CS 543: Computer Graphics

Introduction

Robert W. Lindeman
Associate Professor
Interactive Media & Game Development
Department of Computer Science
Worcester Polytechnic Institute
gogo@wpi.edu

(with lots of help from Prof. Emmanuel Agu :-)

What to Expect

- This course is mainly about how to create ***pretty pictures***
  - Algorithms, mathematics, data structures
  - Over 40 years of research
- Today, a big chunk is available off the shelf
  - Just make **OpenGL** or **DirectX** library calls
  - Use **WebGL** to remove platform dependencies
- We want to learn what is *inside* these libraries
  - We use WebGL as one example of how things could be done
  - At work, you may only use OpenGL, or a Game Engine
  - The ***really interesting*** jobs will ask you to go further!

Summary of Syllabus

- 2 Exams (50%), 4 Projects (50%)
- Projects will use WebGL
- Write code on any platform (Zoo Lab - FL A21)
- Must run in a Web browser
- Program in JavaScript
- Can discuss with others, turn in unique project
- All material on class Website
  - www.cs.wpi.edu/~gogo/courses/cs543/
- Text
  - ***Interactive Computer Graphics: A Top-Down Approach with WebGL*** (7th edition), by Angel and Shreiner, 2015.

Assignments

- Many phases to homework:
  - Understand/design/code/debug/test/eat/test some more
  - Encouraged to discuss approaches
  - Must hand in your own work only
- Cheating:
  - Many reasons *not* to do it!
  - Immediate 'F' in the course
- Advice for doing well:
  1. Do the assigned reading
  2. Come to class
  3. Ask questions (class, office hours, MyWPI discussions)
  4. Make sure you understand before coding
  5. Don't share your code with others!

What to Expect (cont.)

- This course is about Computer Graphics, not WebGL
  - How would one *build* WebGL or OpenGL?
  - Focus on underlying methods
  - Other methods besides WebGL
- This course is heavy on
  - Coding (JavaScript, shaders)
  - Efficiency (speed & space)
  - Pretty pictures

What is Computer Graphics (CG)?

- Computer graphics
  - Algorithms, mathematics, data structures that computer uses to generate PRETTY PICTURES
- Techniques (e.g., draw a line, polygon) evolved over years
- Built into programmable libraries

Computer Generated!
Not a photo!

CHRONOSWISS

Photorealistic vs. Real-Time Graphics

Not this Class

Maker's Mark
WHISKY

- **Photorealistic**
  - High quality
  - Slow to render (days)

This Class

M. BLV
exp sunday
ALL VEHICLES
EXP WAY
NVIDIA

- **Real-Time graphics**
  - Lower quality
  - Fast to render (60 FPS)

Uses of Computer Graphics

- **Entertainment**
  - Games

Courtesy: *Final Fantasy XIV*

Courtesy: *Super Mario Galaxy 2*

Uses of Computer Graphics

- **Entertainment**
  - Movies, TV, books, magazines

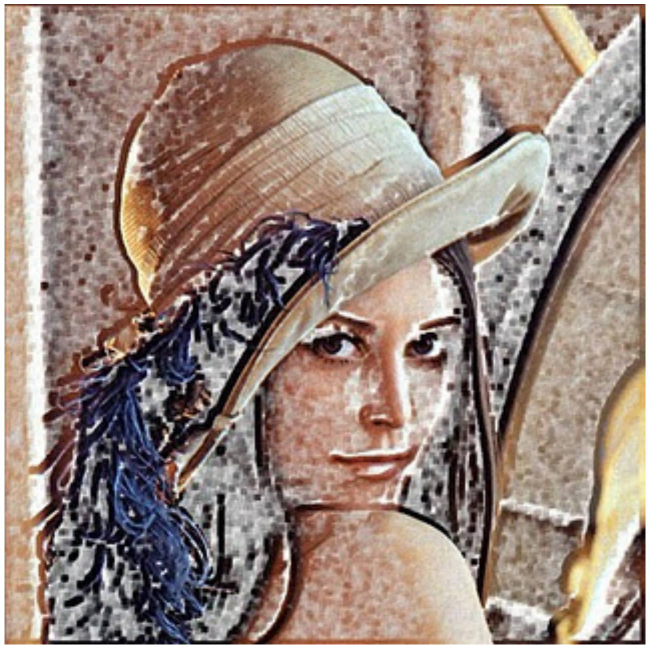
Courtesy: *Shrek*

Courtesy: *Spider-Man*

Uses of Computer Graphics

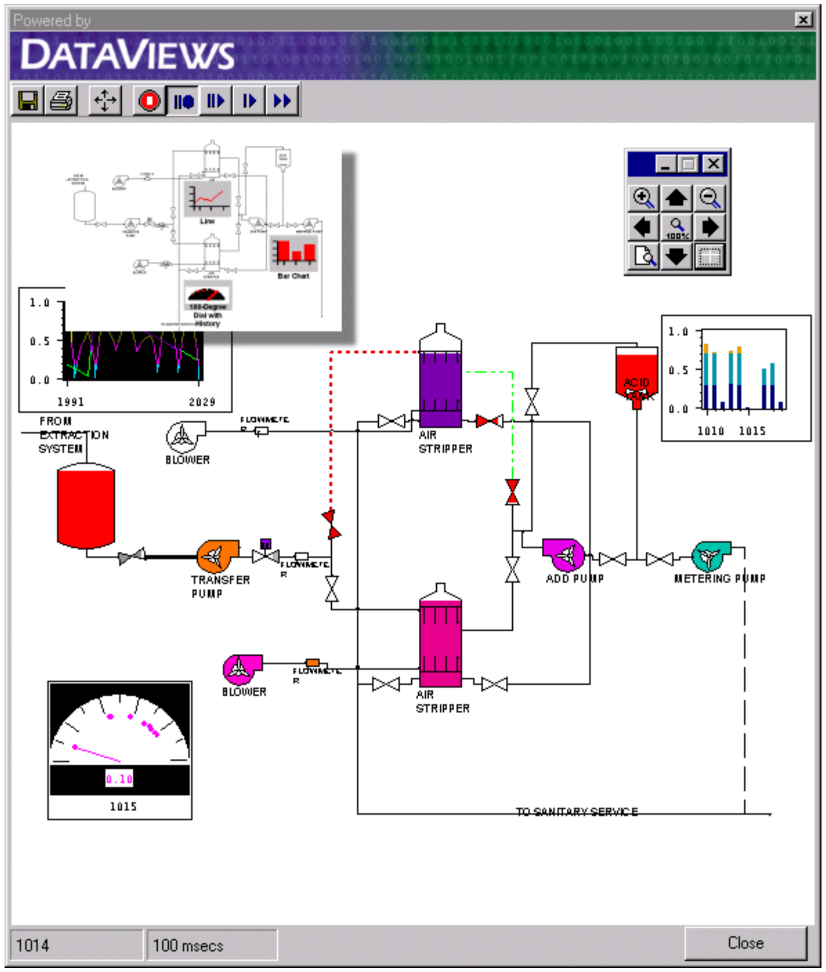
- **Image processing**
  - Alter images, remove noise, super-impose images

Original Image


Sobel Filter

Uses of Computer Graphics

- **Process monitoring**
  - Layout of large systems or plants
  - Monitor manufacturing process
  - User control automatic and manual control



Courtesy: *Dataviews.de*

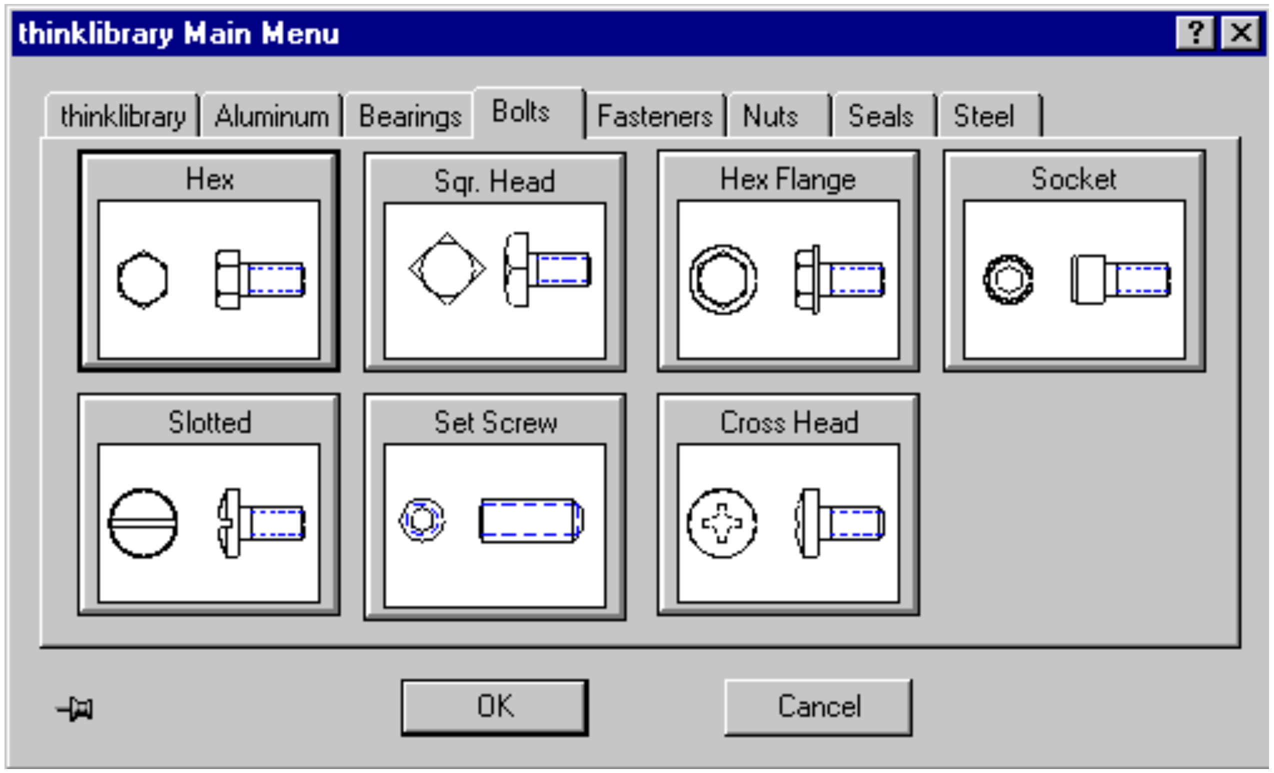
Uses of Computer Graphics

- **Display simulations**
  - Flight simulators, virtual worlds

Courtesy*: Evans and Sutherland*

Uses of Computer Graphics

- **Computer-aided design**
  - Architecture, electric circuit design

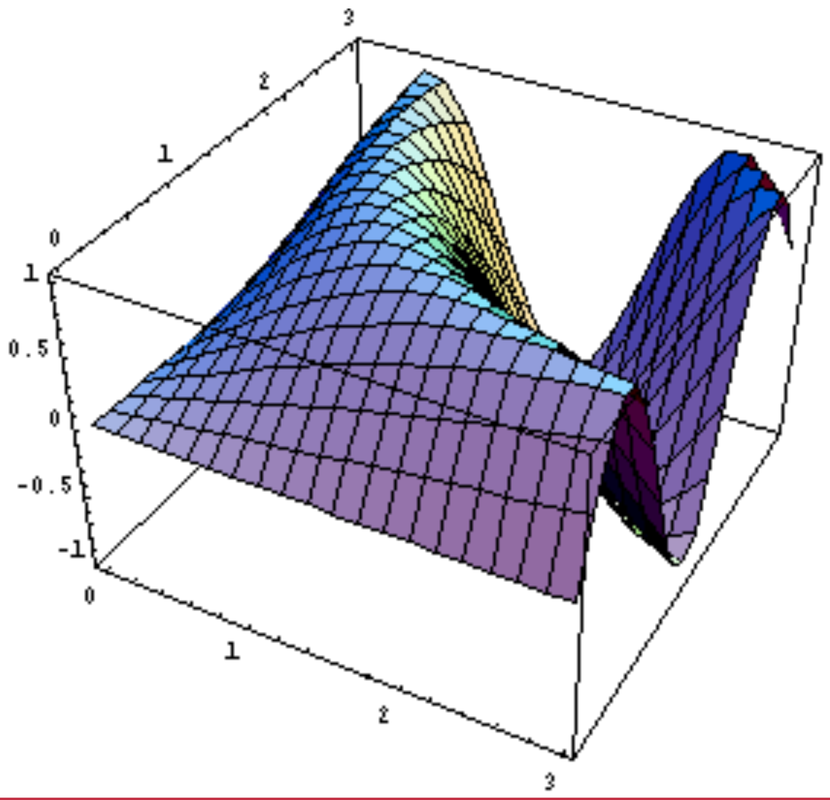


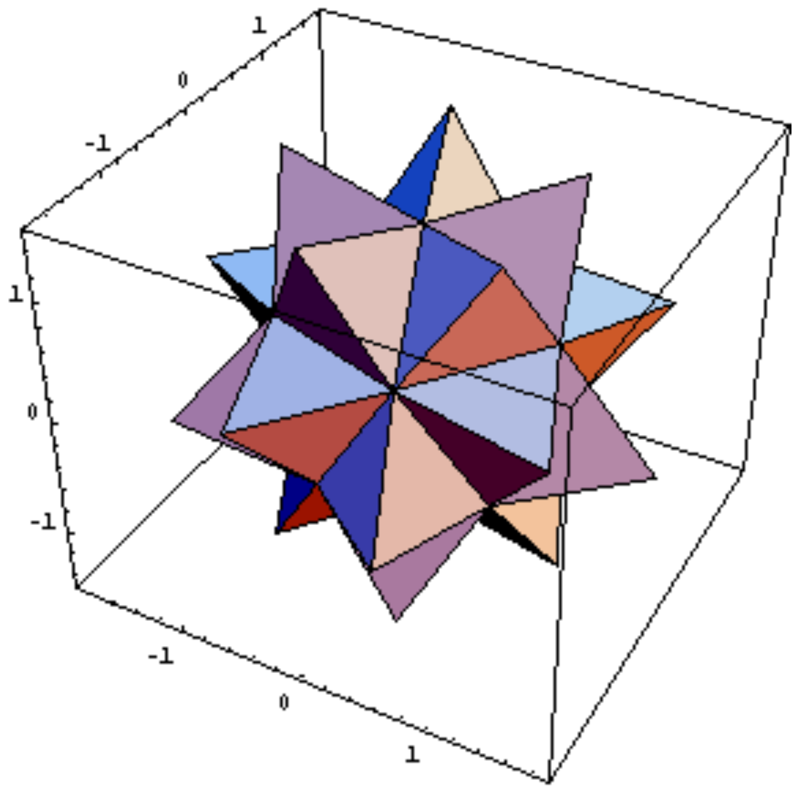
Courtesy: *cadalog.com*

WPI

Uses of Computer Graphics

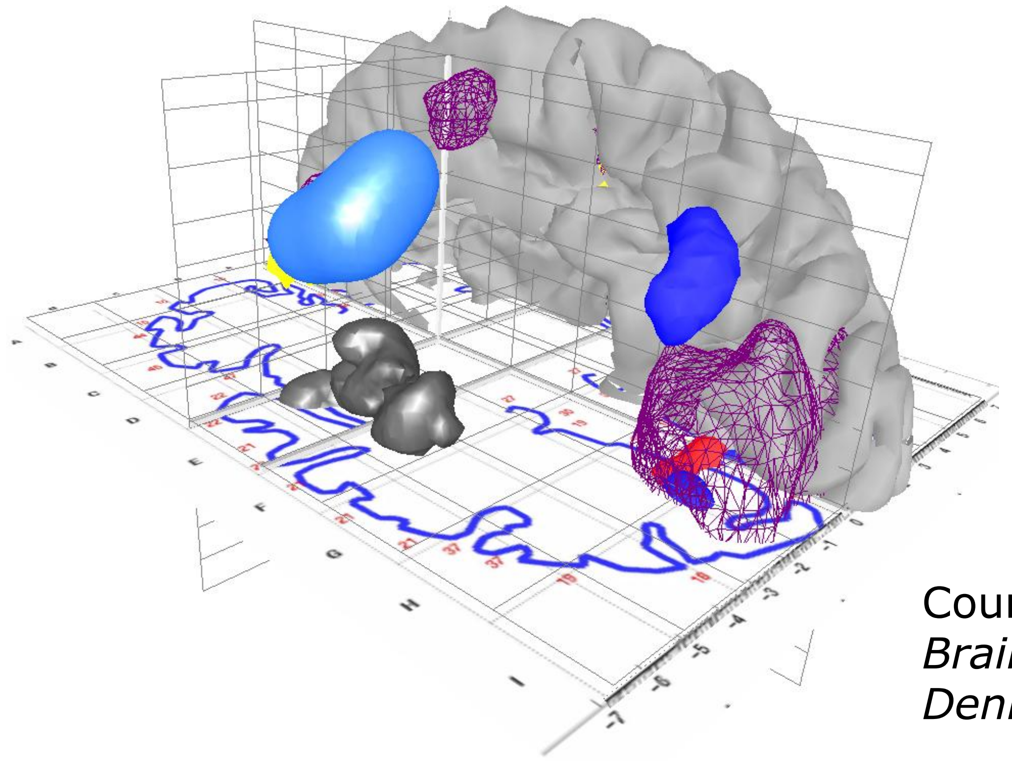
- **Displaying Mathematical Functions**
  - e.g., Mathematica®





Uses of Computer Graphics

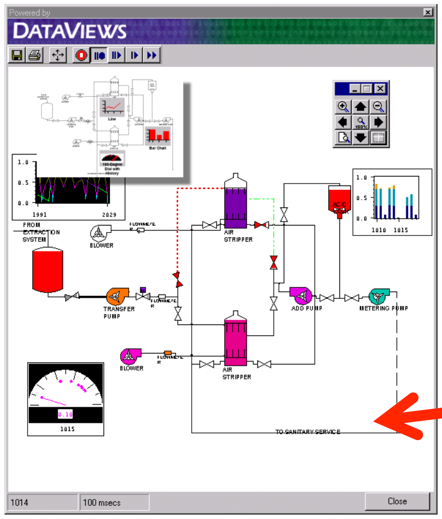
- **Scientific analysis and visualization**
  - Molecular biology, weather, matlab, Mandelbrot set



Courtesy*: Human Brain Project, Denmark*

2 Dimensional vs. 3 Dimensional

- **2D**
  - No notion of distance from viewer
  - Only (x, y) color values on screen
- **3D**
  - Objects have distance from viewer
  - (x, y, z) values on screen



- This class covers both 2D & 3D!
- Also interaction, e.g., clicking, dragging, etc.

M. BLV exp sunday
EXP WAY
ALL VEHICLES
NO TRUCKS
nVIDIA

Related Areas to CG

- Modeling: Shape of objects in a scene
- Shading & Lighting: Surface & Environmental effects
- Post Production: Tweaking the images
- Computer Vision: Extracting info from images
- Scientific Visualization: Making sense of data
- Animation: Making things move over time and space
- HCI: Incorporating user interaction

CG Tools

- Hardware tools
  - Output devices
    - Monitors, projection systems, VR helmets, printers
  - Input devices
    - Mouse/trackball, pen/tablet, keyboard, other
  - Graphics accelerators
- Software tools
  - IDEs (VS, Eclipse)
  - Editor (emacs, vi)
  - Compiler (g++)
  - Debugger
  - Graphics libraries
- Your eyes

What is a CG Library?

- Low-level routines
  - Points, lines, circles, text, *etc.*
- High-level routines
  - Pull-down menus, window management, *etc.*
- Some of this has traditionally been device dependent
  - Difficult to port, error prone
- Now we have device/platform independence (almost)
  - WebGL, OpenGL, DirectX, *etc.*
  - XBOX, PS1/2/3/4/Vita/…, Wii, DS, smartphones, *etc.*

Game Engines

- Game Engines are frameworks that handle many aspects of games at a high level
  - Sit on top of low-level libraries (OpenGL/ DirectX)

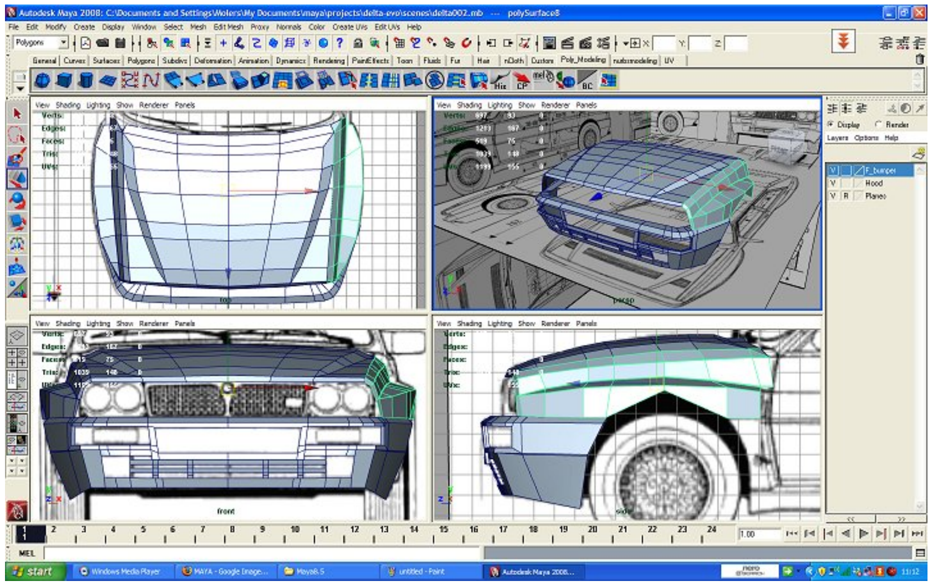
What is a Game Engine?

- A resource manager that supports an entertainment (usually) application
- Graphical (audio, *etc.*) rendering
- A user interface
- Script handling
- Event processing
  - Time, collisions, *etc.*
- File I/O
- Asset-creation tools
  - Models, graphics, sound, *etc.*
- Optional
  - Networking
  - AI

WPI

About This Course

- **Computer Graphics has many aspects**
  - **Computer Scientists**
    - ***Create/program*** CG tools/packages (e.g., Maya, photoshop)
  - **Artists**
    - ***Use*** CG tools/packages to create pretty pictures



About This Course

- **Most hobbyists follow artist path - Not much math!**
  - This Course: Computer Graphics for computer scientists!!!
- **Teaches concepts, uses WebGL as a concrete example**
- **Course is NOT…**
  - just about programming WebGL
  - a comprehensive course in OpenGL/WebGL. (Only covers parts)
  - about using packages like Maya, Photoshop

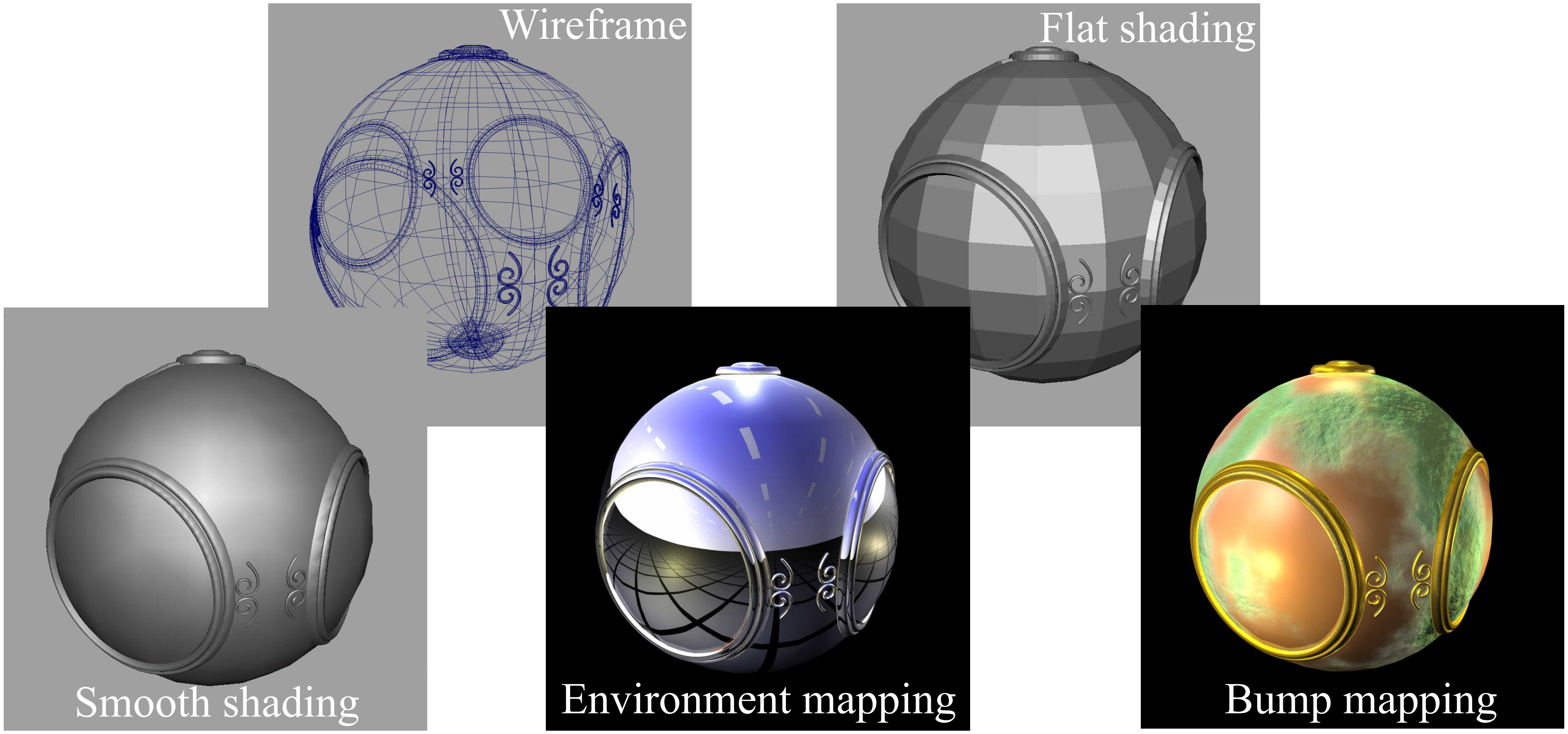
About This Course

- **Class is concerned with:**
  - How to build/program graphics tools
  - Underlying mathematics
  - Underlying data structures
  - Underlying algorithms
- **This course is a lot of work. Requires:**
  - Lots of coding in JavaScript
  - Shader programming
  - Lots of math, linear algebra, matrices
- **We shall combine:**
  - **Programmer's view:** Program WebGL APIs
  - **Under the hood:** Learn OpenGL internals (graphics algorithms, math, implementation)

WPI

Evolution of Rendered Images

- Multiple ways of representing things



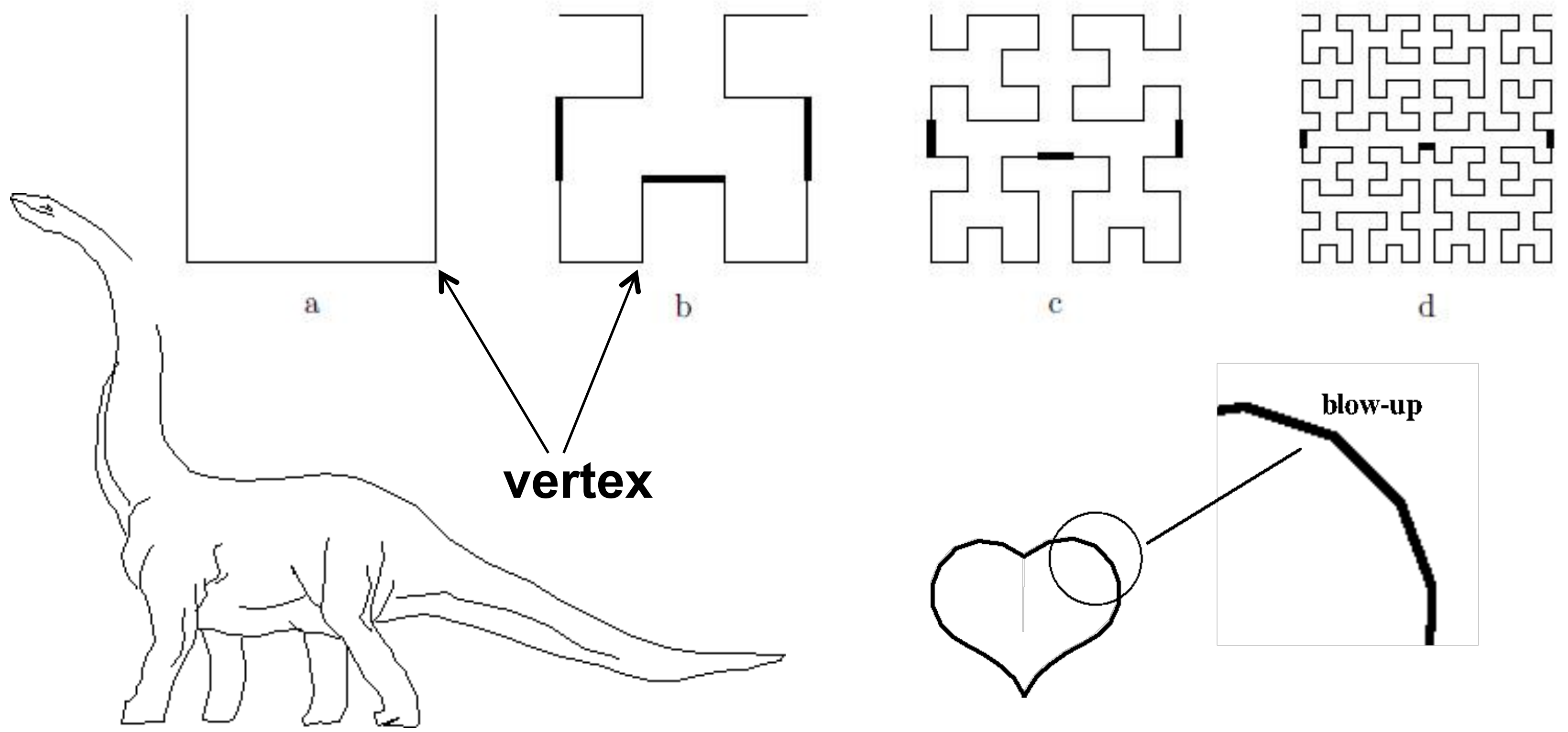
Current State: Things are pretty good right now...

Elements of 2D Graphics

- **Polylines**
- **Text**
- **Filled regions**
- **Raster images (pictures)**

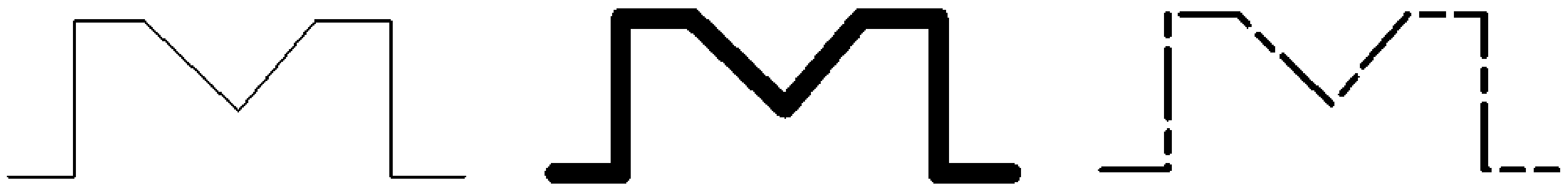
Elements of 2D Graphics

- **Polyline:** connected sequence of straight lines
- Straight lines connect **vertices** (corners)



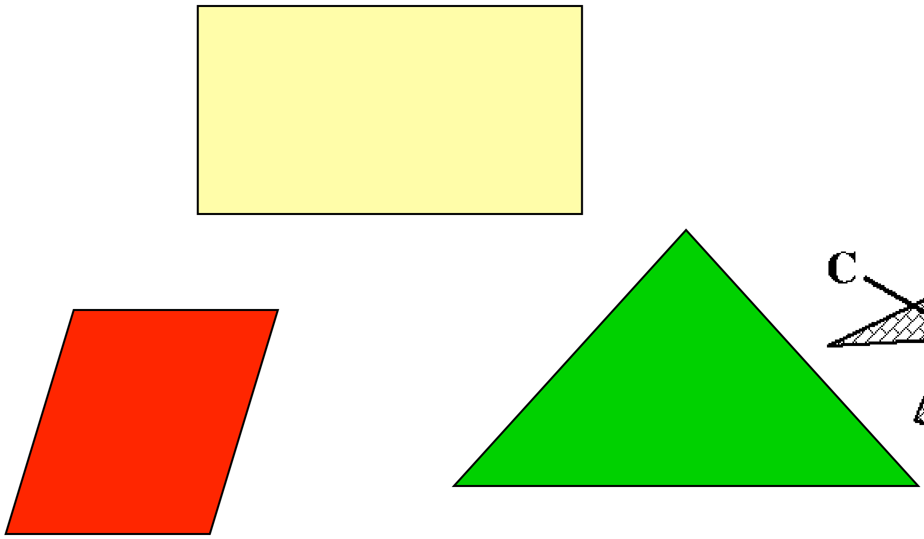
Polyline Attributes

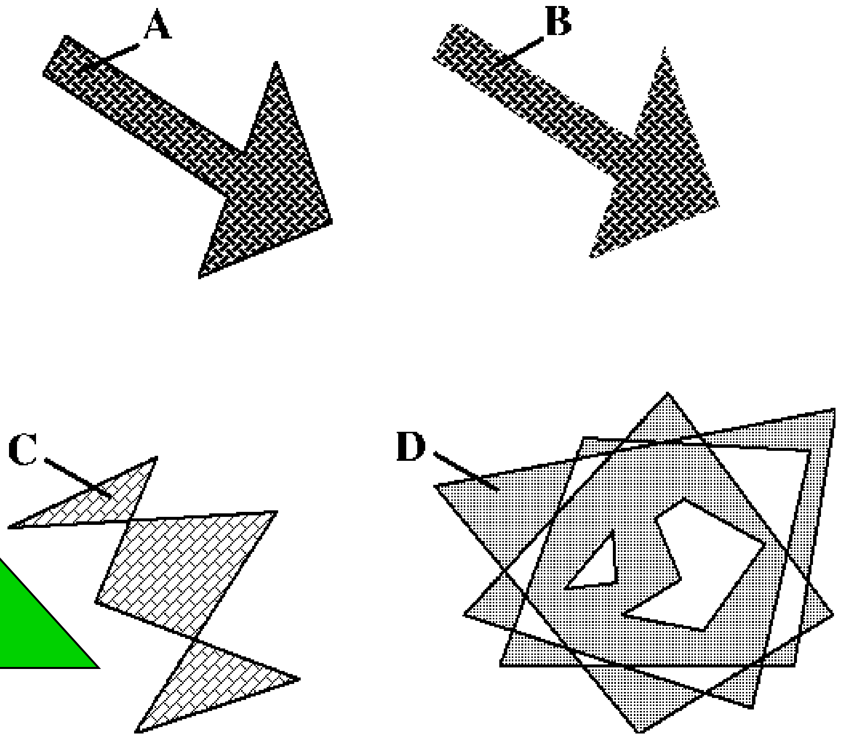
- Color
- Thickness
- Stippling of edges (dash pattern)



Filled Regions

- **Filled region:** shape filled with some color or pattern
- Example: polygons

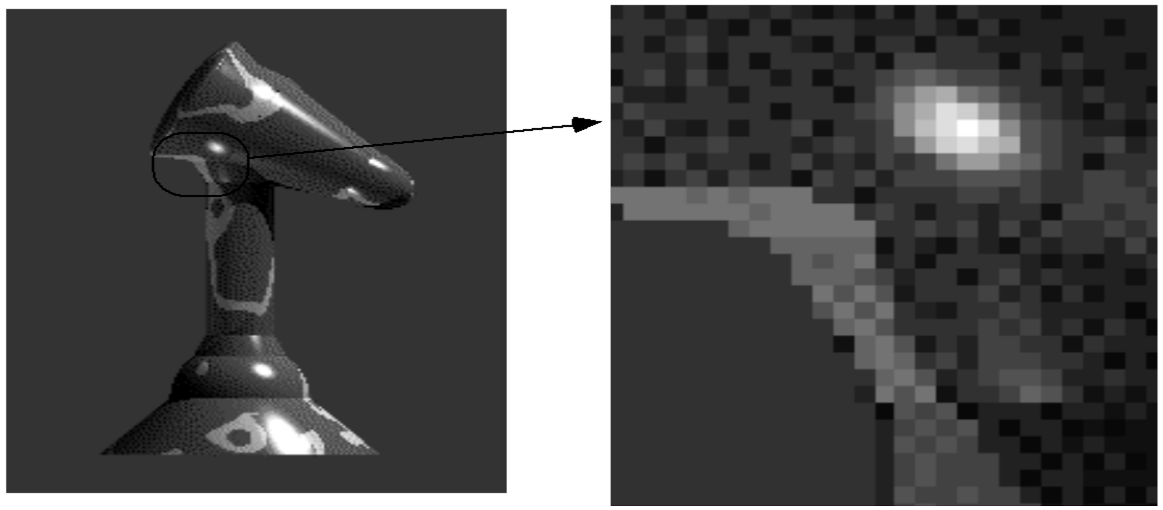




Raster Images

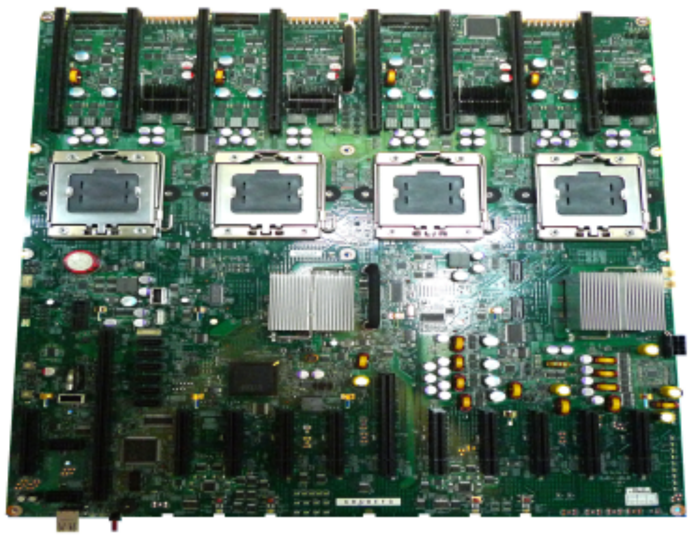
- Raster image (picture) consists of 2D matrix of small cells (pixels, for "picture elements"), in different colors or grayscale.

  Middle image: magnified showing pixels (squares)



Graphics Processing Unit (GPU)

- OpenGL implemented in hardware => FAST!!
- **Programmable:** As shaders
- Located either on PC motherboard (Intel) or Separate graphics card (nVidia or AMD/ATI)


GPU on PC motherboard

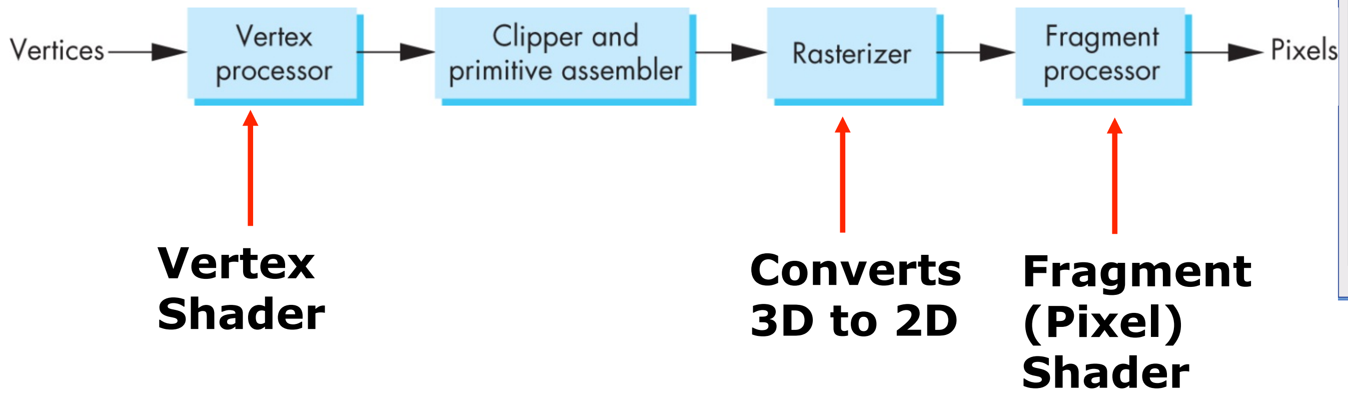
GPU on separate PCI express card

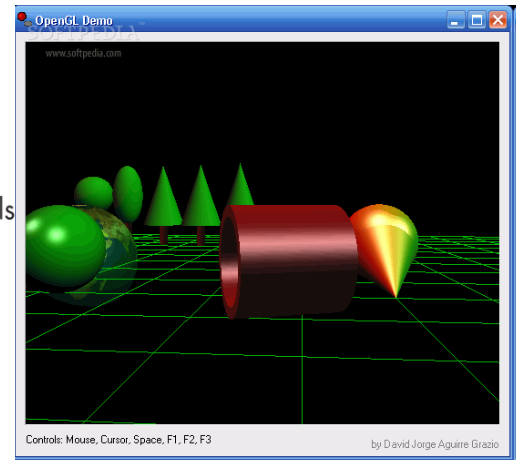
Computer Graphics Libraries

- Functions to a draw line, circle, image, etc.
- Previously device-dependent
  - Different OS => different graphics library
  - Tedious! Difficult to port (e.g. move program Windows to Linux)
  - Error Prone
- Now device-independent libraries
  - **APIs:** OpenGL, DirectX
  - Working OpenGL program minimal changes to move from Windows to Linux, etc.
- Now even more!
  - **Browser as app:** WebGL

Simplified WebGL/OpenGL Pipeline

- Vertices go in, sequence of steps (vertex processor, clipper, rasterizer, fragment processor) image rendered
- **This class:** learn algorithms and order of these steps



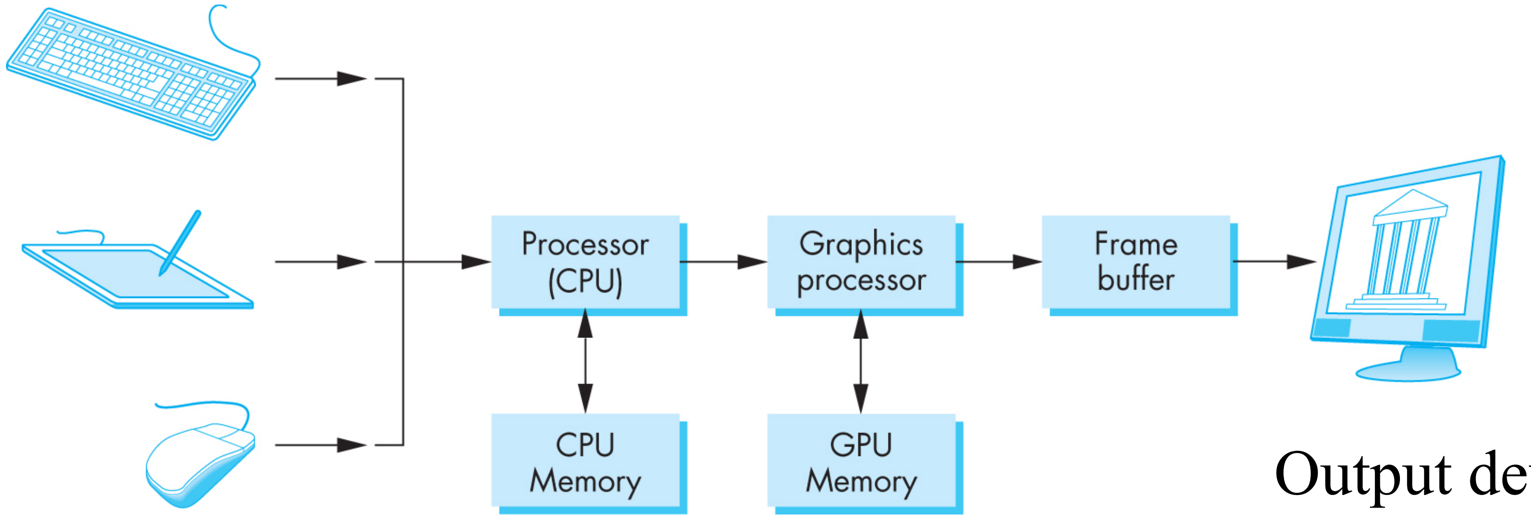


OpenGL Demo
Controls: Mouse, Cursor, Space, F1, F2, F3
by David Jorge Aguirre Grazio

WPI

OpenGL Programming Interface

- Programmer view of OpenGL?
  - Application Programmer Interface (API)
  - Writes OpenGL Application programs



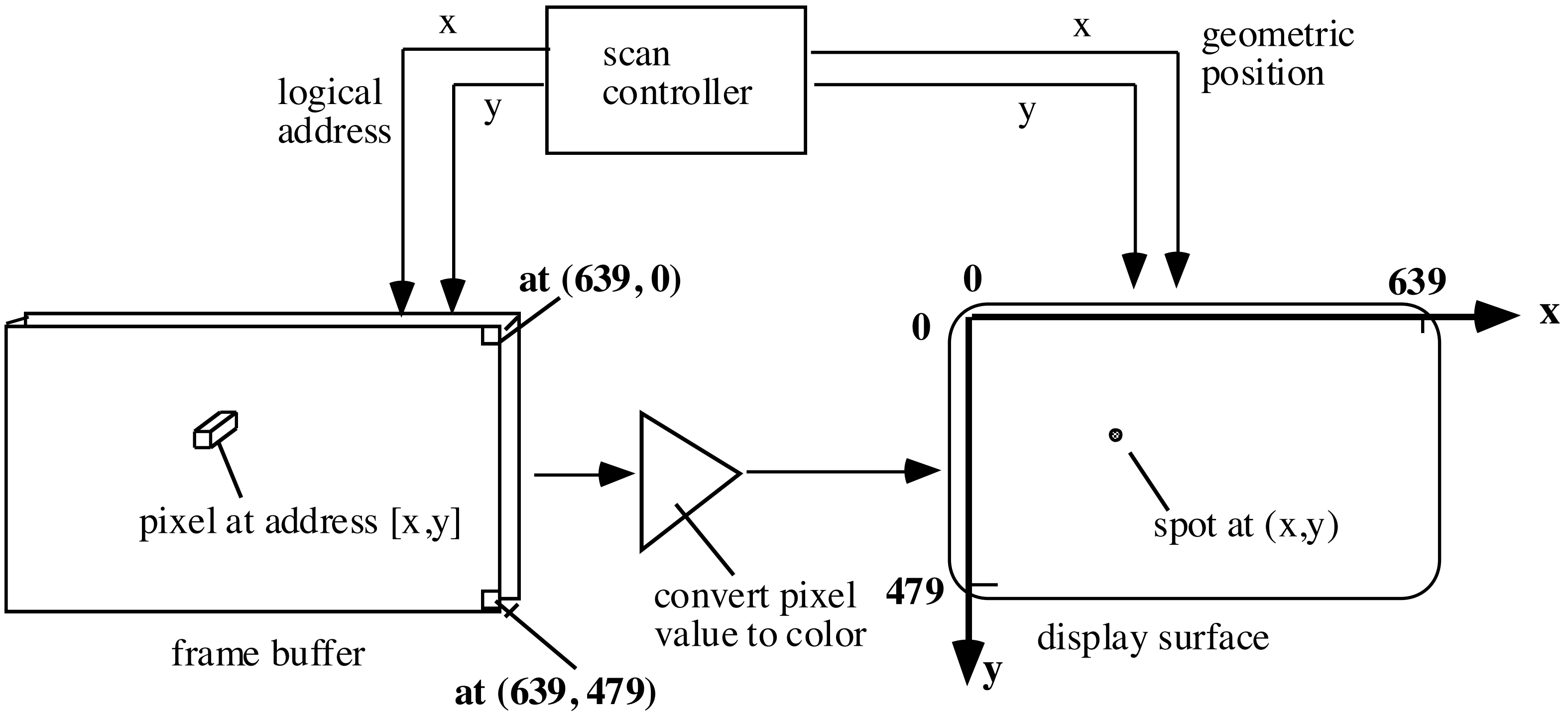
Output device

Input devices

Image formed in frame buffer

Framebuffer

- Dedicated memory location:
  - Draw in framebuffer => shows up on screen
  - Located either on CPU (software) or GPU (hardware)



References

- Angel & Shreiner, Interactive Computer Graphics (7th edition), Chapter 1