Lecture 01 - Anatomy of a C++ program

Philip Caplan

CSCI 0422 - Geometric Modeling (Spring 2022)
Introduction
By the end of this lecture . . .

- You will know what geometric modeling is, and how we will study it in the course.
- You will be familiar with the course structure and policies.
- We will write a small C++ program that covers functions, headers, and pointers.
- You will know what you need to do before next class.
A little about me... 

- Please call me Philip.
- I’m from Montreal, studied at McGill & MIT.
- I like to hike, practice yoga, and play drums.
- I have type 1 diabetes (managing my blood sugar while teaching can be challenging so I might need to eat some candy in class).
What is geometric modeling?

Geometric modeling is all about creating and manipulating digital representations (meshes) of 3d shapes.
Why is this useful?
My research is all about meshes!
Main questions we want to ask.

- **How do we represent meshes?**
  mesh connectivity, half edges

- **How do we create meshes?**
  grids, Delaunay triangulations, Voronoi diagrams, reconstruction, constructive solid geometry

- **How do we modify meshes?**
  smoothing, subdivision, simplification, remeshing

- **What can we do with meshes?**
  parametrization, deformation, simulations

and how can we write efficient C++ code to do all of this?
All course materials will be shared via flux

framework for learning about unstructured meshing

• common code base you will use throughout the semester (exercises, projects),
• visualization done in your browser via flux360 button on course website,
• you will get experience with software development practices, including version control, writing tests, continuous integration,
• you will work in pairs of two throughout the semester:
  • same partner for all exercises and projects,
  • groups will be set at the end of the second week
  • exercises in the first two weeks will allow you to see who you might want to work with.

coursework:

• Daily exercises: I will talk a little bit, you will write code, we will write the solution together.
• Biweekly projects (roughly): C++, approximation, Delaunay, subdivision & simplification.
• Final project: open-ended - we will cover several topics that will make for good final projects.
How should we communicate with you and each other?

- Personal matter? **email** or **private message in Slack**
- Otherwise? **channel in Slack**

join our slack workspace!

https://join.slack.com/t/csci422-s22/shared_invite/zt-12us80i4n-Xo6UxKaZ9bfjQ7VPKsSI9Q

Please use the channels as much as possible:

- Please don’t send me a private message about course material or bugs.
- **announcements**: where I will post information about due dates, updates to flux, etc.
- **features**: is there something you want me to implement in flux? do I have a bug?
- **howto**: for questions about how to use certain components in flux, e.g. "How do I ... ?"
- **projects**: for project-specific questions, e.g. understanding parts of an algorithm.
- **updates**: automatically posts messages about pushes and pipelines for flux-base.
Please think of our class as a software development team.

- We work together, but we have different responsibilities.
- **Your job**: commit to the learning experience; put in your best effort to complete your assignments.
- **My job**: give you the infrastructure to complete your job, but I will not do your job for you.
- I can help you with conceptual and clarification questions, but I **cannot help you find bugs**.

When you’re stuck on a problem: **pause, brainstorm, abstract, sketch, debug**.

Your introduction to the team: Project0 is a LeetCode problem (due 02/22).
Success

Runtime: 48 ms, faster than 5.61% of C++ online submissions for Reverse Nodes in k-Group.

Memory Usage: 11.8 MB, less than 8.36% of C++ online submissions for Reverse Nodes in k-Group.

Next challenges:
- Swap Nodes in Pairs
- Swapping Nodes in a Linked List
- Reverse Nodes in Even Length Groups

Show off your acceptance: 

<table>
<thead>
<tr>
<th>Time Submitted</th>
<th>Status</th>
<th>Runtime</th>
<th>Memory</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/11/2022 09:41</td>
<td>Accepted</td>
<td>48 ms</td>
<td>11.8 MB</td>
<td>cpp</td>
</tr>
<tr>
<td>01/15/2022 11:18</td>
<td>Accepted</td>
<td>37 ms</td>
<td>11.7 MB</td>
<td>cpp</td>
</tr>
</tbody>
</table>
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Projects sneakpeek
Anatomy of a C++ program
Building a C++ program with a Makefile

- Follow along at: https://replit.com/@pcaplan/csci422-exercise01-template and click Fork repl.
- Click on Shell, then type make to build and ./program to run (or make run).

```cpp
int
main() {

    int x = 2;
    float y = 3.14;
    double z = 3.14;

    return 0;
}
```

```makefile
default:
    g++ main.cpp -o program

run:
    ./program

full:
    g++ -c numerics.cpp -o numerics.o
    g++ main.cpp -o program numerics.o

clean:
    rm -rf program
```
Before next class

- familiarize yourself with **course web page**: go/cs422 (https://csci422-s22.gitlab.io/home/calendar.html),
- **review notes** from today: https://gitlab.com/csci422-s22/flux-base/-/tree/main/notes/class01,
- follow instructions (at the end of notes from today) to **install development tools**,  
- complete **background form**: https://forms.gle/mmjyQ3JkJlwyjyCw9.