Computational Thinking Across Curriculum

Two papers on teaching computational thinking to non-CS students

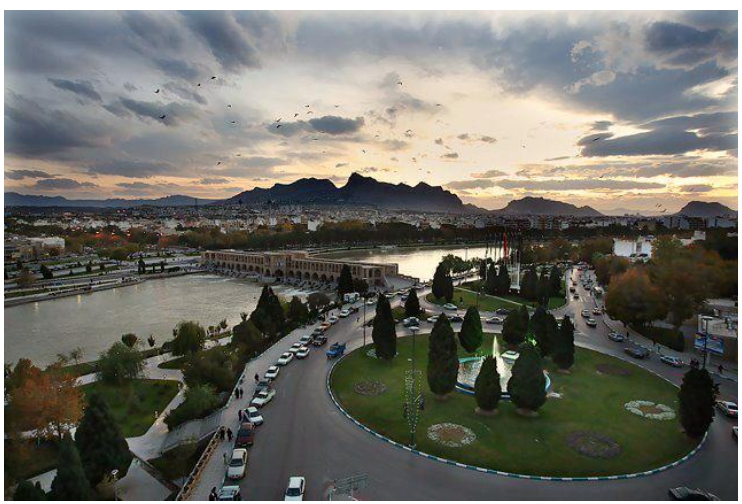
Pejman Khadivi
CS Department, Virginia Tech

Who am I?

Pejman Khadivi
Ph.D. Student, Computer Science
Working with Prof. Ramakrishnan

B.Sc. Computer Engineering, Isfahan University of Technology,1994-1998
M.Sc. Computer Engineering, Isfahan University of Technology, 1998-2000
Ph.D. Electrical Engineering, Isfahan University of Technology, 2000-2006

Visiting researcher, McMaster University, 2003
Lecturer, Isfahan University of Technology, 2000-2006
Assistant Professor, Isfahan University of Technology, 2006-2011



Papers

1. S. Hambrusch, et al. “**A Multidisciplinary Approach Towards Computational Thinking for Science Majors**”, *SIGCSE’09,* March 3–7, 2009.

2. L. Perkoviģ, et al. “**A Framework for Computational Thinking across the Curriculum**”, ITiCSE’10, June 26–30, 2010.

First Paper

- S. Hambrusch, et al. “**A Multidisciplinary Approach Towards Computational Thinking for Science Majors**”, *SIGCSE’09,* March 3–7, 2009.

- Summary:
  - This paper is about a course which has been designed to teach CT to students from Physics, Chemistry, and Bioinformatics majors. During the course students learn how to program in Python. Domain-specific examples (from each field) are presented in class and students have to do term-projects.

Main Principles

- Lay the groundwork for computational thinking
  - Formulating problem, abstraction, algorithms, visualization
- Present examples in a language familiar to the students
  - Science majors, conversant in the basics of the classical disciplines, will comprehend computational concepts more easily if those concepts can be motivated by examples from their scientific subdisciplines
- Teach in a problem-driven way
  - Start description from a real world problem
  - Thermodynamic system → computational perspective → randomized models and Monte Carlo techniques

Main Principles

- The programming language should right away allow a focus on computational principles
  - Write meaningful problems in a short time
  - Libraries used by scientific community
- Make effective use of visualization
  - Better understand scientific questions
  - Better understand computational principles and processes

Disciplines

- Physics
  - Better understanding of computation
  - New computational opportunities for learning and research
  - New perspective on physics and applied mathematics
- Chemistry
  - Computational methods relevant in chemical research (Monte Carlo, Simulated Annealing and Molecular Dynamics)
  - Use and integrate existing Fortran programs
  - Visualizing techniques
- Bioinformatics
  - Use of R for statistical computing and visualization
  - Program in a language for which bioinformatics software packages exist or can easily be integrated.

Course Syllabus

I. Basic Programming Tools (6 weeks)

- Introduction to Python. Elementary values and data types.
- Straight line programs, assignments to variables, type conversion, math library.
- Strings, lists, and tuples. Vectors and arrays.
- Conditionals and loop structures.
- Plotting using MatPlotLib and 3D visualization in VPython.
- Functions, parameters, and scope. Recursion.

Course Syllabus

II. Computational Tools and Methods (6 weeks)

- Arithmetic and random numbers. Using NumPy. Examples of numerical stability and problem stability.
- Introduction to simulations and Monte Carlo methods.
- Computational Physics: Ideal gas and Ising Spin simulations; adapting a generic Demon algorithm and estimating parameters in a physical system.(1 week)
- Trees as a data structure, traversal and exploration.
- Introduction to graphs, graph operations using NetworkX, graphs in science applications.
- Bioinformatics: Modeling protein interactions using tree and graph representations. Visualizing graphs in Cytoscape and analyzing protein interactions using clustering techniques. (1 week)
- Grand challenges in scientific computing.

Course Syllabus

III. Looking Under the Hood at Computer Science (3 weeks)

- Object-oriented design. Use and design of classes, OO concepts. Dictionaries and spatial queries as examples.
- History of computer science.
- Limits of computing, intractability, computability.
- Future models of computation: DNA computing, quantum computing.

Course Projects

- Two parts:
  - Programming part
  - Experimental part
    - Students can use their own program or another program
    - Most of them decided to use their own

- All projects asked students to produce visualizations of computational results and provide a write-up on their observations

Course Projects

- Manipulating Digital Audio
- Computational Experiments on Percolation in Grids
- Simulating Physical Systems
- Analyzing Protein-Protein Interactions

Evaluation

Spring 2008 with only 13 students

10 Physics, 3 Chemistry

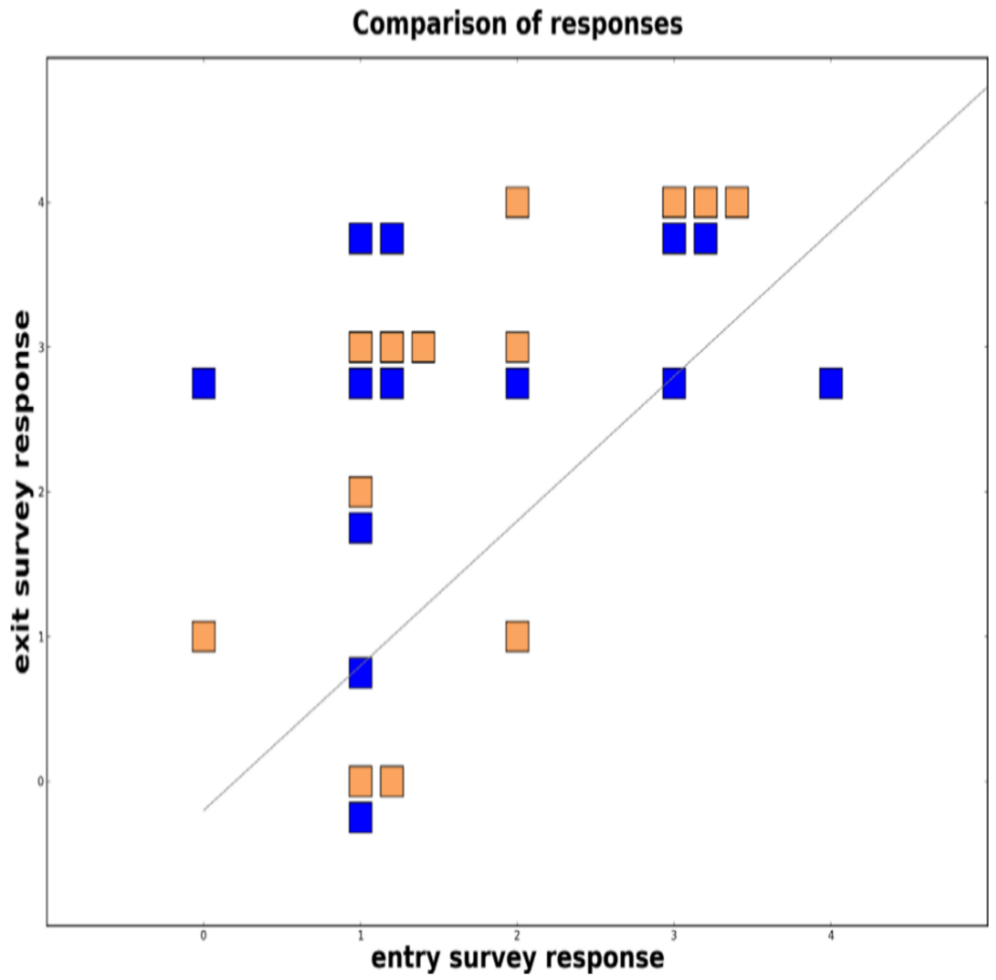
Q1: Taking another computer science course?

Q2: Pursuing a career that requires programming skills?

Answers: 0 to 4

0) not interested

4) very interested



Evaluation

- Students interest in taking other CS courses increased
  - Previous programming experience had no effect
  - Responses indicate that 60% of the students plan to take another CS course
- In other introductory programming courses for majors a decrease in interest in computer science is observed
- This is not a fair comparison
  - Example: Different instructors

Discussion

- This course seems to be a special “Programming Course” with a flavor of applications in science fields, not a CT course.

- “If all you have is a hammer, everything looks like a nail”
  - People do not know how to teach CT, so they teach Programming

Second Paper

- L. Perkoviç, et al. "**A Framework for Computational Thinking across the Curriculum**", ITiCSE'10, June 26–30, 2010.

- Summary:
  - This paper is about a framework that helps to integrate CT into different curriculums based on Denning's great principles of computing. Different domains are considered in arts, sciences, humanities, and social sciences. Authors ultimate goal was to augment people productivity in their fields.

Principles

- **Computation:** execution of an algorithm
- **Communication:** transmission of information from one process or object to another
- **Coordination:** control of the timing of computation at participating processes
- **Recollection:** encoding and organization of data in ways to make it efficient to search and perform other operations
- **Automation:** mapping of computation to physical systems
- **Evaluation:** statistical, numerical, or experimental analysis, and visualization of data
- **Design:** organization (using abstraction, modularization, aggregation, decomposition) of a system, process, object, etc.

Framework

- Common Core
  - First-year, two-course sequence in Math and Technology Literacy
  - Apply quantitative reasoning and information
  - Evaluate real-world problems using modern IT
  - Spreadsheets, databases, programming algorithms, …
  - Authors think that this is necessary to teach CT in other courses

- Domain-specific courses
  - Courses in various learning domains
  - Students are required to take 2-3 courses

Re-worked Courses

Course	Title
Scientific Inquiry	
CSC 233	Codes and Ciphers
CSC 235	Problem Solving
CSC 239	Personal Computing
ECT 250	Internet, Commerce, and Society
ENV 216	Earth System Science
ENV 230	Global Climate Change
ENV 340	Urban Ecology
GEO 241	Geographic Information Systems I
HCI 201	Multimedia and the WWW
IT 130	The Internet and the Web
Arts and Literature	
ANI 201	Animation I
ANI 230	3D Modeling
DC 201	Introduction to Screenwriting
GAM 224	Introduction to Game Design
HAA 130	Principles of European Art
Understanding the Past	
HST 221	Early Russia
HST 250	Origins of the Second World War
First Year Program	
LSP 112	Focal Point Seminar (The Moon)
Honors Program	
HON 207	Introduction to Cognitive Science

Re-worked Courses

Course	Title
Scientific Inquiry	
CSC 233	Codes and Ciphers
CSC 235	Problem Solving
CSC 239	Personal Computing
ECT 250	Internet, Commerce, and Society
ENV 216	Earth System Science
ENV 230	Global Climate Change
ENV 340	Urban Ecology
GEO 241	Geographic Information Systems I
HCI 201	Multimedia and the WWW
IT 130	The Internet and the Web
Arts and Literature	
ANI 201	Animation I
ANI 230	3D Modeling
DC 201	Introduction to Screenwriting
GAM 224	Introduction to Game Design
HAA 130	Principles of European Art
Understanding the Past	
HST 221	Early Russia
HST 250	Origins of the Second World War
First Year Program	
LSP 112	Focal Point Seminar (The Moon)
Honors Program	
HON 207	Introduction to Cognitive Science

Covered concepts

Course	Auto.	Comm.	Comp.
Scientific Inquiry			
CSC 233			XX
CSC 235			XX
ECT 250	XX		
IT 130		XX	
Arts and Literature			
ANI 201	XX		
ANI 230	XX		XX
First Year Program			
LSP 112			XX
Honors Program			
HON 207		XX	

Course	Coor.	Desi.	Eval.	Reco.
Scientific Inquiry				
CSC 233			XX	
CSC 235			XX	
CSC 239			XX	
ENV 216		XX		
ENV 230			XX	
ENV 340			XX	XX
GEO 241		XX		
HCI 201		XX		XX
IT 130		XX		XX
Arts and Literature				
ANI 230		XX	XX	
DC 201		XX		
GAM 224	XX	XX		
HAA 130		XX	XX	
Understanding the Past				
HST 221			XX	
HST 250			XX	
First Year Program				
LSP 112		XX		
Honors Program				
HON 207	XX			

What discussed for each course

- (A) catalog course description
- (B) high-level description of the course and computational thinking concepts covered
- (C) computational thinking learning goal
- (D) a case discussion and several guiding questions
- (E) assessment for the specified learning goal

Geographic Information Systems I

- An introduction to the fundamentals of geospatial information processing
- Introduces basic concepts and methods that underlie information systems designed to deal with geographically referenced data.
- Design (data modeling, …)
- Comparing two different data representations

Introduction to Game Design

- Computer Games from Three Perspectives:
  - Media: elements, structure, interactive appreciation
  - Complex software artifact
  - Cultural artifact
- Game rules:
  - Constituative, operational, and implicit
- Design (Abstracting game rules, …)
  - Relationship between different abstractions of rules, the modeling of game behavior, and the underlying structure of a game

3D Modeling

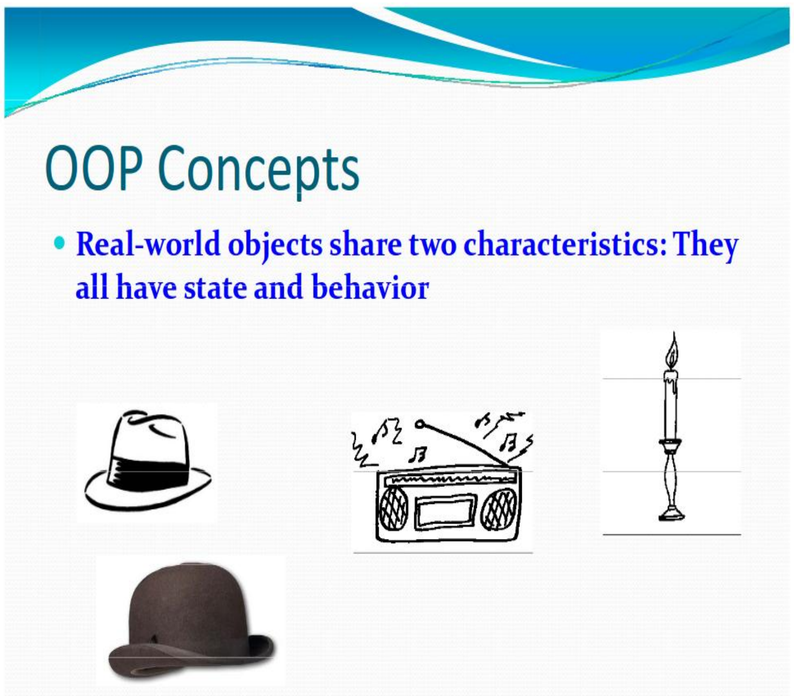
- Introductory modeling and texturing techniques required to construct 3D objects and scenes to be used for animation and gaming
- Modularization in 3D modeling
- Design, Automation, Computation

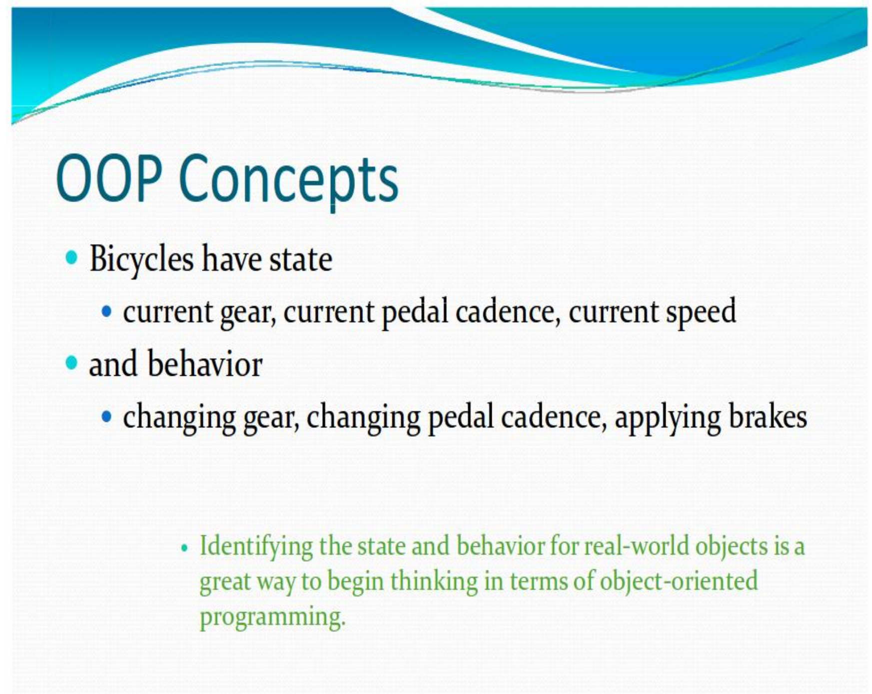
Discussion

- Proposed method: Talk about a domain specific problem with existing computational solutions to teach a CS/CT skill
- Alternative method: which concept is relevant to our problems?
    - Some simple familiar examples help students to absorb CS concepts not as an "only CS" concept but as a multi-domain concept. Then, domain specific examples can show up and do their magic.

Discussion

- Example: Teaching Object Oriented Design





Discussion

- Example of alternative method:

Foundation

- Concepts and methods: Abstraction, simulation, …
- Simple familiar examples

Issues

- What is the performance of that physical system?
- What agricultural process results in more crops?
- What will happen when a Mars rover wants to land on Mars?

Relevancy

- Is there any CS concept useful here?
- e.g. Can we use simulation to answer those questions?

Discussion

- Which approach do you like more: attacking a domain-specific problem (e.g. in Physics) or building a fundation first?
- Should we explicitly tell to students that the goal is to learn computational thinking or not?
- Should we tell the students that "now you are familiar wit abstraction" or any other concept in CT/CS, or not?


Happy
Halloween