Topics:
- Internet and WWW Overview
  - History
  - Technologies
- Web Programming Languages
- Crash course in programming (paradigms)
- Processing

Resources:
- Processing web site: http://www.processing.org/
- Getting started tutorial: http://www.processing.org/learning/gettingstarted/
- Drawing tutorial: http://www.processing.org/learning/drawing/
Networks

- When computers talk to each other, this is called a network.
  - The network can have different kinds of computers and peripherals attached to it.
  - Networks in which computers are physically connected to each other in the close geographical proximity are called local area networks (LANs).
  - Other networks are called wide area networks (WANs).
- The internet is a wide area network.
- The internet is an open system = “a system whose architecture is not a secret”
Protocols

Definition: protocol = set of rules for how computers communicate with each other.

Examples:
Lower Level Protocols
– TCP: transmission control protocol (computer <-> computer)
– IP: internet protocol (computer <-> computer)

Higher Level Protocols
– HTTP: hypertext transfer protocol (program <-> program)
– FTP: file transfer protocol (program <-> program)
– SMTP: simple mail transfer protocol (program <-> program)

For more information:
http://en.wikipedia.org/wiki/Internet_protocol_suite
What is the internet?

The internet is a WAN.

**History**
- Motivated by military desire for secure, reliable, efficient communications in the result of a nuclear war.
- ARPAnet (circa 1971): used “NCP”
- Open internet was standardized in September 1981

**Fundamental idea:**
- Uses "data packets" to move information (packet switching).
- Allows communication without a "dedicated connection".
What is the World Wide Web?

The world-wide web (WWW) is NOT the internet!

History:
- The idea of the world-wide web was conceived by Sir Tim Berners-Lee
- Developed and discussed at CERN in Switzerland from about 1989
- Made public in 1994

Fundamentals:
- The WWW uses the internet, but is not the internet itself
- The WWW is a way of organizing and viewing data that is accessible through the internet
Other Internet Facilities

- FTP (file transfer protocol)
  - download
  - upload
- Bitorrent (File Sharing)
- Email (SMTP)
- Newsgroups
  - posting
  - thread
- VoIP (Digital phone)
- Streaming digital movies
- Botnets
  - Spam
  - DOS attacks
Clients and Servers

- **Server**: computer on a network which carries out some service for another computer.

- **Client**: the other computer for whom the server is carrying out the service

- **Types of servers:**
  - file server
  - database server
  - web server
  - groupware server (lotus notes, Bittorrent)
  - mail server
  - application server (provides access to particular applications e.g., game servers of a web site)
Client-Server Architecture

Advantages:
- Isolates data storage technology.
  - Places more burden on server (instead of client)
  - Allows for distributing tasks amongst server(s)
- Follows object-oriented and modular programming paradigms
  - Example: HTML documents, CSS files, images, video clips, etc.

Disadvantages:
- Multiple points of failure.
- Multiple points of attack.
Client-Server Architecture Types

Two-tier

- Presentation/Interface Layer
- Data Layer

Three-tier

- Presentation/Interface Layer
  - Web Browser == Client
  - Internet
- Processing Layer
  - Application Server, Web Server
  - Internet or LAN
- Data Layer
  - File Server, Database Server, Web Server
Web programming languages

Motivation:
- Right now, your web-pages are pretty stagnant. (nothing really moves).
- There is also no way to dynamically change what user sees (consider the case of a business owner who has 12,000 different products... does he need 12,000 different web pages?)

Interactive Web Programming
1. User initiates some action.
2. Which causes the web page to change in some way.
   - Changes can happen locally, on the “client”.
   - Changes can happen remotely on the server.
Web Programming Languages (Types)

Client Side:
- Programs are run by the clients browser/application.
- Example -> Javascript
  - scripting language based on Java
  - write programs using a text editor,
  - programs are embedded/stored with HTML.

Server Side:
- Programs are run by a web server program.
- Example -> PHP
  - scripting language (HyperText PreProcessor)
  - write programs using a text editor,
  - programs are embedded/stored with HTML.
Crash Course - Programming Languages

- Allow us to "talk" to a computer, in a language that we can understand.
- Require a well-defined syntax.
- Allow us reuse and share code (functions and libraries).
- Fall into a variety of different paradigms:
  - Functional,
  - Logical,
  - **Imperative,
  - **Procedural
  - **Object-Oriented.
- *(Scripting languages are a sub-type)*
Paradigms

- The advantage to learning about programming paradigms is that once you understand a particular paradigm it is easier to learn new programming languages that use that paradigm.

- 3-paradigms will be very important to you in this class:
  1. Imperative
     - "The Smart List"
  2. Procedural
     - "Making Phone Calls"
  3. Object-Oriented
     - Viewing a program as set of interacting objects
     - Designing patterns and templates.
Imperative Programming

- The imperative paradigm views a program as a "list" of things to do.
- Imperative languages need 3 things:
  1. **Sequence** - A predefined order in which to process information.
  2. **Selection** - The ability to make a choice. The "IF" statement.
  3. **Repetition** - The ability to repeat an action. The "WHILE" statement.
The procedural programming paradigm is based upon the concept of the “procedure call”: the ability to “send a message” to another section of a program. Procedural programming allows us to create sections of code that can be reused over and over.

Types of procedural calls:
- Function Calls
- Message Passing
- RPC Calls
- Event Handlers
Object-Oriented Programming

Basic Concepts:
- Visualize a program as a set of interacting objects (easy to do with games).
- Identify the associated properties and functions of these objects.
  - Properties: Where is it?
  - Functions: What can it do?
- Create templates/patterns (classes) for these objects that can be reused.
  - Game-Object -> Enemy Ship
  - Bullet Class - Create LOTS of bullets
Programming Languages vs. Scripting Languages

- For our purposes a full strength "programming language" is a language that provides some means for the programmer to talk directly to a computer's hardware.
- A "scripting language" by contrast only allows the programmer to create code that will talk to an application.
Graphics Programming:

- Utilizing and/or manipulating images within a computer program.
  - Photoshop
  - Game Programming
  - Digital Movie Creation
- Windows and GUI are graphics programming.
- Many programming languages have "graphics libraries" (Direct3D, OpenGL)
  - But the robustness of these libraries makes them difficult to work with.
Processing

Why we are going to use it:

- Simple (Imperative, Procedural, Limited OO) scripting language.
- Has excellent built in graphics capabilities.
- Programs can also be saved as applets and run inside a browser (client side scripting).

Essentials:

- Originally written for artists (create narratives, games).
- Programs in Processing are called sketches.
- Text-based programming language.
- Write and run using an integrated development environment (IDE) that is part of Processing.
Processing Getting Started

Open processing and type in the following commands just as you see them in the window.

Then hit the "run" button in the top bar. (Review buttons)
Coordinate System

All graphics are drawn using the following coordinate system:

- Think of it like a piece of graph paper
- A point fills in one cell on the graph paper
- A line fills in multiple cells, from one endpoint of the line to the other

![Diagram of a coordinate system with points (0,0) and (50,50)]
Processing Programming Basics

1. Each line contains a statement
2. Statements end with a semi-colon ( ; )
3. Comments are contained within /** and */
   - Another way to make comments: //
4. Functions
   - Provide a way to modularize code
   - Makes it easier to read and re-use
   - Also allows you to specify content for functionality built in to Processing
   - Example:
     ```java
     void draw() {
       line( 10, 20, 30, 40 );
     }
     ```
   - void keyword that indicates a function which returns nothing
   - draw() = the name of the function
   - curly brackets ( { and } ) delineate beginning and end of the function
   - with Processing, your sketch has to use no functions or all functions
Drawing Things

- **point( x, y )**
  - draws one point (looks like a dot...)
- **line( x1, y1, x2, y2 )**
  - connects two points
- **triangle( x1, y1, x2, y2, x3, y3 )**
  - connects three points
- **quad( x1, y1, x2, y2, x3, y3, x4, y4 )**
  - connects four points
- **rect( x, y, width, height )**
  - origin + extent; square if width=height
- **ellipse( x, y, width, height )**
  - origin: center of ellipse’s; circle if width=height
- **arc( x, y, width, height, start, stop )**
  - origin: center of arc’s bounding box (see ellipse)
  - start and stop: can be whole (int) or real (float) numbers (float); expressed in degrees or radians, depending on current angle mode; 0 is due east; measured clockwise.
Keyboard Interaction

- keyPressed()
  - handles behavior when user presses a key down
- keyReleased()
  - handles behavior when user releases a key
- keyTyped()
  - handles behavior when user types a key (press and release)
- key
  - indicates which key was pressed/released/typed
  - equals CODED when special key is pressed/released/typed, like an arrow key, shift, control, alt, etc.
- keyCode
  - indicates special key: UP, DOWN, LEFT, RIGHT, ALT, CONTROL, SHIFT
Second Program

// This is a comment
/* So is this */
/**
   So is this
*/

void setup() {
    background( #ffffff );
}

void keyPressed() {
    background( #0000ff );
}

void draw() {
    line( 10, 20, 30, 40 );
    point( 50, 50 );
}
Selection

- one decision—IF something is true:
  
  ```java
  if ( test ) {
    statements
  }
  ```

- two decisions—IF something is true...or ELSE:
  
  ```java
  if ( test ) {
    statements
  }
  else {
    statements
  }
  ```

- joint decisions—IF something is true OR something else is true:
  
  ```java
  if (( test1 ) || ( test2 )) {
    statements
  }
  ```
You may want to repeat an action. Both **for** and **while** loops are supported:

```
while (expression) {
    statements
}
```

**Example:**
```
int i=0;
while(i<80) {
    line(30, i, 80, i);
    i = i + 5;
}
```
void setup() {
    background( #ffffff );
}
void draw() {
    line( 10, 20, 30, 40 );
    point( 50, 50 );
}
void keyPressed() {
    if ( key == 'R' || key == 'r' ) {
        background( #ff0000 );
    } else {
        int i=0;
        while(i<80) {
            line(30, i, 80, i);
            i = i + 5;
        }
    }
}
Don't Panic

- We will have several labs on using processing.
- For those of you with little or no programming experience your final project will not be that hard.
- Anyone can learn to do simple (and fun) things with processing fairly quickly.

NOTE: Processing is FREE software that you can download and run on your machine at home.