Lecture 2

Quick scan through:
- Basics of an image
- Mechanics of drawing (for practice assignment)
- Basic ideas of images/drawing (a scan through Chapter 3)

- A brief touch on a bunch of topics we'll come back to later

A window on the screen

- Some 2D picture

Aside #1
- How did we get a window on the screen in the first place?

- Operating system, window system, toolkit

Basic toolkit questions

- For class: FlTk, GLUT
  - Why? (why not)

- When do I draw (redraw, idle, damage)
  - Event models
- Where do I draw
- How do I draw (double buffering)
- What happens when I draw
- What about user interaction?

How will I draw

- Directly access pixels
  - In image data structures, let toolkit display
  - Or just read/write to files
- Use toolkit (and therefore hardware)
  - Primitives (geometry)
- OpenGL

- Practice assignment – draw something

Things to know about OpenGL (for practice assignment)

- What is X,Y (coordinate system, NDC)
- State model
- Primitives
- Basic Commands

Kinds of things
- Set up coordinate systems
- Draw primitives
- Control appearance of primitives
- Other drawing control

(back to images) Measuring on the image plane

- Want to measure / record the light that hits the image plane
- At every position on the image plane (in the image) we can measure the amount of light
  - Continuous phenomenon (move a little bit, and it can be different)
  - Can think of an image as a function that given a position (x,y) tells "amount" of light at position
    i = f(x,y)
  - For now, simplify "amount" as just a quantity, ignoring that light can be different colors

How to think about sampled images

- Little squares?
  - Little regions of the image?
- Sometimes useful for thinking about

A pixel is not a little square…

- Piecewise linear approximation of an image
- Discrete measurements of continuous thing
  - Individual measurements or samples
  - Usually regular grids

Displays

- Continuous vs. Discrete

- Flicker rate
- Real world vs. Movies vs. TV

- Old fashioned TV (CRT)
  - Raster scan / retrace (discrete lines)
  - Interlace (radio limitations)

Practical Aside Storing images

- Need to store a measurement for each pixel
- X * Y pixels * (# bits per pixel)
- R,G,B
- An extra "A" (transparency)
- 8 bits integer per channel (often OK – more in a minute)

- Lots of data – lots of redundancy

Image formats

- Lots of them
- Often compressed

Practice assignment

- Simple format: TARGA (.tga)
- We provide a really simple library
- No compression

- JPEG, PNG – built into FlTK – but only reading, so use TGA for imaging assigns

Raster Images

- Why is it called a raster
- Pixel order, channels (RGB vs. RRR…GGG)
- Row padding

- What to store
  - RGB vs. I vs. RGBA
  - Fixed point vs. Floating point
  - 8 bits vs. more (later)

What is "A" (alpha)

- Kindof treating like an extra color

- "Opacity" of pixel
- As if image were painted on glass
- Useful for "compositing" one picture over another

What numbers to store?

- Ideally:
  - Continuous amount (nearly, discrete quanta of photons)
  - Huge range (surface of sun vs. dark room)
- Practically:
  - Mainly interested in what we can see
  - What differences we can tell
  - If we're making pictures, not for analysis

How sensitive is the eye?

- Amazing range!
  - Night vision – when eyes adjusted, camping
  - Bright daylight
    - `Sunlight 10000.`
    - `Twilight    10.`
    - `Starlight    0.001`
- Catch: at any given time, can't see this range
  - Adaptation – bright light, iris closes, lets in less light, …
- At any given time, about 100:1 contrast ratio
  - This is a lot more than most displays
  - Better displays = more constrast
    - Often by blacker blacks

High Dynamic Range Imagery

- Most sensors/displays have less range than eye
  - Certainly less range than scenes do
- What happens?
  - Bright areas – all white (no details)
  - Dark (shadow) areas – all black (no details)
- What to do?
  - Adjust exposure (time, aperature, sensitivity) to get the most important stuff
  - Acquire "High Dynamic Range" Imagery
    - Special sensors
    - Multiple exposures (at different settings) – cool thing to do
  - HDR later in the course

Perception of intensity

- Eye senses relative differences
  - Equivalent differences 50:100 20:40
  - Hard to tell absolute differences directly
    - Adaptation to current setting
- Can sense 1% differences
- At any given time 100:1 contrast ratio
- How many levels can you see in an image?
  - 1.01 ^ 463 = 100.2 (e.g. 463 1% differences = 100:1)
  - This is about 8 bits of precision (less than 9)
  - But its VERY non linear 1, 1.01, …. , 99.2, 100.2

Non-linearity of intensity

- Non-linear mapping from "amount of light" to perceived brightness
- Want uniform mapping of intensities -> perception
  - Level 1, 2, 3, …. 255 -> 1, 1.01, 1.02, … 99, 100
- Worse: displays are non-linear too
  - Voltage -> amount of light is non-linear
  - Different displays are different
- Want to linearize the system
  - Intensity levels map nicely to perceived levels

Gamma correction

- Idea: put a non-linear function between intensity and output
  - Done as the last step (usually) – after all computations
- Could create arbitrary functions for mapping
  - Too cumbersome
- Exponential is a good approximate model
  - Exponential non-linearity of perception
  - Exponential power laws in CRTs

Modeling a display device

- 5/2 power law (five-halves)
  - Models physics of a CRT
  - Real CRTs are close, LCDs designed to be similar
- L = M (i+ε)^γ
  - i = input intensity value
  - L = amount of light
  - ε = since zero isn't really black
  - M = maximum intensity
  - γ = specific property of display

Linearizing the display

- Define a function **g** that corrects for non-linearity
- L = M (g(i))^γ   (ignoring ε)
  - G = 1/ γ
- Where do we get γ from?
  - Pick it so things look right
- Note: 1st order approximation (very simple)
  - Only 1 parameter to specify (γ), many factors

Gamma correction

- Want value 0 = minimum intensity
- Want value max (1 or 255) = maximum intensity

--- those 2 are easy to get

- Pick one more point
  - Midpoint should be 50%
  - Easy – show 50% black white + 50% gray
  - Adjust gamma until it looks the same
- All this happens "behind the scenes"
- Everything gets harder when we deal with color

What to store in the frame buffer?

- Frame Buffer = rectangular chunk of memory
- Intensity measurements
  - Deal with color later, basically store multiple monochrome
- Continuous range of intensities
  - 8-9 bits of precision ideally
    - More since can't get exactly right (10-12 bits)
    - More since want more dynamic range (12-14 bits)
    - More since want linear space to make math easy (16-32 bits)
- Discrete set of choices – **QUANTIZATION**
  - Inks, palettes, color tables, …
  - Less storage cost + Color table animation

Geometry to Images

- How do we draw?
  - Set pixels / alter existing values
  - Convert geometry
- Rasterization: convert geometry to pixel values
  - Line drawing, Triangle drawing
- Taken care of in hardware nowadays
  - Hardware uses different algorithms

Line-Drawing algorithm Brezenham's or Midpoint

- Requirements
  - No skipped pixels
  - No floating point
- Key Ideas:
  - Limit to 1 octant (0->45 degrees)
  - Get others by symmetry
  - 1 pixel per column
  - Each step – either horizontal, or up one
  - Decision rule: if pixel is above "midpoint"

Aliasing

- Line is a continuous thing
- Pixels are discrete measurments
  - Imperfect representation
- Jaggies, Crawlies
- Line-weights
- Sub-pixel positions

Aliasing

- Lost information because using a continuous representation
- Many “continuous” things = 1 discrete thing
  - They are “aliases” of each other
- Lots of theory (later)

Anti-Aliasing

- Once you’ve aliased you’ve lost
- Can do drawing to try to minimize the visual artifacts
  - Simplistic: soften hard edges
  - Not “all in 1 bucket” – spread it out
- We’ll look at this a lot – mainly in context of photo processing