Lecture 04 - flux

Philip Caplan

CSCI 0422 - Geometric Modeling (Spring 2022)
Learning objectives

By the end of this lecture you will be able to:

- commit, push to and pull from your flux repository,
- configure and compile the flux library, develop and run unit tests,
- compile and run executables where we will develop our class work,
- debug the code you write in the flux framework.
Getting started

You don’t need to LiveShare just yet: both group members should obtain a copy of the repository.

- Sit next to your partner (see https://gitlab.com/csci422-s22/flux-base/-/tree/main/notes/class04 for your group number and partner).
- Navigate to where you would like to save your flux repository (this is where you will work for the rest of the semester).
- Clone the repository (replace N with your group number, e.g. group03):

  ```bash
  $ git clone https://gitlab.com/csci422-s22/groupN/flux.git
  ```

- Build the library and tests:

  ```bash
  $ cd flux
  $ mkdir build
  $ mkdir build/debug
  $ cd build/debug
  $ cmake ../
  $ make
  ```

  You can use `make -j 4` (or more) if you want to speed up the build.
Some practice with git.

<table>
<thead>
<tr>
<th>Group</th>
<th>Host</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Adam</td>
<td>Issy</td>
</tr>
<tr>
<td>02</td>
<td>Annika</td>
<td>Nick</td>
</tr>
<tr>
<td>03</td>
<td>Ben</td>
<td>Peter</td>
</tr>
<tr>
<td>04</td>
<td>Cynthia</td>
<td>Nellie</td>
</tr>
<tr>
<td>05</td>
<td>Dan</td>
<td>Sam</td>
</tr>
<tr>
<td>06</td>
<td>Ian</td>
<td>Sammy</td>
</tr>
<tr>
<td>07</td>
<td>Jack</td>
<td>Steven Shi</td>
</tr>
<tr>
<td>08</td>
<td>Jay-U</td>
<td>Mead</td>
</tr>
<tr>
<td>09</td>
<td>Otis</td>
<td>Mihir</td>
</tr>
<tr>
<td>10</td>
<td>Noel</td>
<td>Steven Jin</td>
</tr>
<tr>
<td>11</td>
<td>Ruben</td>
<td>Youssef</td>
</tr>
<tr>
<td>12</td>
<td>Siri</td>
<td>Toby</td>
</tr>
</tbody>
</table>

**Host:** (within some build directory)

- `$ make update`
- `$ git status`
- `$ git commit -a -m "updated flux-base"
- `$ git push`

**Client:** (somewhere within flux)

Wait for host to push! Then

- `$ git pull`
A tour of flux!

- **src**: any .cpp files you add here will be compiled into the library
- **src/flux-base**: git submodule containing core functionality
- **projects**: where you will develop your projects
- **test**: where you will write your unit tests during the semester
- **exercises**: where in-class exercises will be developed
- **data**: where some example meshes are saved

Whenever you need to update flux-base:

- `$ cd build/debug` or `build/release` or some other build directory
- `$ make update`
- usually, you will need to re-run CMake after updating:
  `$ cmake .` (from within a build directory)
Developing and running an in-class exercise

- navigate to your build directory
- $ make template_class04_flux
- this will copy over the template in src/flux-base/notes/class04/flux.cpp to your repository (only the host should do this).
- to build and run the template, type $ make class04_flux.

In most cases, the solution will be available after class (today it’s already available):
- $ make class04_flux_sol
- Nothing will be copied to your repository (the solution will always be in src/flux-base/notes/).
Developing and running unit tests.

You can run existing unit tests provided (and updated) in `src/flux-base/test`, e.g.:

```
$ make core_mat_ut
```

Note: directory separators are replaced with underscores, so this is in `src/flux-base/test/core/mat_ut.cpp`. Take a moment and inspect this file.

To run all the unit tests together (including ones you will write):

```
$ make unit
```
Checking code coverage

- You can run this directly on your computer, but flux is automatically configured to run a "pipeline" when you push to the repository - which we did earlier!
- Navigate to https://gitlab.com/csci422-s22/groupN/flux (replace N with your group number)
- See the readme for today’s class for instructions on how to run code coverage locally.

I have to pay for CI minutes, but I would rather pay and see you push often (within reason).
keep your unit test runtime low!
1000 minutes = $10 for our entire group (csci422-s22)
Exercise: write a unit test with your partner.

In VS Code:

- Click on the Live Share button at the bottom-left.

- Sign in with Middlebury email address even if you have a GitHub account.
- Your name should then appear at the bottom left where it previously said Live Share.
- Click on your name, you should see this:

- Host: Send the invite link to Client (via private message in Slack).
Let’s write an example unit test.

- check that 1+1 is equal to 2 using UT_ASSERT_EQUALS( value1 , value2 )
- check that 0.1*3 is close to 0.3 using UT_ASSERT_NEAR( value1 , value2 , tol )
- write a function which throws an exception via flux_assert, using
  UT_CATCH_EXCEPTION( function_call_which_throws_exception )
Let’s create a segmentation fault.

```cpp
void segfault_function() {
    std::vector<double> x;
    x[10] = 2;
}
```

Then call `segfault_function()` in your unit test. Try to run with

```
$ make example_ut
```

To debug with `gdb`:

```
$ make example_ut_gdb
```

then press r to "run".
Visualizing your first mesh!

Navigate to your build directory (e.g. build/debug or build/release):

```
$ bin/flux360 ../..data/spot.obj
```

Then click on the flux360 button at the top of the course website.
When you’re done writing your test.

Host:

```
$ git commit -a -m "your commit message here"
$ git push
```

If you need to add files you should use `git add some_file_to_add` and then `commit` and `push`.

Check your code coverage at: https://gitlab.com/csci422-s22/groupN/flux
(replace N with your group number)

Client:

```
$ git pull
```

You will take turns hosting the repository throughout the semester.
Before next class...

be curious and have fun!

- download a (free) mesh in OBJ format from TurboSquid: (look for pictures with the "free" badge)
  https://www.turbosquid.com/Search/3D-Models/free/obj
- visualize this mesh using flux360. In some build directory:
  
  ```
  $ make flux360
  $ bin/flux360 yourdownloadedmesh.obj
  ```

Then navigate to https://csci422-s22.gitlab.io/home/tools/flux360.html
- Click the export button and post the image of your mesh to the #gallery channel in Slack.
  (this will be part of Project 1)