Teaching Open Source

Ralph Bean

Oct 08, 2018

Contents

CHAPTER 1

Syllabus

1.1 Projects Seminar in FLOSS Game Development

- Syllabus - http://ritfloss.rtfd.org/ – (subject to change)
- Course Number - 4080.590.01
- Room - Orange Hall, Room 1380 (013-1380)
- Tuesday, Thursday – 10:00am-11:50am
- Instructor - Ralph Bean <rjbpop@rit.edu>
  - Office: Building 17, Room 3110 (17-3110)
  - Office Hours: Friday, 1:00pm-3:00pm
- IRC - irc.freenode.net, `#floss-seminar`
- Email list - floss-seminar@lists.rit.edu
- Blog Planet - http://threebean.org/floss-planet
- The source for this syllabus can be found at http://github.com/ralphbean/tos-rit-projects-seminar

1.2 Text Books

You can `download the textbooks here`.

1.2.1 Casual Game Design

Casual Game Design: Designing Play for the Gamer in ALL of Us.

1.2.2 Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript

Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript.

1.3 Goals of the course

Having taken this course, students should be able to:

- Demonstrate competence with modern FLOSS development tools and conventions (git, public forges, unit tests, bug trackers, wikis, etc..).
- Demonstrate competence with modern web development technologies (HTML5, Javascript, CoffeeScript, CSS3, etc..).
- Document their work progress, accomplishments, and pitfalls by way of weekly blog posts.
- Work with and contribute to existing open source projects.
- Build and manage a new project using open source tools.
- Deploy a web application to the cloud.
- Have a fun, open-source web-based game for their portfolio and/or to show off to their friends.

1.4 The spirit of the course

While still a course where you will receive a letter grade, the spirit of the course is intended to be both *open* and *fun*. This is a seminar course, so an experimental approach will be taken.

An *open* course – students will have access to the 'document source' for the syllabus and grading rubric. While you are reading *the syllabus* right now, as a student of the class you have a right to fork the upstream repository, make modifications, and submit patches for review. Barring a troll festival, this can create a fun, dynamic environment in which the course curriculum can develop by the very same mechanism being taught during the quarter (community-driven).

A *fun* course – while the primary deliverable for the course is a working web-based game, we are going to subject the course itself to *gamification*. Instead of grading students' final projects individually, projects will be pitted against one another through a scheme developed by the students themselves, called the final_project_rubric.

For example, one way this could work is through simple accumulation of weighted point values awarded for the presence of certain features: a game that works in all modern browsers as well as on mobile devices gets +5 points, a game that includes a velociraptor gets +3 points, etc…

1.5 Licensing

All code developed by students in the course must be licensed (by the student) under any one of the licenses approved by the open source initiative.

Your code that you write is your code, with which you can do what you will; true. However, if you're unwilling to license code you write for an open source course with an open source license, you're in the wrong course.

1.6 Schedule

Week	Day	Topic	Reading	Assigned	Due
1	1	Introductions, Syllabus, Mailman, IRC, git, github	*The Syllabus*	*Homework - First Flight*	
	2	Bugfix deep dive	*The Open Source Way*	*Homework - Bugfix*	
2	1	Casual Games: Matching, Sorting, and Seeking	*Casual - Week 2*		*Homework - First Flight*
	2	Introduction to HTML5	*Isometric - Week 2*	*HTML5 - Programming Assignment #1*	
3	1	Casual Games: Managing, Hitting, and Chaining	*Casual - Week 3*		
	2	Audio, WebWorkers, and CoffeeScript	*Isometric - Week 3*		
		Holiday Break			
4	1	Pitch Session : Talk about your game.			*HTML5 - Programming Assignment #1 Homework - Bugfix*
	2	Paper Prototypes : Lecture and Build			
5	1	Paper Prototypes : (con't) Project Decisions		*Homework - Rubric*	
	2	Server choices, Social APIs, and *le Cloud*. (#openshift)	*Isometric - Week 4*	*HTML5 - Programming Assignment #2*	
6	1	**No class**			
	2	Casual Games: Constructing, Socializing, and Physics	*Casual - Week 4*		*Homework - Rubric*
7	1	Digital Prototype : Build			
	2	Digital Prototype : Play			
8	1	Digital Prototype : Report and Revise			*HTML5 - Programming Assignment #2*
	2	Guest Lecture			
9	1	Digital Prototype : Build			
	2	Digital Prototype : Report and Revise			
10	1	Play Testing/Development			
	2	Play Testing/Development			
11	?	Final Presentations			

1.7 Required Reading

1.7.1 The Syllabus

- You're reading the syllabus right now. It is posted at http://ritfloss.rtfd.org/

1.7.2 The Open Source Way

- What they didn't teach me in college
- How to Start Contributing to Open Source Projects
- Understanding Open Source Licensing
- Revitalizing Computing Education Through Free and Open Source Software
- Why Open Source Misses the Point of Free Software

1.7.3 Casual - Week 2

- casual
  - chapters 1-6 (139 pages). It's light reading, trust me.

1.7.4 Isometric - Week 2

- *Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript*
  - chapters 1-3 (65 pages). This reading is not quite so light.

1.7.5 Casual - Week 3

- casual
  - chapters 7-9 (36 pages)

1.7.6 Isometric - Week 3

- *Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript*
  - chapter 4 (18 pages)

1.7.7 Casual - Week 4

- casual
  - chapters 10-12 (56 pages)

1.7.8 Isometric - Week 4

- *Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript*
  - chapter 5 (25 pages)

1.8 Grading

Assignments are due at midnight of the day they are marked as due.

Late submissions will be deducted 10% per day they are late.

Your final grade for the quarter will be derived from the following weights.

Component	Weight
In-Class Participation	10%
FLOSS Dev Practices (Blogging, patching, writing, IRC)	15%
Homework Assignments	10%
Programming Assignments	15%
Paper Prototype	10%
Final Project	40%

Class partitipation is speaking in class, answering questions, etc. . .

Blog updates – students are required to keep a blog to which they post updates about their investigations, progress, success, and pitfalls. This blog can be hosted anywhere, but must be added to the course planet (there are instructions on how to do this in *Homework - First Flight*).

- You must make at least one blog post per week to receive full credit.
- You must participate regularly in the course's IRC channel: asking and answering questions.
- You must participate in the course's mailman list, floss-seminar@lists.rit.edu.
- Contributions to the course curriculum, syllabus, and rubric are factored in here as well.

Blogging is good for you and good for the FLOSS community at large.

The *homework assignments* are listed in the syllabus. You will be able to complete some of these in class.

Programming assignments are more in depth, but will amount to two deliverables derived from one of the course's two textbooks, Making Isometric Social Real-Time Games with HTML5, CSS3, and Javascript.

There are two assignments:

- *HTML5 - Programming Assignment #1*
- *HTML5 - Programming Assignment #2*

Students' *paper prototypes* are presentations to the rest of the class on their idea for their game, *before a single line of code is written*.

These are 'play sessions'. You will need to bring some playable version of your game so we can all try it out. For instance, if you're thinking about a first-person-shooter, come with a set of rules for playing 'pointing tag' and we'll all really play it, in person.

The rest of the students will comment on your prototype. Take notes and:

- Use them to improve your design
- Turn in a copy for your grade

Your *final project* will be the culmination of the quarter's work and will be graded according to the final_project_rubric.

Additionally, graduate students are expected to complete some extra work as described in hw/gradproj.

1.9 Lightning Talks - Extra Credit

Every Tuesday for the first portion of class, any student has the opportunity to give a lightning talk on a topic of their chosing. Your lightning talk must be less than 5 minutes in length and must be at least remotely related to the course material.

You will receive +1 extra credit points towards your final grade for every lightning talk you give. Only the first three lightning talks offered will be allowed during a given class. Talks will be chosen from among those offered by students on a FIFO basis.

CHAPTER 2

Notes for Class Sessions

2.1 Week 01, Day 1: First Flight

- Introductions
- Covering the Syllabus
- *Homework - First Flight*

2.2 Week 01, Day 2: Guided Bugfix

- Lightning Talks?
- Review last class (*Homework - First Flight*)
- Review Schedule (When are homeworks due, how are we with the reading schedule?)
- Guided Deepdive into pandas
- Talk about *Homework - Bugfix*

2.3 Week 02, Day 1: Matching, Sorting, and Seeking

- Graduate student proposals are due
- *Homework - First Flight* is due. How was it?
  - Review http://threebean.org/floss-planet
- Slides - http://prezi.com/1qe6g-pvye_q/floss-games-matching-sorting-and-seeking/

2.4 Week 02, Day 2: Introduction to HTML5

- Lightning Talks
- Introduction to HTML5
  - The book
    - clone from github - https://github.com/ralphbean/Making-Isometric-Real-time-Games
    - examples/ex2-fps-requestAnimationFrame.html
    - examples/ex13-isogrid-buildings.html
    - examples/ex14-gui.html
      - Modernizr.js
  - jQuery
  - spritely
    - Here's the spritely source code
  - DuckHunt
- Javascript Game Frameworks
  - gameQuery - http://gamequery.onaluf.org/
  - limeJS - http://badassjs.com/post/3200945950/limejs
  - melonJS - http://www.melonjs.org/
  - processingJS - http://processingjs.org/
  - akihabara - http://www.kesiev.com/akihabara/
  - effect - http://www.effectgames.com/effect/
- *HTML5 - Programming Assignment #1*

2.5 Week 03, Day 1: Managing, Hitting, and Chaining

- Managing
  - Diner Dash http://www.playfirst.com/game/dinerdash
  - Cake Mania http://www.bigfishgames.com/download-games/898/cakemania/index.html
  - Insaniquarium http://www.popcap.com/games/insaniquarium/web
    - Requires ActiveX
- Chaining
  - Revisit,
    - Diner Dash
    - Insaniquarium
    - Tetris
    - Scrabble
- Hitting

  - Whac-a-mole vs Wii Tennis

2.6 Week 03, Day 2

Class was cancelled for the STEM/CSI hackathon!

2.7 Week 04, Day 1

- Welcome back from break.
- Homeworks due. How'd it go?
- Game Pitches

- `<audio>` tags
- WebWorkers
- CoffeeScript
  - Online interpreter
  - Observations
    - Python style whitespacing
    - Ruby styled lightweight syntax
    - Concise function declarations
    - JSLint approved
    - Class based inheritance
    - Comprehensions!
  - Hangman

2.8 Week 04, Day 2 - Paper prototypes

- Paper prototypes

2.9 Week 05, Day 1 - Settling on projects

- Paper prototypes revisited.
- Decide on top three projects.
- Votes -
  - How many per team?
    - 2 teams of 6
    - 3 teams of 4

        * 4 teams of 3
    - Which games? Vote on the clipboard site.
        * Rabenvald - Robocode++
        * kaeedo - Eco
        * PhilMoc - Haunted House
        * JaceTwice - Arrangamajig
        * Crystick - Gold Rush
        * LakeEffect - Helicopter Race
        * trose/decause - FOSS
        * Lo-Rin - Dragonfire + Maths
        * rossdylan - Pip3z!!1
        * Qalthos - Myst
        * Chips545 - Moar LaZ0rs
- *Homework - Rubric* assigned.

2.10 Week 05, Day 2 - Openshift

- Revisit last class
    - Teams and *Homework - Rubric*
- Due homeworks
    - *Homework - Rubric* due next Thursday.
    - *HTML5 - Programming Assignment #2* due the Tuesday after that.
- Class this coming Tuesday will be a working session on *HTML5 - Programming Assignment #2*.
- Walk through *HTML5 - Programming Assignment #2*

2.11 Week 06, the Valley of the Shadow of Openshift

:(

2.12 Week 07, Day 1: TurboGears

Setting up your environment (on `typhon.csh.rit.edu`):

```
$ virtualenv ~/myenv
$ source ~/myenv/bin/activate
$ pip install tg.devtools Pylons==1.0 WebOb==1.0.8
$ paster quickstart roflapp

# Yes you prefer mako templates
```

(continues on next page)

(continued from previous page)

```
# Yes you need authentication

$ cd roflapp
```

At present, the current release of TurboGears doesn't know it, but it needs

- Pylons==1.0
- WebOb==1.0.8

```
$ python setup.py develop
$ paster setup-app development.ini
```

Since we're on a shared machine `typhon.csh.rit.edu`, we'll need to pick different ports to serve our respective roflapps on. Edit `development.ini` accordingly.

Once you've made your edits, serve your app with:

```
$ paster serve --reload development.ini
```

2.12.1 Understanding Modern Web Frameworks

It's all about MVC – model, view, controller. Modern frameworks separate your code out into these three distinct, yet interdependant chunks.

- *model* (`rolfapp/model/*.py`) - contains all the database-related code
- *view* (`roflapp/template/*.mak` and `roflapp/public/*`) - contains all the presentation-related code, html markup, css, javascript, etc.
- *controller* (`roflapp/controllers/*.py`) - all the control-logic (or *business* logic). Who can access what urls? Validation of data? Did you win an iPad?

If you look inside `roflapp` you'll see these directories and a few other secondary ones.

1. Add `roflapp/public/testing123.html` and browse to `/testing123.html`.
2. Edit `roflapp/templates/index.html` and browse to `/`.
3. Edit `roflapp/controllers/root.py`. Edit the `def index(..)` method to return a random number. Display it in the template.
4. Look at `roflapp/model/`. Edit `roflapp/controllers/root.py` to return the number of users.
5. Throw an exception.
6. Use `tg.flash()`.

2.13 Week 07, Day 2: More TurboGears - AJAX - Back to the Cloud - Facebook

1. Edit `roflapp/controllers/root.py`.

   1.1) Add one method that JSON returns info about users. 1.2) Add another method that JSON returns {'success': True} but adds a new user

2. Add `roflapp/public/javascript/rofl.javascript`.

   1.1) Add one function that given JSON, updates the DOM. 1.2) Add one function that queries the `/query_users` URL. 1.3) Add one function that POSTs to create a random user. 1.4) Add $(document).ready(..) to kick it all off.

2.13.1 Facebook, if we have time

1. Look at hanginwit-threebean for the example. In particular, check out `auth-fb.coffee`.

2.14 Week 08, Day 2: Facebook Auth

External docs for Facebook:

- get an appID - https://developers.facebook.com/apps
- general docs on fb auth - http://developers.facebook.com/docs/authentication/

Modifications to your openshift app:

- tg2app/models/stuff.py - https://gist.github.com/1779952
- tg2app/controllers/root.py https://gist.github.com/1779931
- tg2app/templates/waiting.mak - https://gist.github.com/1780020
- tg2app/public/js/waiting.js - https://gist.github.com/1779989
- tg2app/lib/base.py - https://gist.github.com/1780206
- tg2app/templates/master.mak - https://gist.github.com/1780188
- tg2app/public/js/auth-faked.js - https://gist.github.com/1780093
- tg2app/public/js/auth-fb.js - https://gist.github.com/1780065

2.15 Week 09, Day 1

Presentations!

2.16 Week 09, Day 2: Facebook Graph API

- https://developers.facebook.com/tools/explorer
- http://developers.facebook.com/docs/reference/api/
- http://threebean.org/gitlog.html

CHAPTER 3

Helpful Hints – A list of external resources

3.1 git

- git cheat sheet

3.2 vim

- vim cheat sheet

CHAPTER 4

README.rst – Tools for teaching the open source projects seminar @ RIT

This is an all-purpose repository for storing some content, but mostly tools for teaching the open source projects seminar @ RIT.

Future tools could include things like scripts to produce blog/commit/unittest statistics. This is also a place the syllabus could live, where students could fork and produce pull requests.

4.1 Setting up your environment

Before you can do anything with this (build the documentation or run any of the scripts) you'll need to setup and activate a python virtualenv. Run the following at the command prompt...

4.1.1 On Linux/Mac OS X

```
$ virtualenv --no-site-packages -p python2 sphinxenv
$ source sphinxenv/bin/activate
$ git clone git@github.com:YOUR_USERNAME/tos-rit-projects-seminar.git
$ cd tos-rit-projects-seminar
$ python setup.py develop
```

4.1.2 On Windows

At the windows command prompt:

```
$ virtualenv --no-site-packages -p python2 sphinxenv
$ sphinxenv/Scripts/activate.bat
```

In msysGit or git-bash:

```
$ git clone git@github.com:YOUR_USERNAME/tos-rit-projects-seminar.git
```

Back in the windows command prompt:

```
$ cd tos-rit-projects-seminar
$ python setup.py develop
```

4.2 Building the "Documentation"

The "documentation" for the course (the syllabus, all the homework assignments, notes on the lectures) are all kept in the `doc/` directory of this repository. The files all end with the extension `.rst` which is the file extension for the reStructuredText markup language. They are all furthermore tied together by the *sphinx* framework for building integrated docs.

You might notice that the syllabus, et. al. is hosted on http://readthedocs.org/. The upstream github repository has a hook installed that automatically triggers a `git pull` at http://readthedocs.org from http://github.com. Thus, every time we change the docs here, they are automatically re-built into HTML for us and posted online. Awesome!

This however means that we should be careful before we push anything to github, or it will 'go live'. To be careful, you should rebuild the documentation locally (on your machine) to check that whatever modifications you made to the `.rst` files actually renders into the HTML that you want.

In order to do that, first make sure you have your virtualenv activated.

Being certain of that, in the root directory, simply run:

```
$ sphinx-build -b html doc html-output
```

The html documentation will be generated in `html-output/`. Check `html-output/html/index.html` to see if it exists.

Note: If your machine complains that 'sphinx-build' is a command that could not be found, try running "$ python setup.py develop" in the root of the tos-rit-projects-seminar repository first. That `setup.py` file contains information about all *other* open source projects that are *required* for this project, and will automatically install them from http://pypi.python.org/

4.3 Validating the `data/students.yaml` file

The `data/students.yaml` file is a structured data file that keeps track of all the students in the class and metadata about them. Using this file and the bindings in `lib/ritfloss/model/students.py` we can build scripts that count how many lines of code each student modifies each week, or how many words/blogpost, or whatever we like.

The data format (YAML) can be a little prickly though. It is *whitespace-sensitive*, meaning that how many spaces you put before an entry on each line has an impact on how the data is interpreted. It also means that tabs and spaces are distinctly different in their meaning. It also means that editing such a file is easy to mess up.

In order to ensure that you don't introduce any unparseable errors into the file, there is a script in `lib/ritfloss/model/validate.py` that reads in the file and checks each entry. You should run it after every time you edit `data/students.yaml`.

In order to run the `validate.py` script, make sure you have your virtualenv activated.

In the root of the cloned source directory, run:

```
$ python lib/ritfloss/model/validate.py
```

CHAPTER 5

Homework - First Flight

The purpose of this homework assignment is to introduce students to their first FLOSS practices. Read it in full, there are a number of graded deliverables.

The due-date is listed in the *Syllabus*.

5.1 Fill out the survey

***Tasks*:**

- Your first task is to fill out the survey located here: https://clipboard.rit.edu/take.cfm?sid=E958FABE

It's anonymous, but the results will be published back to the whole class so we have a feel for the technical level and preferences coming into the course.

5.2 IRC

IRC is one of the primary means of communication for a FLOSS community, particularly for *informal* communication.

There is a course *IRC* channel on `irc.freenode.net`. The channel is `#floss-seminar`. Communicating regularly in *IRC* factors into the *FLOSS Dev Practices* component of your final grade.

***Tasks*:**

- Download and install an IRC client on your development machine.
  - Windows: mIRC
  - Mac OS X: Colloquy
  - Linux: irssi
- Choose a nick and register yourself with the NickServ.
- Connect to `#floss-seminar` on `irc.freenode.net` and introduce yourself.

  - The instructor's nick is `threebean`.

It is a good practice to "hang out" in IRC channels of projects that you use and especially of projects that you contribute to. Here you can find early alerts regarding any upcoming major changes or security vulnerabilities. It is also the easiest (lowest overhead) method for getting your questions answered.

Note: Only for the brave – if you want to be completely awesome, you can setup a proxy node so you are always logged in. People can leave you messages this way.

If you want to be completely completely awesome, you can setup BitlBee so you can tweet from your IRC client.

5.3 Mailman

Discussion mailing lists are a more formal mechanism of communication for FLOSS projects. More formal than *IRC*, less formal than bug trackers. Discussion mailing lists are often used to ask questions, announce upcoming releases and beta tests, and to debate redesigns and refactors. The advantage here is that mailing lists are typically archived and indexed by Google; discussions that should be preserved for posterity should occur here.

There is a GNU Mailman discussion list for the course hosted by RIT.

***Tasks*:**

- Subscribe to it at https://lists.rit.edu/mailman/listinfo.cgi/floss-seminar
- Write your first email to floss-seminar@lists.rit.edu, introducing yourself. Include your name, major, hometown, and favorite color.

Communicating regularly over the course *mailman* list (asking and/or answering questions) factors into the *FLOSS Dev Practices* component of your final grade.

5.4 Blogging

Setup a blog if you don't have one. Much like mailing lists, blogs are archived, indexed by Google, and therefore preserved for posterity. When you encounter a technical challenge, typically you google for a solution and you typically find that solution in a blog post of some developer who has run into a similar situation. Blogging about your attempts, successes and failures (and writing tutorials!) is a best practice for increasing the general body of searchable knowledge available, for increasing the Wisdom of the Ancients.

Blogs around a topic are also typically aggregated by a planet (an RSS feed aggregator). This way, all developers blogging about `Project X` can have their blog posts fast-tracked to a readership subscribed to `Planet X`. For instance, here's a link to Planet Python.

The Planet for the course is hosted at http://threebean.org/floss-planet/. There are instructions for how to subscribe your blog to it in the *Patch the Course Project* section below.

You must create a blog (if you don't have one already) and write at least one post per week about your progress, attempts, successes, failures, reflections, and/or all of the above.

***Tasks*:**

- Create a blog if you don't already have one. There are lots of free services available. You might try http://wordpress.com or http://blogspot.com.
- Write an introductory post relevant to the course. The topic is your choice!

5.5 github

Code *forges* are service sites around which FLOSS development orbits, some of the more popular sites are github, bitbucket, sourceforge, and launchpad.

For your own enlightenment, review the following comparisons of the different forges:

- Timeline
- Metadata
- Artifacts
- Features
- Revision control
- Policies

You'll need to create your own account on github.com. All development for this course should be tracked on that forge. Github is, after all, the most popular forge.

***Tasks*:**

- Create a github account if you don't already have one.

5.6 Patch the Course Project

Check out the source repository for this course; it's hosted at https://github.com/ralphbean/tos-rit-projects-seminar.

Inside the repository, we'll keep an index of all the students in the course and metadata about them (you!).

***Tasks*:**

- Load up the git cheatsheet listed at *Helpful Hints – A list of external resources* and keep it nearby.
- Work through this git tutorial if you don't have any experience with git.
- Fork the repository (link to github help on this).
- Clone a local copy.
- Follow the instructions in `README.rst` to setup your environment.
- Edit the file `data/students.yaml`. Perhaps obviously, it is a YAML file. Add yourself to the file with the necessary keywords.
- Verify that you added yourself correctly by running the script located at `lib/ritfloss/model/validate.py`
- Edit the file `planet/config.ini`. Look at the very bottom of the file and there will be the beginnings of a list of subscribed blogs. Add your blog's RSS feed (or a topical sub-feed) to this list. Make sure its a working RSS URL! (Once the patch is accepted upstream and pushed to production, this should add your blog feed to the course planet.)
- If everything checks out, then
  - Commit your change
  - Push to your github repository
  - Issue a pull request through the web interface.

CHAPTER 6

Homework - Bugfix

Real learning in computing comes more from doing and less from studying. Debugging, figuring out how some software works and how it doesn't, is an interactive process that develops basic engineering practices and, in the open source context, community engagement and collaboration.

6.1 Pick a Project

You can choose any project you like. The best to pick is something you *already use*, something with which you're already familiar. If you can't think of any projects to investigate off the top of your head, here's a list of suggestions.

- The Battle for Wesnoth
- js.io
- The Linux Kernel (bugzilla is down, you could try looking here
- liveusb-creator
- The Mozilla Project
- node.js
- pandas, a data manipulation library
- Pyramid (web framework)
- Toscawidgets 2 (web widgets)
- Turbogears 2 (web framework)
- The Ur-Quan Masters
- The ManaWorld MMORPG for Linux

A little more focused, here is a list of open source javascript/html5 game engines and links to their bug trackers.

- gameQuery
- limeJS

- melonJS
- akihabara
- effect

You might also find something neat over at bounty sites like:

- Gun.io http://gun.io/

Really, the sky is the limit here.

Note: For background, you might want to also check out the project on http://ohloh.net/. It can help you characterize what kind of community orbits around your choice.

6.2 Find a bug

A bug can be anything: an unintended side-effect in a low-level routine, a user-interface cleanup, a feature enhancement, grammatical errors or lack of clarity in the project's documentation.

Broadly, you have two different options here. You can

- Find a known bug (or feature enhancement) listed in the project's bug tracker.
- Find a bug yourself by using the software.

In the event of the second case, make sure you file the bug in the project's tracker before proceeding.

6.2.1 OpenHatch.org

OpenHatch.org bills itself as an "Open source Involvement Engine." It's mission is *explicitly* to reach out to communities of developers just like the one in this very course, and make it trivial for new contributors to effectively cycle on upstream projects!

You can read more about OpenHatch and it's vision here on their wiki

For this homework assignment, you should check out:

- OpenHatch, particularly
- The OpenHatch Volunteer Opportunity Finder for Bite-sized Bugs
- If you want *all* the bugs, you can find them through OpenHatch's Search

6.3 Use the Source, Luke

Once you've identified a bug that needs fixing, you'll need to get ahold of the source. In most cases, the code for a project will be hosted on a forge and the process of forking and cloning the source will be straightforward. If you forget how to do this for github, you can refer to *Homework - First Flight*.

For whatever project you've chosen, you should ask that project's community for help. Look for *IRC* channels and project mailing lists. You'll be communicating with developers who have a lot on their plate so make sure to be polite and leave your ego at the door.

Find the code related to the bug; use whatever code navigation tools you're more familiar with. The instructor's favorite method is: `grep -inr "some string" project_src/`.

Fix the bug, this may require some thinking, and some more asking around.

Test your fix! Project maintainers hate nothing more than receiving a patch that breaks every other function of the project. Often, projects have built-in test suites. If yours does, run it!

Prepare your patch with descriptive commit messages. Follow the method for submitting patches recommended by your project and submit!

Make sure the project community can easily understand what you did and why you did it.

Make sure there is a reference in the tracked bug ticket to your patch (that is, if the project maintains a bug tracker).

6.4 The Deliverable

Write a blog post about this process and provide relevant links where possible to documentation.

- A link to the patch(es) hosted somewhere on the web, usually forges provide the ability to link to changesets.
- A link to any mailing list discussions archived on the web
- Snippets of any relevant IRC conversations.

You will be graded on your post and the explanation of your process. Extra kudos (but not extra credit) for super epic patches.

6.5 An Afterthought (not required)

Once your patch has been accepted, mosey on over to http://ohloh.net.

- Create an account
- Find the project you patched
  - If it doesn't exist, you can add it yourself
- "Claim your position" as the author of the commit(s) you sent in to increase your rank among open source developers of the world!

CHAPTER 7

HTML5 - Programming Assignment #1

For this assignment you'll be combining examples 13 and 14 from the book into a tech demo (still not actually a game).

- Example 13
- Example 14

While they are both good examples that demonstrate some of the basic features of the `<canvas>` element and its 2d context, they suffer from a few deficincies. Fix these as you rewrite and combine them. Do as you see fit, you'll be expanding on these examples for the second homework (*HTML5 - Programming Assignment #2*), so better to code well now than suffer through bad code later.

Use FLOSS development practices while working on this. From the start, keep your code in a git repository linked to github. "Commit early and commit often." You'll need to create a new repository on github for this.

As far as combining the examples goes, your demo will need to feature the ability to:

- Place and destroy buildings.

As far as fixing up deficiencies, you need to:

- Move all javascript out of the `<script>` tags and into its own `.js` files. Your main page can still have one or two javascript make calls to initialize your demo.
- Convert the code from a half-procedural/half-object-oriented pattern to be fully object-oriented. You should have differenet objects here and not just cram everything into the `Game` object from example 14.
- Replace the event handling and listening with jQuery.
- Replace the style attribute manipulation on DOM elements outside the `<canvas>` element also with jQuery.
  - For example, the lines where they set `className = null` and use `.setAttribute(...)` are so five years ago.

7.1 Deliverable

Write a blog post explaining what changes you made and why you made them.

Simply include a link to your github repository for proof of your work.

7.2 Optional - use a framework

You may, if you like, move ahead and use a javascript game framework instead of jQuery and the examples from the book. Your project will need to have the same functionality as if it were composed of the two examples, but this will require a larger rewrite (although with much cleaner, denser code). Definitely make note of this in your blogpost if you choose this route.

Here are some examples of frameworks you could use:

- gameQuery - http://gamequery.onaluf.org/
- limeJS - http://badassjs.com/post/3200945950/limejs
- melonJS - http://www.melonjs.org/
- processingJS - http://processingjs.org/
- akihabara - http://www.kesiev.com/akihabara/
- effect - http://www.effectgames.com/effect/

CHAPTER 8

Homework - Rubric

An excerpt from *Syllabus*:

> An *open* course – students will have access to the 'document source' for the syllabus and grading rubric. While you are reading *the syllabus* right now, as a student of the class you have a right to fork the upstream repository, make modifications, and submit patches for review. Barring a troll festival, this can create a fun, dynamic environment in which the course curriculum can develop by the very same mechanism being taught during the quarter (community-driven).
>
> A *fun* course – while the primary deliverable for the course is a working web-based game, we are going to subject the course itself to *gamification*. Instead of grading students' final projects individually, projects will be pitted against one another through a scheme developed by the students themselves, called the ../final_project_rubric.

Part of the experience of being an open source developer is the *instrinsic motivation* that drives you to build open software. That motivation can be different for each person. Its not something that is ever taught and is difficult to develop into a course curriculum for sure!

Your assignment here is to take part in creating ../final_project_rubric.

You should have divided up into your teams for the final project by this point. Do the following with your github accounts to setup both for the final project and for this homework:

- Designate one person from your team whose github account will be the *primary* account for your team. The primary designee will be responsible for merging *pull requests* from their teammates both for the final project and for this homework.
- All other members of your team should:
  - Delete their `tos-rit-projects-seminar` repositories on github.
  - Fork the `tos-rit-projects-seminar` repo from their team's *primary* account (the designee's account).

You will be responsible in this homework for forking your team's repo as described above, committing patches to the document (../final_project_rubric), pushing to your own github repo and issuing pull requests. You may add to the document as well as delete from it.

Although open source development is typically thought of as a *cooperative* mode of production, some participants derive their motivation from a competitive outlook towards their peer developers. As an experiment, you will all be graded in competition with one another for this assignment.

- There are 100 possible points.
- The student with the highest *impact* will receive 100 points.
- All other students who contribute patches will receive a *weighted* grade between 100 and 75 points based on their *impact*.
- Students who submit no patches will receive 0 points.

Your *impact* is defined as the number of lines added + the number of lines deleted. You can see a graph of your *impact* so far here.

Good luck! And make an awesome *rubric* worthy of awesome projects!

CHAPTER 9

HTML5 - Programming Assignment #2

9.1 Required technology overview

- Take a look at openshift `rhc` tools.
  - Create an openshift account.
  - Install them using these instructions.
    - Follow only steps 1 and 2.
- Take a look at lmacken's openshift-quickstarter
  - Use it to create your first app.
- Take a look at the Facebook API.
  - Register your site.
  - Use the API to auth against your app, get friends lists.

9.2 Assignment

For this you'll be taking your product from *HTML5 - Programming Assignment #1*, hosting it on the cloud, and connecting it to Facebook authentication.

Requirements:

- Your app must be hosted on http://YOURAPP-YOURDOMAIN.rhcloud.com
- All client-side code must be written in coffee-script.
- Users must login with their Facebook account.
- Whatever map they work on must be saved to a database.
- Whatever map they were working on must be re-loaded once they login.

Extra Credit

- Share maps with other users on Facebook

Deliverable

- Write a blog post including a link to your game and its source on http://github.com