# CENG 477
# Introduction to Computer Graphics

Fall 2007-2008

# Instructors & TA

- Section 01 - Veysi Isler (isler@ceng) B-208
- Section 02 - Tolga Can (tcan@ceng) B-109

- TAs
  - Burcin Sapaz (burcin@ceng) B-204
  - Hande Celikkanat (hande@ceng) A-301

# Class Web Page & Newsgroup

http://www.ceng.metu.edu.tr/courses/ceng477/

- Lecture slides posted
- Syllabus
- Tutorials on OpenGL
- OpenGL, GLUT installation files
- Project related documents

metu.ceng.course.477

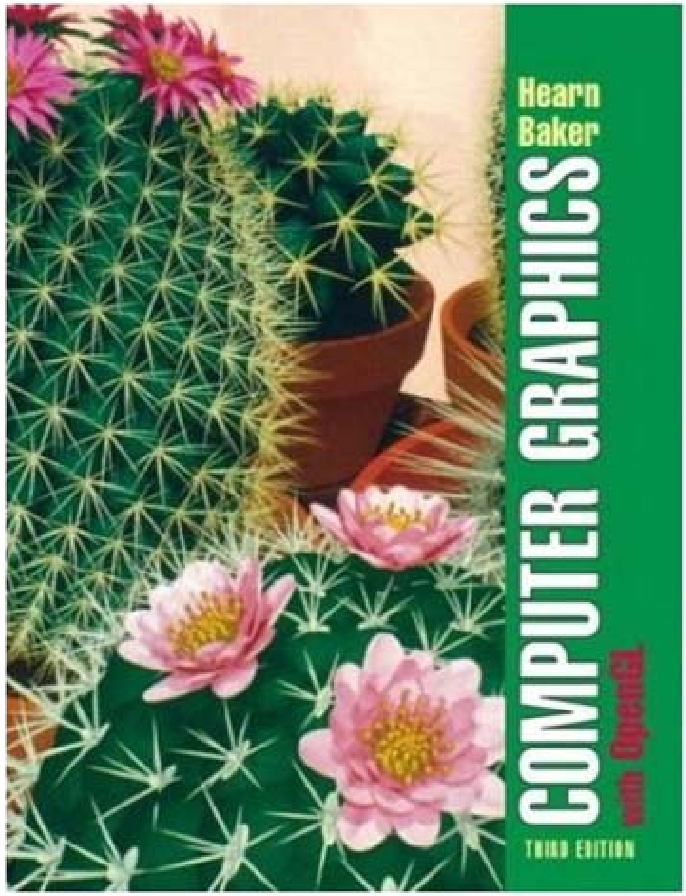
- You should follow the newsgroup on a daily basis for announcements and discussions

# Prerequisites

- C/C++ programming: For the project, you will have to do a lot of programming using advanced data structures
- Basic linear algebra and analytic geometry: You will learn that Computer Graphics involves a lot of mathematics.

# The textbook

- D. Hearn, M.P. Baker, "Computer Graphics with OpenGL", 3rd Edition, Prentice Hall, 2004, ISBN 0-13-015390-7
- Available at the bookstore



# Grading

- Term project : 35%
- Warm-up homework : 5%
- Quizzes: 10%
- Midterm: 20%
- Final: 30%

# Project

- 3 phase project with a main theme that involves 2D and 3D components.
- Project details will be announced next 1-2 weeks.
- Warm-up
  - Prepare your CG development environment
  - Install OpenGL and GLUT
  - You may use standard C/C++ compilers like gcc/g++ to compile and link your programs.
  - If you use MS Visual C/C++, **do not use** MS Visual C/C++ specific directives as your programs will be compiled and tested under Linux.

# Computer Graphics History, Hardware and Software, and Applications

# What is Computer Graphics?

- Different things in different contexts:
  - pictures, scenes that are generated by a computer.
  - tools used to make such pictures, software and hardware, input/output devices.
  - the whole field of study that involves these tools and the pictures they produce.
- Use of computer to define, store, manipulate, interrogate and present pictorial output.

# Another definition

- Computer graphics: generating 2D images of a 3D world represented in a computer.
- Main tasks:
  - *modeling*: creating and representing the geometry of objects in the 3D world
  - *rendering*: generating 2D images of the objects
  - *animation*: describing how objects change in time

# Involves

- How pictures are represented in computer graphics,
- How pictures are prepared for presentation,
- How interaction within the picture is accomplished.

# Computer Graphics Applications

- Art, entertainment, and publishing
  - movie production, animation, special effects
  - computer games
  - World Wide Web
  - Book, magazine design, photo editing
- Simulations (education, training)
- CAD architectural, circuit design etc.
- Scientific analysis and visualization
- Graphical User Interfaces
- CG versus Computer Vision (syntesis vs. analysis)

# Graphics Applications

- Entertainment: Movies

Pixar: Monster's Inc.

Square: Final Fantasy

# Entertainment

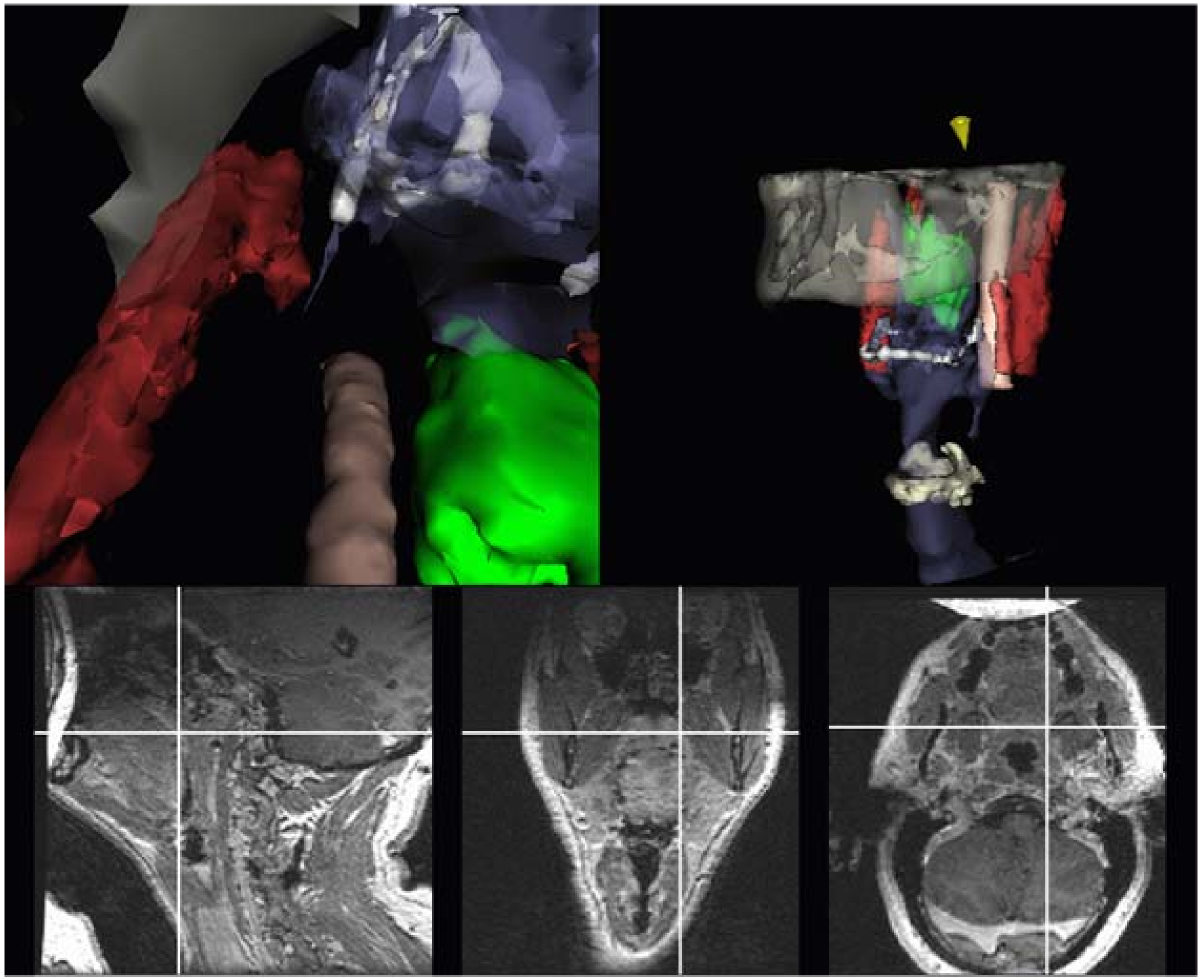

Final Fantasy *(Square, USA)*

# Entertainment

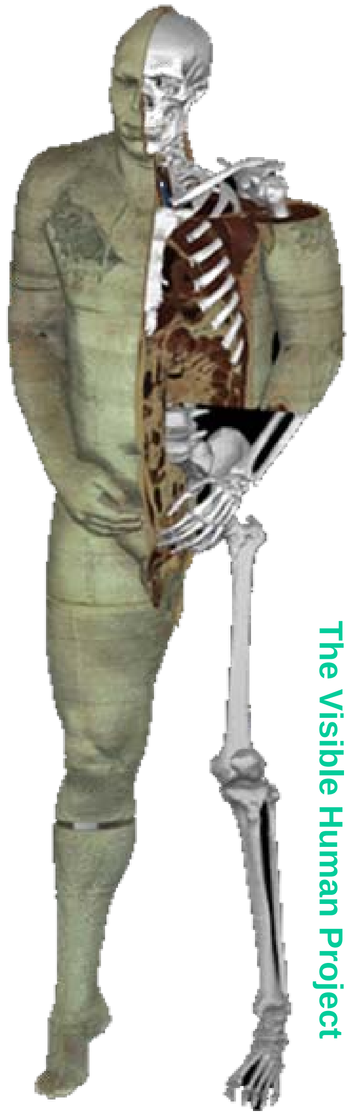
A Bug’s Life *(Pixar)*

# Graphics Applications

- Medical Visualization

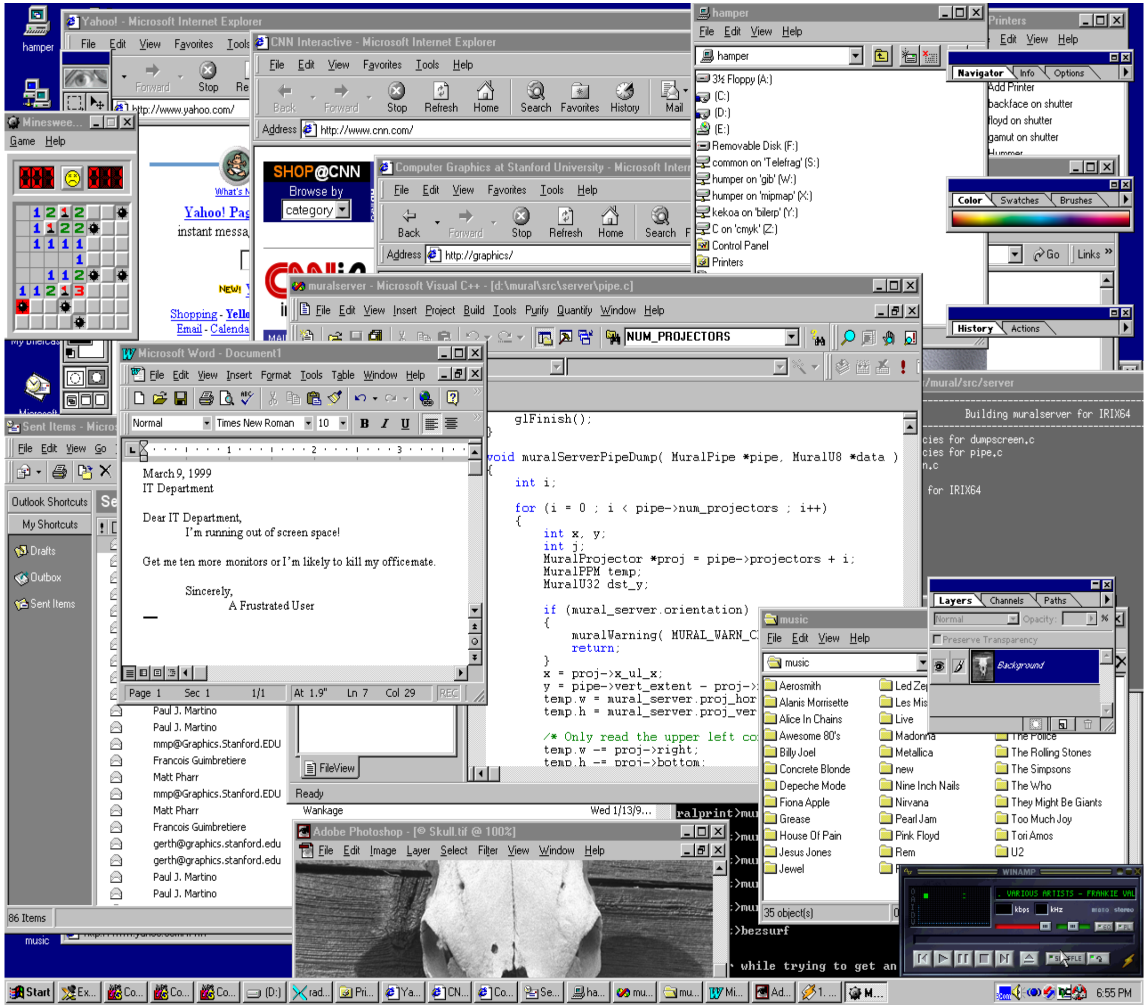


MIT: Image-Guided Surgery Project



The Visible Human Project

# Everyday use

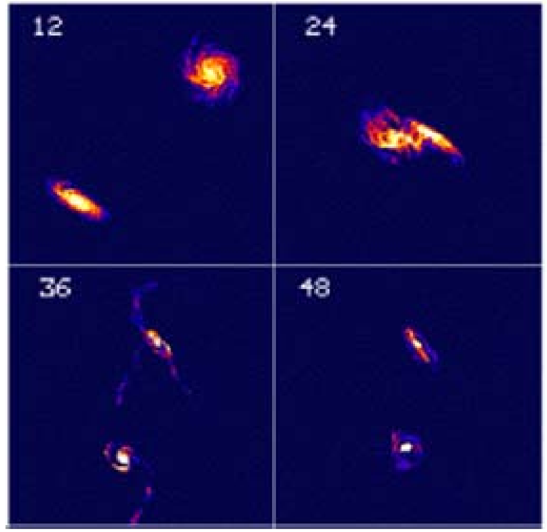


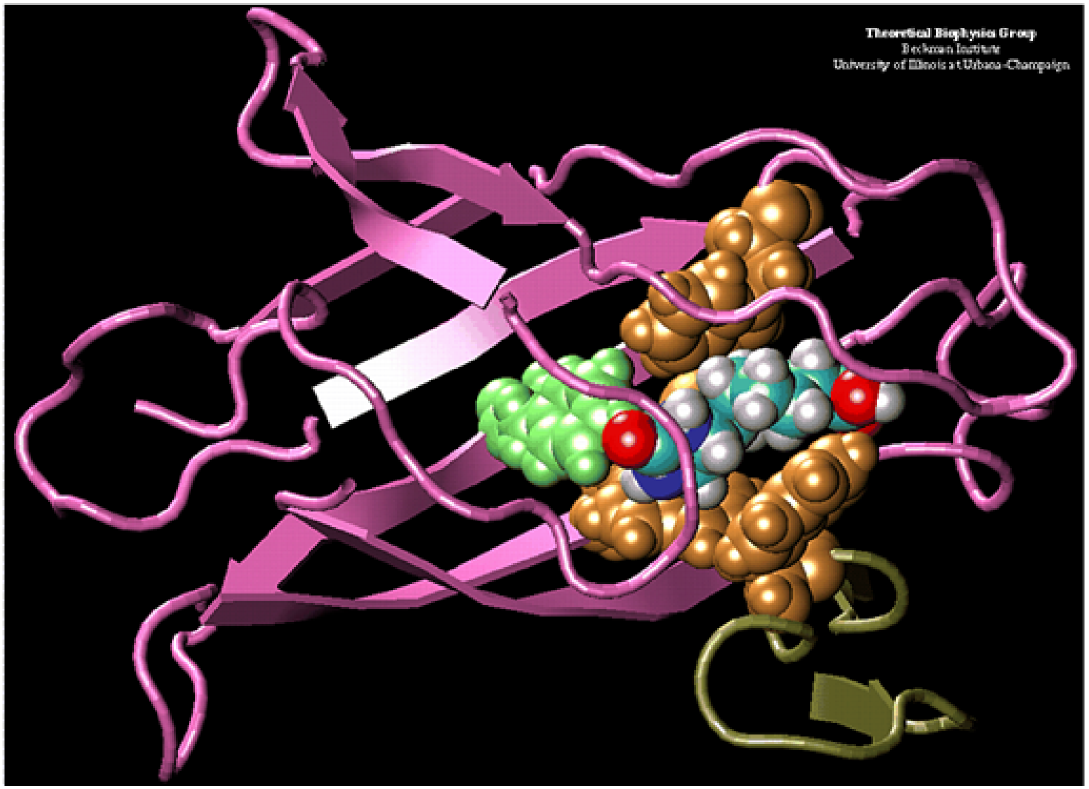
# Everyday use

Window system and large-screen interaction metaphors *(François Guimbretière)*

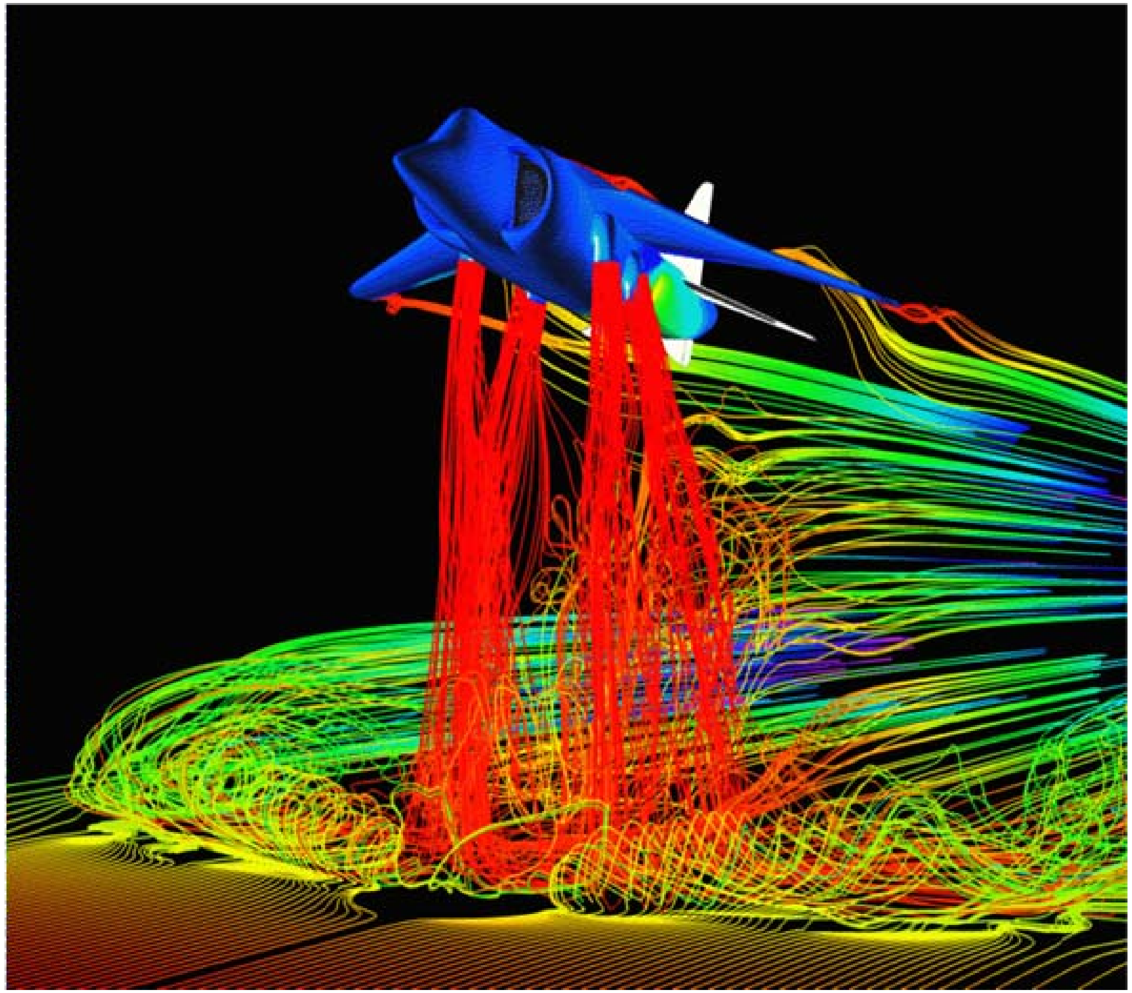
# Graphics Applications

- Scientific Visualization



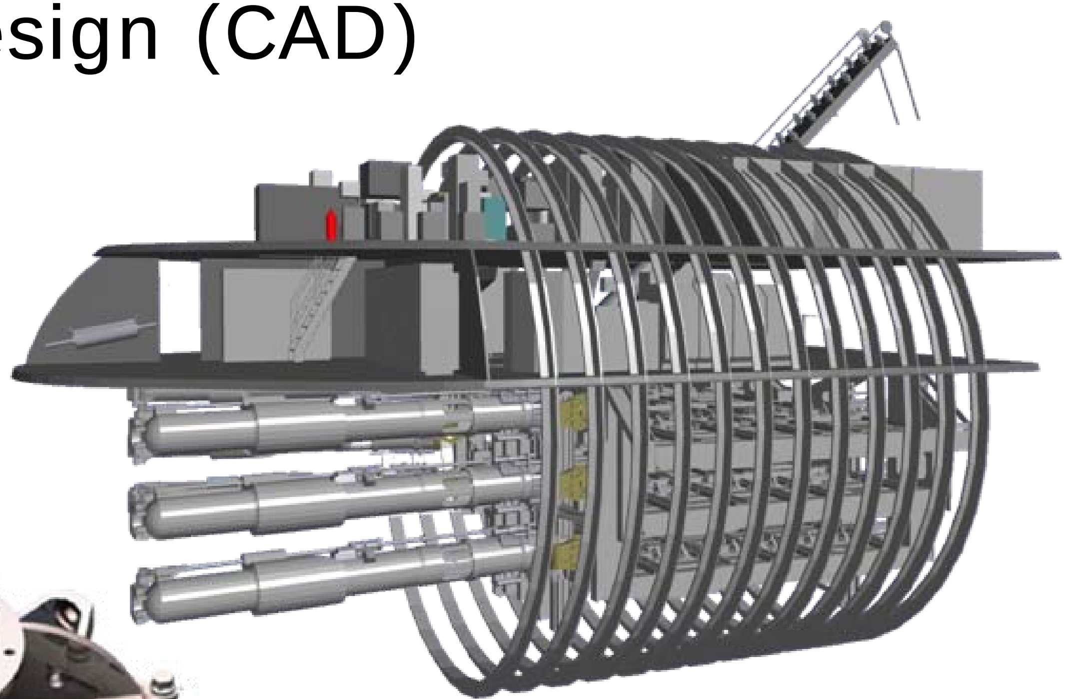
# Scientific Visualization

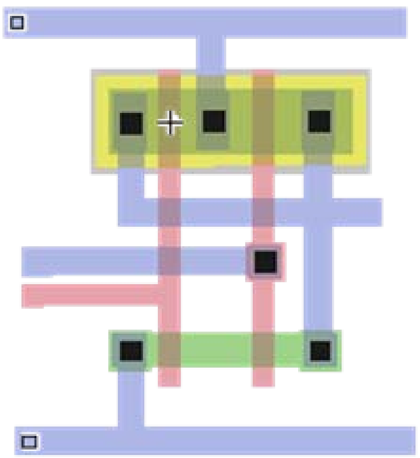


Airflow around a Harrier Jet *(NASA Ames)*

# Graphics Applications

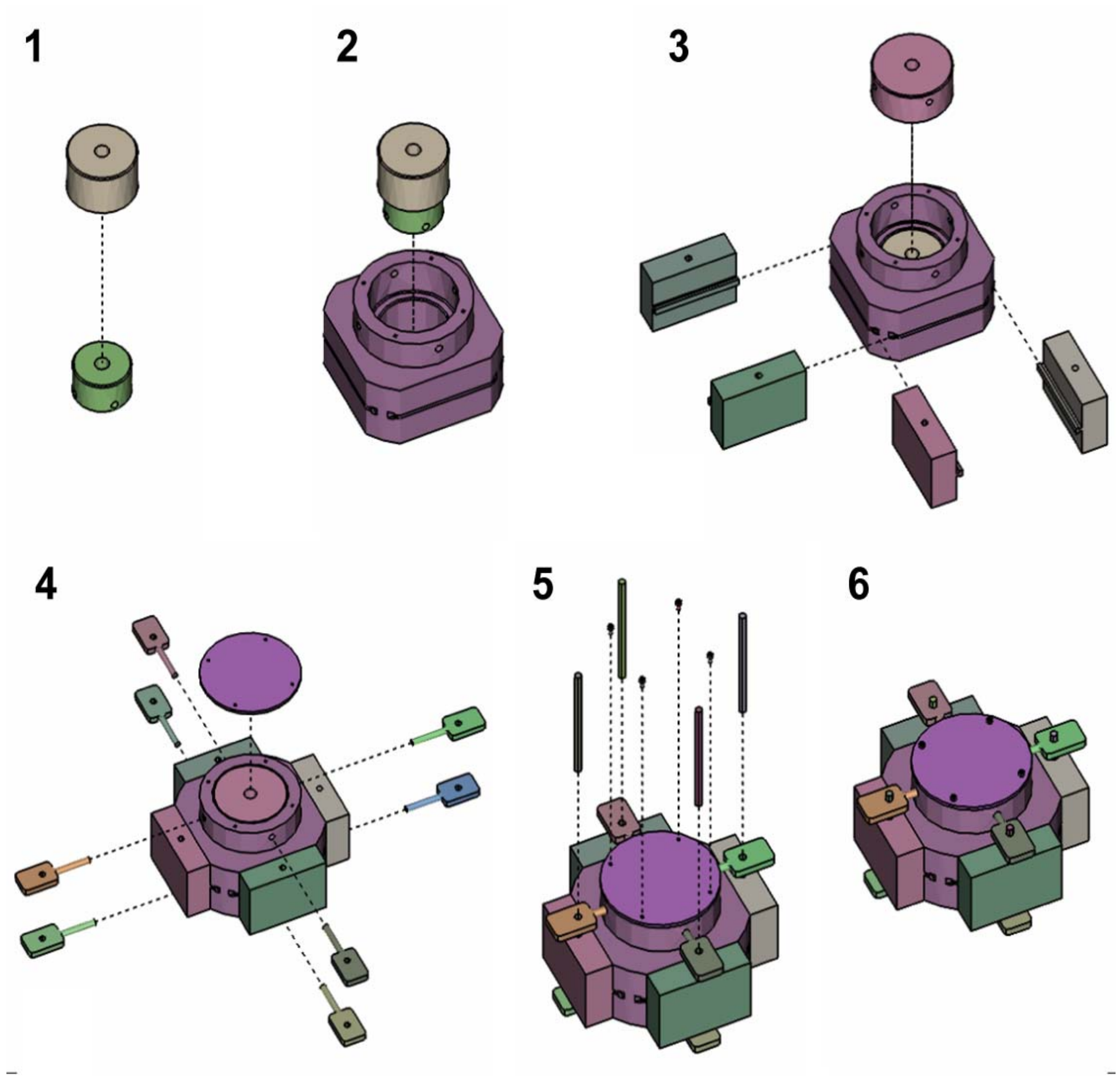
- Computer Aided Design (CAD)





# Graphics Applications

- Training




Designing Effective Step-By-Step Assembly Instructions *(Maneesh Agrawala et. al)*

# Graphics Applications

- Entertainment: Games



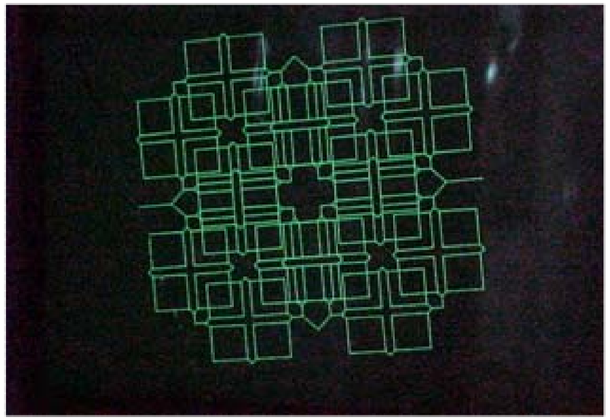
Polyphony Digital: Gran Turismo 3, A Spec

# Training

View from the ship’s bridge in the virtual environment at Dalian Maritime University.*(Courtesy Xie Cui )*

# Short History of Computer Graphics

# Early 60's:



- Computer animations for physical simulation; Edward Zajac displays satellite research using CG in 1961
- 1963: Sutherland (MIT)
  Sketchpad (direct manipulation, CAD)
  Calligraphics (vector) display devices
  Interactive techniques
  First mouse (Douglas Englebart)
- 1968: Evans & Sutherland founded
- 1969: First SIGGRAPH

# Late 60's to late 70's:

- Utah Dynasty
  - 1970: Pierre Bezier develops Bezier curves
  - 1971: Gouraud Shading
  - 1972: Pong (first computer game) developed
  - 1973: Westworld, the first film to use computer animation
  - 1974: Ed Catmull develops z-buffer (Utah)
    First Computer Animated Short, *Hunger*.
    Keyframe animation and morphing.
  - 1975: Bui-Toung Phong creates Phong Shading (Utah)
    Martin Newel models a 3D teapot with Bezier patches (Utah)

# Mid 70's -80's:


- Quest for realism. Radiosity shading; mainstream real-time applications.
  - 1982: Tron, Wrath of Kahn. Particle systems and obvious CG.
  - 1984: The Last Star Figher, CG replaces physical models. Early attempts at realism using CG.
  - 1986: First CG animation nominated for and Academy Award: Luxo Jr. (Pixar)
  - 1989: Tin Toy (Pixar) wins Academy Award.

– 1995: Toy Story (Pixar/Disney), the first full length fully computer generated 3D animation. The first fully 3D CG cartoon Babylon 5. First TV show routinely using CG models.

# Late 90's:

- Interactive environments, scientific and medical visualization, artistic rendering, image based rendering, path tracing, photon maps, etc.

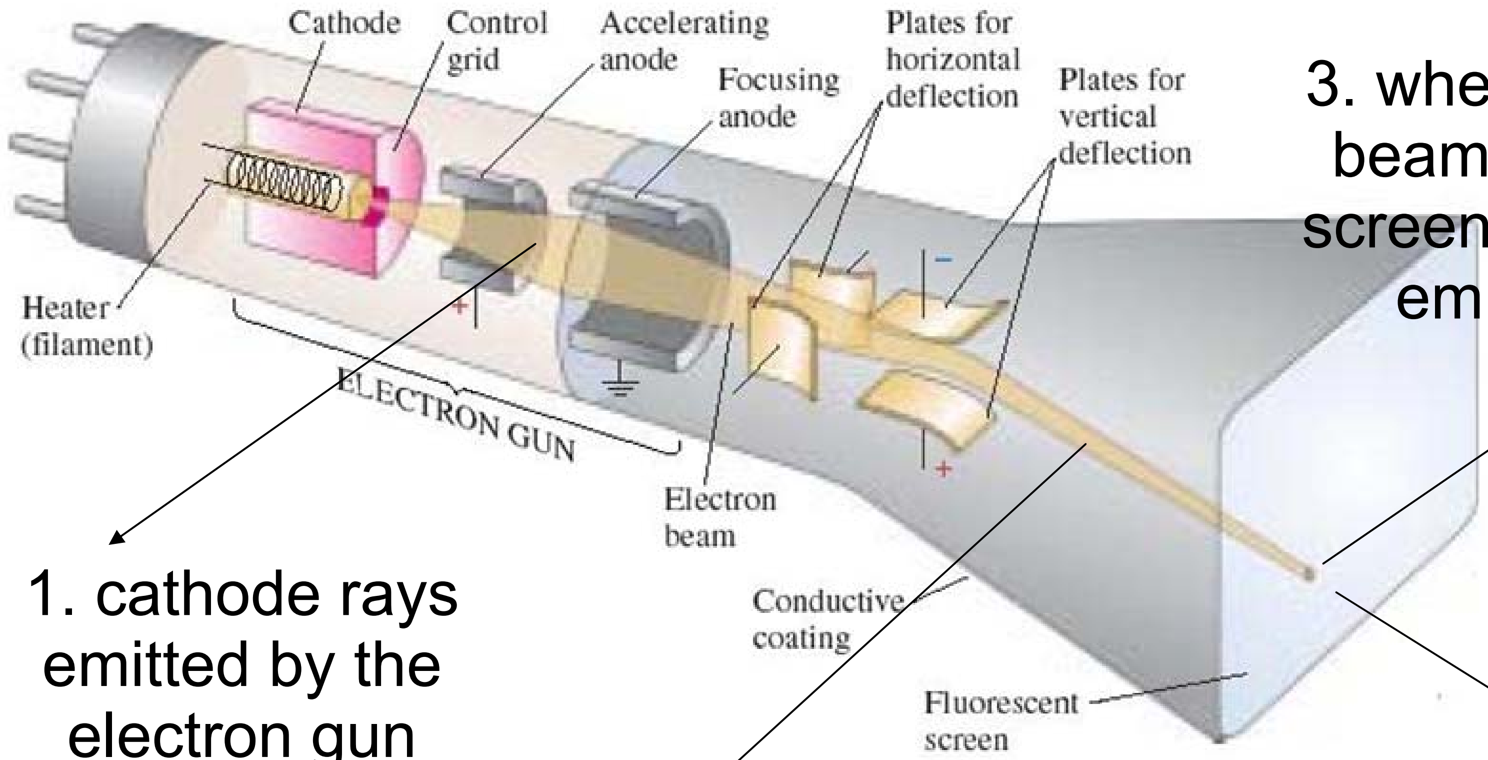
# 2000's:

- Real-time photorealistic rendering on consumer HW? Interactively rendered movies? Ubiquitous computing, computer vision and graphics.

# Display (Video Display Device)

- Most CG on video monitors
- Still most popular: Cathode Ray Tube (CRT)
- Other popular display types:
  - Liquid Crystal Display
  - Plasma display
  - Field Emission Displays
  - Digital Micromirror Devices
  - Light Emitting Diodes
  - 3D display devices (hologram or page scan methods)

# CRT




1. cathode rays emitted by the electron gun

2. focusing and deflection

3. when electron beams contact screen phosphor emits light

4. light fades, redraw required in a small period (refresh)

# CRT types

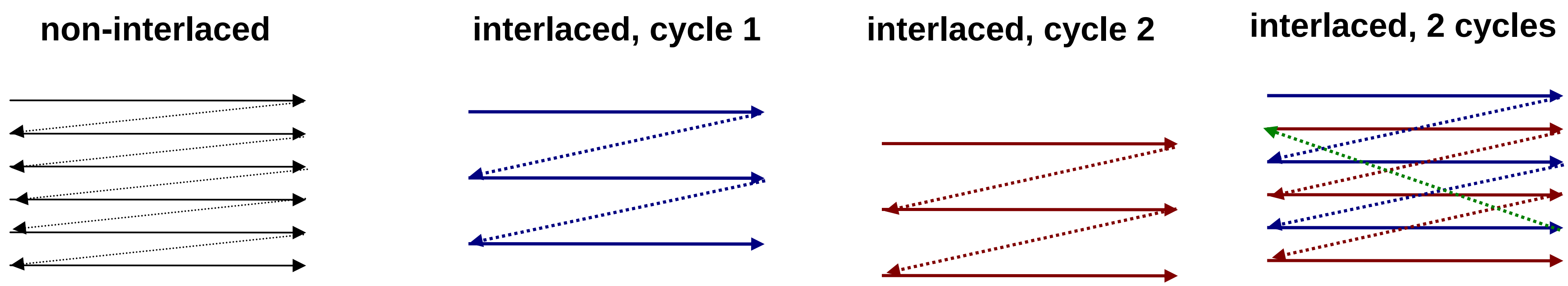
- Direct View Storage Tubes (not CRT, no need for refresh, pictures stored as a permanent charge on phosphor screen)
- Calligraphic refresh CRT (line drawing or vector random scan, need refreshing)
- Raster-scan (point by point refreshing)
- **Refresh rate:** # of complete images (frames) drawn on the screen in 1 second. Frames/sec.
- **Frame time:** reciprocal of the refresh rate, time between each complete scan. sec/frame

# Vector Scan

- Also referred to as Random-Scan Displays
- Picture definition is stored as a set of line-drawing commands in a refresh buffer.
- to display a picture, the system cycles through the set of commands in the buffer
- Designed for line drawing applications (CAD)

# Raster Scan

- Screen is a regular grid of samples called **pixels (picture element)**
- **Screen is refreshed line by line**




- **Interlacing: Avoid flickering affect for small refresh rates.**
  **interlaced 50Hz: actually 25Hz**

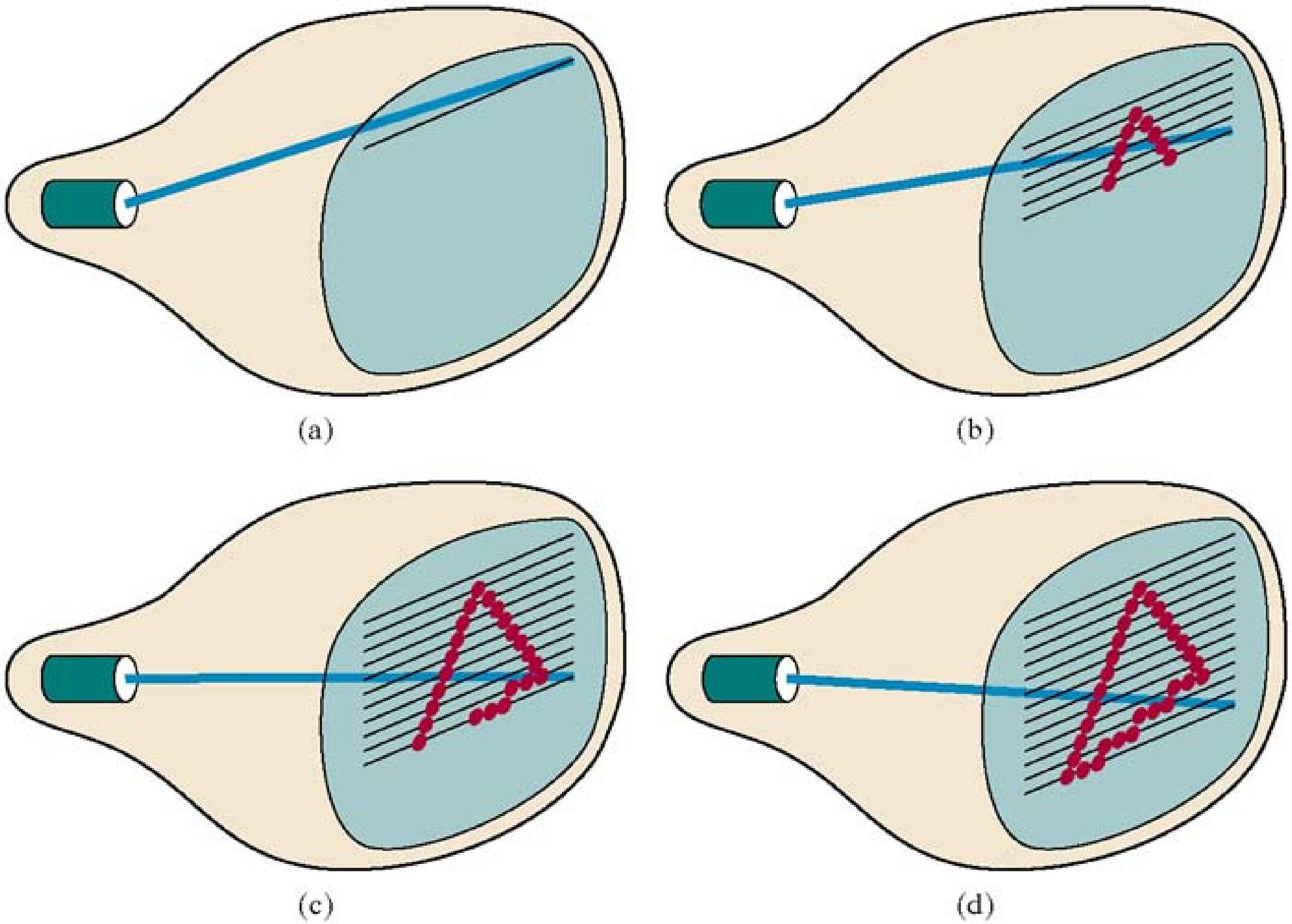


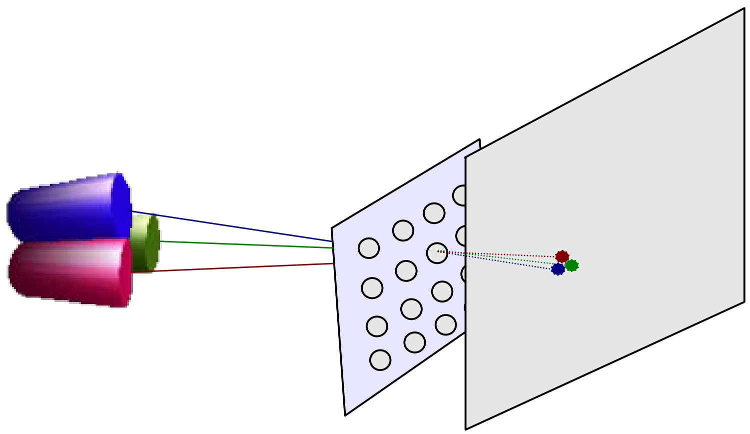

Figure 2-7
A raster-scan system displays an object as a set of discrete points across each scan line.

*Computer Graphics with Open GL*, Third Edition, by Donald Hearn and M.Pauline Baker.

- **resolution: a 2D term that measures the number of scan-lines and the number of pixels on each line (maximum number of points that can be displayed without overlap on a CRT)**
- **black and white display only binary pixels.**
- **intensity of a pixel can be achieved by the force of electron beam (gray scale)**
- **color display?**

# Color Displays

- Beam penetration method:
  special phosphors emitting different colors for different intensity of electron. Slow, limited colors.
- Shadow mask method:
  3 electron guns + a shadow mask grid. Intensities of 3 colors result in an arbitrary color pixel. (most TVs and monitors)



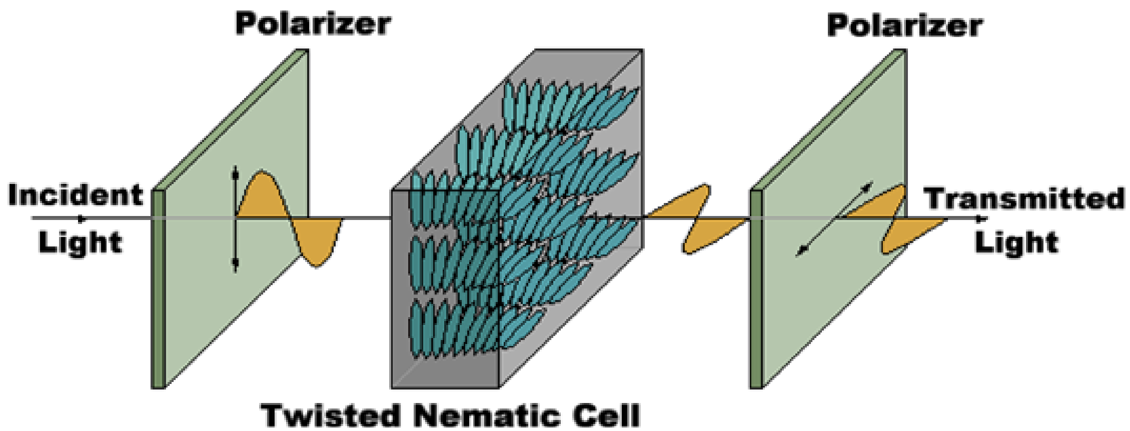
- black and white: 1 bit per pixel.
- gray scale: 1 byte per pixel (256 gray levels)
- true color: 3 bytes=24pits per pixel ($2^{24}$ colors)
- indexed color frame buffer: each pixel uses 1 byte, an index entry in a colormap table matching the color to the actual color.

# Vector vs. Raster Scan

- raster scan monitors:
  - inexpensive
  - filled areas, patterns
  - refresh process is independent (constant for any complex scene)
- vector scan monitors:
  - Smooth lines. no need for scan conversion: lines to pixels. (raster scan solution antialiasing)
  - sometimes memory and CPU efficient 1000 lines: Vector scan: 2000 endpoints and 1000 operations Raster scan: whole frame buffer 1000 scan conversions.

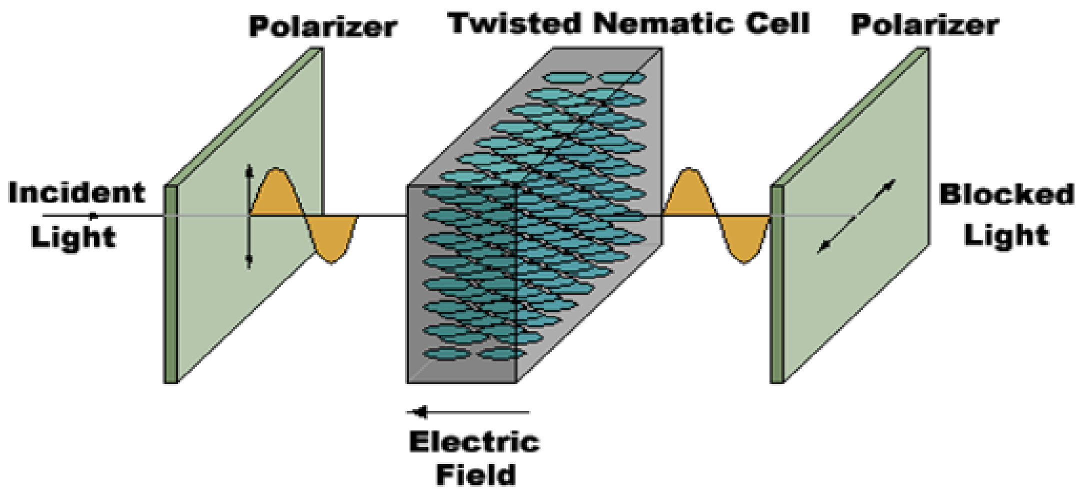
# LCD Displays

- Thinner and lighter. No tube or electron beams.
- Blocking/unblocking light through polarized crystals. Crystals liquefy when excited by heat or E field.
- A matrix of LC cells one for each pixel.
- No refresh unless the screen changes.
- Color 3 cells per pixel.



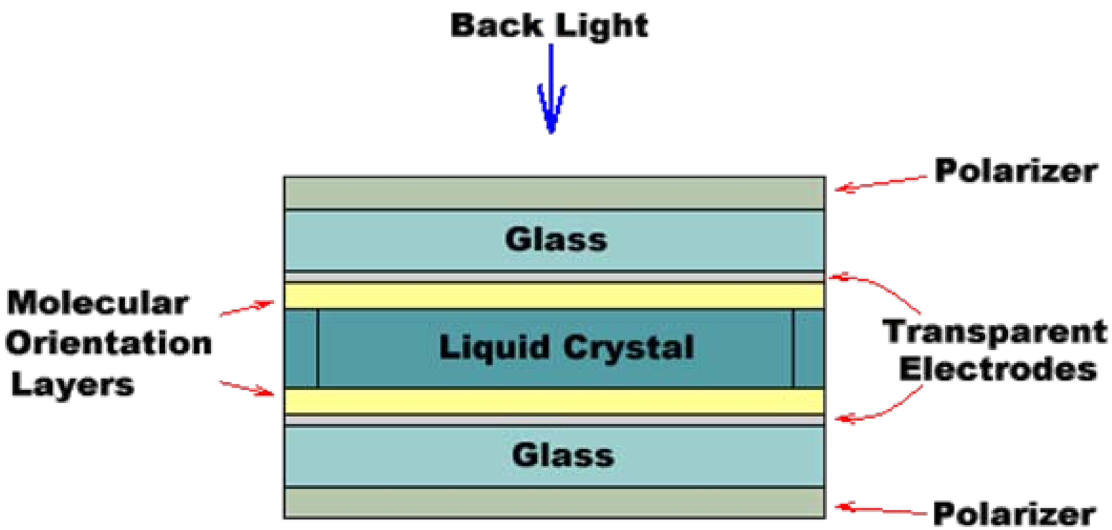
# LCD Displays

- Thinner and lighter. No tube or electron beams.
- Blocking/unblocking light through polarized crystals. Crystals liquefy when excited by heat or E field.
- A matrix of LC cells one for each pixel.
- No refresh unless the screen changes.
- Color 3 cells per pixel.



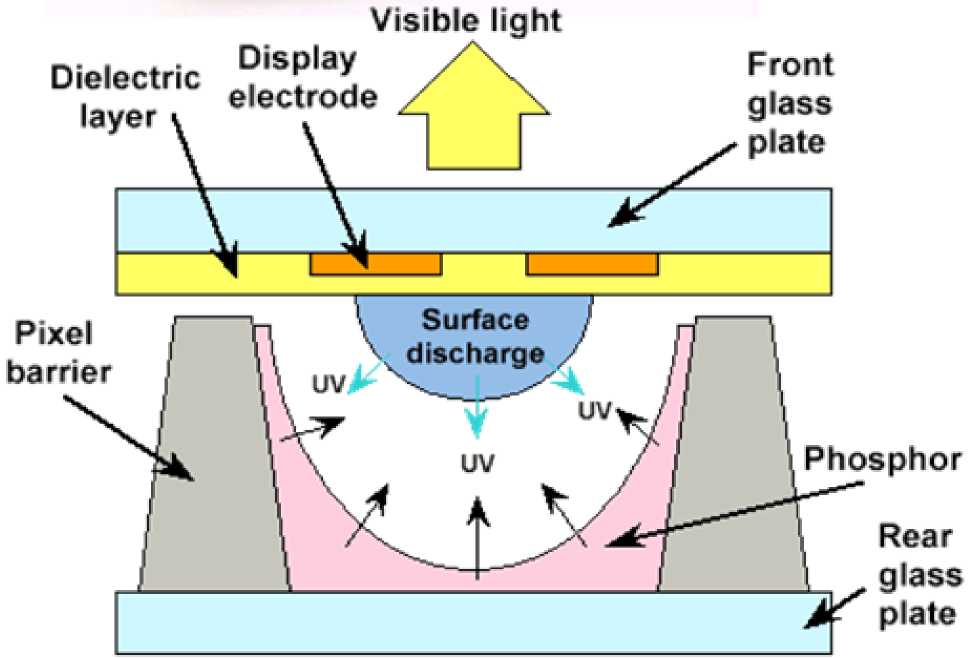
# LCD Types

- Transmissive & reflective LCDs:
  - LCDs act as light valves, not light emitters, and thus rely on an external light source.
  - Laptop screen: backlit, *transmissive display*
  - Palm Pilot/Game Boy: *reflective display*



# Plasma Displays

- Plasma display panels
  - Similar in principle to fluorescent light tubes
  - Small gas-filled capsules are excited by electric field, emits UV light
  - UV excites phosphor
  - Phosphor relaxes, emits some other color

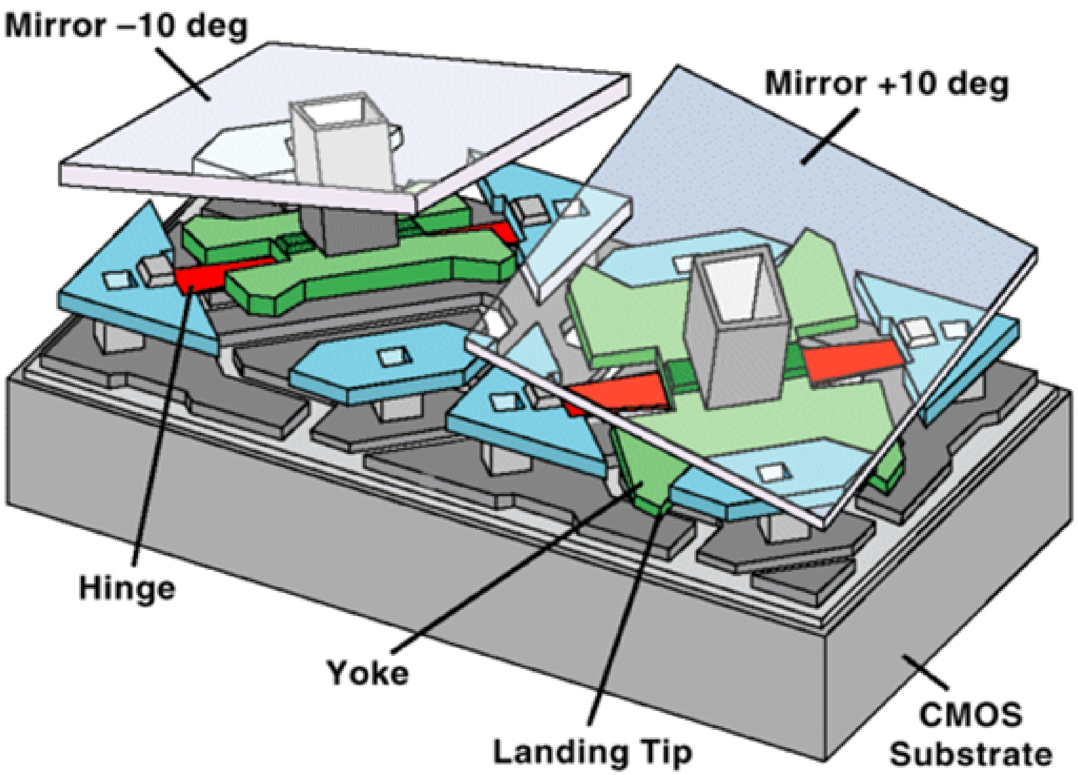


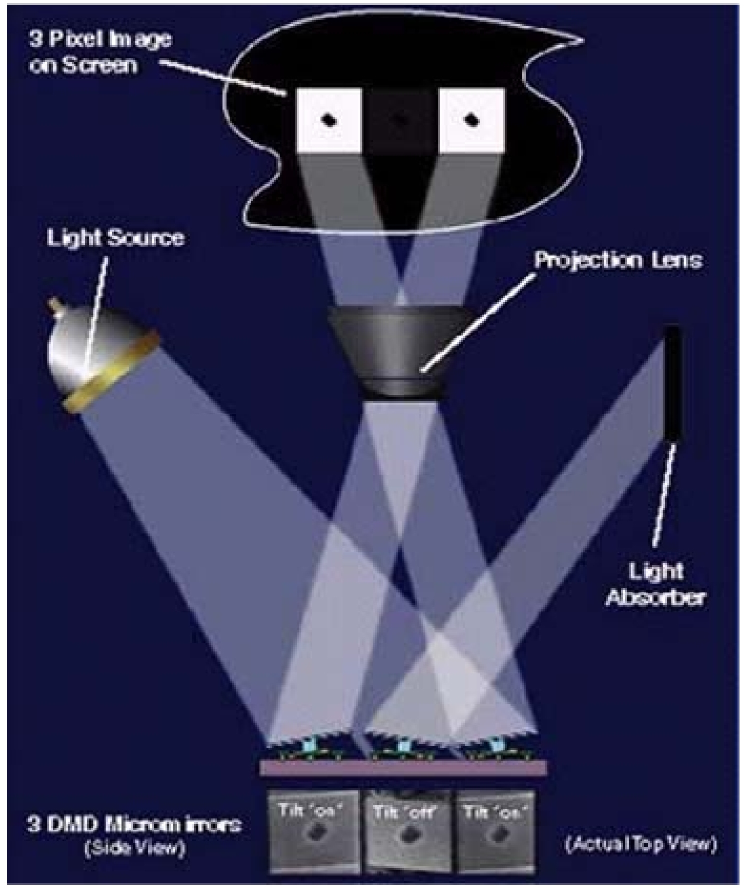
# Plasma Displays

- Plasma Display Panel Pros
  - Large viewing angle
  - Good for large-format displays
  - Fairly bright
- Cons
  - Expensive
  - Large pixels (~1 mm versus ~0.2 mm)
  - Phosphors gradually deplete
  - Less bright than CRTs, using more power

# Display Technology: DMD / DLP

- Digital Micromirror Devices (projectors) or Digital Light Processing
  - Microelectromechanical (MEM) devices, fabricated with VLSI techniques



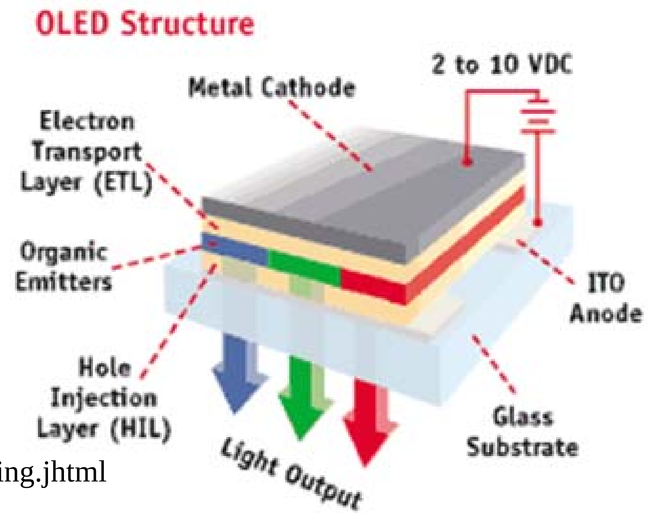



# Display Technology: DMD / DLP

- DMDs are truly digital pixels
- Vary grey levels by modulating pulse length
- Color: multiple chips, or color-wheel
- Great resolution
- Very bright
- Flicker problems

# Display Technologies: Organic LED Arrays

- Organic Light-Emitting Diode (OLED) Arrays
  - The display of the future? Many think so.
  - OLEDs function like regular semiconductor LEDs
  - But they emit light
    - Thin-film deposition of organic, light-emitting molecules through vapor sublimation in a vacuum.
    - Dope emissive layers with fluorescent molecules to create color.



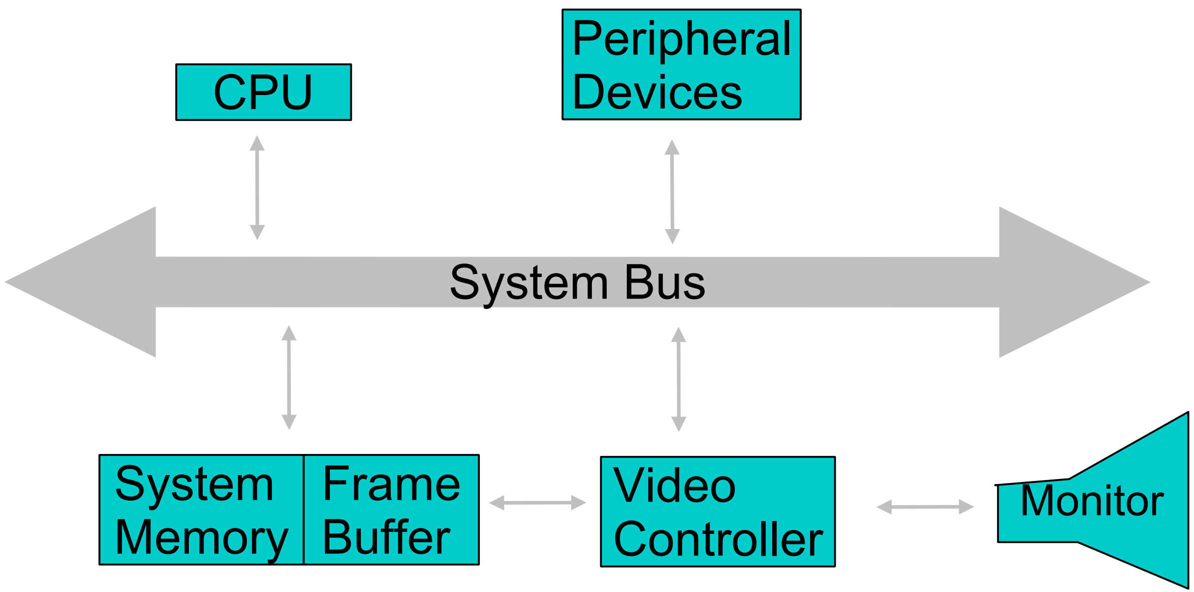

http://www.kodak.com/global/en/professional/products/specialProducts/OEL/creating.jhtml

# Display Technologies: Organic LED Arrays

- OLED pros:
  - Transparent
  - Flexible
  - Light-emitting, and quite bright (daylight visible)
  - Large viewing angle
  - Fast (< 1 microsecond off-on-off)
  - Can be made large or small
  - Available for cell phones and car stereos
- OLED cons:
  - Not very robust, display lifetime a key issue
  - Currently only passive matrix displays
    - Passive matrix: Pixels are illuminated in scanline order, but the lack of phospherescence causes flicker
    - Active matrix: A polysilicate layer provides thin film transistors at each pixel, allowing direct pixel access and constant illum.

# Simple Raster Display System

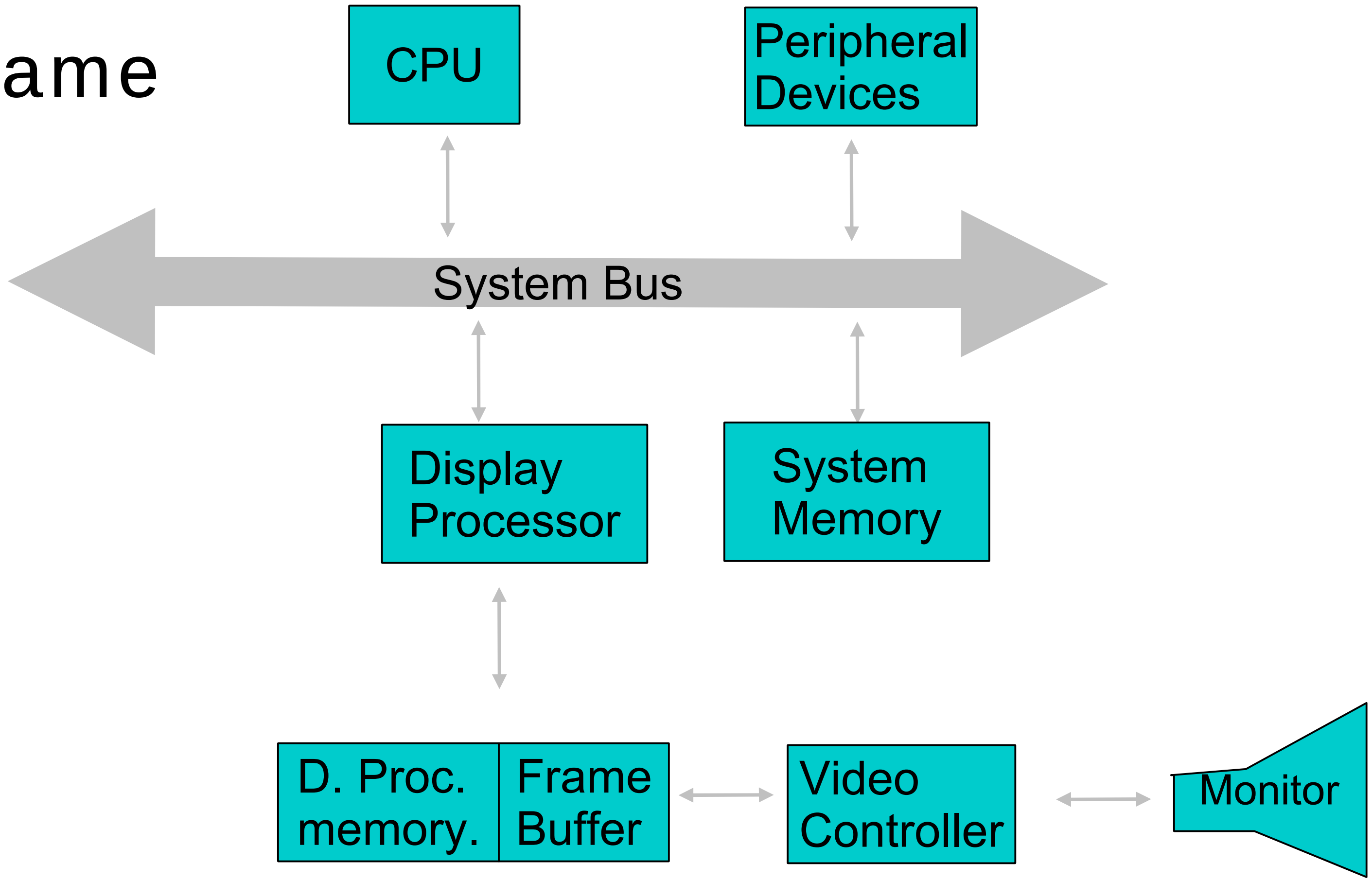
- Frame buffer: stored pixel map of screen
- Video controller just refreshes the frame buffer on the monitor periodically.



- Inexpensive
- Scan conversion of output primitives (lines, rectangles etc.) done by the CPU. Slow.
- As refresh cycle increases, memory cycles used by the video controller increases. Memory is less available to CPU.
- Solution: Graphics Display Processor

# Graphics Display Processor

- Scan conversion, output primitives, raster operations (double buffering)
- Separete frame buffer



# Computer Graphics Software

- Rendering Primitives
  - Models are composed of, or can be converted to, a large number of **geometric primitives**.
  - Typical rendering primitives directly supported in hardware include:
    - Points (single pixels)
    - Line segments
    - Polygons (perhaps simple, triangle, rectangle)

- Modeling primitives include these, but also
  - Piecewise polynomial (spline) curves
  - Piecewise polynomial (spline) surfaces
  - Implicit surfaces (quadrics, blobbies, etc.)
  - Other…
- Software renderer may support modeling primitives directly, or may convert them into polygonal or linear approximations for hardware rendering

# Algorithms

- A number of basic algorithms are needed:
  - **Transformation:** Convert representations of models/primitives from one coordinate system to another
  - **Clipping / Hidden surface removal:** remove primitives and part of primitives that are not visible on the display
  - **Rasterization:** Convert a projected screen space primitive to a set of pixels.

- Advanced algorithms:
  - **Picking:** select a 3D obejct by clicking an input device over a pixel location.
  - **Shading and illumination:** Simulate the interaction of light with a scene.
  - **Animation:** Simulate movement by rendering a sequence of frames.

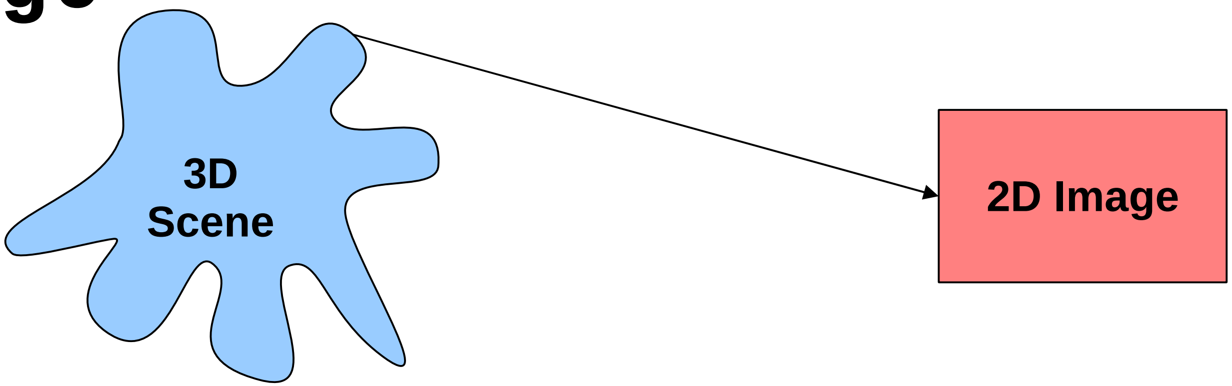
# Application Programming Interfaces

- X11: 2D rasterization
- Postscript, PDF: 2D transformations, 2D rasterization
- Phigs+, GL, OpenGL, Direct3D: 3D pipeline
- APIs provide access to rendering hardware via conceptual model.
- APIs abstract the hardware implementations and algorithms in standard software calls.

- For 3D interactive applications, we might modify the scene or a model directly or just the change the attributes like viewing information.
- We need to interface to input devices in an event-driven, asynchronous and device independent fashion. APIs and toolkits are also defined for this task. GLUT, Qt, GTK, MFC, DirectX, Motif, Tcl/Tk.

# Graphics Rendering Pipeline

- **Rendering:** conversion from **scene** to **image**




- **Scene is represented as a model composed of primitives. Model is generated by a program or input by a user.**
- **Image is drawn on an output device: monitor, printer, memory, file, video frame. Device independence.**

- **Typically rendering process is divided into steps called the graphics pipeline.**
- **Some steps are implemented by graphics hardware.**
- **Programmable graphics accelerator, GPU:**
  **programmable pipelines in graphics hardware**

# The basic forward projection pipeline: