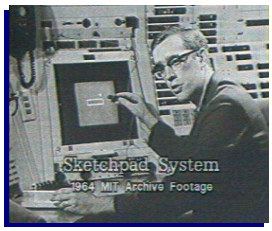
Introduction to Computer Graphics

Chapter 1 Objectives

- To understand the basic objectives and scope of computer graphics
- To identify computer graphics applications
- To understand the basic structures of 2D and 3D graphics systems
- To understand evolution of graphics programming environments
- To identify common graphics APIs
- To understand the roles of Java language, Java 2D and Java 3D packages
- To identify computer graphics related fields

Computer Graphics: A Brief History

- In The Beginning… 1963
  Ivan Sutherland's ***Sketchpad***
- Modified oscilloscope for drawing
- The original CAD system



Graphics Hardware History

Display Hardware
- vector displays
  - 1974 – E&S Picture System
- raster displays
  - 1975 – E&S frame buffer
  - 1980s – cheap frame buffers → bit-mapped personal computers
  - 1990s – liquid-crystal displays → laptops
  - 2000s – micro-mirror projectors → digital cinema
- Other developments
  - stereo, head-mounted displays
  - autostereoscopic displays
  - tactile, haptic, sound

Input Hardware
- 2D
  - light pen, tablet, mouse, joystick, track ball, touch panel…
  - 1970s & 80s - CCD analog image sensor + frame grabber
  - 1990s & 2000's - CMOS digital sensor + in-camera processing → high-X imaging (dynamic range, resolution, depth of field,…)
- 3D
  - 3D trackers, 3D scanners
  - multiple cameras
  - active rangefinders
- other
  - data gloves
  - voice

Computer Graphics from 66,000ft

- User Interaction/GUI
- 2D graphics and image processing
- 3D graphics/modeling
- Animation/Simulation
- Photorealism
- Virtual Reality
- Graphics Hardware
- Graphics programming environments

Example of a JPG image.

2D Graphics

- **Raster:**

  Pixels
  - X11 bitmap, XBM
  - X11 pixmap, XPM
  - GIF
  - TIFF
  - PNG
  - JPG

  Lossy, "fuzzy" when transforming, good for photos.

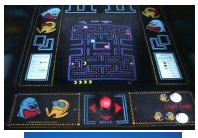
- **Vector:**

  Drawing instructions
  - Postscript
  - CGM
  - Fig
  - DWG

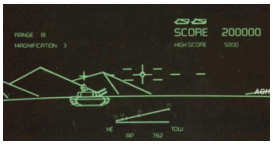
  Non-lossy, smooth when scaling, good for line art and diagrams.

2D Graphics

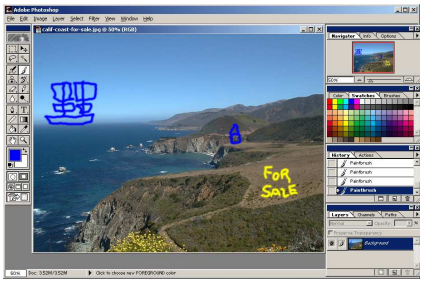
- **Raster:**



- **Vector:**

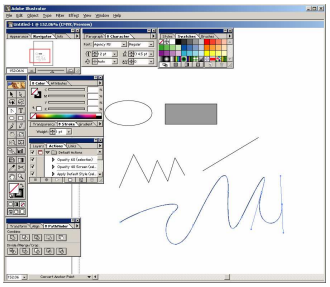


Adobe Photoshop: 2D Raster Graphics

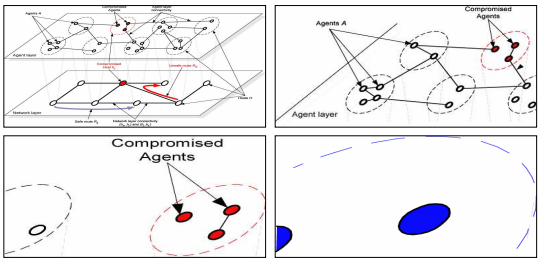


2D Raster Graphics

Adobe Illustrator: 2D Vector Graphics

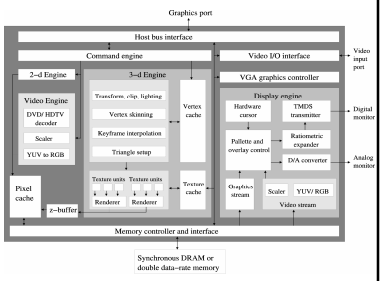


2D Vector Graphics



Graphics Displays

- Nearly all video displays today are RASTER
- How do we go from bits and shapes to pixels on a raster display?

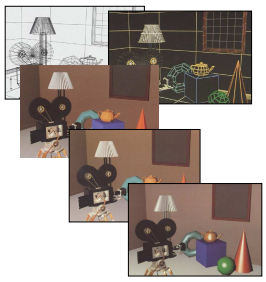


The Frame Buffer

- Video memory holding pixels from which the video display is refreshed
  - I.e. essentially a pix/bit map, a Raster image
- Usually implemented on hardware cards
  - Smart frame buffers
    - Accelerated 2D and 3D interaction
    - Color depth (1, 8, 16, 24, 32 bit), Z-buffering
- Double buffering: use of a second memory space to reduce visual artifacts, i.e. swap in-and-out screen buffers

3D Rendering

- 1960s - the visibility problem
  - Roberts (1963), Appel (1967) - hidden-line algorithms
  - Warnock (1969), Watkins (1970) - hidden-surface algorithms
  - Sutherland (1974) - visibility = sorting
- 1970s - raster graphics
  - Gouraud (1971) - diffuse lighting
  - Phong (1974) - specular lighting
  - Blinn (1974) - curved surfaces, texture
  - Catmull (1974) - Z-buffer hidden-surface algorithm
  - Crow (1977) - anti-aliasing



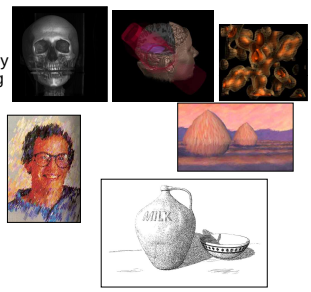
3D Rendering

Toward Reality

- early 1980s - global illumination
  - Whitted (1980) - ray tracing
  - Goral, Torrance et al. (1984), Cohen (1985) - radiosity
  - Kajiya (1986) - the rendering equation
- late 1980s - photorealism
  - Cook (1984) - shade trees
  - Perlin (1985) - shading languages
  - Hanrahan and Lawson (1990) - RenderMan

Present Developments

- early 1990s - non-photorealistic rendering
  - Drebin et al. (1988), Levoy (1988) - volume rendering
  - Haeberli (1990) - impressionistic paint programs
  - Salesin et al. (1994-) - automatic pen-and-ink illustration
  - Meier (1996) - painterly rendering



Application Areas

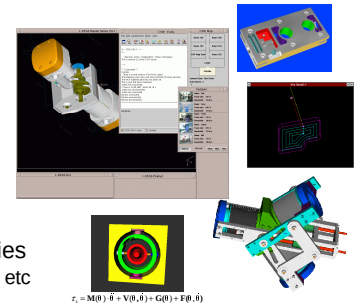
- ***Entertainment***
- CAD/CAM
- Scientific & Medical visualization
- Training & Education
- Synthetic Realities
  - VR, commerce, etc
- Art and design

Pixar

Lord of the Rings Troll

Application Areas

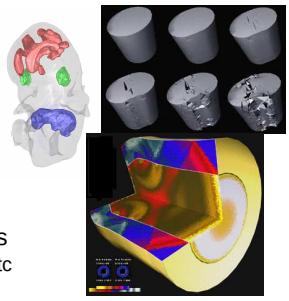
- Entertainment
- ***CAD/CAM***
- Scientific & Medical visualization
- Training & Education
- Synthetic Realities
  - VR, commerce, etc
- Art and design



Regli et al @ Drexel

Application Areas

- Entertainment
- CAD/CAM
- ***Scientific & Medical Visualization***
- Training & Education
- Synthetic Realities
  - VR, commerce, etc
- Art and design

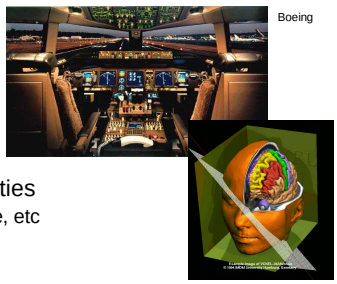


Lombeyda, Breen @ CalTech

http://space.jpl.nasa.gov/

Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- ***Training & Education***
- Synthetic Realities
  - VR, commerce, etc
- Art and design

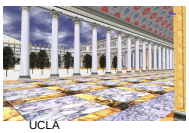


Boeing

Hamburg U, Germany

Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- Training & Education
- ***Synthetic Realities***
  - VR, commerce, etc
- Art and design



UCLA

Telepresence

Id Software

FakeSpace Cave

Application Areas

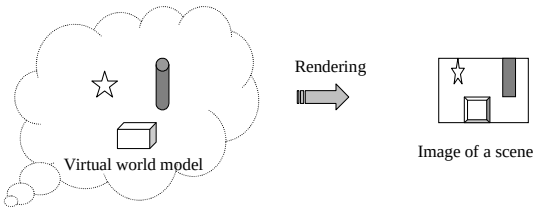
- Entertainment
- CAD/CAM
- Scientific visualization
- Training & Education
- Synthetic Realities
  - VR, commerce, etc
- ***Art and design***

Kimmel Center

Computer Graphics

Modeling: Creating a virtual world.

Rendering: Generating a visual image of a scene.



Modeler and Renderer

- **Modeler:** responsible for construction and maintenance of virtual world model
- **Renderer:** performs the rendering of a scene from a specific view on the graphics device
- Note: some systems don't have a clear delineation between the two

World Space

- **World Space:** Either a 2D or 3D space in which objects are modeled
- The output of a graphics system is usually in 2D
- Although output in 2D form for each, 3D graphics considerably more complex

Transformations

- Different types:
  - **Object transformations:** geometric transforms applied to achieve proper placement of the objects in virtual space
  - **Viewing transformations:** transforms used for viewing

Transformations

- Affine transformations:
  - Geometric transforms including:
    - Translations
    - Rotations
    - Scalings
    - Reflections
  - Projective transformations:
    - Used for 3D viewing ("projecting" a 2D rep of a 3D object)

Views

- **Views** are used to "see" the model from some specific perspective
- Viewing in 2D is simple
  - Object transformations and viewing transformations are usually the same
- 3D Viewing is far more complex
  - Process involving mapping a 3D model to a 2D plane

3D Viewing Issues

- Depends on position of the view, orientation, field of view, etc
- Hidden object issues
  - Hidden lines, hidden surfaces, etc
  - Hidden objects should not be shown
- Light sources should be considered
- The type of "material" the objects are made of should be modeled and considered with lighting
  - E.g., shiny materials, etc
- All of these things require computational overhead

Graphics System: Components and Functions

- Modeler
- Renderer
- Hardware device
- Virtual World
- View
- Geometry
- Transformation
- Illumination
- Interaction
- Animation

Graphics Programming Environment

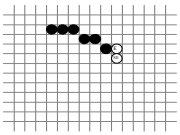
Platform Independent (Java 2D and Java 3D)
Graphics Standard (GKS, PHIGS, OpenGL)
OS (WIN32, X, Mac OS)
Hardware (direct register/video buffer programming)

Hardware Level

- Program the graphics hardware directly
- Typically written in low-level languages
- Manipulate the hardware registers and video buffers
- Highly machine-dependent
- Example: MS-DOS graphics program

Source

Determination of the pixels on a circle. From the current pixel, the next pixel will be either to the "east" or to the "southeast".



Operating System Level

- Program through OS graphics support
- Do not directly manipulate graphics hardware
- Portable on the same platform
- Example: WIN32

```
hdc = BeginPaint (hwnd, &ps);
GetClientRect (hwnd, &rc);
cx = (rc.left + rc.right)/2;
cy = (rc.top + rc.bottom)/2;
if (rc.bottom - rc.top < rc.right - rc.left)
        r = (rc.bottom - rc.top) / 2 - 20;
else
        r = (rc.right - rc.left) / 2 - 20;
Ellipse(hdc, cx-r, cy-r, cx+r, cy+r);
EndPaint (hwnd, &ps);
```

Source Run

GKS and PHIGS

Graphics Kernel System

- International standard (ISO 7942 1985)
- 2D graphics
- Common language binding: FORTRAN
- Example: A FORTRAN GKS program to draw a circle
  - Bindings in C and Pascal are also available

Source

Programmer's Hierarchical Interactive Graphics System

- ISO 9592 1991
- 3D graphics

OpenGL

- Popular 2D/3D graphics API
- over 200 functions
- Common language binding: C
- Example: An OpenGL program to draw a circle

Source

Run

GLUT
GLU
GL

OpenGL

3D Example: A spinning sphere

Source

Run

Java

Java 3D based graphics systems

Graphics application	
Java 2D and other Java APIs	Java 3D
Java VM	OpenGL (or DirectX)
OS	
Display driver	
Graphics card	
Display	

Java Programming Language

- Simple
- Object Oriented (OOP)
- Write once, run anywhere
- Multithreaded

Java graphics: AWT / Swing

AWT Example

Source

Run

JOGL

- OpenGL Java language binding
- No OOP modeler

1. Create a GLCanvas or GLJPanel object through the GLDrawableFactory class.
2. Add a GLEvent listener to the canvas object.
3. Implement the listener by implementing the four methods: init, display, reshape, and displayChanged.

Source

Run

Java 2D

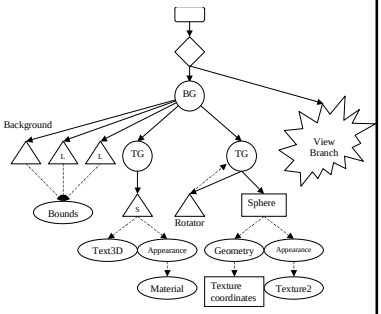
- Standard package of Java 2 platform
- Improvements over AWT
- Graphics2D class → Java 2D's rendering engine

Source

Run

Java 3D

- High-level API
- Scene graph
- Modeler/Renderer
- Java Integration



Source

Run

Other Fields Related to Graphics

- Image processing
- Computer vision
- Mathematics
  - Analytic geometry
  - Linear algebra