Course Topics, 1

How to Program, specifically **object-oriented programming** using:

- Processing - a subset/extension of an older version of Java for media programming
- Java - a high level objected oriented language
- C++ - a high level objected oriented language

How to use and choose between the various **data structures**

- Lists, arrays, maps, sets, queues, stacks, trees, graphs, ...

How to navigate a **Unix** environment

How to use a **Version Control System**

- git & github

How to create a **GUI**

- Swing (for Java)
- QT (for C++/Java)
- from scratch
- GLV (MAT OpenGL project)

Course Topics, 2

How to create interactive **2D graphics** & **3D graphics**

- Processing renderers
- Swing/Java2D renderers
- OpenGL library
- GLSL shaders

How to interact with the file system; **I/O** fundamentals

How to store information in and query information from a **database**

- SQL (MySQL, PostGRES)
- NoSQL (Mongo)
- Graph (Neo4J)
- via webservices APIs (Amazon, Flickr, etc)

How to send and receive information to other computers, **networking** fundamentals

Course Topics, 3

How to design **algorithms**
- searching & sorting

How to write **threads** to manage concurrent processes
- multithreading
- synchronization
- concurrent data structures

How to use **audio** libraries
- Minim (Processing)
- JavaSound / JSyn (Java)
- PortAudio, rtAudio (C/C++)
- OSC messages to dedicated sound engines (SuperCollider, etc)

Today's Agenda

> To provide an overview of the course

> Look at syllabus

> To sign up for the class forum

> To install Processing, play with some sample code, and explore the Processing web site

> To become familiar with the Unix filesystem and environment

> To install git and to set up an account on github.com

> To create a new git repository for a Processing sketch

> To show you your first assignment

Unix - Common Commands

cd = change directory

```
cd ..                    move up to parent directory
cd ~                     change to home directory
cd /                     change to top level "root" directory
cd /Users/angus/Music    change to a directory by providing the absolute path
cd src                   move into a child folder by providing a relative path
```

pwd = print working directory

```
cd ~                     change to home directory
pwd                      prints out "/Users/angus"
```

ls = list files and directories

```
ls                       print out all files in current directory
ls -alh                  print out all files include hidden files in a long format
ls -R mydir              recursively list all files in mydir
```

cat = print out file

```
cat myText.txt           prints out contents of "myText.txt"
```

echo = print out from command line

```
echo hello there!        prints out "hello there!"
```

Unix - Common Commands

cp = copy file

```
cp fileA fileB      copies the contents of "fileA" into a new file named "fileB"
cp src/* ../dest/   copies all files in "src" into "dest" inside the parent directory
cp –r src/ dest/    copies all files and directories in "src" recursively into "dest"
```

mv = move file

```
mv fileA fileB      renames "fileA" to "fileB"
mv dirA ~/dirZ      moves "dirA" into the home directory, renaming it "dirZ"
```

mkdir = make a new directory

```
mkdir newdir        makes a new directory "newdir" under the current directory
```

rmdir = delete an empty directory

```
rmdir emptydir      empty dir is deleted
```

rm = permanently removes a file or directory

```
rm fileA            deletes "fileA"
rm –rf dirA         VERY DANGEROUS! recursively deletes everything in "dirA"
```

Unix - Common Commands

more = page through text

```
more bigfile.txt        pages through bigfile.txt (by pressing space)
```

head / tail = print out first or last lines of a file

```
head -n 5 file.txt      prints out first 5 lines of file.txt
tail -n 5 file.txt      prints out last 5 lines of file.txt
```

find = find files

```
find .                  recursively list all files from the current directory
find ~/logs/*2010*      list all files in the logs dir that contain "2010"
```

grep = print lines matching a pattern

```
grep "A" cell*          print lines containing "A" from files starting with "cell"
grep -i "a" file.txt    print lines containing "a" or "A" from file.txt
grep -v "a" file.txt    print lines that don't contain "a" from file.txt
```

man = get the manual for a command

```
man grep                page through documentation about the grep command
```

Unix - I/O Redirection

pipe | = funnel the output of one program into the input of another one

```
    cat bigfile | more           pages through bigfile
    ls mydir | grep a*           only prints out filenames that start with an "a"
    cat afile | grep a* | more   pages through lines that start with "a"
```

redirect output > = funnel the output of a program into a file

```
    cat myfile > text.txt        write the output of "cat myfile" into the file text.txt
    cat myfile >> text.txt       append the output to the end of text.txt
```

Unix - Connect to Other Machines

ssh = open a secure shell to a remote server

```
ssh my.name@my.server        logs you into your home directory on "my.server"
```

sftp = secure file transfer protocol (using ssh)

```
sftp my.name@my.server       browse through and transfer files from "my.server"
```

Unix - Misc.

There are hundreds of other more specialized helper programs bundled with the various Unix distributions (FreeBSD for OSX).

You can see what these are by typing

> echo $PATH

from the command line, which will list all of the places the terminal will look for a command

you can list out the contents of one these directories, e.g.

> ls /usr/bin

most of these are little C programs and you can find out how they work by looking at their *man* page.

If you want to know where a program you use lives, you can use the *which* command, e.g.

> which ls *//prints out /bin/ls*
> which which *//prints out /usr/bin/which*

Unix - Misc.

you can also get other command line programs or write your own.

MacPorts (http://www.macports.org/) lets you download other command line programs for OSX. Cygwin (http://cygwin.com/) is a Linux emulator for Windows that is bundled with lots command line programs

Version Control System

A Version Control System, or VCS, is a useful system for managing projects.

- keeps track of all of your changes to a project
- manages multiple people working on the same project
- if used via a project hosting site, will provide a backup your project

Some examples of VCS systems: Bazaar (or bzr), Mercury (or Hg), Git (written by inventor of Linux), Subversion (or svn), CVS (totally outdated)

Some examples of free project hosting sites: github.com (for git), Google Code (supports svn and Hg), launchpad.com (for bzr), sourceforge.net (for everything), gitorious.com (git), bitbucket.com (Hg), lots of others.

Also, The MAT servers have git and svn installed. Larry Zins can set up a project hosting repo and a front end wiki/bug-tracker called "Trac" if you want.

git

We will all be using **git** in this course.

git is a fast, distributed VCS, also known as a DVCS (distributed version control system), or SCM (source control manager)

"Distributed" = there is not necessarily one single place that controls the project.

Can work on your code offline, keep track of all your changes, and then push it to the other "clones" of the project when you are back online

makes it very easy to merge changes to the project by different users, even if they are working on the same file

General workflow:

pull from remote repo ➔
edit local code ➔
commit to local repo ➔
push to remote repos ➖

git - Basic Commands

clone a repository:

```
> git clone git://github.com/angusforbes/MAT201B_F2010.git
    //downloads a copy of the course website repository into a local directory
```

add a new file "newfile.txt" to the staging area of the local repo:

```
> git add "newfile.txt"
```

commit the file to the local repo:

```
> git commit "newfile.txt"              //open an editor for you to describe changes
> git commit -m "my message" file.txt   //commits file with a message describing changes
```

at anytime you can check the status of your local repository:

```
> git status
    //will tell you if there are files in the repo that are not in the staging area,
    //if there are files in the staging area that are not committed,
    //and how out-of-synch your local repo is from where you cloned it from
```

push your changes to the remote "origin" repository (if you are allowed!):

```
> git push origin
```

pull other peoples changes to your local repository:

```
> git pull origin
```

github.com

Github.com:
- A project hosting site for managing git repositories
- Free for smaller open source projects
- Includes an issue tracker
- Has some social networking features...

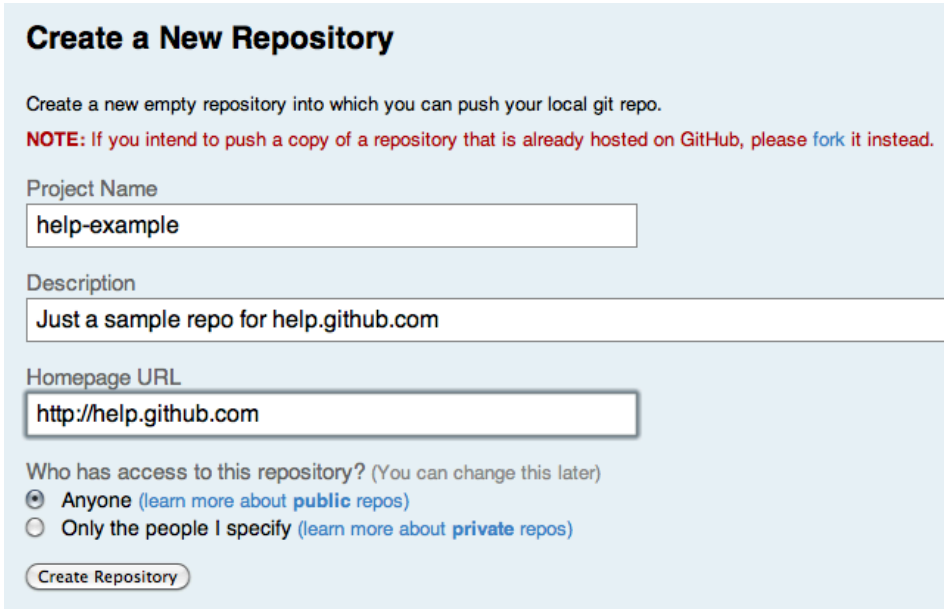
a) Create an account

b) Associate an ssh-key with your account (http://help.github.com/mac-key-setup/)
- Create a key:
  > ssh-keygen -t rsa -C "yourname@email.com"
  //press return until finished (we will explain and use a passphrase later),
  //you will get the following message:
  //"Your public key has been saved in /Users/yourname/.ssh/id_rsa.pub."
  > man ssh-keygen //if curious, or see references on syllabus for info
- Tell github about the key
  go to "Account Settings" → SSH Public Keys → Add a public Key
  copy the contents of ~/.ssh/id_rsa.pub into the text box

c) Create a Repository

Click on "New Repository" from the front page and follow instructions...

github.com


Create a New Repository
Create a new empty repository into which you can push your local git repo.
NOTE: If you intend to push a copy of a repository that is already hosted on GitHub, please fork it instead.
Project Name
help-example
Description
Just a sample repo for help.github.com
Homepage URL
http://help.github.com
Who has access to this repository? (You can change this later)
Anyone (learn more about public repos)
Only the people I specify (learn more about private repos)
Create Repository


Global setup:
Download and install Git
git config --global user.name "Tekkub"
git config --global user.email tekkub@github.com
Next steps:
mkdir help-example
cd help-example
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@github.com:tekkub/help-example.git
git push origin master
Existing Git Repo?
cd existing_git_repo
git remote add origin git@github.com:tekkub/help-example.git
git push origin master
Importing a Subversion Repo?
Click here