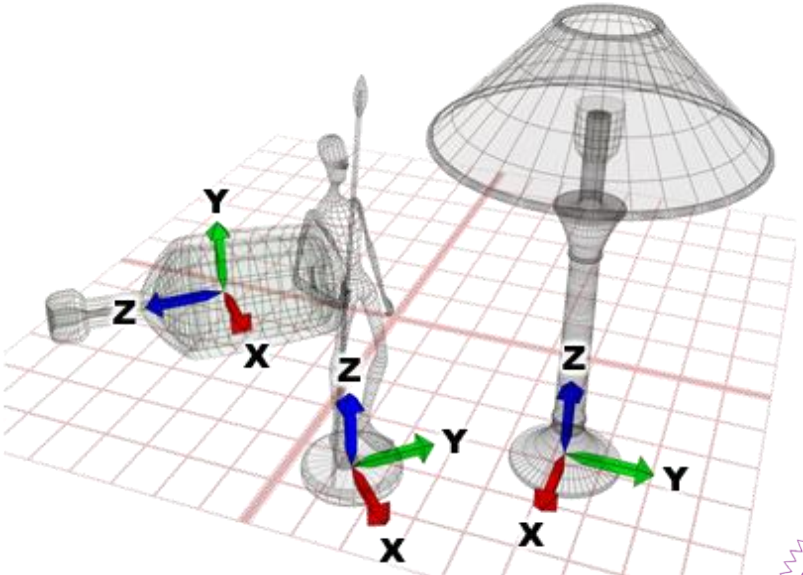
- A.K.A. OpenGL Next
- Idea similar to AMD Mantle and Direct3D 12
- A Fully cross-platform API
  - Windows, Linux, Android…
  - Google has acquired the mobile group of LunarG, the develop team of the Vulkan driver


Preparing for the adventure of
3D World !

Cartesian Coordinate System

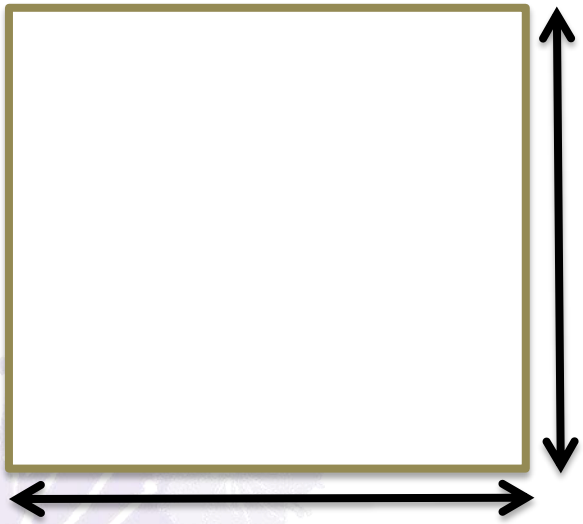
- A space where you draw things.
- 3D world/local coordinate.
- 2D screen coordinate.



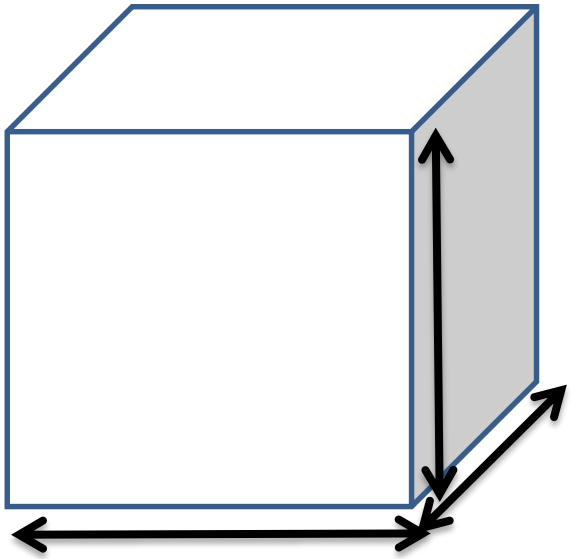


3D Primitives

- Objects with “three” dimensional measurements.



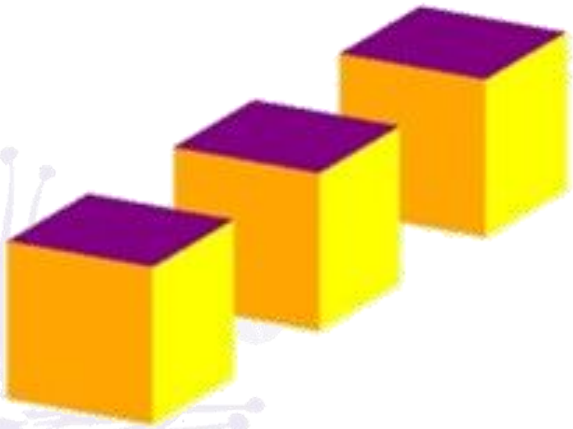
2D Measurement



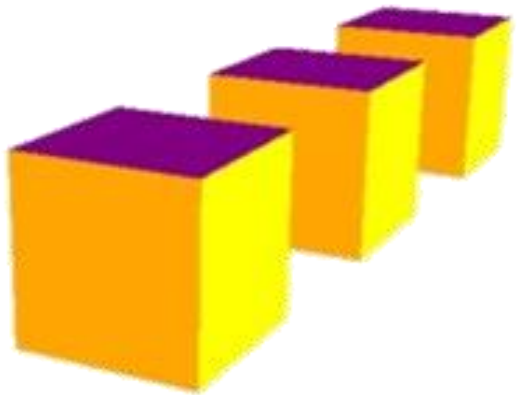
3D Measurement

A Viewing Camera

- Orthographic view
  - The length of a projected 3D line segments is invariant to its distance from camera.
- Perspective view
  - Distant objects look smaller than near ones.

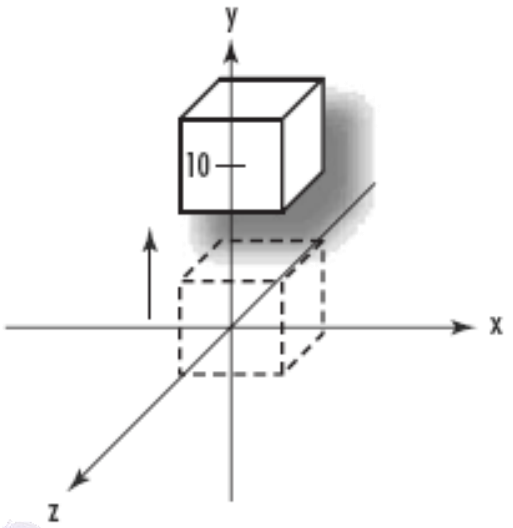


orthographic view

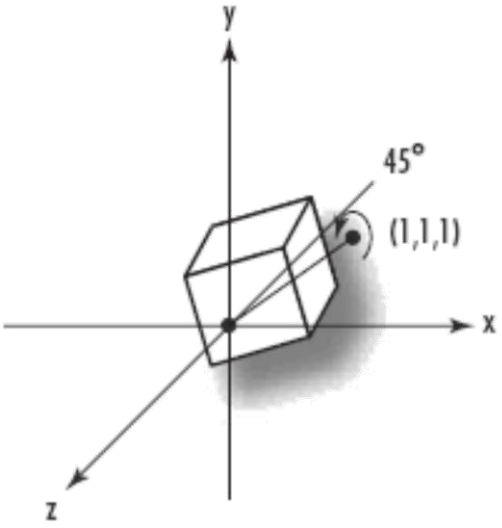


perspective view

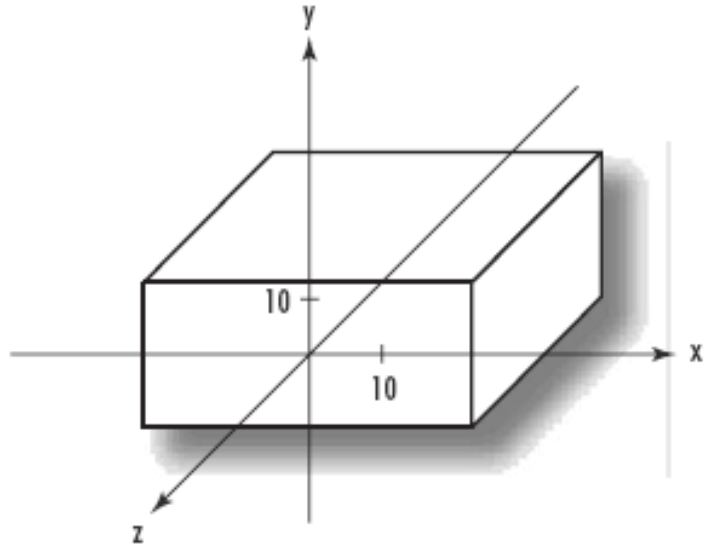
3D Transformations



translation



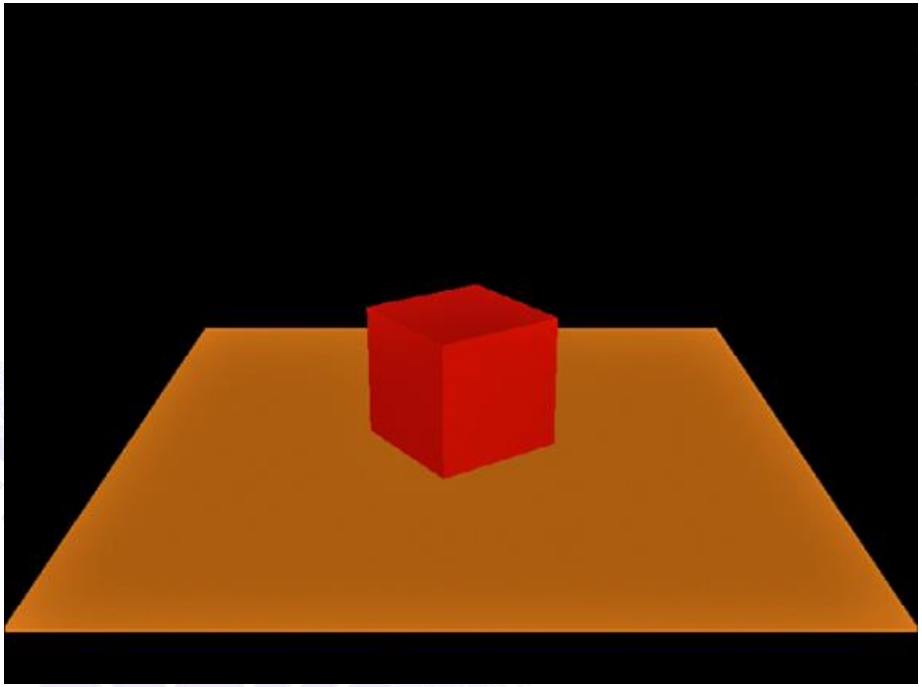
rotation



scaling

Material and Lighting

- Color, texture, shading…etc.



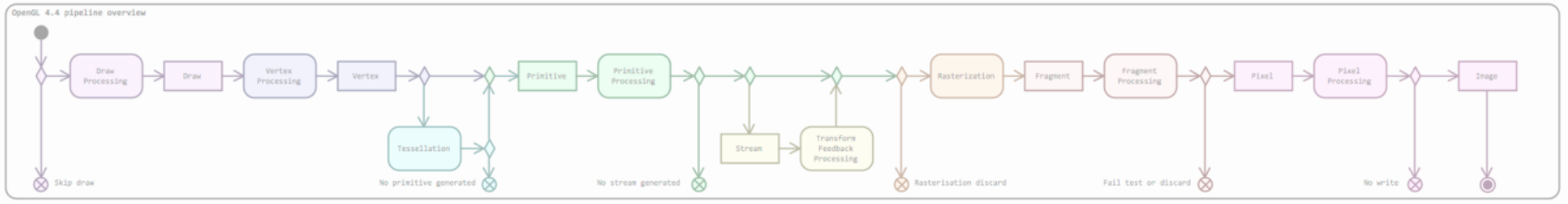
Render to Frame-Buffer (Screen)

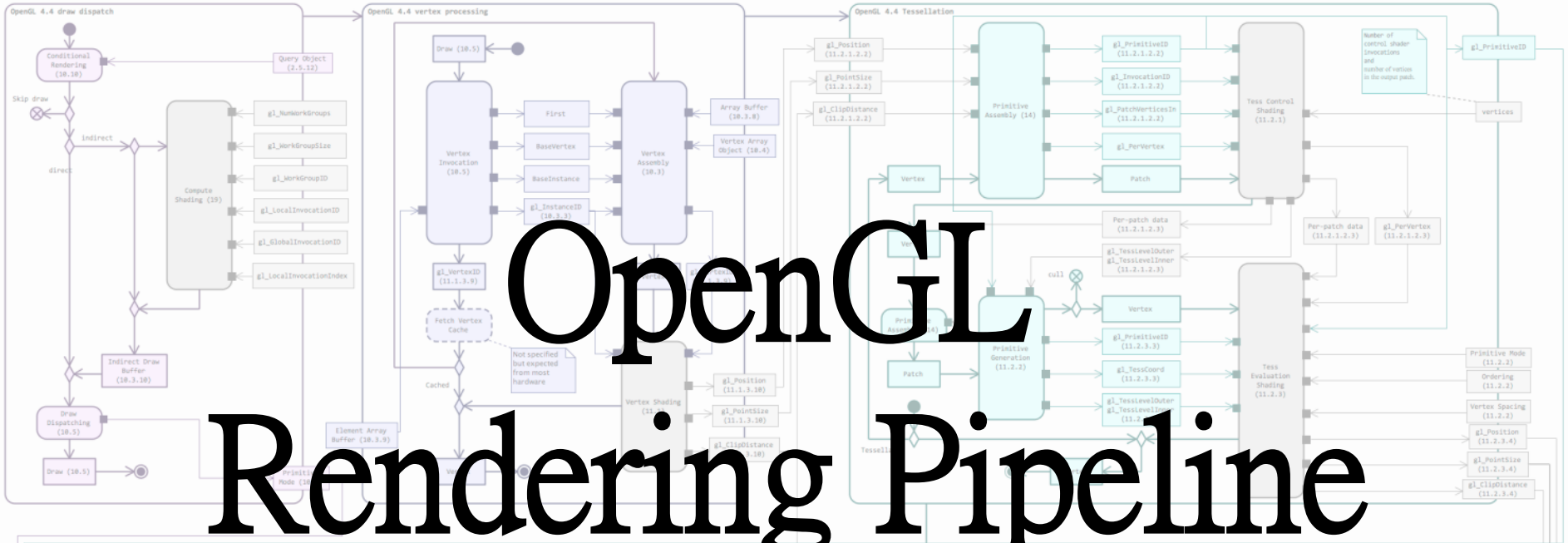
So…What does OpenGL Concern?

The Spirit of OpenGL

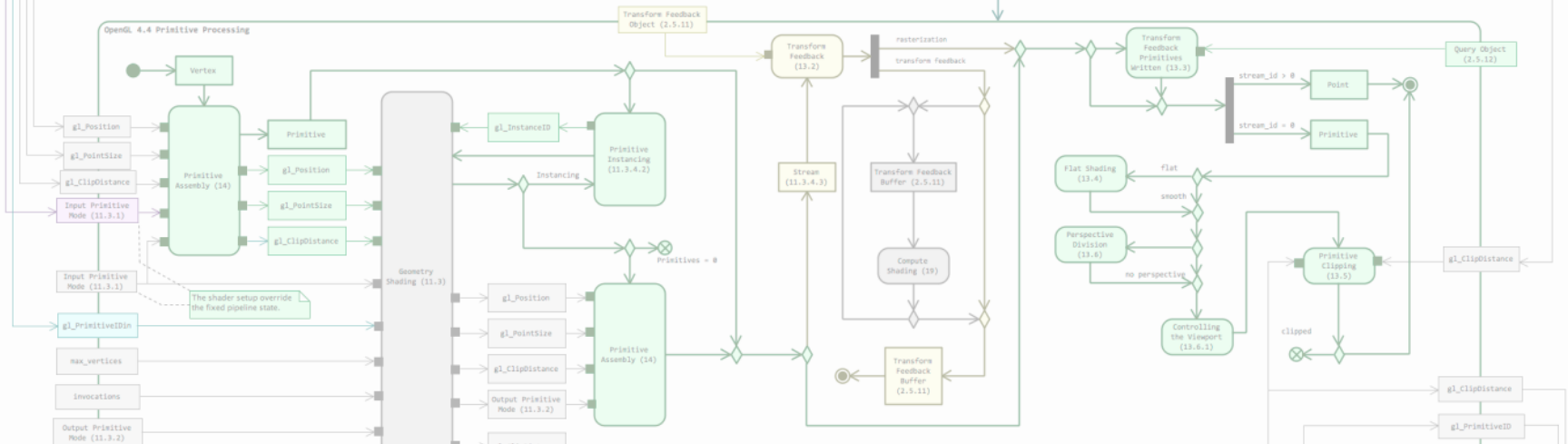
- OpenGL works as a ***state machine***
- There are roughly two types of APIs:

APIs to Specify Objects	APIs to Configure the Pipeline
• Vertex or Index Buffers • 1D, 2D, 3D Textures • Shader Programs • Framebuffer Objects	• State of Fixed Function Stages • Specify the Shader Program begin Used • Specify the Render Target being Used





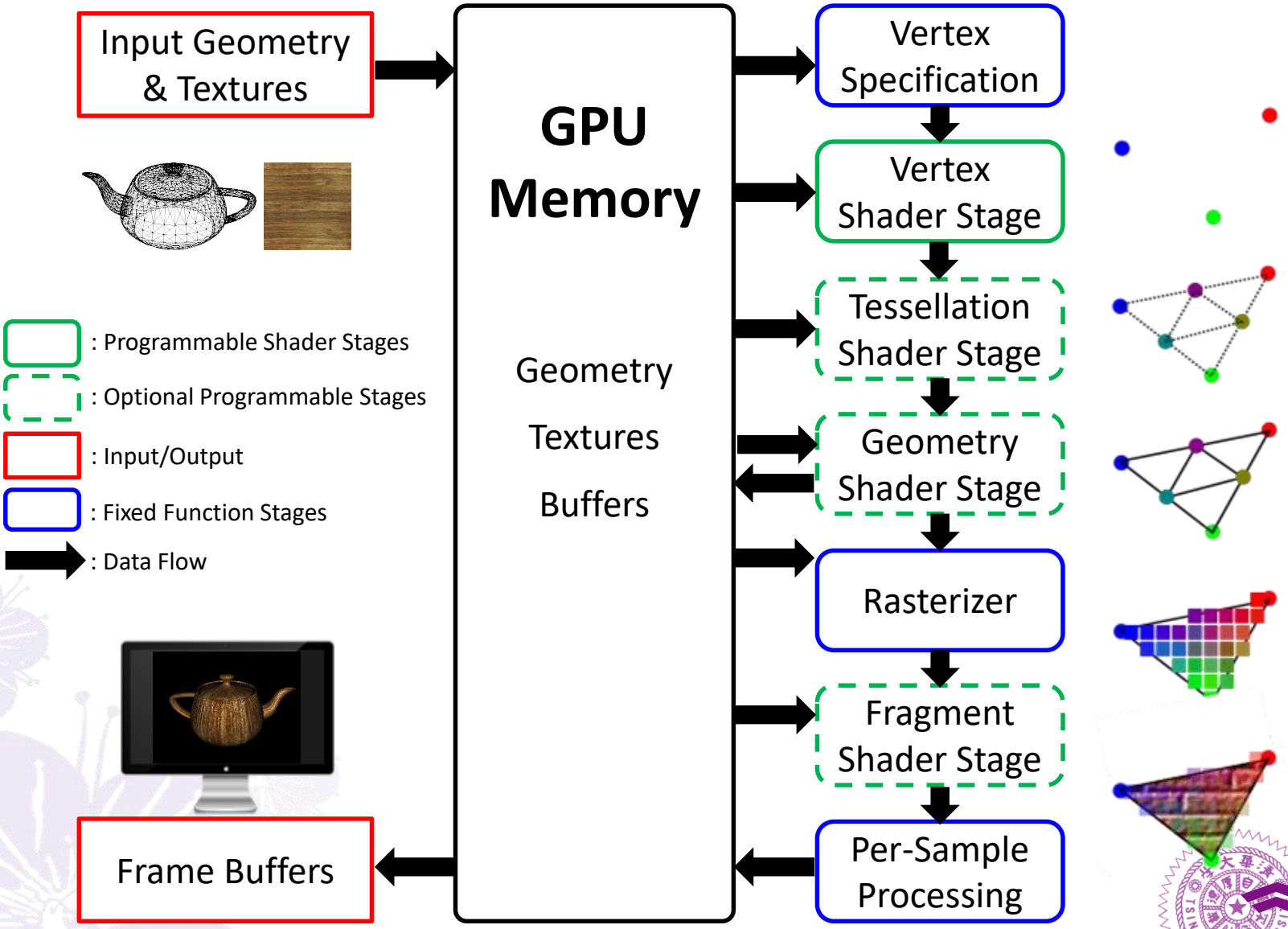
OpenGL Rendering Pipeline



Rendering Pipeline

- Pipeline: consists of multiple stages. Data flows in, being processed in each stages, then flows out
- Stages: each stage represents an unique function to process the input data
  - **Fixed function** stages: limited customization capability, typically exposes states for configuration
  - **Programmable shader** stages: allow custom shader programs to be executed within, providing broader capability of customization

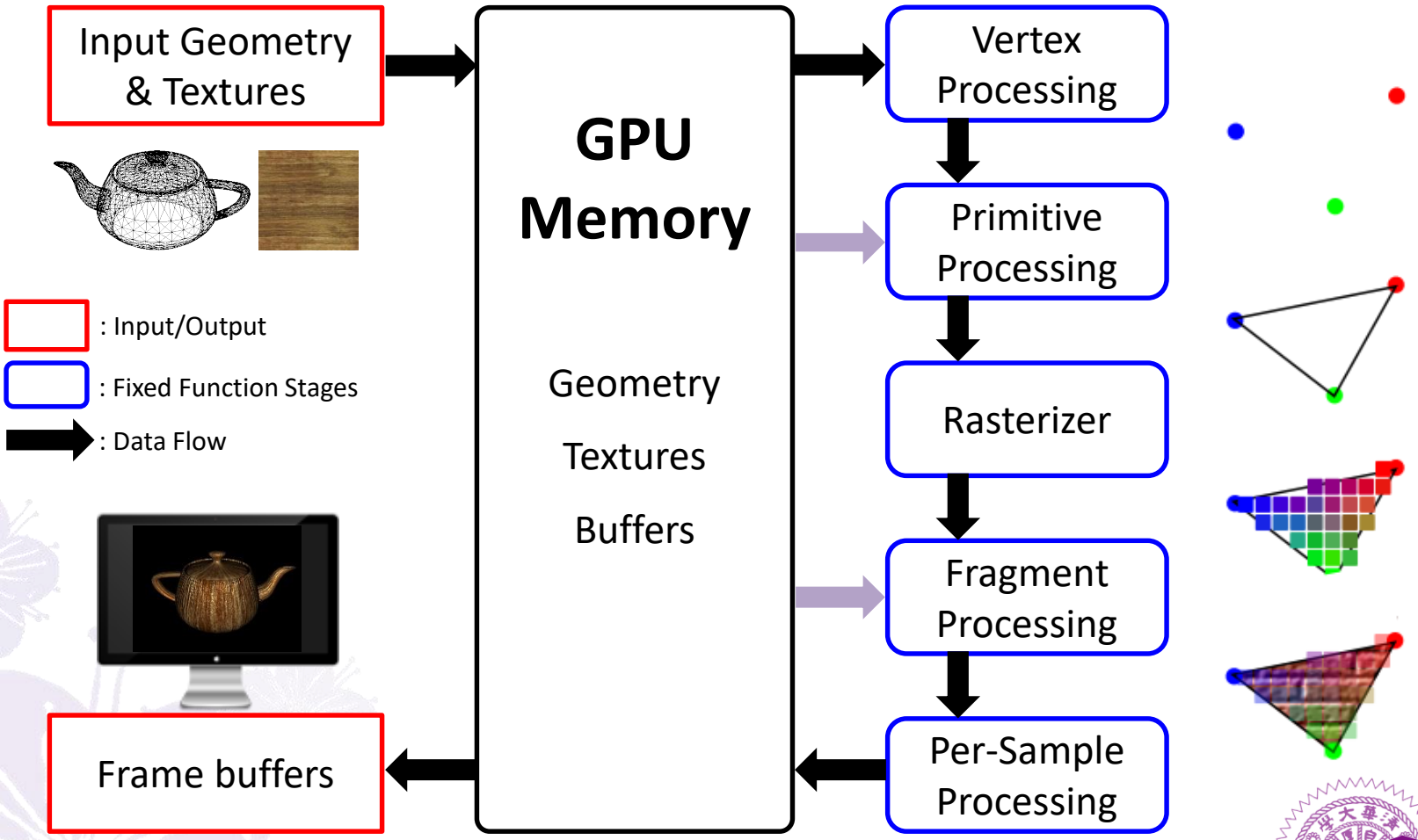
The OpenGL Rendering Pipeline



Fixed Function Pipeline (Legacy)

- Before OpenGL 3.0, OpenGL rendering is done in a fixed function pipeline
- Fixed pipeline is like an machine with a lot of switches/values to configure
- One cannot change how the function is implemented as well as the order of execution

Fixed Function Pipeline (Legacy)



Fixed Function Pipeline: Metaphor

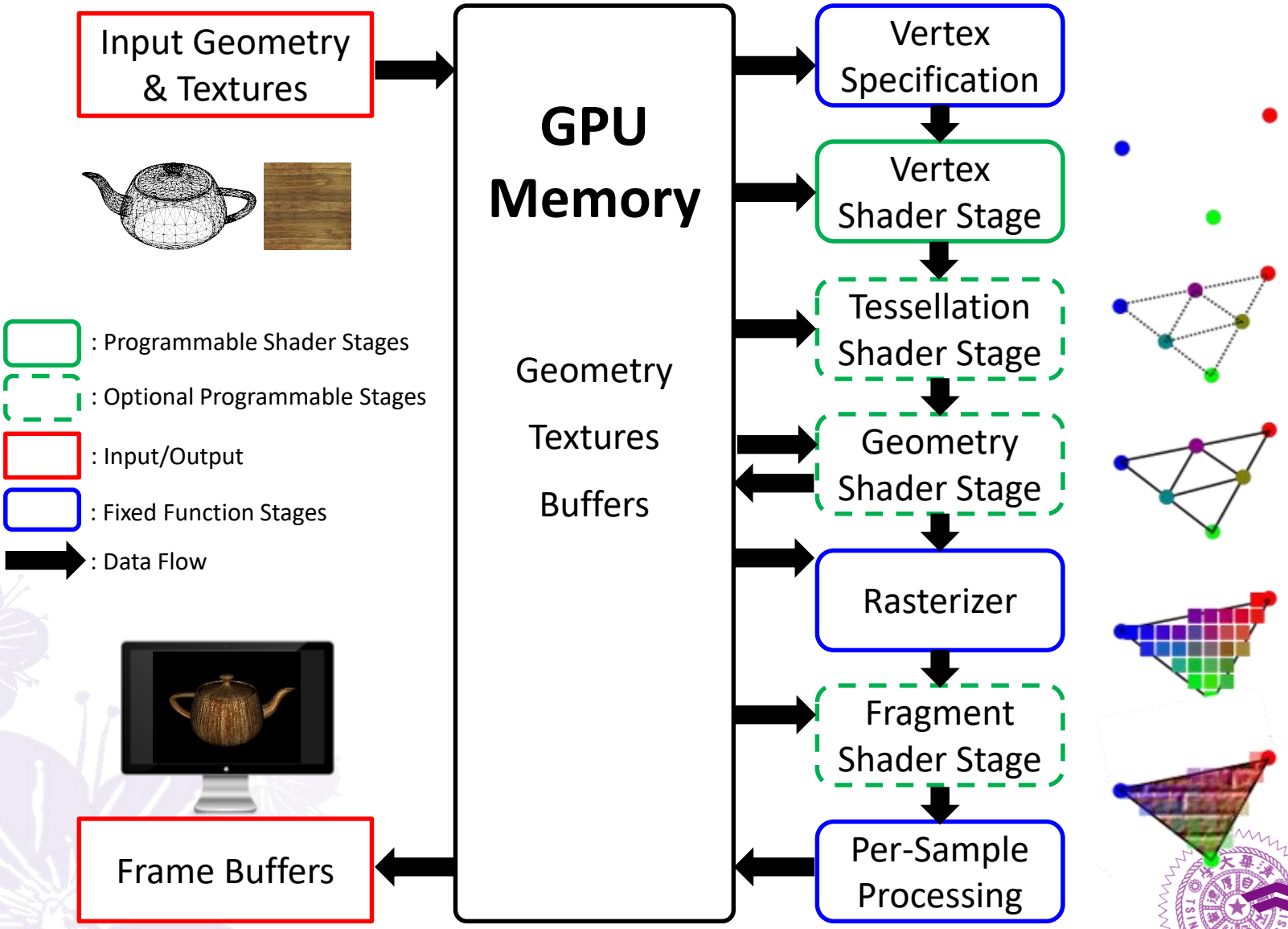
OpenGL Fixed Function Pipeline

How do I press these buttons to get desired effect?

Programmable Pipeline

- Shader programs are introduced in OpenGL 2.0, and included in the core profile in OpenGL 3.0
- Fixed function pipeline is deprecated since OpenGL 3.0
- Shader programs, written in OpenGL Shading Language(GLSL), allow the programmers to customize certain stages in the OpenGL rendering pipeline

The OpenGL Rendering Pipeline



Programmable Pipeline: Metaphor

Shader Programs

```
uniform float t;     // Time (passed in from the application)
attribute vec4 vel; // particle velocity

const vec4 g = vec4( 0.0, -9.80, 0.0 );

void main()
{
    vec4 position = gl_Vertex;
    position += t*vel + t*t*g;

    gl_Position = gl_ModelViewProjectionMatrix * position;
}
```

Let's write a program to create the effect!

Programmable Pipeline V.S. Fixed Function Pipeline

	Programmable Pipeline	Fixed Function Pipeline
Flexibility	+implement various algorithms in shader programs	-limited customization capability
For Simple Application	-must configure the whole pipeline	+performs simple task with less configurations
For Complex Application	+can achieve various effects	-most advanced effects are impossible
Learning Curve	-one must have full knowledge of the pipeline and GLSL before writing an application	+can create simple applications without much knowledge
Deploy	-should consider all graphics driver environment	+works in most graphics driver environment


OpenGL®
SDK

OpenGL Library (1.1)

- OpenGL core library
  - OpenGL32 on Windows
  - GL on most Unix/Linux systems (libGL.a)
- OpenGL Utility Library (GLU)
  - Providing functionality in OpenGL core but avoiding having to rewrite code
  - NURBS, tessellators, quadric shapes, etc
- Links with windows system
  - GLX for X windows system
  - WGL for Windows
  - AGL for Macintosh

GLUT (OpenGL Utility Toolkit)

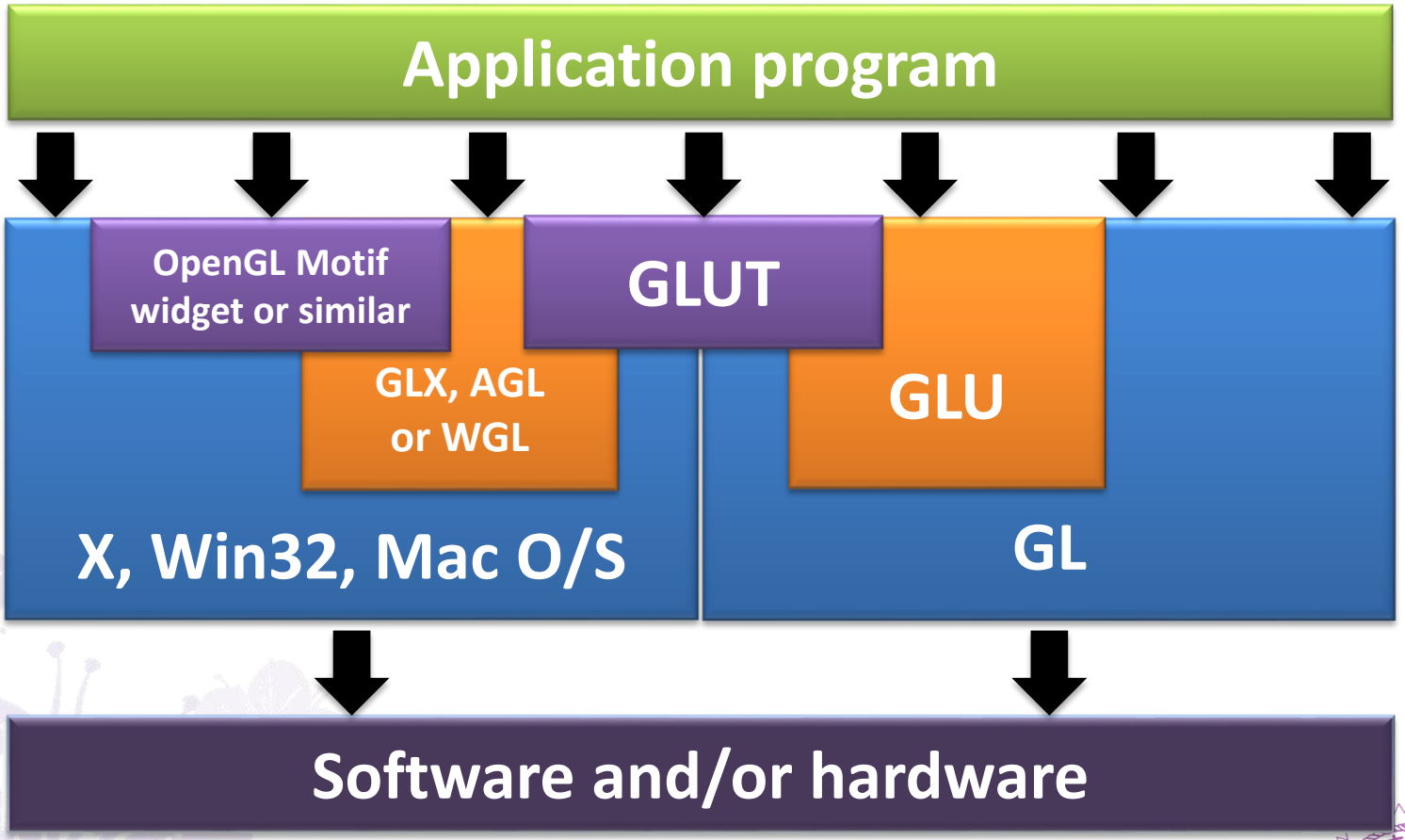
- Configuring a windows program
- Registering input callback functions
  - Rendering
  - Resizing
  - Input: keyboard, mouse, etc
- Entering event processing loop
- freeglut
  - http://freeglut.sourceforge.net/

GLUT Callback Functions

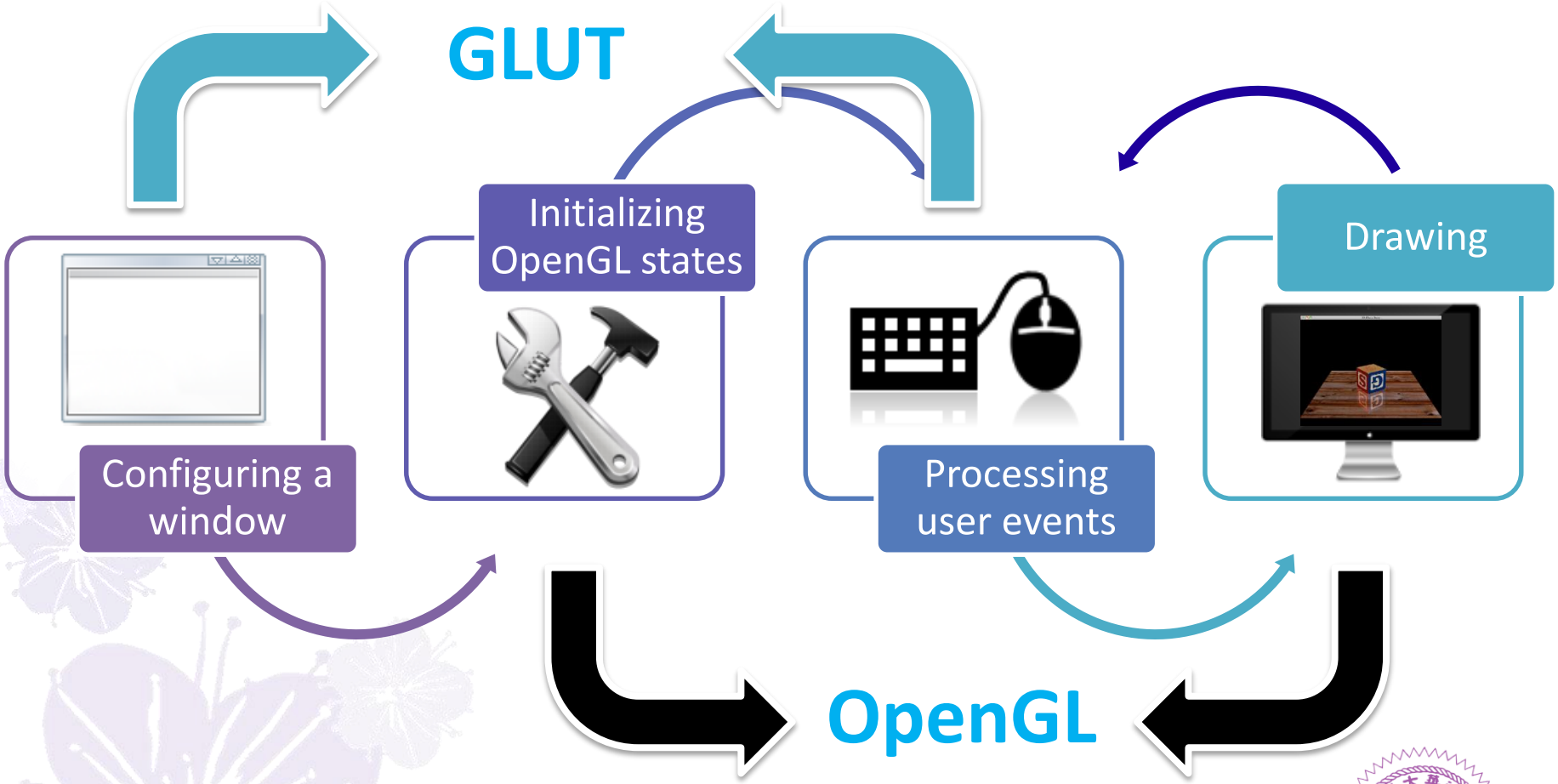
- Routines to call when something happen
  - Window resize or redraw
  - User inputs
  - Animation
- "Register" callbacks with GLUT

```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutKeyboardFunc( keyboard );
```

Software Architecture



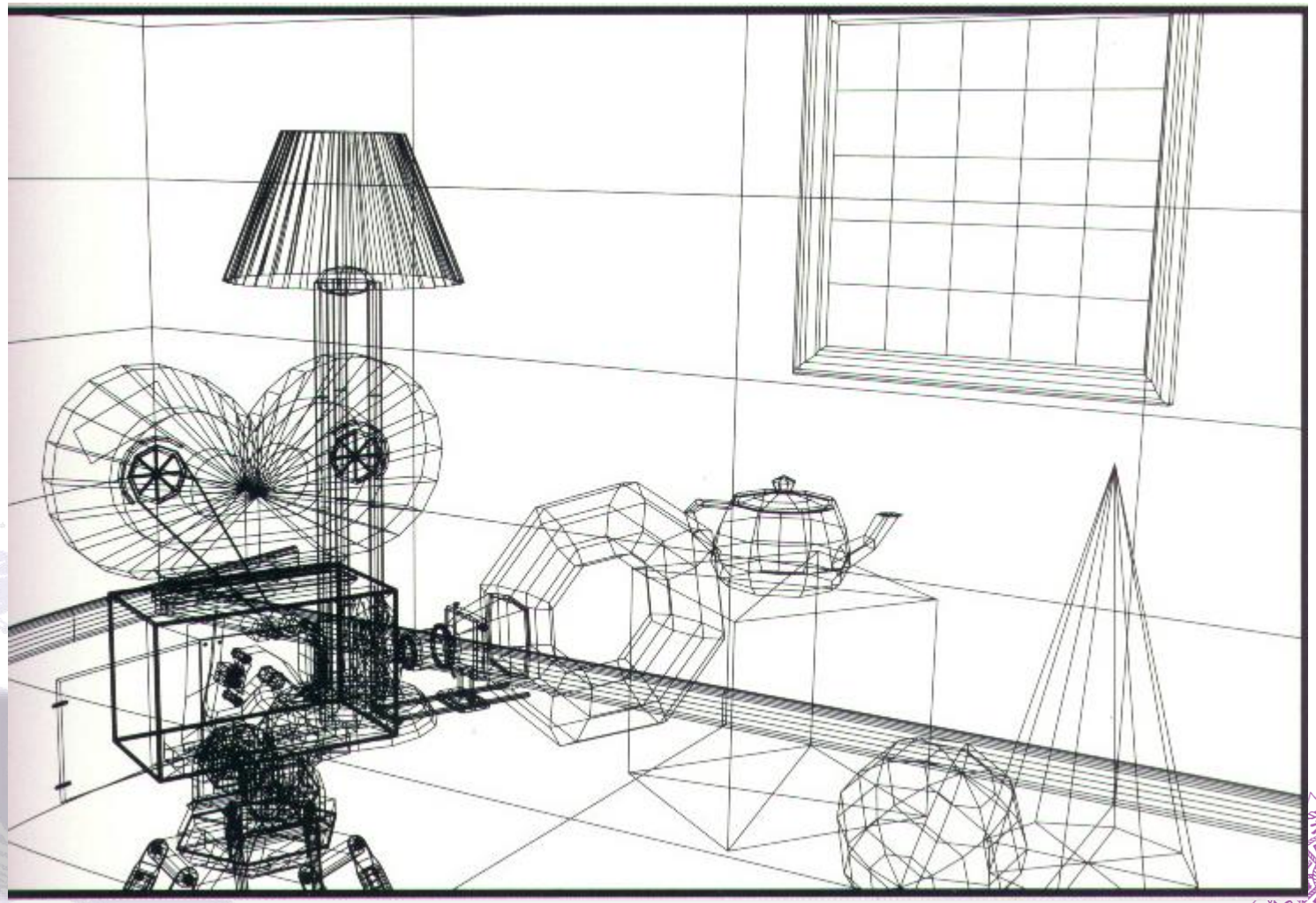
General Structure of an OpenGL Program



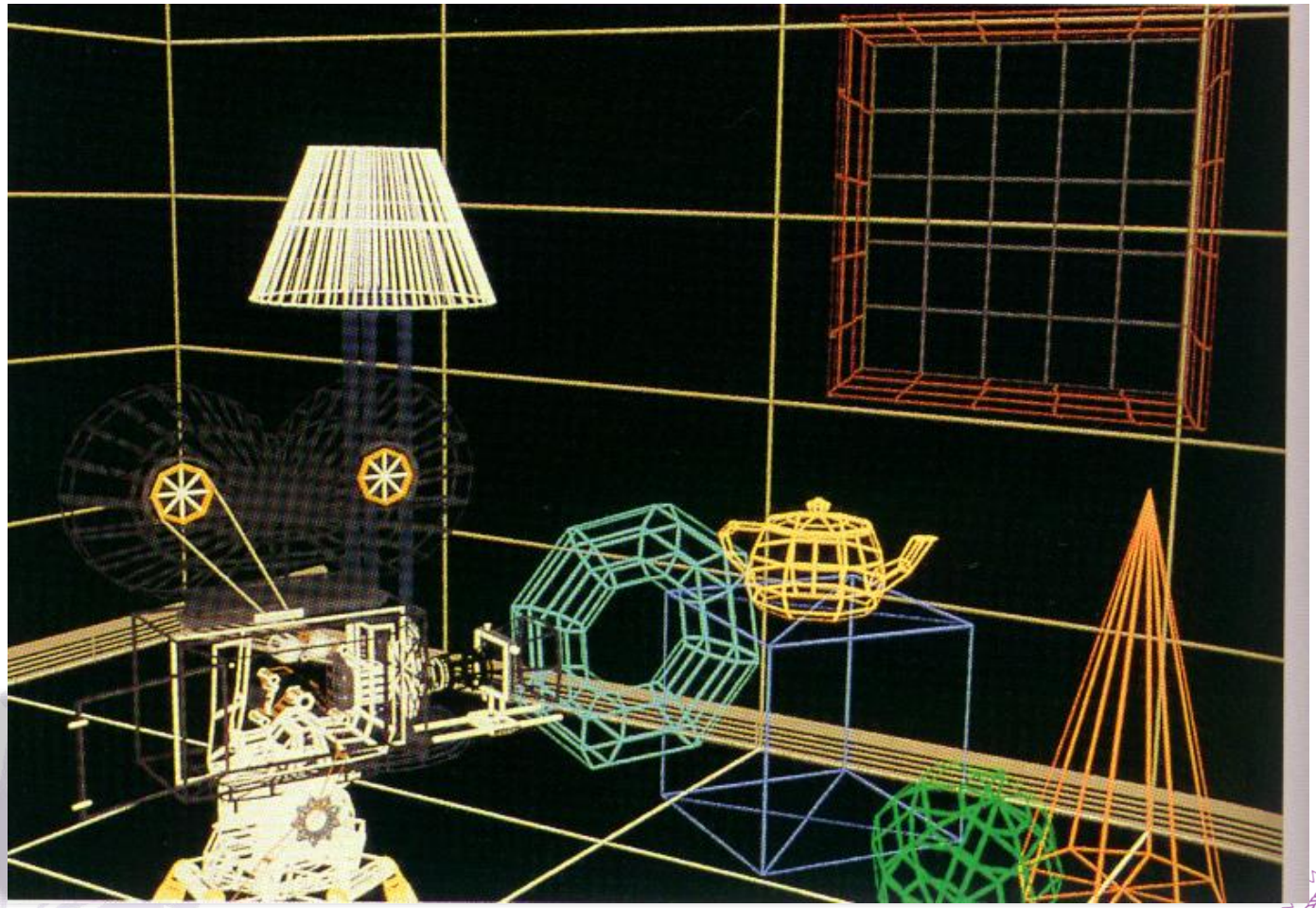
Steps toward reality

A STEP-WISE ILLUSTRATION

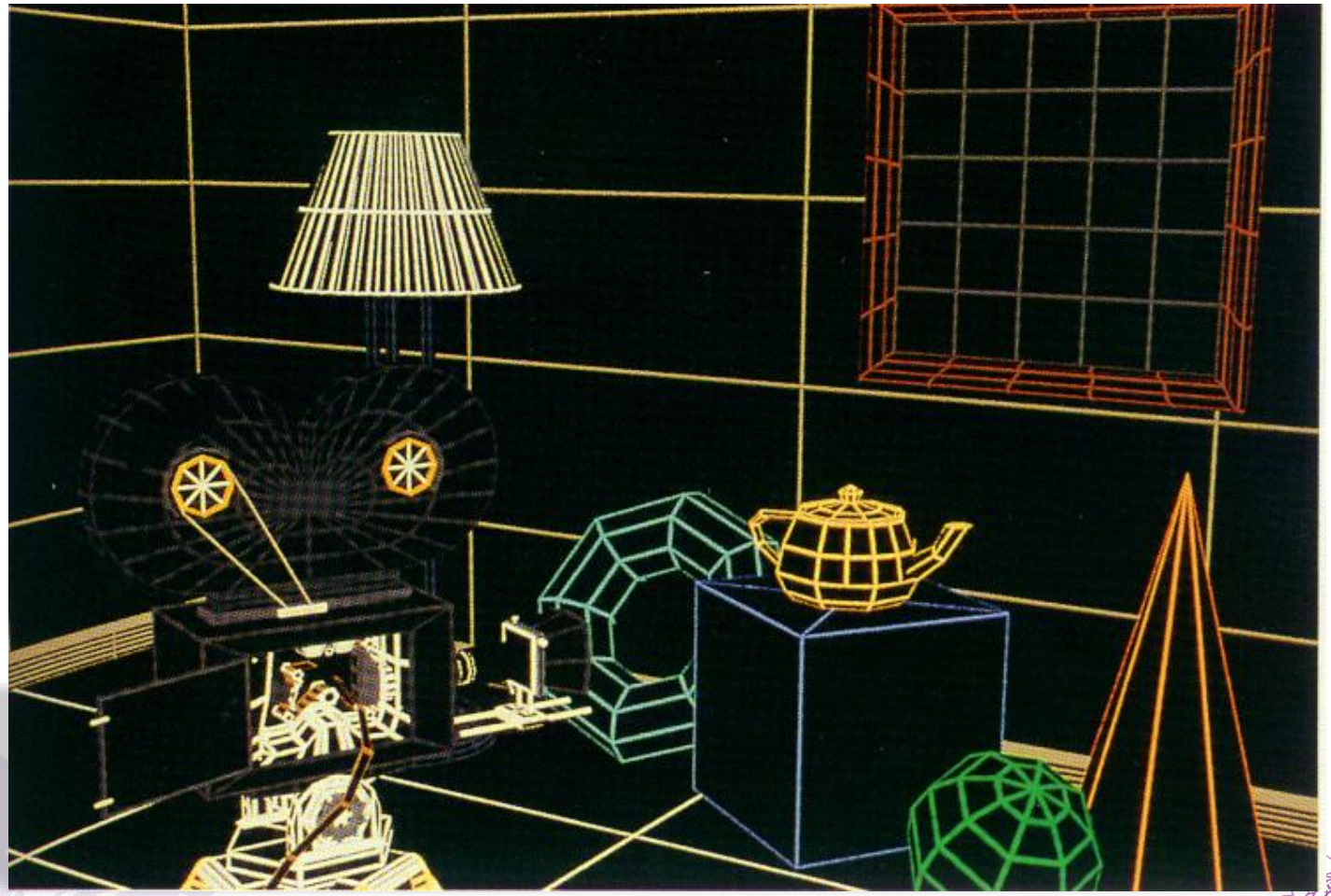
Primitive Wireframe Rendering



Colored Wireframe



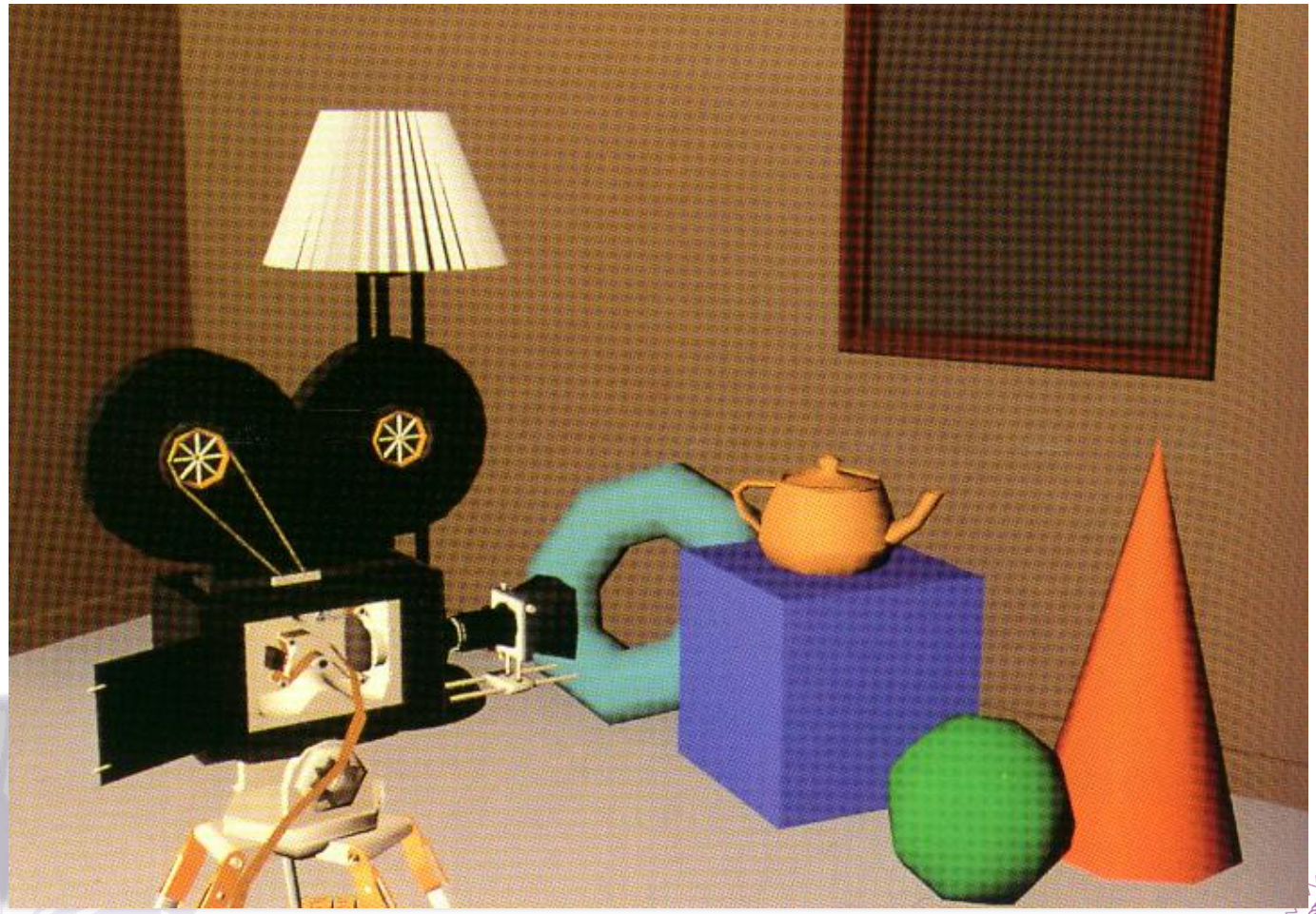
Solid Rendering with Wireframe



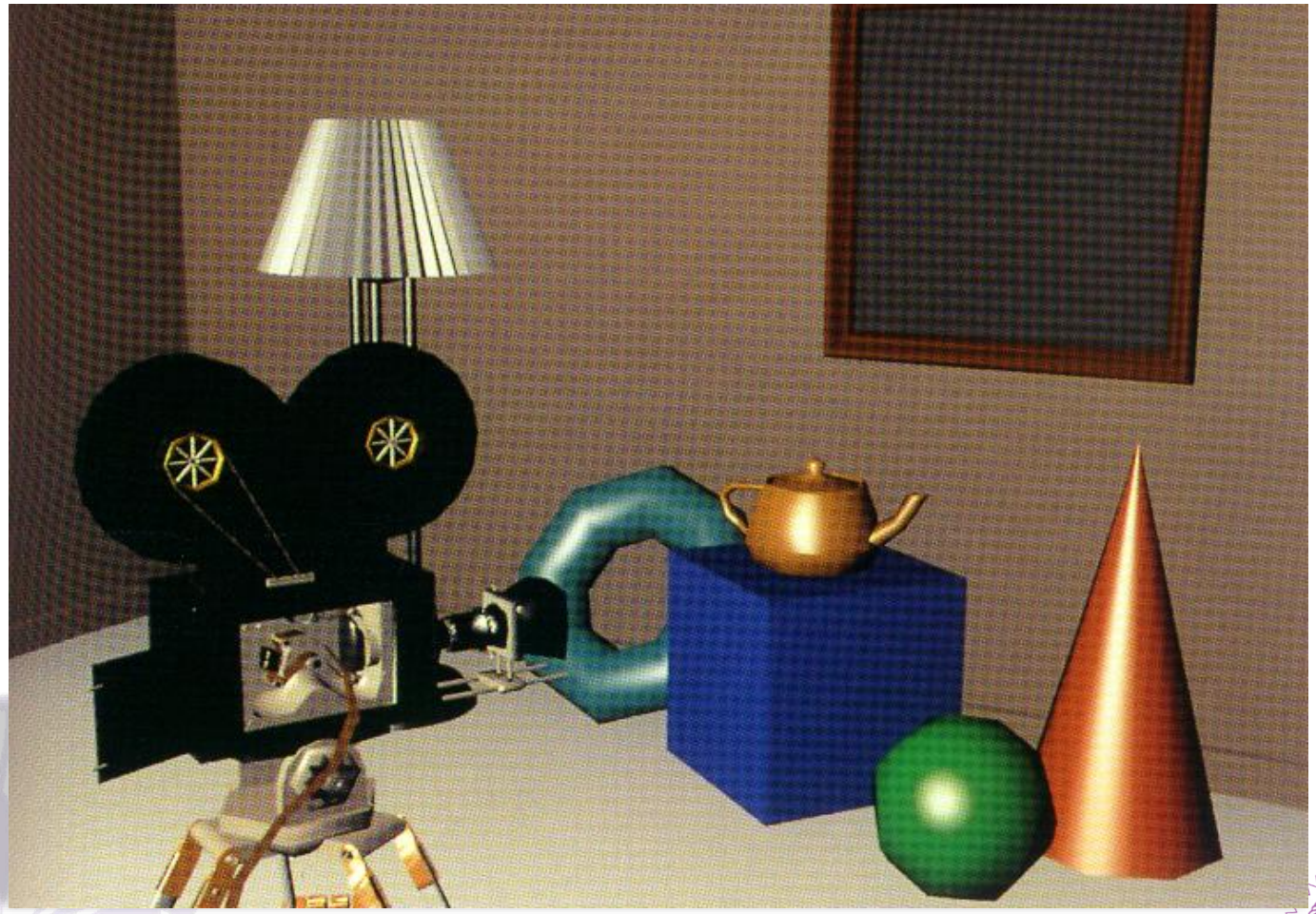
Constant Shading

Flat Shading

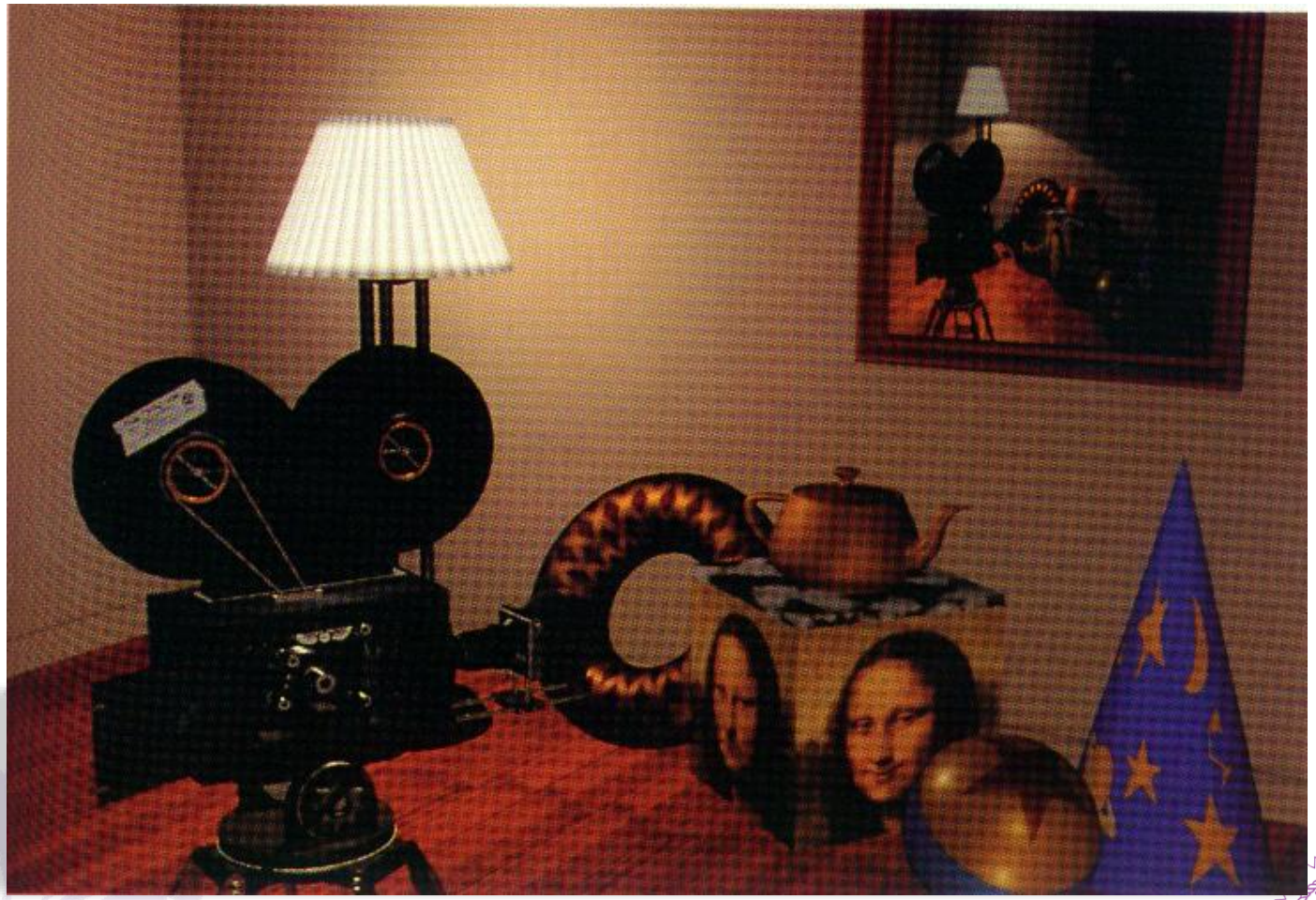
Smooth Shading



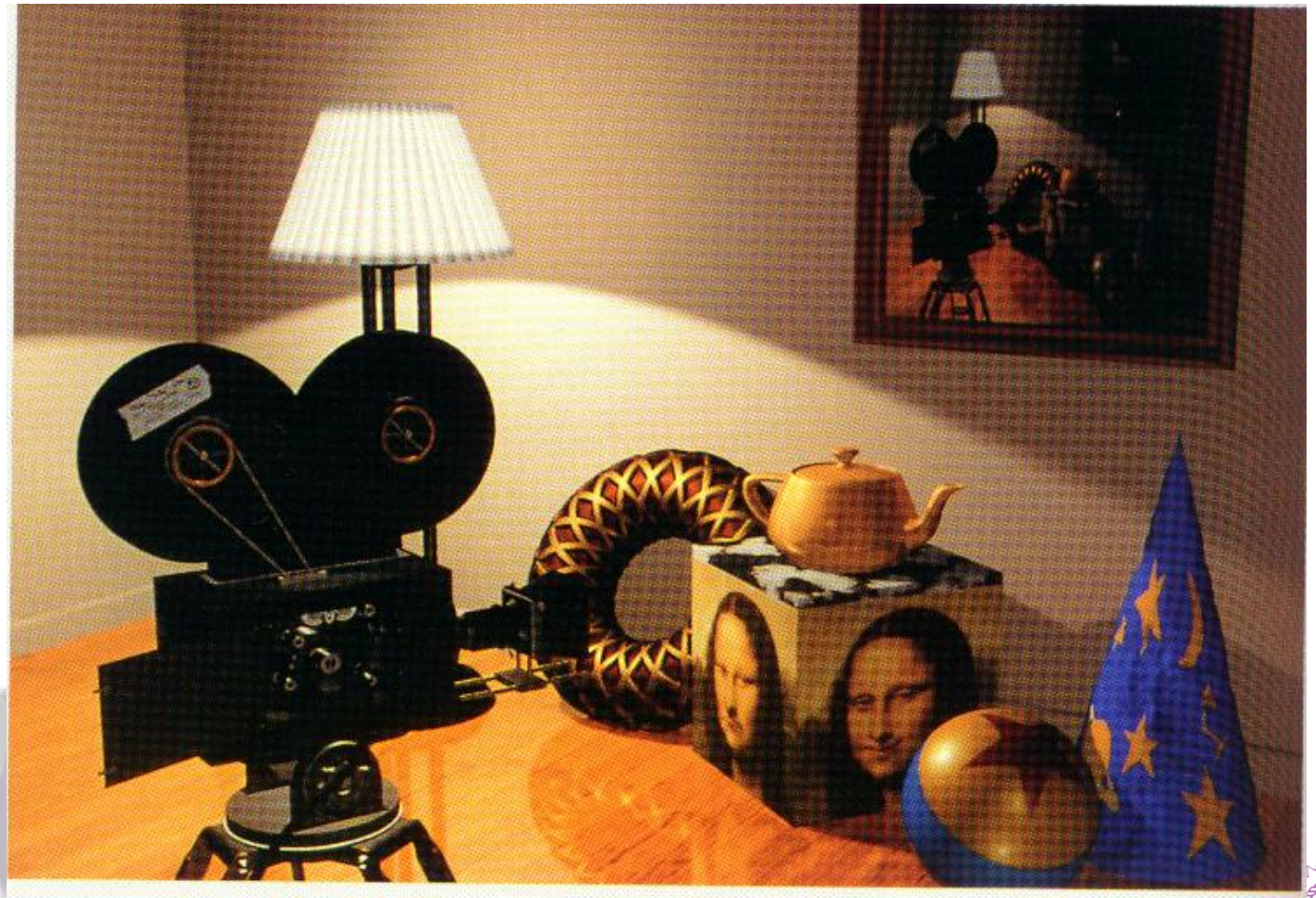
Materials



Textures



Complex Lighting Effects

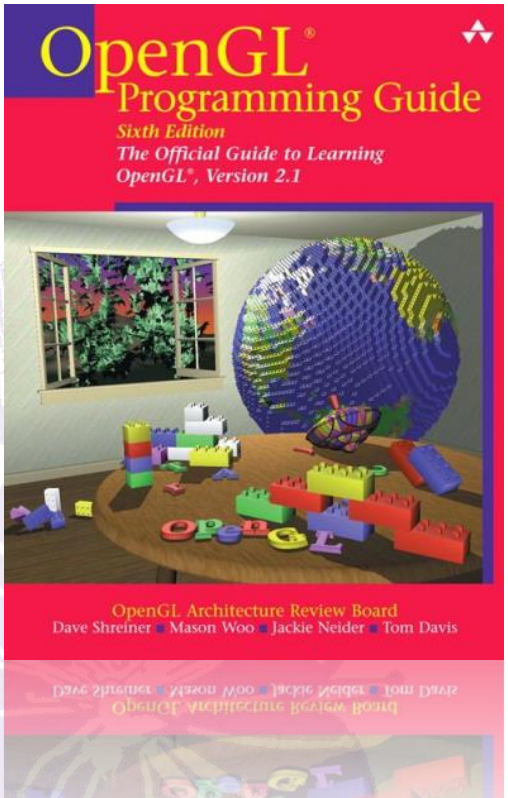


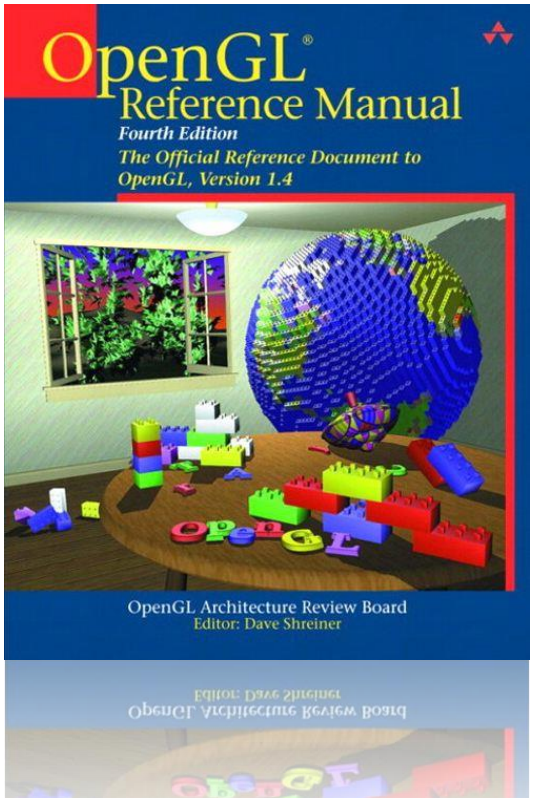
Resources

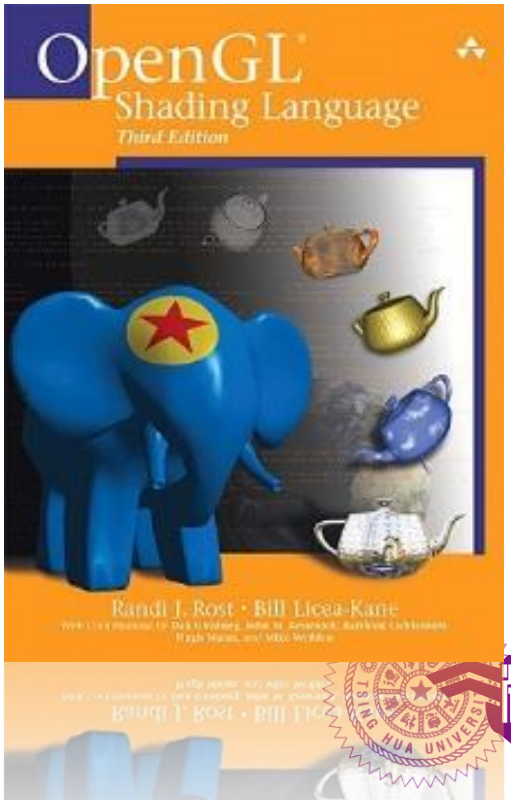
- OpenGL
  - http://www.opengl.org
  - http://openglinsights.com/
  - https://www.khronos.org/
  - https://www.shadertoy.com
- glut
  - http://www.opengl.org/resources/libraries/glut/
- freeglut
  - http://freeglut.sourceforge.net/

Reference Books

- The OpenGL Programming Guide
  - http://www.opengl.org/documentation/red_book/








thank you!
Question?