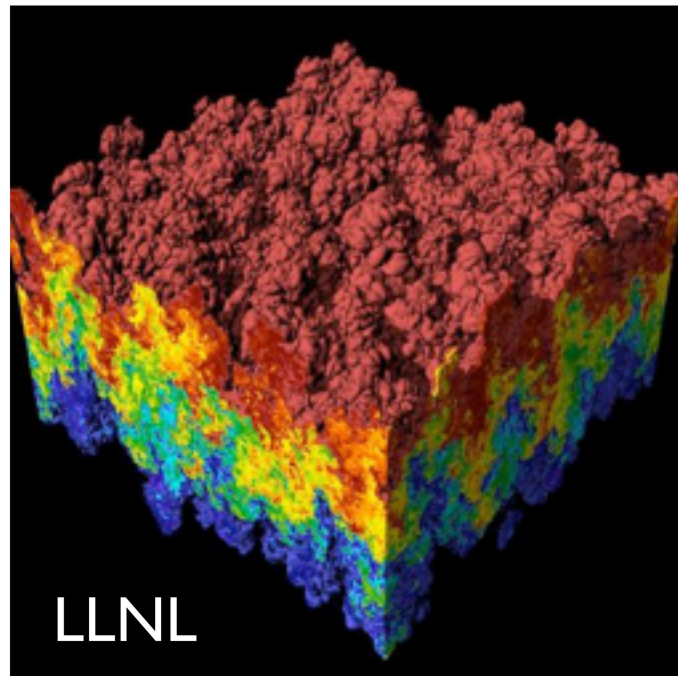
CS130 : Computer Graphics

Fall 2017

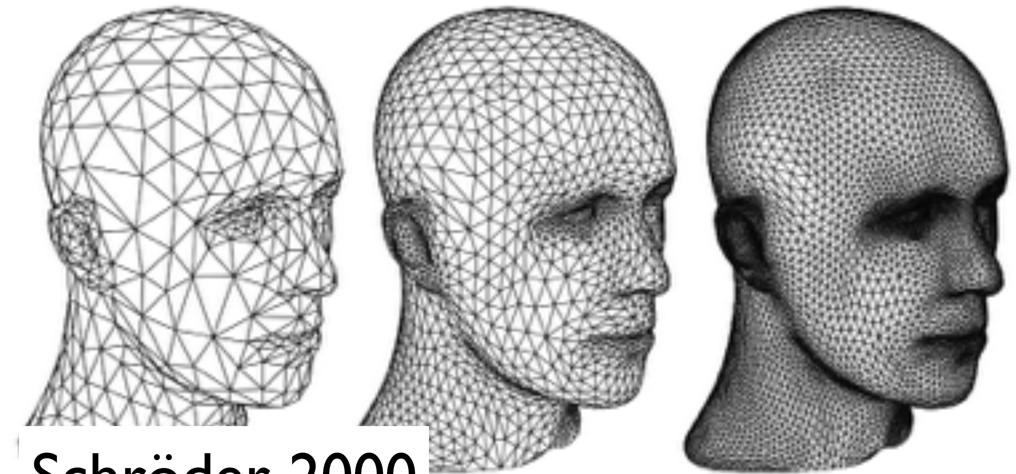
Tamar Shinar
Computer Science & Engineering
UC Riverside

Welcome to CS130!

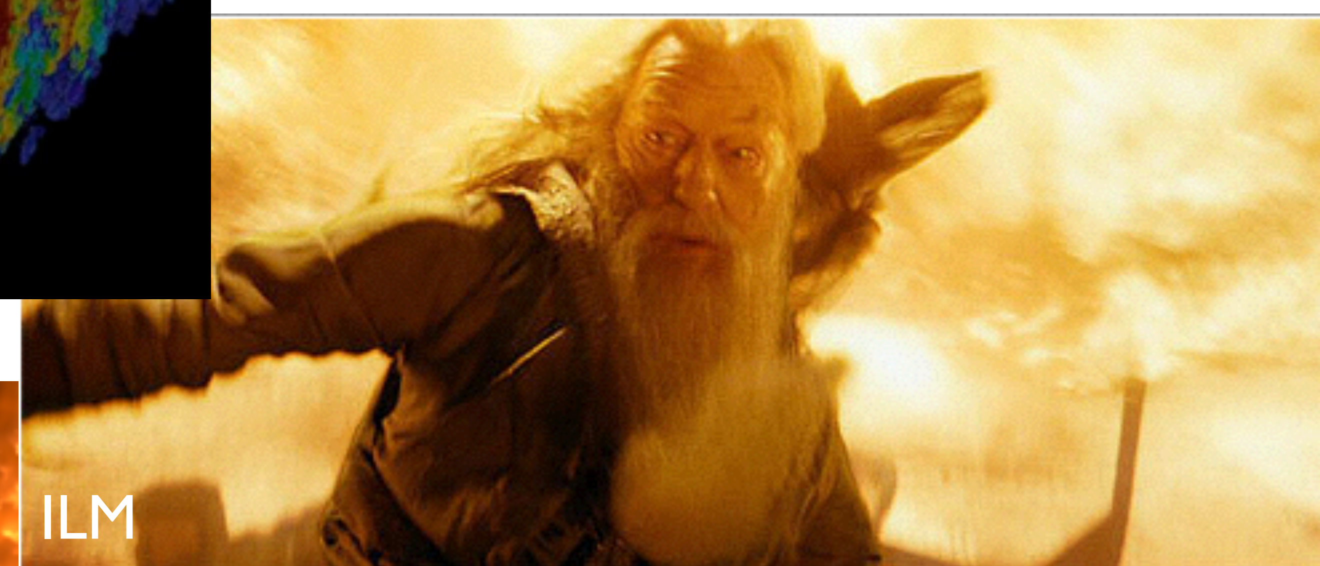
Talton et al., 2011



LLNL



Schröder, 2000



ILM

Henrik Wann Jensen

Hong et al. 2007

Pixar

Today's agenda

- Course logistics
- Introduction: graphics areas and applications
- Course schedule
- Introduction to OpenGL

Course Overview

- Learn fundamental 3D graphics concepts
- Implement graphics algorithms
  - make the concepts concrete
  - expand your abilities and confidence for future work

Course Logistics

- Professor: Tamar Shinar
- TAs: Muzaffer Akbay (Muzo), Cassio Elias
- Website: http://www.cs.ucr.edu/~shinar/courses/cs130
- Lectures: MWF 9:10am-10:00am, CHASS 1002
- Lab: M 1:10-4:00pm, or 4:10-7:00pm, WCH 129
- Announcements (assignments, etc.) made in class and through ilearn
- Questions and discussions: Piazza

Course Logistics

- Grading
  - 10% labs
  - 10% homework
  - 30% assignments (2 assignments, 15% each)
  - 50% tests (2 midterms, 15% each, 1 final, 20%)
- Detailed schedule on class website

Course schedule

advance schedule is subject to change

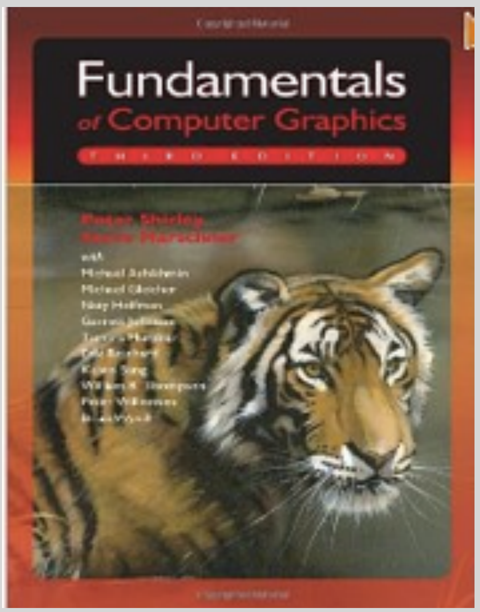
- Home
- Schedule
- Assignments
- Labs
- Homework
- Exams

Schedule

Advance schedule is provisional and subject to change.

Class	Date	Topic	Reading	Assigned	Due
1	9/29	Introduction, OpenGL	Sections 1.1, 1.2	Homework 1	
2	10/2	Math review	Sections 2.1.0-2.1.2, 2.3.0-2.3.2, 2.4 slides		
Lab 1	10/2	Introduction to OpenGL			
3	10/4	Images, graphics pipeline	Sections 3.0, 3.1.1, 3.2.0, 3.2.1, 3.3, 3.4 Section 8.0		
4	10/6	Line rasterization	Sections 8.1, 8.1.1 and Subsection "Implicit 2D Lines" of Section 2.5		Homework 1
5	10/9	Triangles and triangle rasterization	Section 2.7, Section 8.1.2 Sections 8.1.3, 8.1.6, 8.2.0-8.2.3 (except "Precision Issues")		
Lab 2	10/9	Line rasterization			
6	10/11	Matrix transformations			
7	10/13	Viewing transformations			
8	10/16	Perspective transformation			
Lab 3	10/16	OpenGL matrix stack			
9	10/18	Lighting and shading			
10	10/20	Shading (cont.)			
11	10/23	Shading (cont.)			
Lab 4	10/23	OpenGL programmable shading			
12	10/25	Review			
13	10/27	Midterm 1			
14	10/30	Texture mapping			
Lab 5	10/30	Texture mapping			
15	11/1	Texture mapping (cont.)			

Textbook

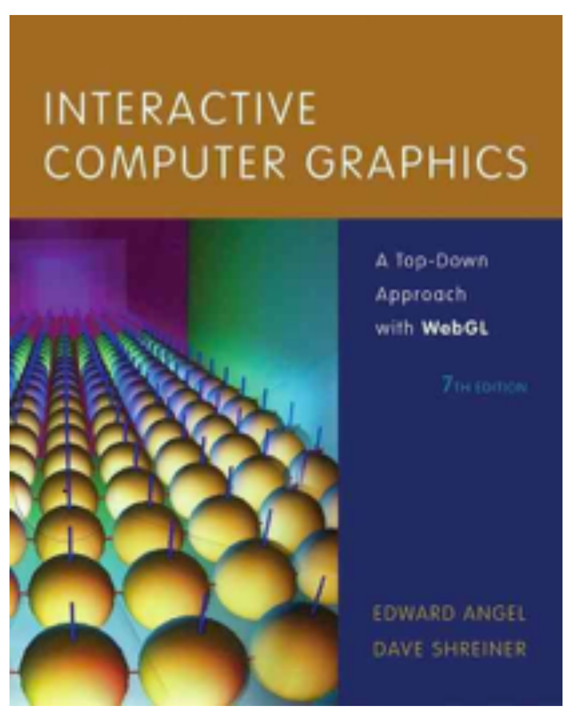


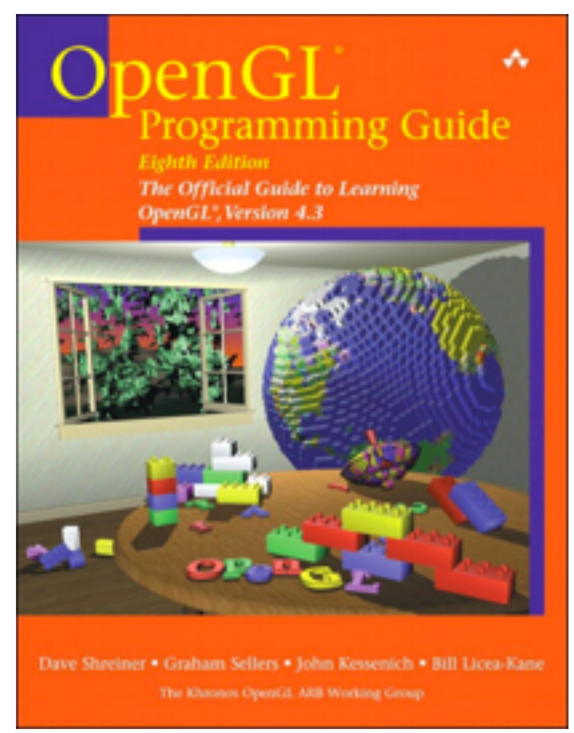
Fundamentals of Computer Graphics

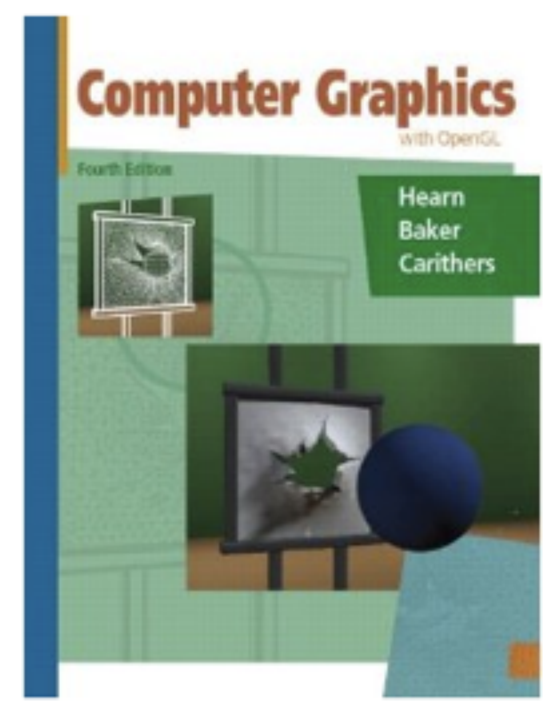
Shirley and Marschner

(3rd or 4th edition)

Additional books







About the professor

- B.S., University of Illinois in Urbana-Champaign, Mathematics, Computer Science, Fine Art
- Ph.D., 2008, Stanford University on simulation methods for computer graphics
- NYU postdoc on computational biology
- Joined UCR CS&E department in the Fall 2011
- Work in graphics simulation and biological simulation

http://www.cs.ucr.edu/~shinar

About the TAs

- Cassio Elias
- Muzaffer (Muzo) Akbay

Introduction

Graphics applications

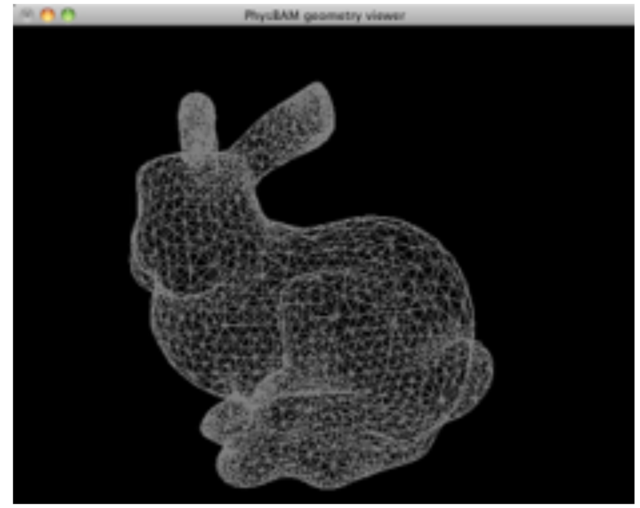
- 2D drawing
- Drafting, CAD
- Geometric modeling
- Special effects
- Animation
- Virtual Reality
- Games
- Educational tools
- Surgical simulation
- Scientific and information visualization
- Fine art

Graphics areas

- **Modeling -** mathematical *representations* of physical objects and phenomena
- **Rendering -** creating a *shaded image* from 3D models
- **Animation -** creating motion through a sequence of images
- **Simulation** - physics-based algorithms for animating dynamic environments

Modeling

Talton et al., 2011



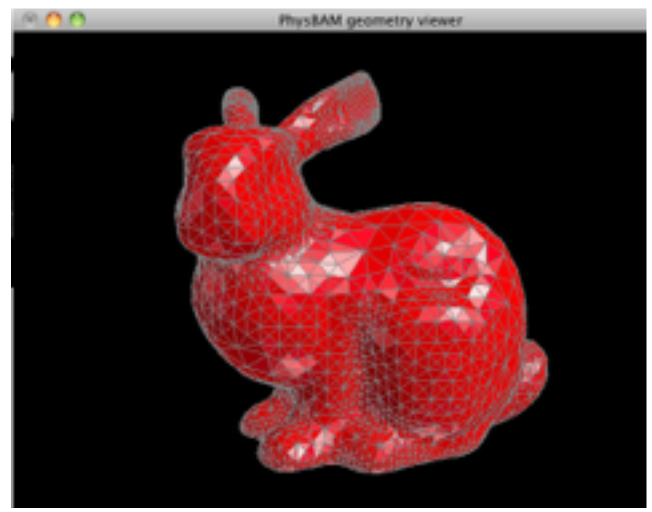
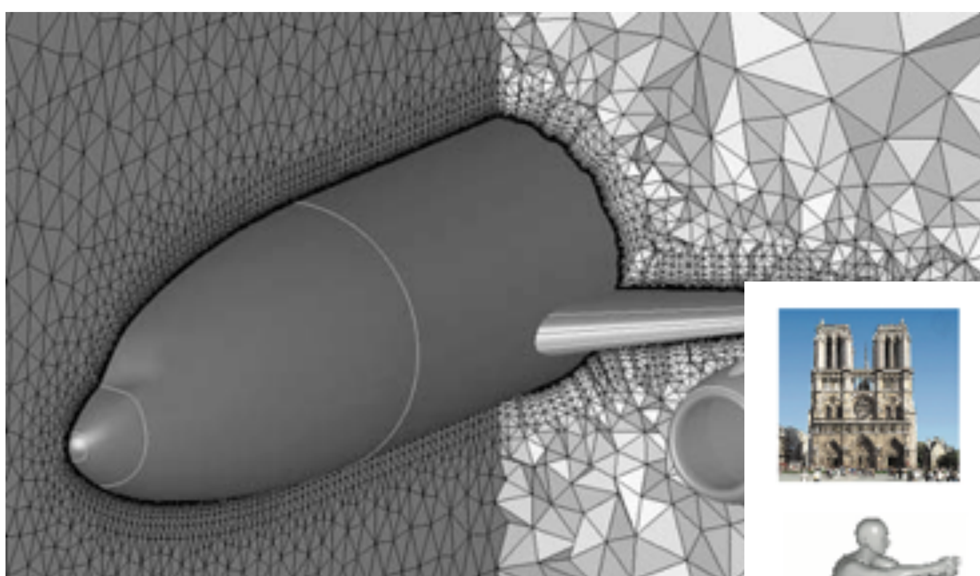


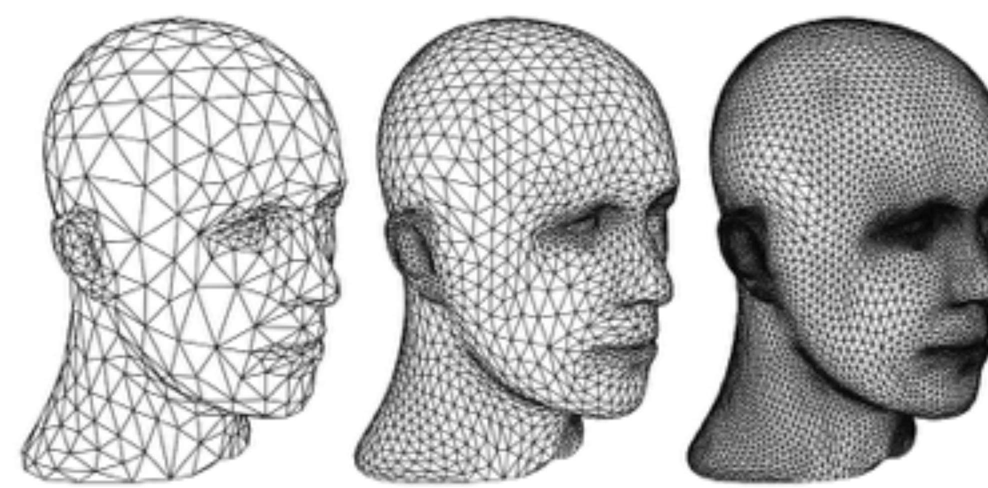
Figure1: Teddy in use on a display-integrated tablet.

Igarashi et al., 2007



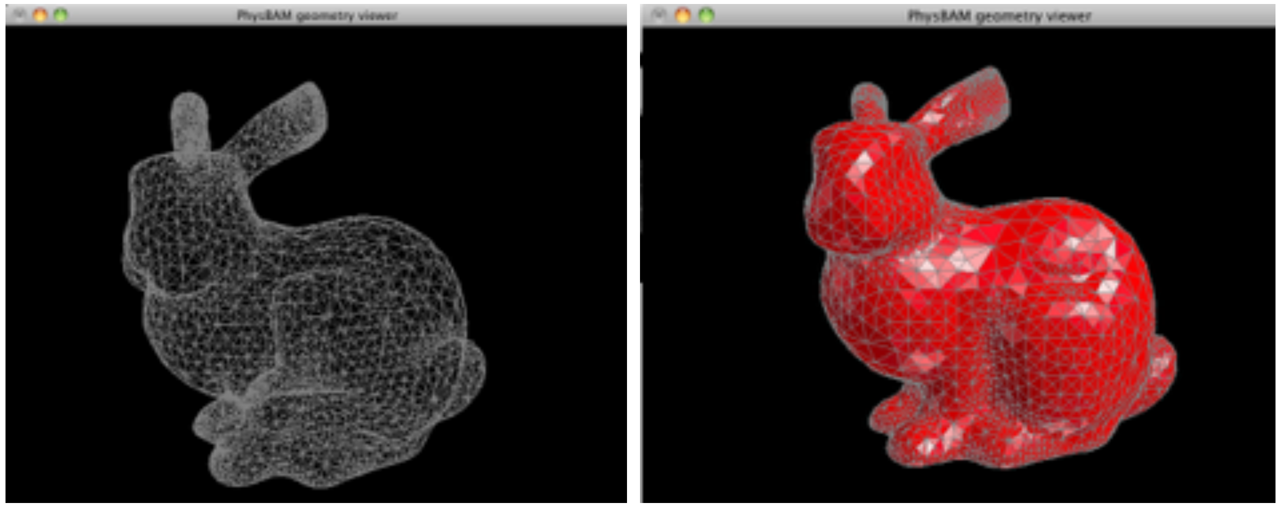
CFD Technologies

Bronstein et al., 2011



Schröder, 2000

Rendering



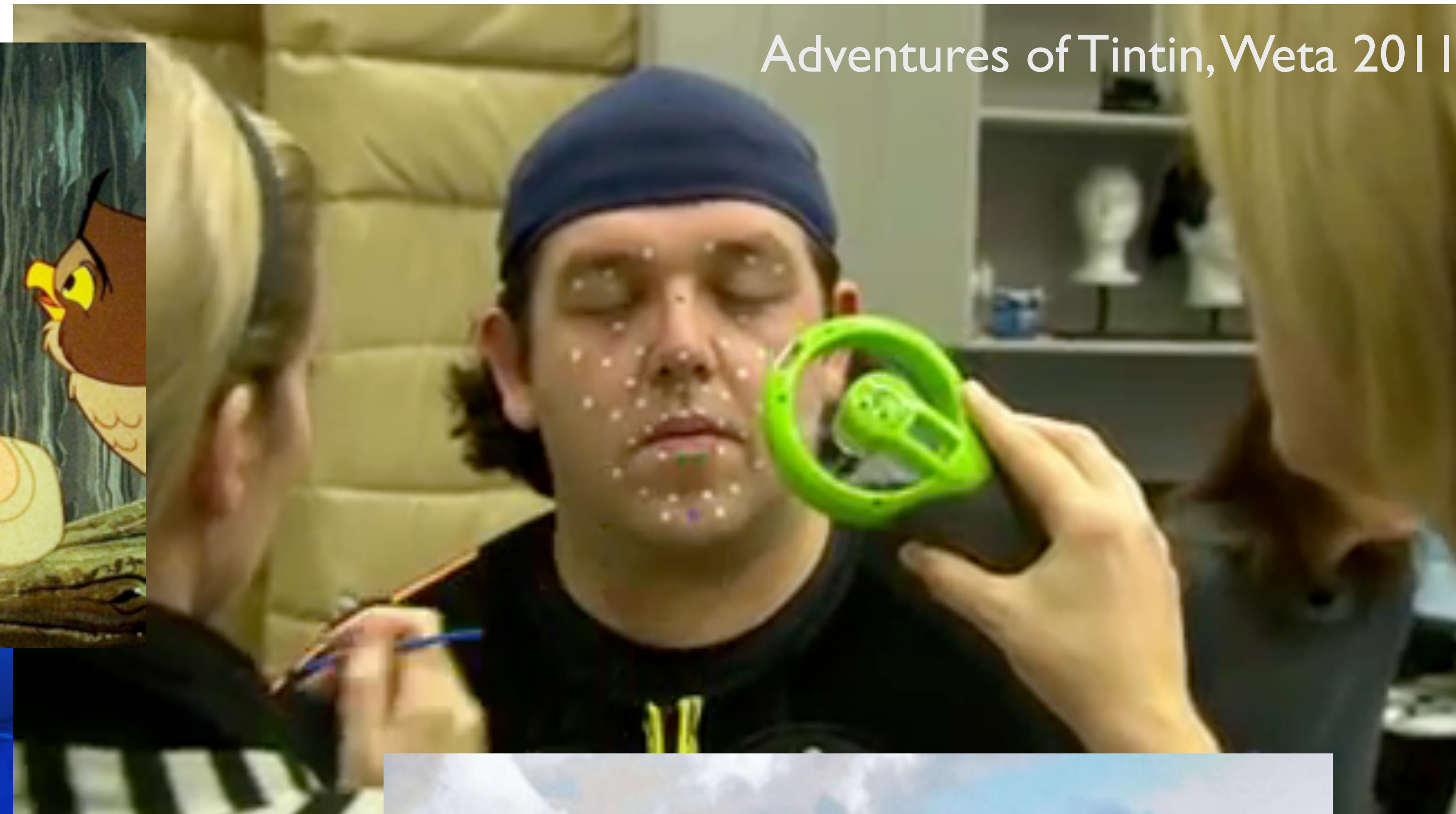
Henrik Wann Jensen

Hong et al. 2007

d'Eon and Irving, 2011

Animation

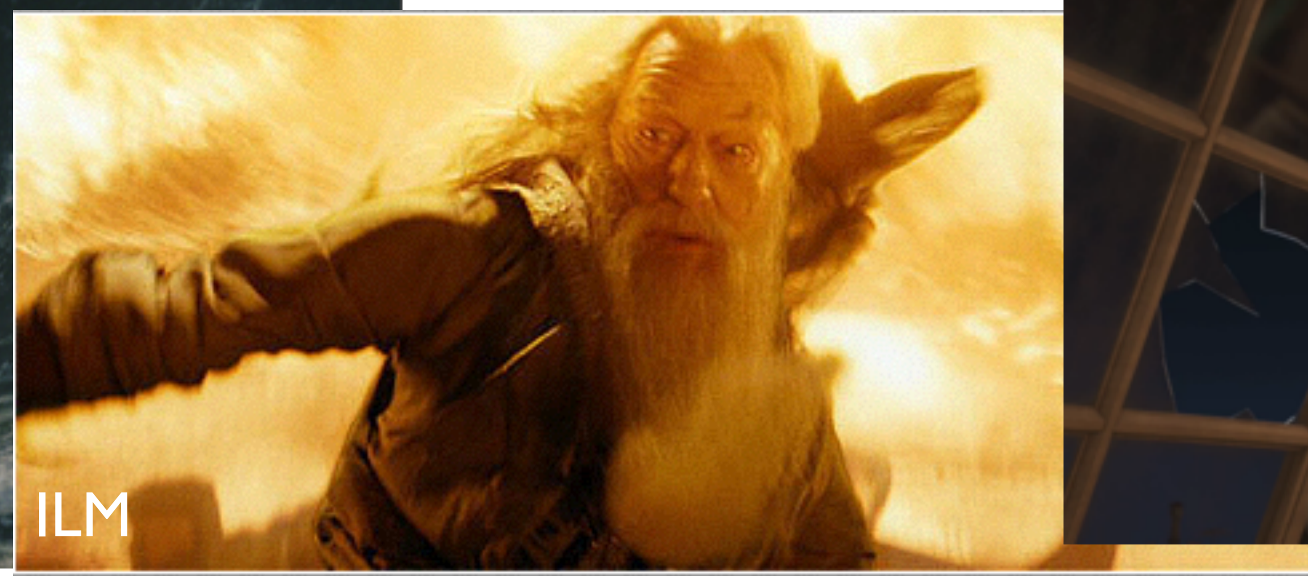

Sleeping Beauty, Disney, 1959

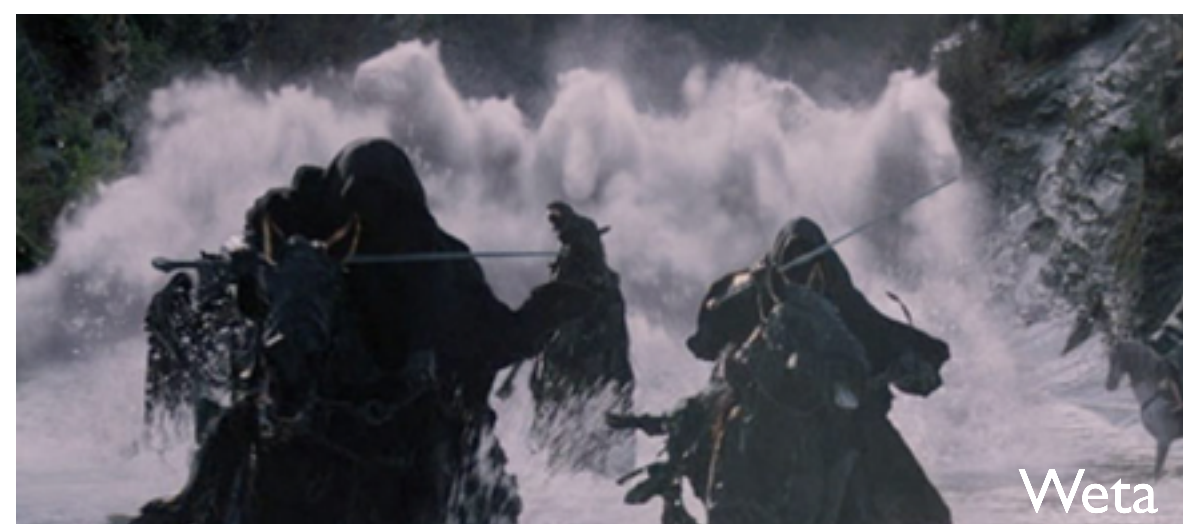

Adventures of Tintin, Weta 2011


Monsters Inc, Pixar, 2001


Life of Pi, 2012

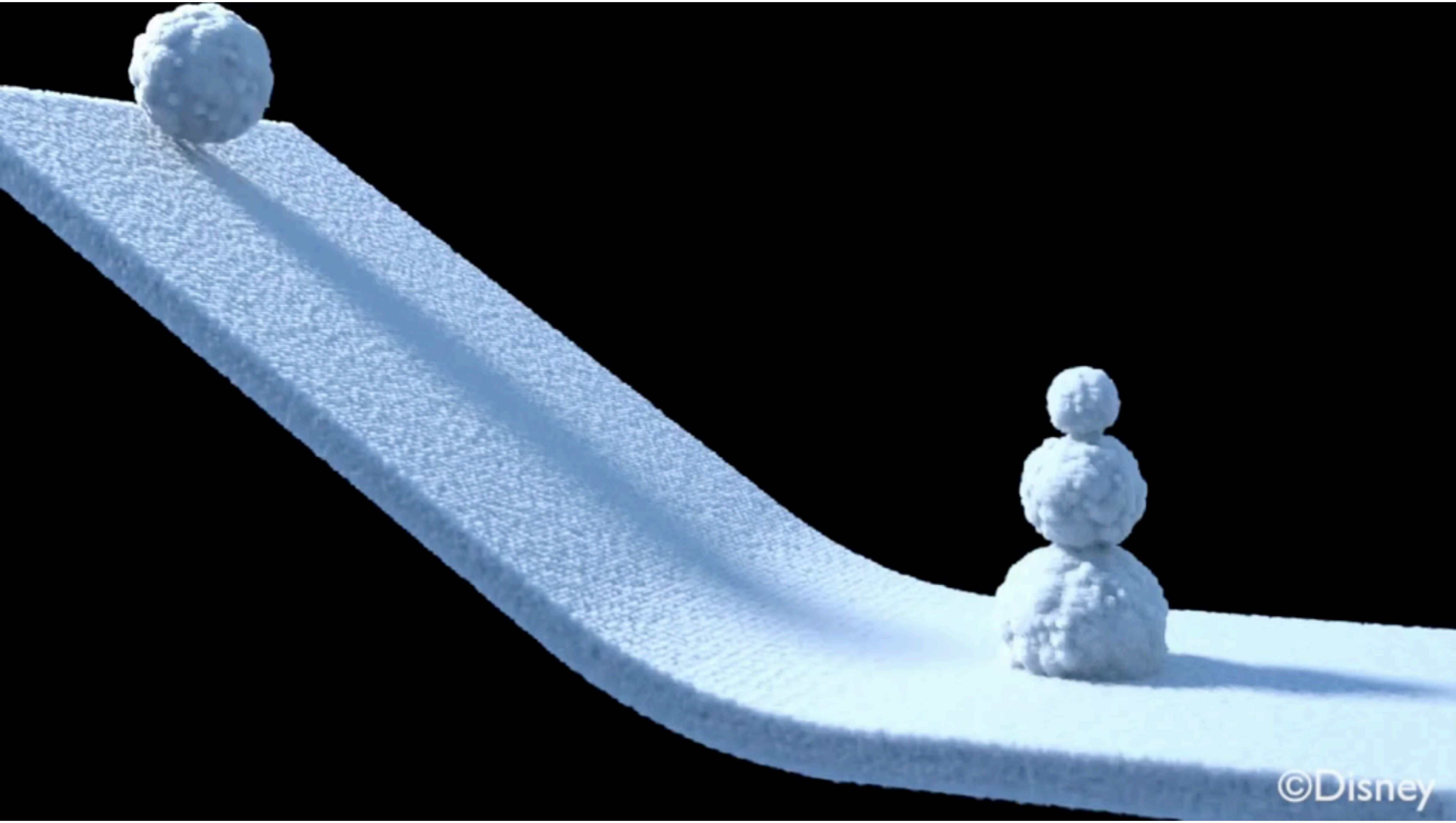
Simulation



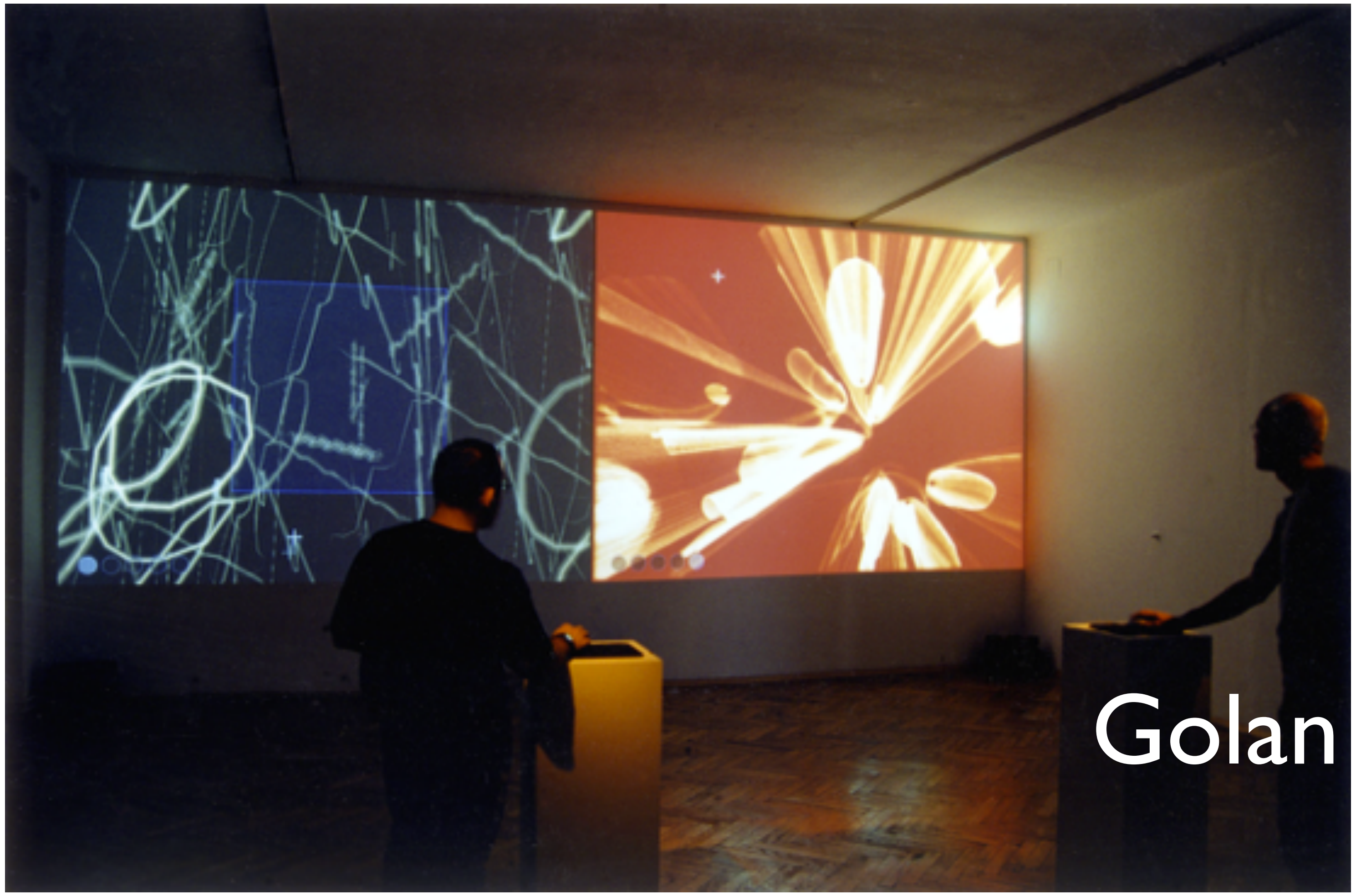


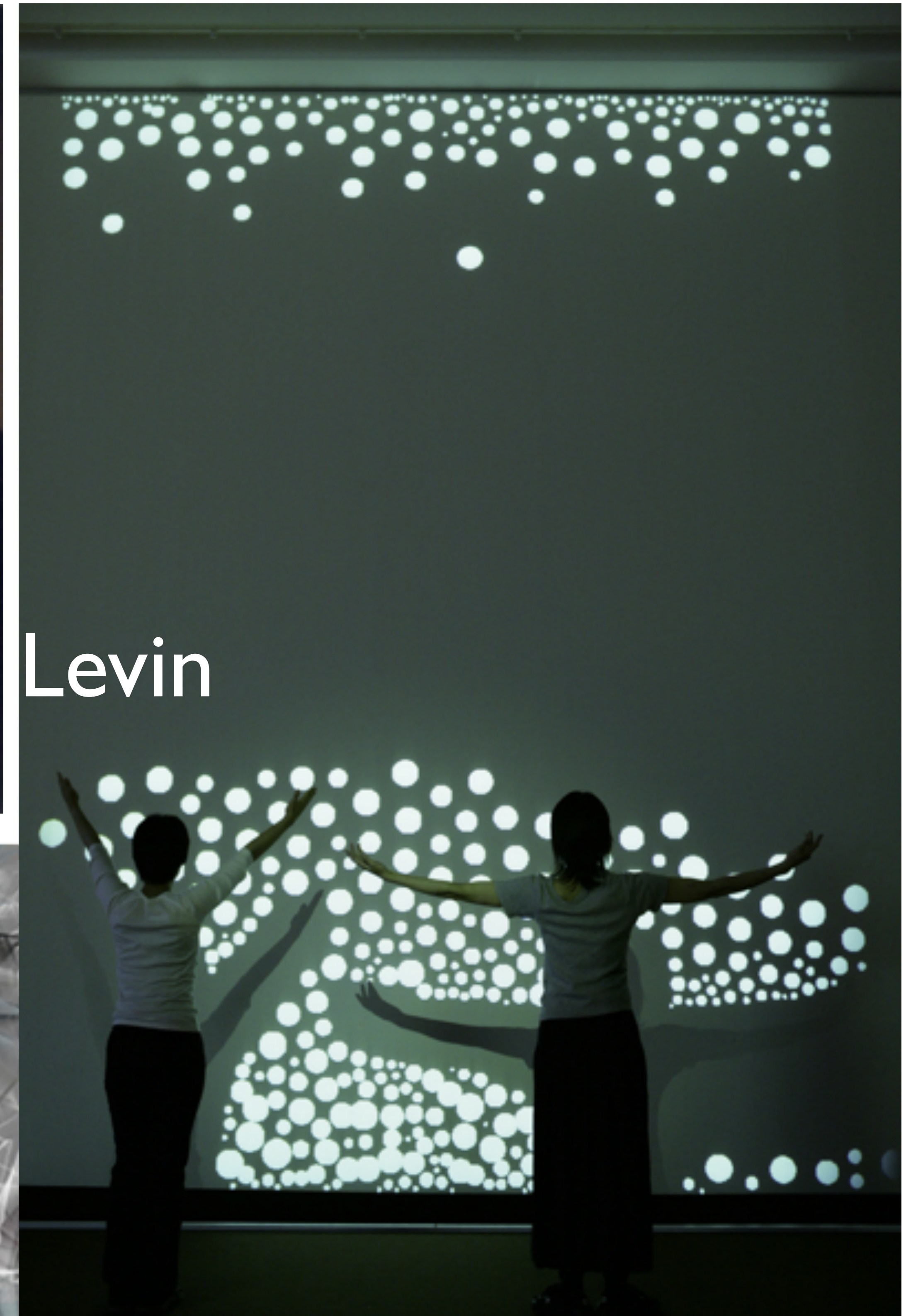
Firestorm
Harry Potter and the Half Blood Prince
Industrial Light + Magic

fluid simulation in Pixar's *Ratatouille 2007*

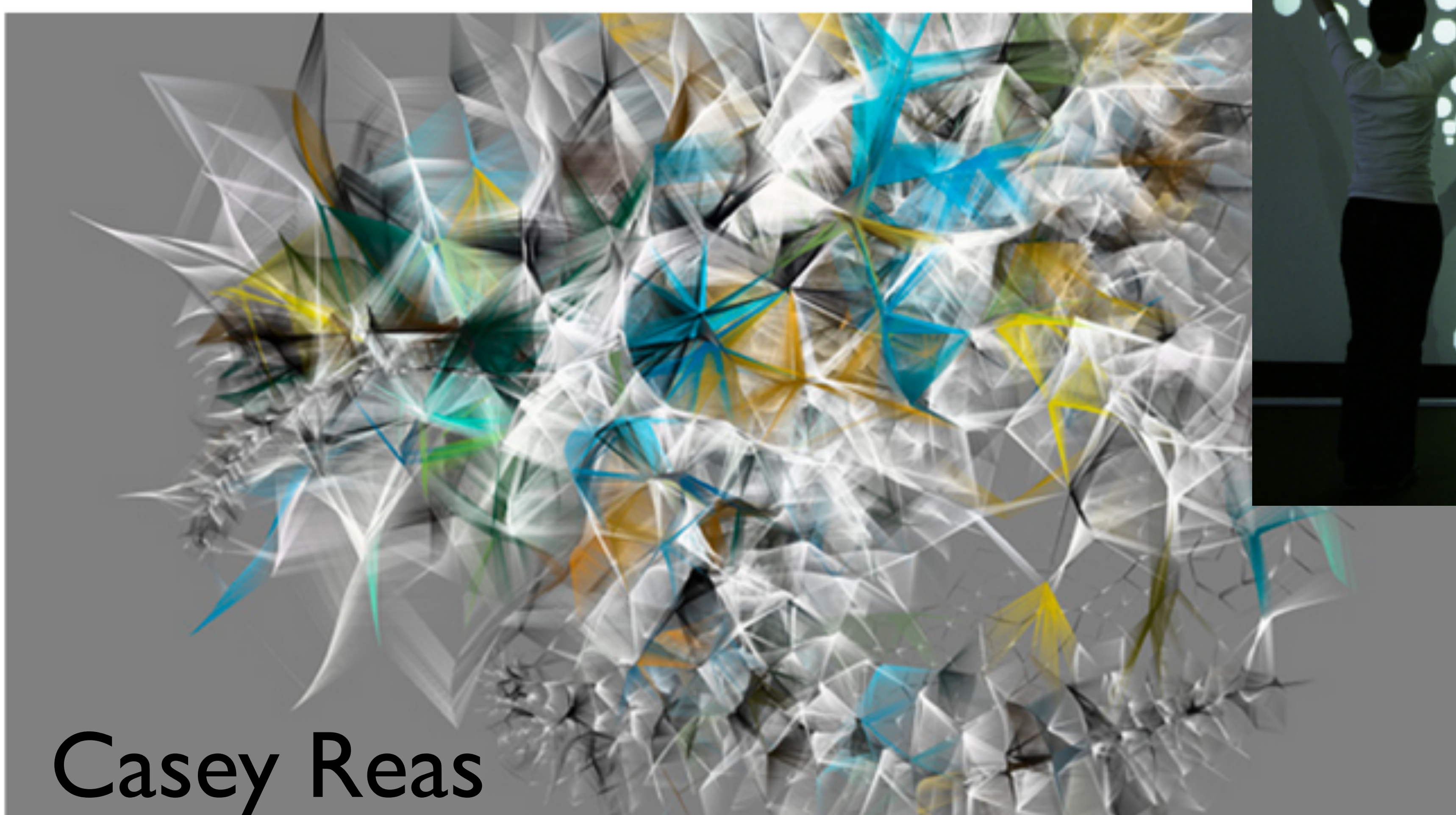


Stomakhin et al. 2013





Golan Levin

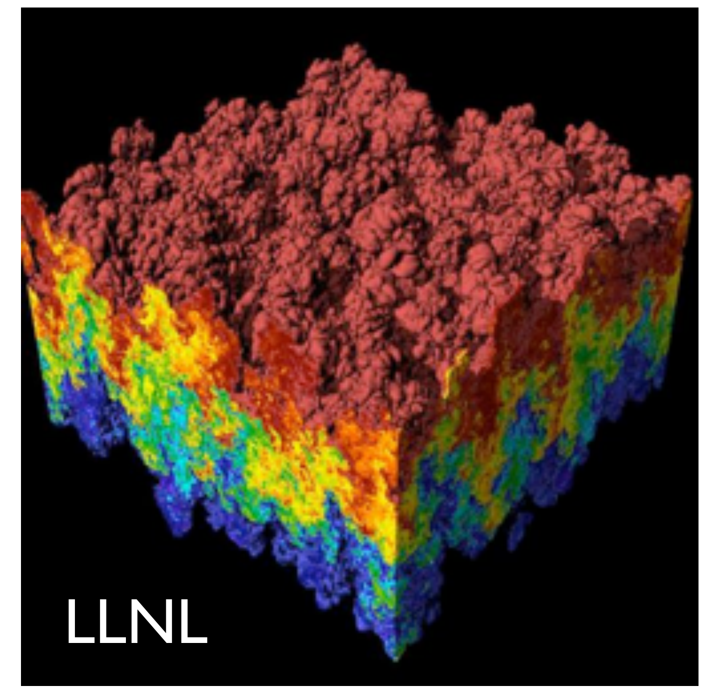


Casey Reas

Other areas...

- Interactivity (HCI)
- Image processing
- Visualization
- Computational photography


L-1 Identity Solns


LLNL


Lytro

Microsoft Kinect

Introduction to OpenGL

OpenGL - Software to Hardware

- Silicon Graphics (SGI) revolutionized the graphics workstation by putting graphics pipeline in hardware (1982)
- To use the system, application programmers used a library called GL
- With GL, it was relatively simple to program three dimensional interactive applications

OpenGL

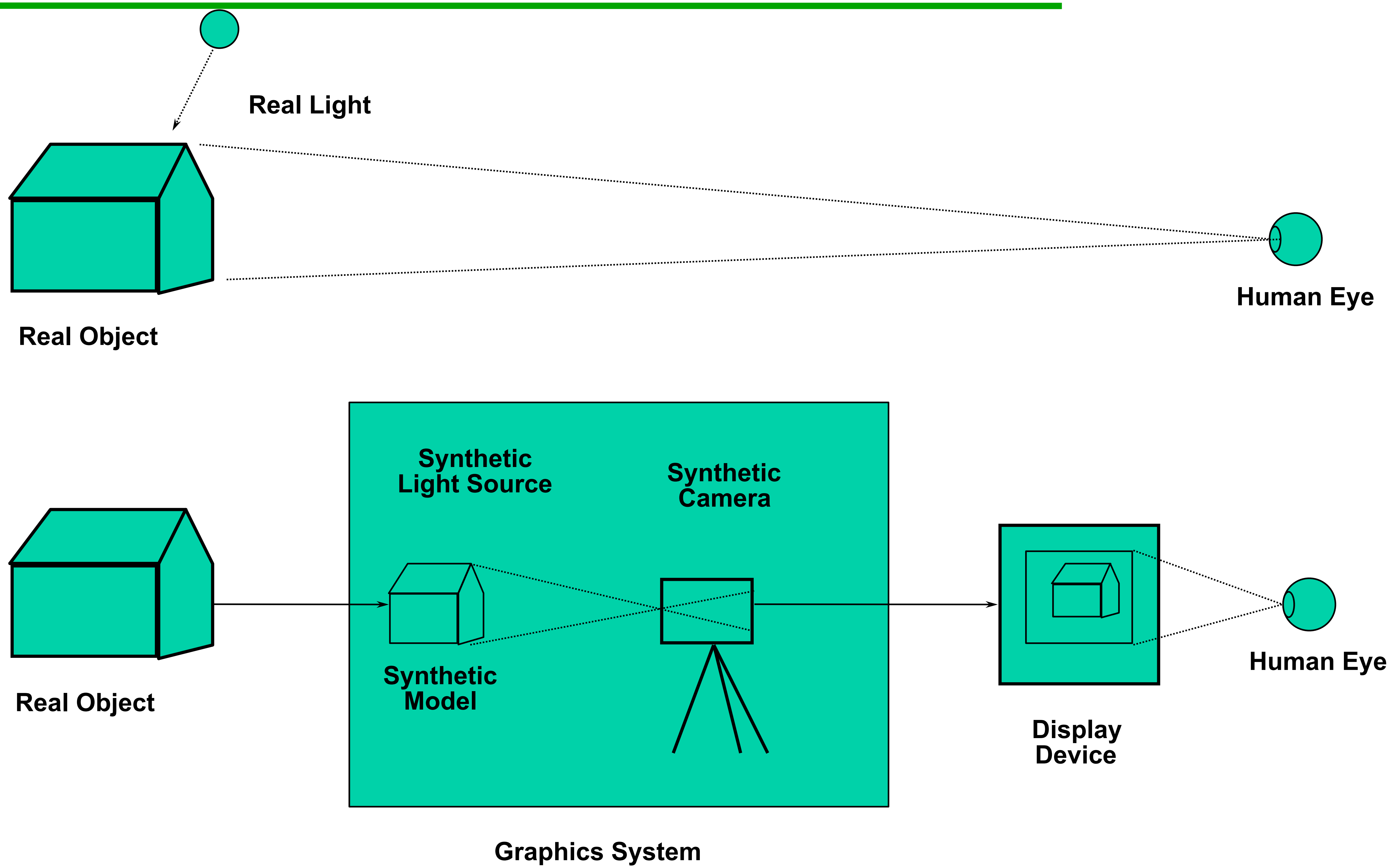
- The success of GL lead to OpenGL (1992), a platform-independent API that was
  - Easy to use
  - Close to the hardware - excellent performance
  - Focus on rendering
  - Omitted windowing and input to avoid window system dependencies

Introduction to OpenGL

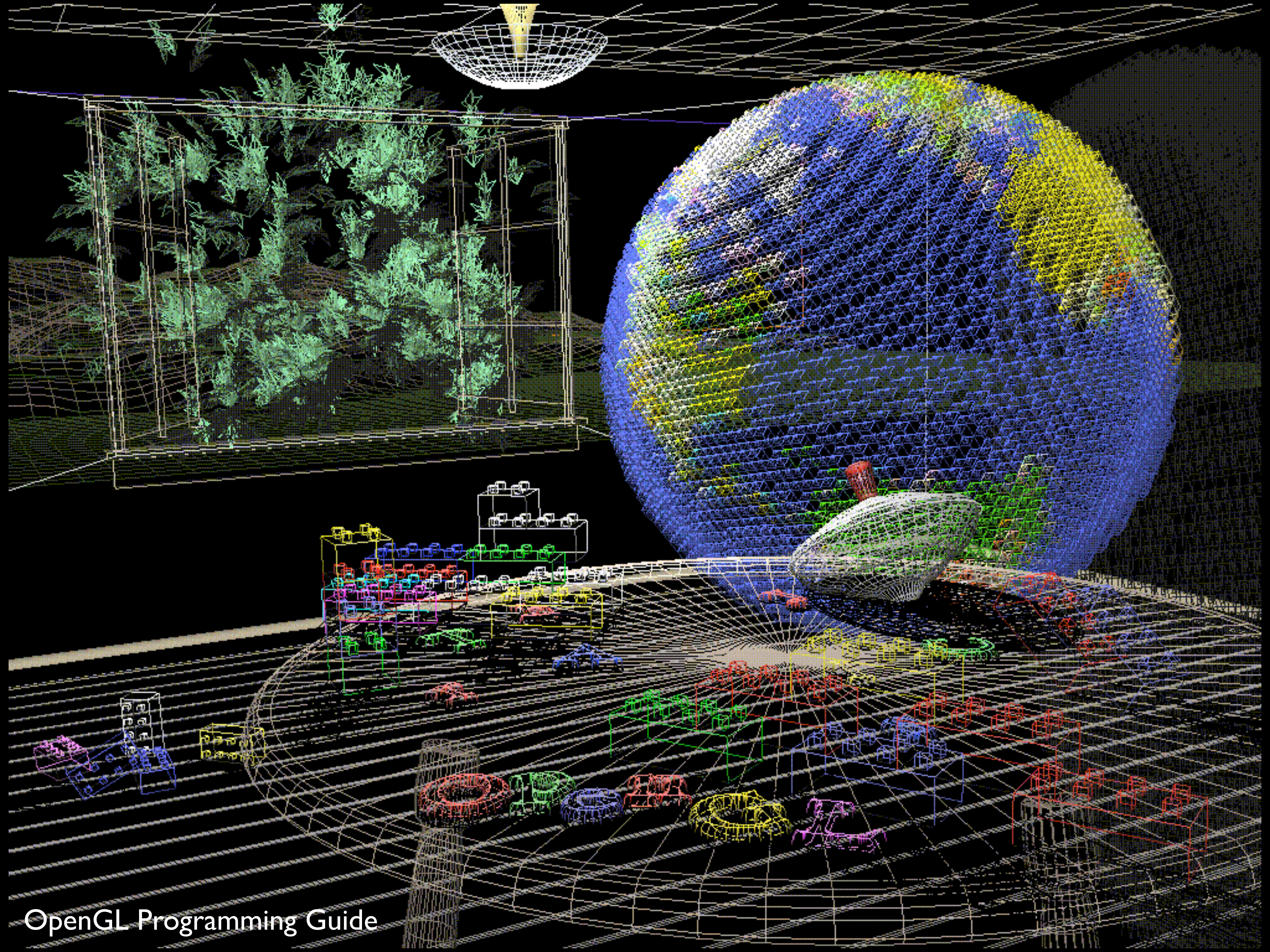
- **Open G**raphics **L**ibrary, managed by Khronos Group
- A software interface to graphics hardware (GPU)
- Standard API with support for multiple languages and platforms, open source
- ~250 distinct commands
- Main competitor: Microsoft's Direct3D
- http://www.opengl.org/wiki/Main_Page

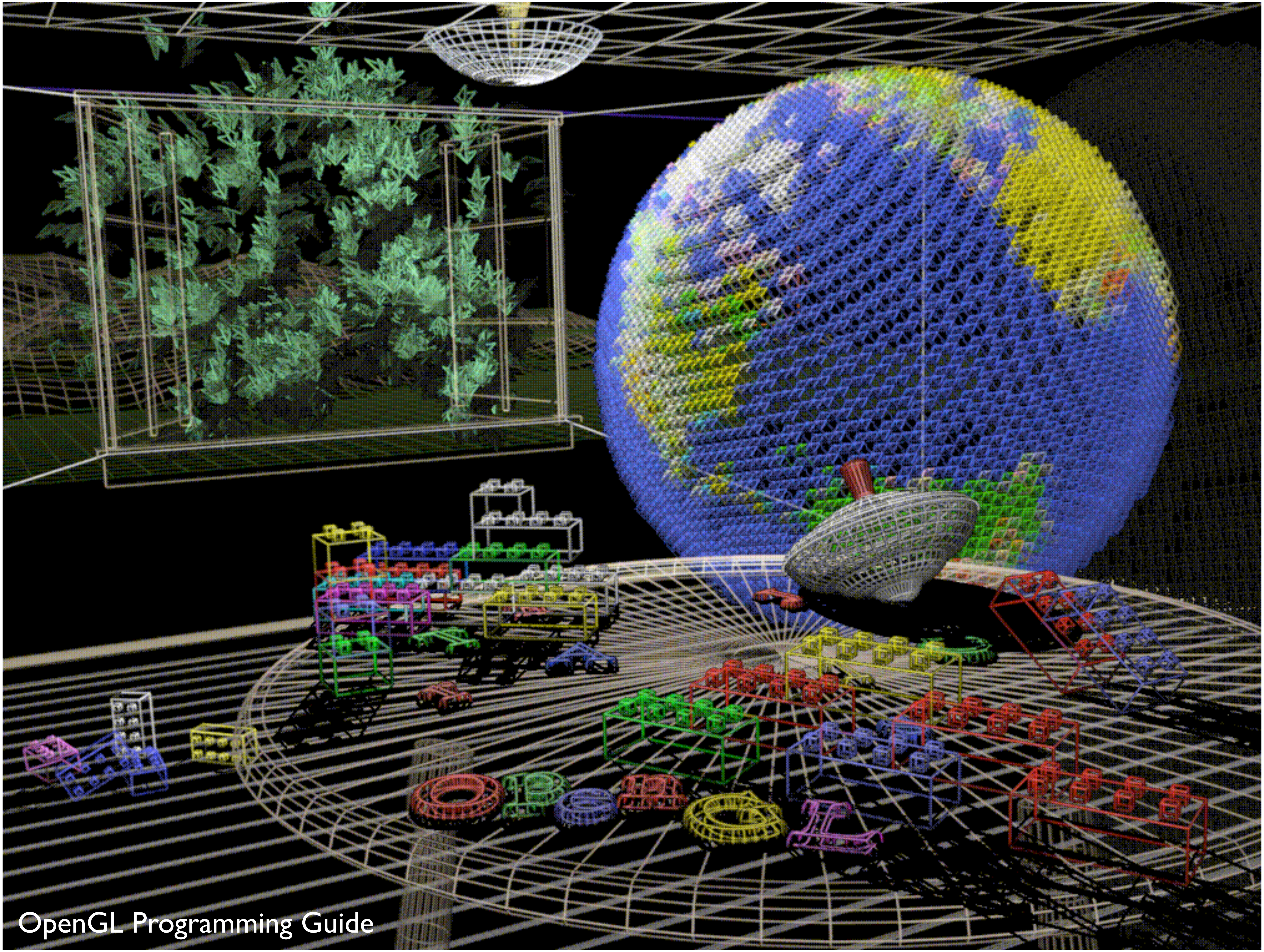
OpenGL: Conceptual Model

OpenGL: Conceptual Model "synthetic-camera model"



What can OpenGL do?
Examples from the
OpenGL Programming Guide (“red book”)


OpenGL Programming Guide


OpenGL Programming Guide

OpenGL Programming Guide


OpenGL
OpenGL Programming Guide


OpenGL
OpenGL Programming Guide


OpenGL
OpenGL Programming Guide


OpenGL Programming Guide

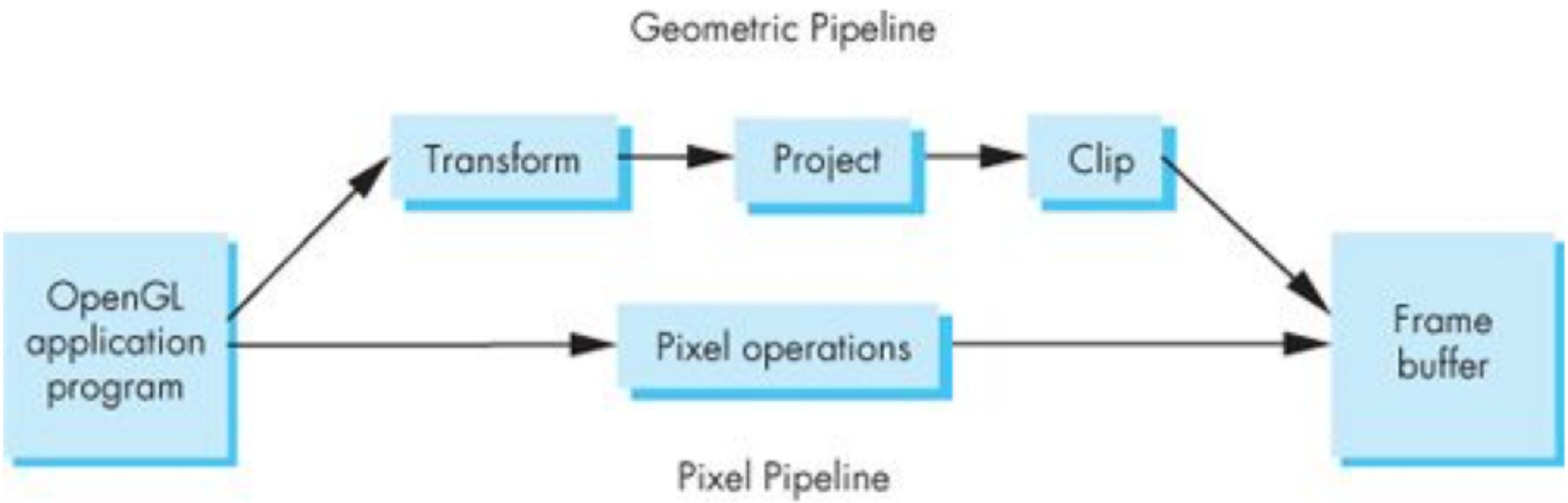
OpenGL Context

- contains all the information that will be used by OpenGL in executing a rendering command
- OpenGL functions operate on the "current" context
- local to an application
- application may have several OpenGL contexts

OpenGL State

- context contains “state” information
- put OpenGL into various states
    - e.g., current color, current viewing transformation
    - these remain in effect until changed
    - glEnable(), glDisable(), glGet(), glIsEnabled()
    - glPushAttrib(), glPopAttrib() to temporarily modify some state

OpenGL Rendering Pipeline



- sequence of steps OpenGL takes when rendering an object

OpenGL Shaders

- Pipeline has evolved from fixed-function to programmable
  - programs are called “Shaders”
  - execute on the GPU
  - enables a lot of functionality (e.g., geometry compression, bump mapping, skinning)
- In modern OpenGL, every program must provide at least **vertex** and **fragment** shaders
- Written in the OpenGL Shading Language (GLSL)

OpenGL command syntax

- commands: **gl**ClearColor();
  - glVertex**3f**()
- constants: **GL**_COLOR_BUFFER_BIT
- types: GLfloat, GLdouble, GLshort, GLint,

Simple OpenGL program

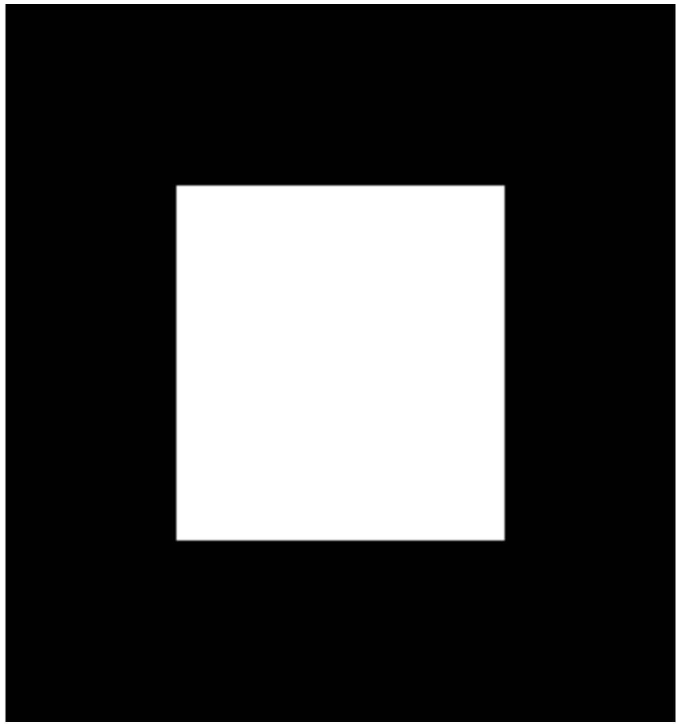
```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f(0.25, 0.25, 0.0);
      glVertex3f(0.75, 0.25, 0.0);
      glVertex3f(0.75, 0.75, 0.0);
      glVertex3f(0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```

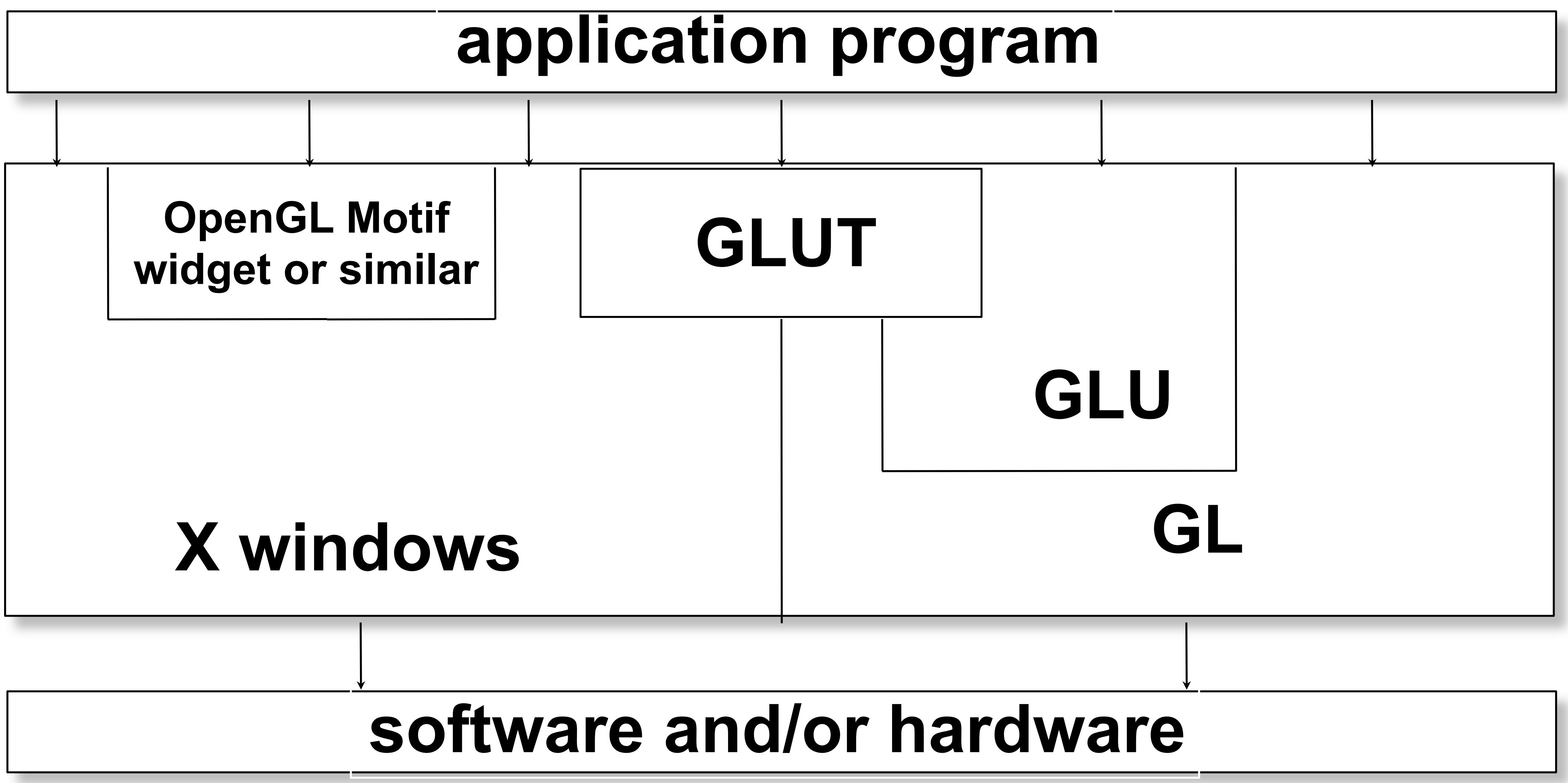


OpenGL Programming Guide, 7th Ed.

OpenGL Libraries

- OpenGL core library (gl.h)
  - OpenGL32 on Windows
  - GL on most unix/linux systems
- OpenGL Utility Library -GLU (glu.h)
  - avoids having to rewrite code
- OpenGL Utility Toolkit -GLUT (glut.h)
  - Provides functionality such as:
    - Open a window
    - Get input from mouse and keyboard
    - Menus

Software Organization



Simple OpenGL program

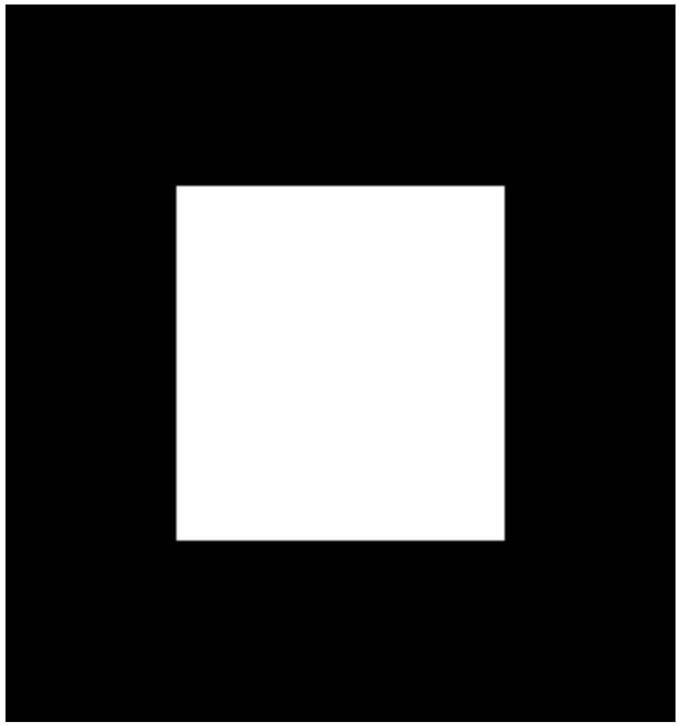
```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f(0.25, 0.25, 0.0);
      glVertex3f(0.75, 0.25, 0.0);
      glVertex3f(0.75, 0.75, 0.0);
      glVertex3f(0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```



OpenGL Programming Guide, 7th Ed.

Simple OpenGL program

```
#include<GL/glut.h>

void init() {
   glClearColor(0.0, 0.0, 0.0, 0.0);
}

void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f(0.25, 0.25, 0.0);
      glVertex3f(0.75, 0.25, 0.0);
      glVertex3f(0.75, 0.75, 0.0);
      glVertex3f(0.25, 0.75, 0.0);
   glEnd();
   glFlush();
}
main() {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (FB_WIDTH, FB_HEIGHT);
    glutCreateWindow ("Test OpenGL Program");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
}
```