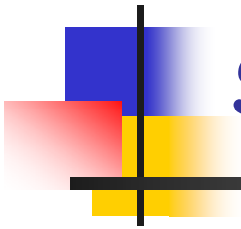
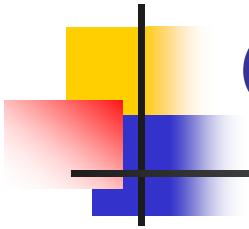


Computational education for scientists and engineers

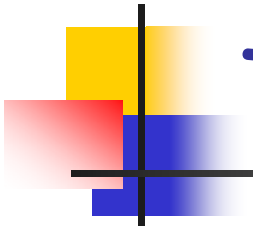
Devika Subramanian
Rice University



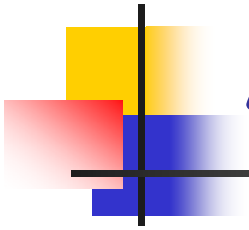
Questions

- Should computer scientists teach computation to scientists and engineers?
- What are the key components of a good computational training program?
- How can computation be introduced into K-12 and undergraduate curricula in science and engineering?
- How do we know if we making progress in teaching computation to science and engineers?

Should *we* teach computation to science and engineers?

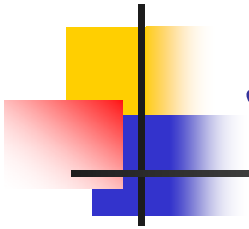


- Standard model: outsourcing of computational training to CS departments.
  - 2 course sequence, typically the first two courses taken by CS majors (functional programming in Scheme, object-oriented programming in Java or C++).
  - Reinforce the myth that "CS=programming".
- Increasing dissatisfaction among engineers about quality and type of training
  - No experience in computational modeling
  - No experience in programming either!
    - Cannot handle languages they use: C/Fortran
    - Cannot integrating code into existing code bases
    - Have no knowledge of parallel/distributed computing



A new model

- Need a 4 course sequence in undergraduate curriculum --- like mathematics for engineers and scientists
  - Computational thinking (Wing) taught in conjunction with scientists/engineers.
  - Computational mechanics
    - Discrete modes of failures
    - Peeling layers of abstraction
  - Interleave modeling and mechanics



Start early!

- Computational modeling should start in the 5th grade
  - Introduce along with algebra (functional programming ideas can be taught here) --- computational algebra
  - Introduce combinatorics/set theory/propositional calculus/finite state machines early
  - Introduce them in the context of problems they already solve and games they already play (Chicago Math concrete approach)
  - Recognize that the hardest part is teaching them how to map problems in the world onto mathematical and computational structures they are learning about.

We know we are succeeding when …

- Our colleagues in science and engineering understand that computational modeling is a distinct intellectual endeavor which is crucial for successful use of computation, and which can be taught by us.
  - Corollary: CS != programming
- Our colleagues in science and engineering understand that designing a good software system requires a unique set of engineering skills that can be taught by us.
  - Corollary: Robust systems cannot be built by writing out specifications and throwing them over the wall to a CS graduate.