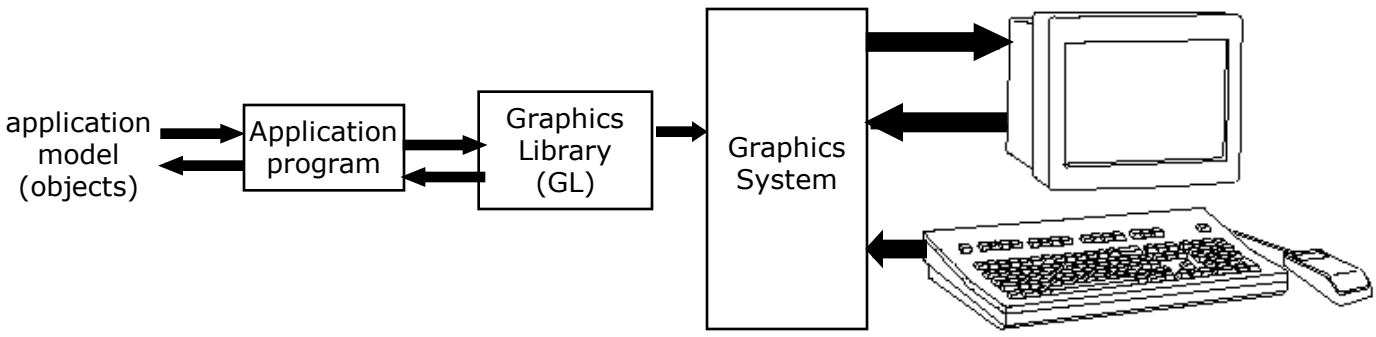
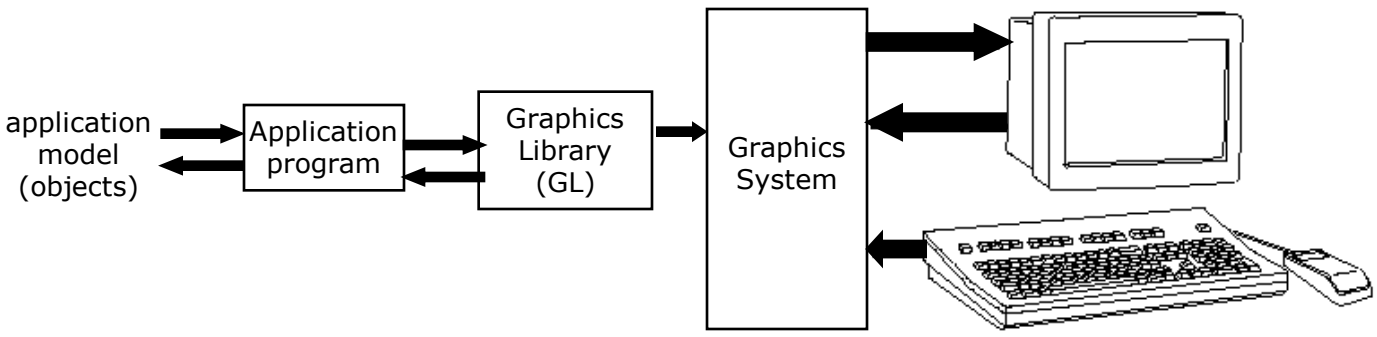
Graphics Framework



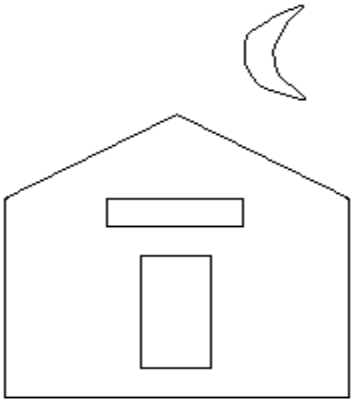
Conceptual Framework for Interactive Graphics

- Graphics library/package (e.g., OpenGL) is intermediary between application and display hardware (Graphics System)
- Application program maps application objects to views (images) of those objects by calling on graphics library
- This hardware and software framework is more than 4 decades old but is still useful, indeed dominant

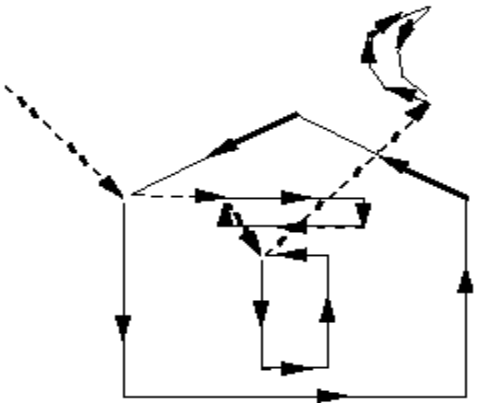


Graphics Display Hardware

- Vector (calligraphic, stroke, random-scan)
  - still used in some plotters

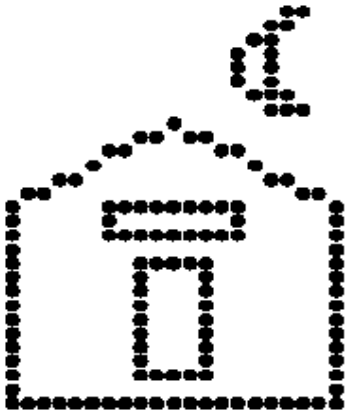


Ideal Drawing

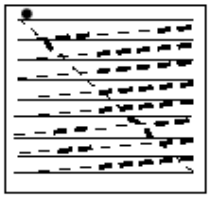


Vector Drawing

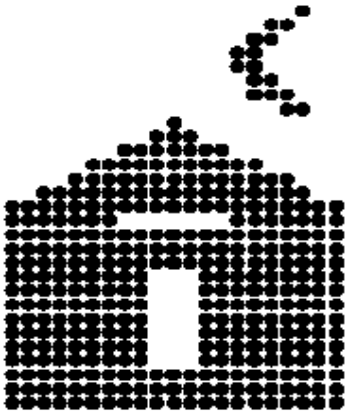
- Raster (TV, bitmap, pixmap), used in displays and laser printers



Outline primitives

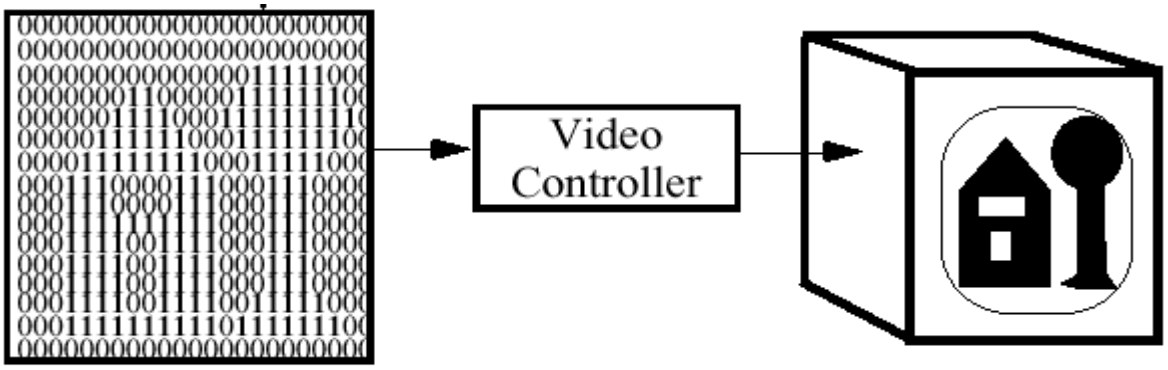


Raster



Filled primitives

2D Raster Architecture

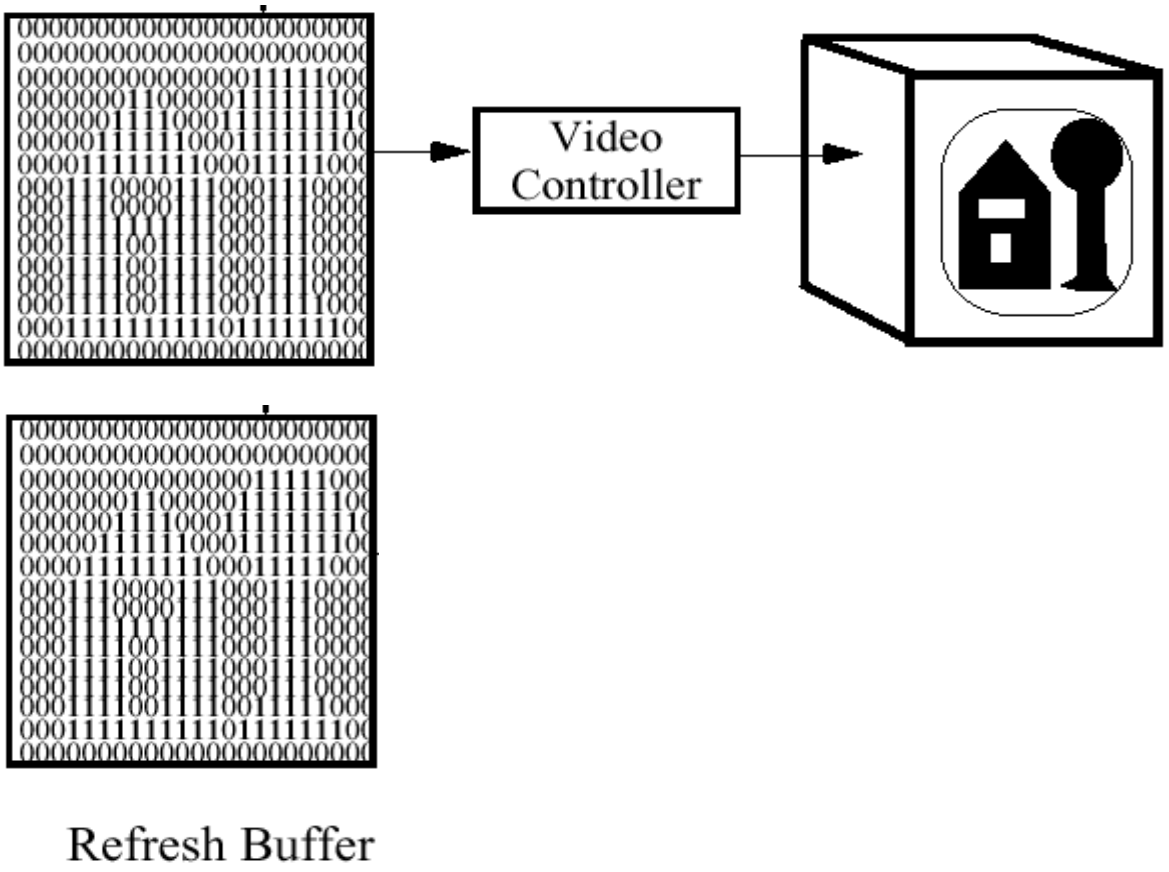


Frame buffer

Display

- Raster displays store images (pixmaps, or bitmaps) in a *frame buffer*, also known as bitmap buffer, or refresh buffer
- The *frame buffer* is a chunk of memory located either in separate hardware (VRAM) or in CPU's main memory (DRAM)
- The *frame buffer* can be accessed directly, through memory operations
- Video controller draws all scan-lines at consistent $\geq$ 60 Hz;
  - separates update rate of the frame buffer and refresh rate of the display

Single vs. Double Buffering

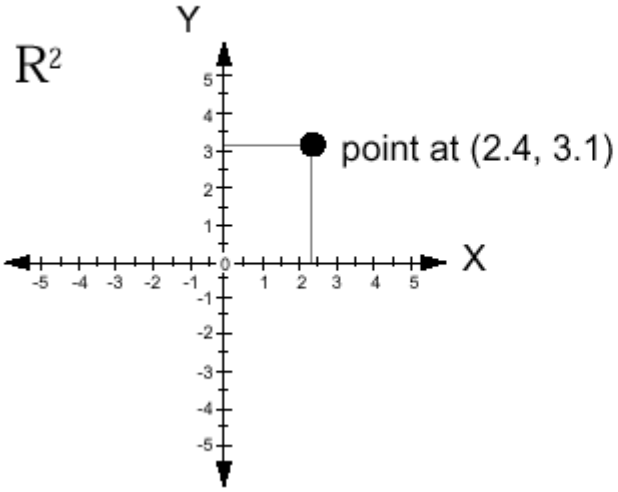


- Use **two** buffers (two chunks of memory)
- Draw always on the back buffer
- When done drawing, **swap buffers**

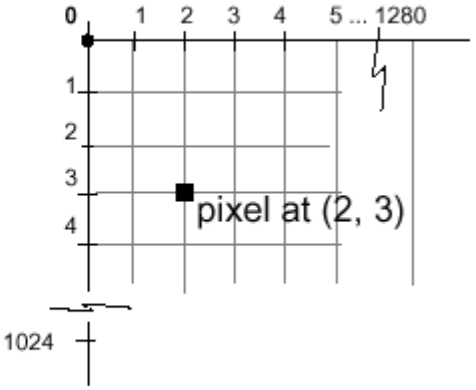
Display Coordinates vs Model

Cartesian Coordinates

- Typically modeling space is floating point, screen space is integer

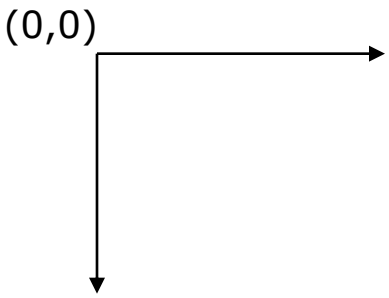


x, y Cartesian grid

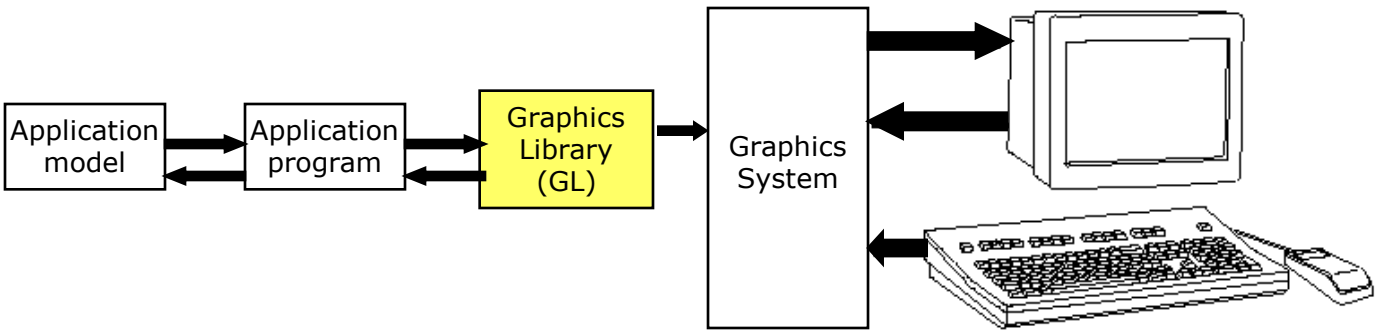


Integer Grid

NB: Often, screen coordinates are measured top to bottom, based on raster scan



The Graphics Library



Graphics Library

- May be: OpenGL™, DirectX™, RenderMan™, **your own code,** proprietary company code, …

- Provides representations and support for:
  - primitive elements (vertices, triangles etc)
  - attributes (color, line style, material properties)
  - lights
  - transformations

What is OpenGL?

The ***Open G****raphics* ***L****ibrary*

- 3-D graphics API specification
  - specification publicly available
- "A software interface to graphics hardware"[1]
- Raster graphics library
- pass in vertices and other scene data
- get pixels out
- Industry standard
- supported across many platforms
  - Mac OS, Windows, Linux, iPhone, PSP…
  - WebGL for 3D web graphics

[1] *"The OpenGL Graphics System: A Specification" Version 3.0*

OpenGL Architecture (1/2)

- OpenGL uses a client-server model
  - client sends commands to the server
  - server interprets, processes commands
  - note: client and server usually on the same computer, but need not be
    - your program = client
    - OpenGL = server
  - example interaction:

program	OpenGL
begin triangle normal (0, 0, -1) vertex (-1, 1, -1, 1) vertex (1, -1, -1, 1) vertex (-1, -1, -1, 1) end triangle	<scan-converts the given triangle with normal (0,0,-1) at all vertices>

OpenGL Architecture (2/2)

OpenGL is state-full and procedural

- the current OpenGL state (collectively called a *context*) contains data describing how rendering should proceed
  - ex: current color, lights, textures, etc
- state does not change until explicitly set
  - once some state is set, it remains in effect until changed
- procedural model
  - usually accessed through a plain C API
  - **NOT** object-oriented (though this changes gradually in OpenGL 3.1 and beyond)

OpenGL, GLU, and GLUT

- OpenGL (Graphics Library)
  - offers a platform-independent software interface with graphics hardware
  - supports basic geometric primitives (points, lines, polygons)
  - sets up an environment for graphical programming
    - May be useful to think of it as a state machine

- GLU (OpenGL Utility Library)
  - higher-level features (e.g. curved surfaces) built out of basic OpenGL functions

- GLUT (OpenGL Utility Toolkit)
  - separate library by Mark Kilgard, platform independent
  - hides details about basic window operations and context creation
  - a simple framework for writing OpenGL applications independent of any particular platform

Recap

- Gfx framework
- Display hw: raster scan & frame buffer
- Graphics libraries
- OpenGL, GLU, GLUT overview