Today’s class

- Introductions
- Graphics system overview
Introductions
Instructor

- Cary Laxer
- Visiting lecturer
- Home institution is Rose-Hulman Institute of Technology, Terre Haute, Indiana, USA
- Professor and Head of Computer Science and Software Engineering
- Bachelor’s degree in computer science and mathematics from New York University
- Ph.D. in biomedical engineering from Duke University
Lab instructors

- Tony Meijer
- Martin Ericsson
Course

- Information is maintained on the course website: [www.it.uu.se/edu/course/homepage/grafik1/ht07](http://www.it.uu.se/edu/course/homepage/grafik1/ht07)
- 14 lecture meetings and 4 lab meetings
- We will cover chapters 1-8 and 11
- I will try to have some in-class exercises to help reinforce the material and to break up the long lecture periods
Introduce yourselves

Tell us:
- Your name
- Your hometown
- Your computer background
- Something interesting about yourself
Graphics system overview
Computer graphics

- Definition from Foley & van Dam
- The creation, storage, and manipulation of models of objects and their pictures via computer
- It takes hardware and software to accomplish this
Examples and applications

- Graphs and charts (can be done using simple programming in your favorite language)
- CAD (computer aided design)
- Presentation graphics and computer slide shows
- Process control
Examples and applications

- Image processing
  - satellite
  - medical
  - spacecraft
- Computer user interfaces (icons)
- Video games
- Flight simulation and other training
Examples and applications

- Computer art and animation
- Virtual reality
- Others you can think of?
- Watch “Surly Squirrel” from SIGGRAPH 2005 Animation Theatre (available online at YouTube: http://www.youtube.com/watch?v=5y5ZRcYPYo)
Output devices

- **Definitions**
  - **pixel**: picture element
  - **resolution**: # of points displayable in a direction without overlap
  - **aspect ratio**: # points vertically divided by # points horizontally to give equal length lines in both directions
Cathode ray tubes (CRTs)

- Electron flow: cathode $\Rightarrow$ control grid (intensity) $\Rightarrow$ focusing system $\Rightarrow$ vertical deflection $\Rightarrow$ horizontal deflection $\Rightarrow$ phosphor coating

- Light emitted (just for a few milliseconds) when excited electrons of the phosphor coating return to unexcited state

- Need to refresh image at least 50 times per second for human to see a steady image
Raster scan CRTs

- Electron beam covers the entire screen
- Image is a set of intensity values for all points on the screen
- Memory to hold this image is called a frame buffer
Frame buffer

- Usually implemented with a special type of memory
  - VRAM (video random-access memory)
  - DRAM (dynamic random-access memory)
- The number of bits used for each pixel is the **depth** of the frame buffer
Interlacing

- Image drawn one row (scan line) at a time
- Sometimes scan lines are interlaced - even numbered rows drawn first, then odd numbered rows
- Thus, only half the image is updated on each refresh
- Non-interlaced CRTs draw all scan lines on each refresh
Display processing units (DPUs)

- Special purpose CPUs
- Decode opcodes and data
- Instruction counters and registers present
Raster display

- CPU → image creation system → frame buffer → image display system → CRT
- Image creation system - scan conversion algorithms to load image into frame buffer
- Image display system - an autoincrement system in x, y addresses; when scan completed, CPU is interrupted, image modifications are made (in flyback time, about 1.3 ms)
How fast does memory need to be?

- A typical workstation display can have a resolution of 1280 x 1024 pixels.
- If it is refreshed 72 times per second, how fast does memory need to be? (How much time can we take to read one pixel from memory?)
Pipeline architectures

- Increase throughput by allowing multiple arithmetic operations to occur in one clock cycle
- Have multiple adders and multipliers
- Consider how this can speed up multiplication of $4 \times 4$ matrices, the size used for projection of points to the screen in computer graphics
The graphics pipeline

- Vertex processing – coordinate transformations and color
- Clipping and primitive assembly – determines what can be seen
- Rasterization – determining which pixels go with each primitive
- Fragment processor – fragments are potential pixels that convey location, color, depth
Installing GLUT on your Windows computer

- The three GLUT files you need are on the course web site
- Place copies of them as follows (for Visual C++):
  - `glut.h` in `C:\Program Files\ Microsoft Visual Studio 8\VC\Include\GL`
  - `glut32.lib` in `C:\Program Files\ Microsoft Visual Studio 8\VC\Lib`
  - `glut32.dll` in `C:\Windows\System`
Making Visual C++ do OpenGL graphics

- Launch Visual C++
- Create a new console project - call it testline
- Download file testline.c from the course web site to your testline project folder
- Add the file to the project
- Build and run the project as you normally would in Visual C++