Welcome to CS 249A

http://web.stanford.edu/class/cs249a

Object-Oriented Programming from a Modeling and Simulation Perspective

Mark Linton
The modeling and simulation perspective

- What CS 249A is about
- How the course works
- SOS methodology
Software is everywhere

- Media production and delivery (Adobe, LucasArts, Netflix)
- Communications (Verizon, Cisco)
- Financial services (Intuit, Goldman Sachs)
- Hardware development (chip simulators)
- Order fulfillment (Amazon, Fedex)
- Transportation (driverless cars, drones)
- … and lots more
Software is getting better (some of it anyway)

- IBM TSO: “Slow but hard to use”
- Internet empowerment
- Windows is better than DOS, Windows 7 better than Vista
- Amazon order fulfillment
- HTML5
- WebSockets
But still have a long way to go

• 80% of requested features in MS Word are already there

• Searching the App Store

• From Earthlink to Comcast

• SSL/TLS bugs

• … and many more
Apple SSL bug

```c
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
err = sslRawVerify(...);
```

We need better software.

What does “better” mean?
Better software is

1. Easier to use

2. More widely applicable

3. Less expensive (includes time as well as effort and materials)
What about performance and reliability?

- Lower performance requires more powerful devices, raising the cost to the user
- Buggy or unpredictable behavior makes software more difficult (or sometimes impossible!) to use
- Insecure software systems are really expensive (just ask Target)
So how to we get better software?

Use a better *methodology* to develop software

**Methodology** — a set or system of methods, principles, and rules for regulating a given discipline, as in the arts or sciences. — *Dictionary.com*
The “modeling and simulation perspective” is the foundation of developing better software.
Programming as modeling and simulation

• Applications simulate models of data and behavior
• Users directly manipulate (some of) the models
• Programs define the models
• Models include notifications to simplify extensibility
• Simulation runtime framework manages discrete events and timing
Models are everywhere

• Media production and delivery (Adobe, LucasArts, Netflix)

• Communications (Verizon, Cisco)

• Financial services (Intuit, Goldman Sachs)

• Hardware development (chip simulators)

• Order fulfillment (Amazon, Fedex)

• … and lots more
The modeling and simulation perspective

• What CS 249A is about

• How the course works

• SOS methodology
Learning the material

- Read, watch, listen, ask: lectures, online chapters, office hours
- Write and read code (3 assignments to develop simulations)
- Explain code
Coding in C++

• Quirky language, to say the least

• Our methodology works in most other languages: Java, Swift, JavaScript, Python, C#, C, Ruby, Scala, Go, …

• I will not present C++ language details in class …

• … but help is available whenever you need it!
Programming is (not) a DIY activity

• Hands-on experience is extremely valuable, so everyone does the assignments.

• Helping others is very important, so you are encouraged to help each other, but
  • You must cite assistance you get on assignments.
  • Although you may look at each other’s code and give advice, you may not share files at all.
  • You must be able to explain all code you turn in.
Exams will be like shorthand assignments

• Problems involving modeling and simulation

• Questions about both a model and its implementation

• Full or correct C++ code will not be required
Getting a good grade

• Programming assignments (60%)

• Midterm in class Thu. Oct. 23 (15%)

• Final (25%) Tentatively Wed. Dec. 10 7-10pm

• Class participation (Bonus)

• Typically grades are about 1/2 As, 1/2 Bs
Instructor: Mark Linton

• Experienced teacher:
  Stanford CS Faculty 1983-1990, CS 249A in 2006

• Experienced developer:
  Compilers, Debuggers, UI toolkits, Interactive TV,
  Application integration, VM management,
  Big Data, Telemedicine, …

• C++, Java, C, JavaScript, Modula-2, Basic, …

• Worked at small, medium, and large companies
“I’ve been programming for more than 30 years, and I love it!”

— Professor Cheriton, CS 249A Course Reader, p. 2
Attitude is important
Schedule (1st half)

• Sept. 23: Ch. 1 (pp. 2-33), Ch. 2 (pp. 58-85)

• Sept. 30: Ch. 3 (pp. 117-142), Ch. 1 (pp. 24-57)

• Oct. 7: Ch. 2 (pp. 85-116), Ch. 3 (pp. 143-173)
  Assignment 1 due Fri. Oct. 10 5:00pm PT

• Oct. 14: Ch. 4 (pp. 174-215)

• Oct. 21: Ch. 5 (pp. 216-243)
  Midterm in class Thu. Oct. 23
The modeling and simulation perspective

• What CS 249A is about
• How the course works
• SOS methodology
SOS software development

• **S**ource code representation
• **O**utside-in development
• **S**hort cycle time
Software development is multi-generational

• Software breeds new software (evolve, not rewrite)

• Forward spiral of analyze-design-implement-use

• Natural selection will lead to extinction of some of the software (or sometimes entire companies)
Business needs and opportunities also evolve

• What are people willing to pay for? how much?
• Direct or indirect, service or license, ad-based or not
• Appeal—pictures and words
• System testing—will it work everywhere?
• Packaging— product or feature?
• Setup, documentation, help
Objectives for each generation

• New features and functionality

• R&D cost and time to market

• Quality for now and the future ("if it ain’t fixed it’s gonna be broken")
Maintaining quality across generations

• Mindset of the culture—clean up the graffiti
• Invest continuously—read code, test, integrate, automate
• Know the territory—learn the framework
• Keep it simple—small set of concepts
Complexity kills.
It sucks the life out of developers,
it makes products difficult to plan, build and test,
it introduces security challenges, and
it causes end-user and administrator frustration.

– Ray Ozzie
Key aspects of a software development methodology

• Notation — what is the central definition of the product

• Process — how do programmers coordinate their work

• Timing — when to schedule the process steps
Common pitfalls

• Career advancement — *The Peter Principle*

• Software architects as “potted plants”

• Speed bumps to avoid mistakes just slow things down

• Counterculture—all that matters is the result
One good approach is SOS

• Focus communication around the Source

• Push development from the Outside with use cases

• Make everyone effective with Short time frames

• Lots of overlap with Agile Programming, though SOS is simpler
Notation: Source is the common objective

• The code *is* the software that you deliver

• The code defines and uses the terminology of the application (e.g., networks, packets, hosts)

• A feature isn’t implemented until it is checked into the tree
Example

class Cell {
public:

    Kilograms mass() const {
        ...
    }
};
What else would it be?

- Architecture (pictures)
- Specification (words and/or special language)
- Meeting notes (CYA)
Advantages of focusing on the source tree

- Less to learn
- More efficient
- Fewer errors
Another advantage of focusing on the source tree

Socializes design and terminology
Some documents are useful

- User interface mockups
- Design choice writeups
- Implementation discussions
- Marketing requirements
- Support constraints (platforms, i18n)
But the source code is the truth.
Reading assignments

• Chapter 1, pp. 2-33 (today’s class)

• Chapter 2, pp. 58-85 (for Thursday’s class)

• Chapter 3, pp. 117-142 (for next Tuesday’s class)

Programming Assignment 1: Malware Simulation

http://web.stanford.edu/class/cs249a/asgn1.html

Due Friday October 10 at 5:00pm Pacific Daylight Time.