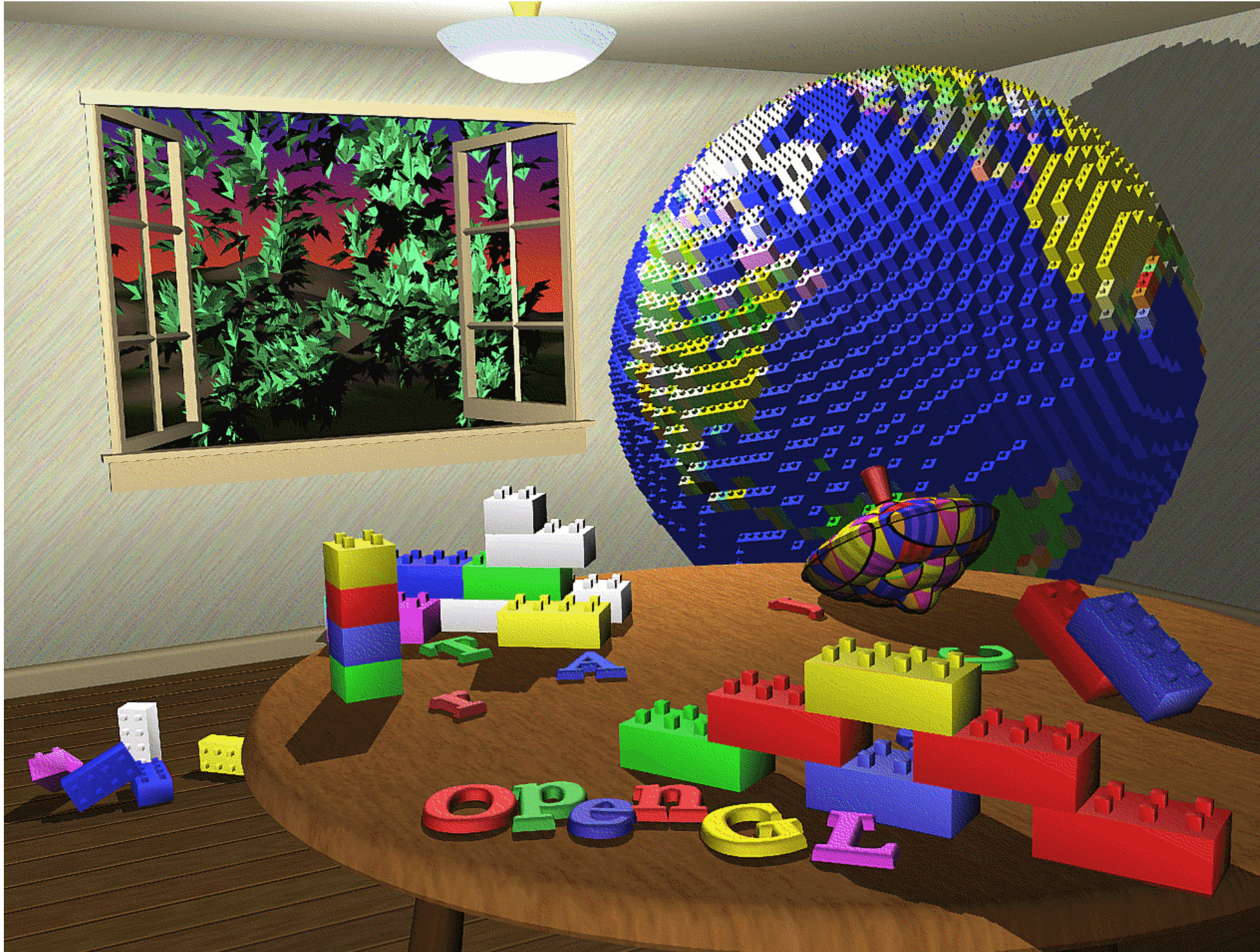


OpenGL and GLSL

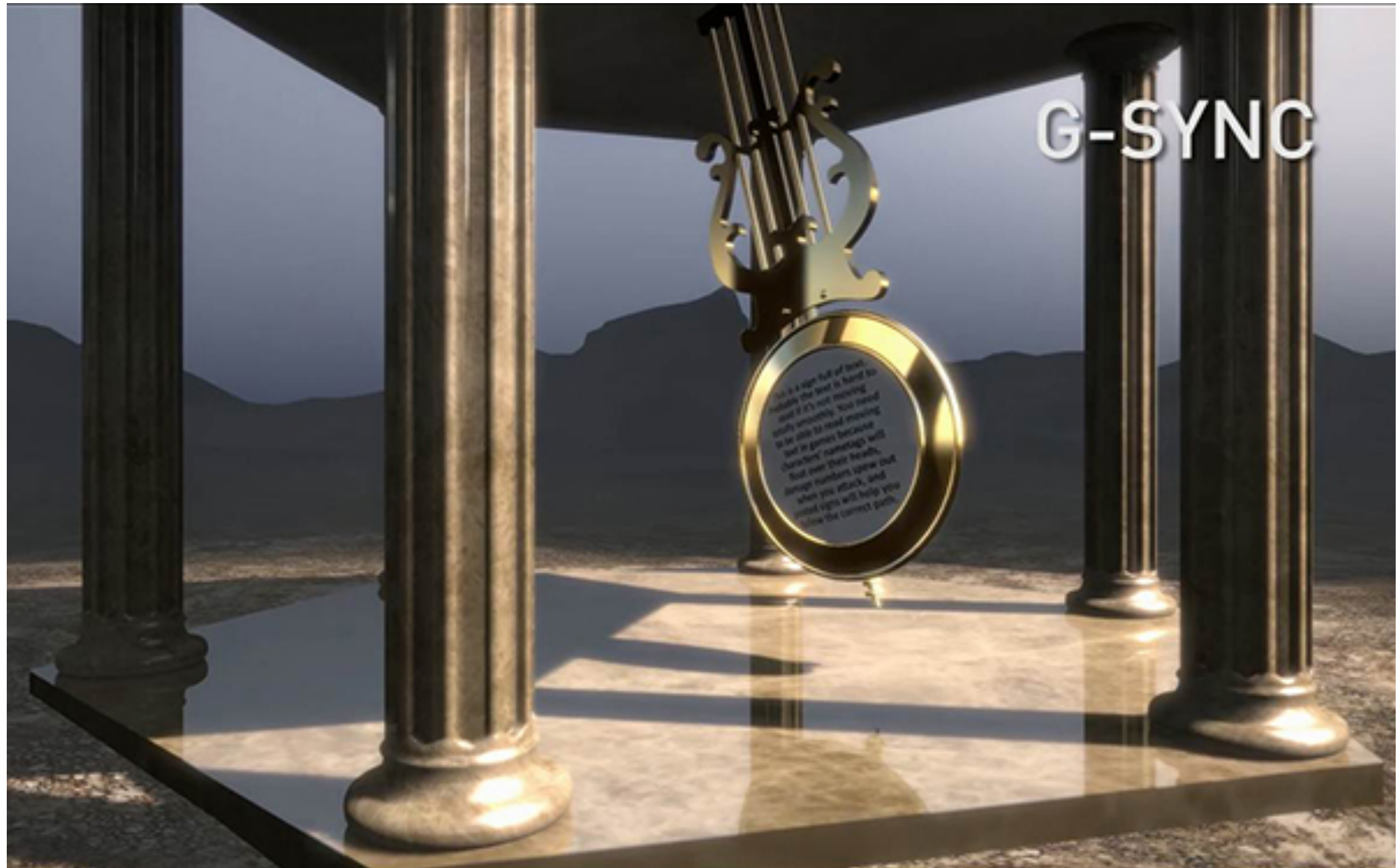
CS 4620 Lecture 14

OpenGL 25 years ago



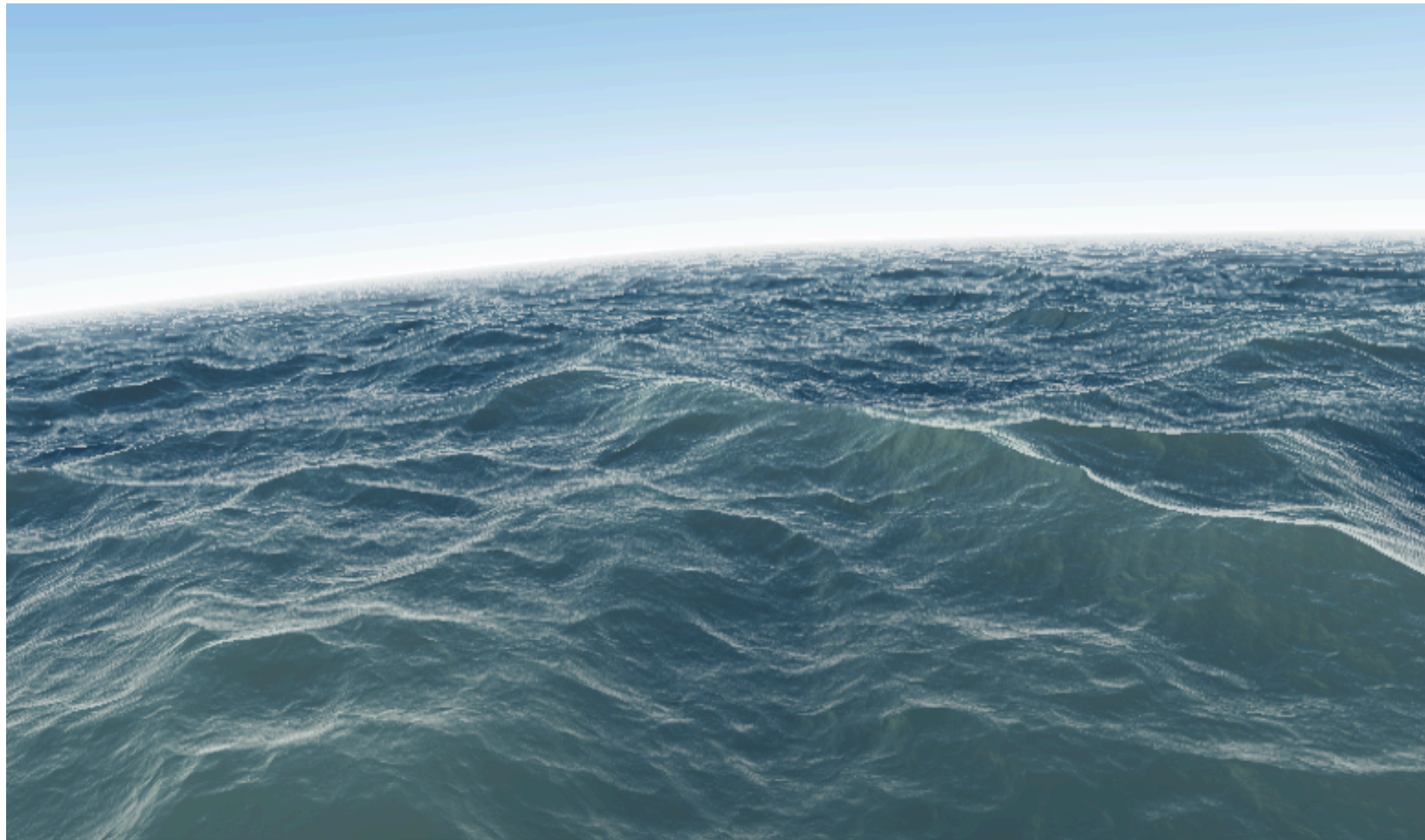
Modern OpenGL

(OK, this is rendered with DirectX, but you get the idea)



NVIDIA demo <https://www.youtube.com/watch?v=s6T9jlwQBSM>

Modern WebGL



"Seascape" by Alexander Alekseev

Sources for more examples:

<http://glslsandbox.com>

<http://shadertoy.com>

What changed?

25 years ago:

- Vertex transformation/fragment shading hardcoded into GPUs

Now:

- More parts of the GPU are programmable (but not all)

What changed?

25 years ago (Fixed pipeline):

- Transform vertices with modelview/projection matrices
- Shade with Phong lighting model only

Contemporary (Programmable hardware):

- Custom vertex transformation
- Custom lighting model
- More complicated visual effects
- Shadows
- Displaced and detailed surfaces
- Simple reflections and refractions

GLSL

GLSL : **G**raphics **L**ibrary **S**hading **L**anguage

- Syntax similar to C/C++
- Language used to write shaders
 - vertex, tessellation, geometry, fragment, compute
 - We only cover vertex and fragment shaders today
- Based on OpenGL
 - First available in OpenGL 2.0 (2004)
- Alternatives: Nvidia Cg and Microsoft HLSL

What is a Shader Program?

- A small program to control parts of the graphics pipeline
- Consists of 2 (or more) separate parts:
 - Vertex shader controls vertex transformation
 - Fragment shader controls fragment shading

GLSL Program

- Specifies how OpenGL should draw geometry
- Program: A collection of shaders that run together
 - At least one vertex shader or one fragment shader
- At any time, the GPU runs only one program
 - Must specify program to use before drawing geometry

Pipeline overview

you are here →

APPLICATION

COMMAND STREAM

3D transformations; shading →

VERTEX PROCESSING

TRANSFORMED GEOMETRY

conversion of primitives to pixels →

RASTERIZATION

FRAGMENTS

blending, compositing, shading →

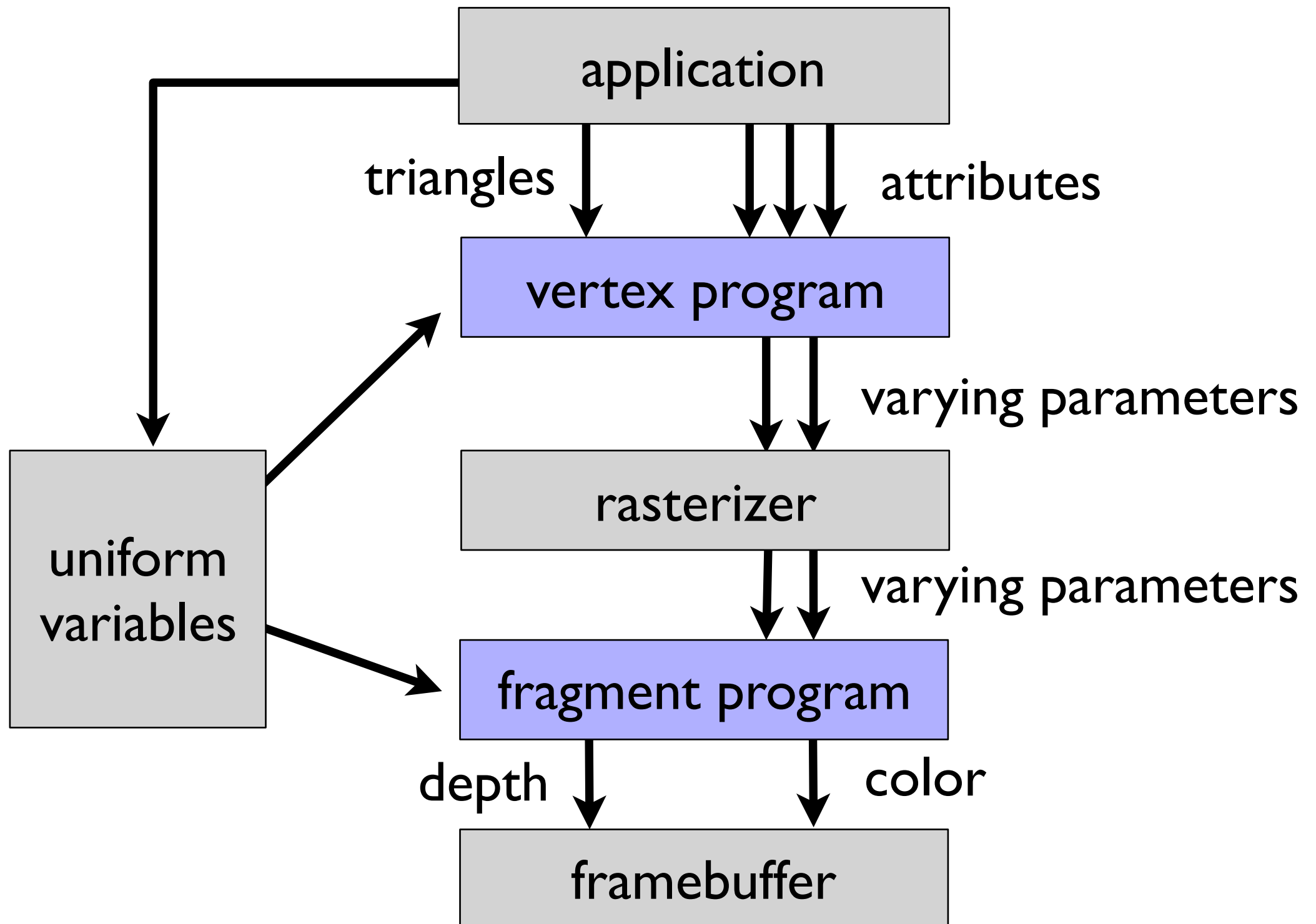
FRAGMENT PROCESSING

FRAMEBUFFER IMAGE

user sees this →

DISPLAY

GLSL Shaders



Varieties of OpenGL and GLSL

- **OpenGL versions 1, ..., 4.5**
 - for desktop/server GPUs
 - latest features, best performance
- **OpenGL ES 1, ..., 3.2**
 - for mobile
 - older, more stable set of features
- **WebGL 1, 2**
 - bindings for OpenGL ES in browser-based JavaScript
 - 1.0 widely supported (OpenGL ES 2); 2.0 (ES 3) in latest browsers
- **GLSL versions 1, ..., 4.5**
 - numbers don't always correspond to OpenGL version (later they do)

In this class

- **We will use WebGL 1**
 - This means OpenGL ES 2 and GLSL 1.2
 - Supported in all modern browsers
 - We officially recommend Chrome
- **We will use three.js on the Browser side**
 - a popular library providing matrix math, scene graph, convenience functions, etc.
 - You won't have to write code that uses three.js but it will help to understand how it works

Good reference materials

- the classic book
 - its website
 - Cornell library eBook
- lighthouse3d.com tutorials
 - GLSL 1.2 tutorial
- webglfundamentals.org tutorials
- Mozilla WebGL API doc
- Official material from Khronos standards organization
 - OpenGL ES 2 and GLSL ES 1.0 Specifications
 - OpenGL ES 2 / GLSL ES 1.0 Reference Card
 - OpenGL wiki