Development of Computer Graphics

- 1951
  - Whirlwind, Jay Forrester (MIT)
  - CRT displays
- Mid 1950s
  - SAGE air defense system
  - Command & control CRT, light pens
- Late 1950s
  - Computer Art, James Whitney Sr.
  - Visual Feedback loops
- 1962
  - Sketchpad, Ivan Sutherland
  - Data structures, light pen for drawing and choices
Development of Computer Graphics

- Early display devices (mid-60s):
- Vector, stroke, line drawing, calligraphic displays
- Architecture of Vector Display
Development of Computer Graphics

Architecture of a vector display - random scan

- vector generator converts digital coordinates to beam deflections
Development of Computer Graphics

1964
- CAD and CAM
- General Motors DAC, Itek Digitek for Lens Design

1964-1970s
- Photorealism at University of Utah
- Sutherland, Evans, Catmull, Blinn

1968
- Evans & Sutherland
- commercial company - flight simulators
- 3D vector pipeline, matrix multiplier, clipping

1969
- First SIGGRAPH
Development of Computer Graphics

1970

- Pierre Bezier - Bezier curves
Development of Computer Graphics

- 1971:
  - Gouraud Shading
Development of Computer Graphics

1974-1977

- Catmull - Z-buffer
- Bui-Young Phong creates Phong Shading (Utah)
- Martin Newell's teapot (Utah)
- Computer graphics at NYIT - computer animation
- Raster Graphics (Xerox PARC, Shoup)
Development of Computer Graphics

- Architecture of a raster display
Development of Computer Graphics

- Architecture of a raster display - raster scan
  - beams (3 beams) intensity set to reflect pixel intensity
  - scan speed: originally 30Hz now 60Hz
Development of Computer Graphics

- **Raster Scan**
  - need to store whole image
  - 1024 x 1024 x n - n bits per pixel
  - mono 1 bit, color 8 (256 color), 24 (16 million)
  - 32 to 96 bits used (double buffering, z-buffering)
  - 1280 x 1024 x 24 needs only 3.75 MB video RAM
Development of Computer Graphics

- Random Scan versus Raster Scan
  - note ragged lines
Development of Computer Graphics

Random Scan versus Raster Scan

Raster advantages:
- low cost, superior fill ability, refresh rate independent of complexity, 70Hz sufficient to avoid flicker

Raster disadvantages:
- discrete nature of pixel representation, need for scan conversion in software or RIP chips
  - real-time dynamics more demanding
- approximation of lines by sequence of pixels
  - aliasing - jaggies or staircasing
  - manifestation of sampling error in signal processing
  - need for anti-aliasing
Development of Computer Graphics

1976
- Image and texture mapping (Blinn)

1977
- 3D Core Graphics System, first “standard” for device independent graphics package
- allowed portable graphics programming
- ACM SIGGRAPH committee including Foley, Van Dam, Feiner
- baseline specification - many implementations
Development of Computer Graphics

1982

- Clarke, Geometry Engine
  - hardware support for transforms (matrix-vector multiplies), clipping (variant of Sutherland-Hodgman algorithm)
- IRIS - Integrated Raster Imaging System, SGI
  - high-end workstation
  - hardware acceleration of graphics pipeline

1982

- TRON - 'non-realism' and relatively low technical quality special effects
- Star Trek - Genesis Effect; Lucasfilm's computer graphics division (later split into Industrial Light and Magic, and Pixar)
  - used key technical effects (such as particle systems and caustics)
Development of Computer Graphics

1982
- Ray Tracing, Turner Whitted
- Good at rendering reflections, refractions and shadows

1983
- VRAM, Video random access memory, Texas Instruments
  - Can read out all pixels in one memory cycle
Development of Computer Graphics

1983

- Fractals
- Allowed generation of the key components of natural-looking landscapes
Development of Computer Graphics

- **1985**
  - Radiosity, Don Greenberg (Cornell)
  - GKS, Graphical Kernel System
    - first ANSI standard
    - elaborated cleaned up version of CORE but only 2D
- **1986**
  - Renderman - an extensible 'procedural language' for controlling the animation/rendering process
- **1988**
  - GKS, Graphical Kernel System - 3D version
Development of Computer Graphics

1988
- PHIGS, PHIGS+
- Programmer’s Hierarchical Interactive Graphics System
- More complex than CORE
PHIGS v GKS

- GKS allowed grouping of primitives into “segments”
  - no nesting of segments
- PHIGS allowed nested hierarchical grouping of 3D primitives into “structures”
  - all primitives subject to geometric transformations
  - editable database of structures
  - auto-update of screen when database altered
- PHIGS+
  - extension for pseudo-realistic rendering on raster devices
- PHIGS, PHIGS+ large packages
  - run best with hardware support of transformations, clipping and rendering
Development of Computer Graphics

- **1993**
  - OpenGL - Open Graphics Library
  - derived from SGI’s GL library
- **1993**
  - Open Inventor, OO layer on OpenGL
- **1995**
  - QuickDraw 3D, Apple
- **1995**
  - Direct3D, Microsoft, game playing API
Development of Computer Graphics

- Input Devices
  - early light pens to modern mice
  - data tablet
  - touch sensitive screens
  - 3D input devices (spaceballs etc.)
  - button and dial boxes
Describing Scene to be viewed

- Application Program - creates application
- Application Model - independent of display system
  - program must extract geometry and convert to primitives of graphics system
  - primitives: points, lines, (rectangles, ellipses, text, polygons, polyhedra, spheres, curves, surfaces)
  - application must convert geometry to primitives supported
- attributes (line style, color, line width, fill style)
Graphics Systems

Typically libraries: output subroutines

- user programs in *logical display device* terms
- graphics library converts to device dependent instructions
- abstraction of display device
  - *locator* - mouse, tablet, joystick etc
  - *sample* - return from locator
  - *event* - generated by user input
Interaction Handling - *event driven loop*

```java
while (!quit) {
    enable selection of commands/objects
    wait for user selection
    switch (selection) { process selection, updating model and
                        screen as necessary}
}
```

User interaction

- change in screen appearance - does not involve update of
  model: application updates state and calls graphics package
- change in model: must recalculate