Panel:

NSF-Sponsored Innovative Approaches to Undergraduate Computer Science

Stephen Bloch (Adelphi University)
Amruth Kumar (Ramapo College)
Stanislav Kurkovsky (Central CT State University)
Clif Kussmaul (Muhlenberg College)
Matt Dickerson (Middlebury College), moderator
<table>
<thead>
<tr>
<th>Project</th>
<th>Web site(s)</th>
<th>Intervention</th>
<th>Delivery</th>
<th>Supervision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program by Design</td>
<td><a href="http://programbydesign.org">http://programbydesign.org</a> <a href="http://picturingprograms.org">http://picturingprograms.org</a></td>
<td>curriculum with supporting IDE, libraries, &amp; texts</td>
<td>in class; software and textbook are free downloads or web-based</td>
<td>normally active, but can be done other ways</td>
</tr>
<tr>
<td>NSF awards 0010064 &amp; 0618543</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problets</td>
<td><a href="http://www.problets.org">http://www.problets.org</a></td>
<td>in- or after-class problem-solving exercises on programming concepts</td>
<td>applet in a browser</td>
<td>none - teacher not needed, although some adopters use it in active mode too</td>
</tr>
<tr>
<td>Amruth Kumar</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSF award 0817187</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Game Development</td>
<td><a href="http://www.mgdcs.com/">http://www.mgdcs.com/</a></td>
<td>in-class or take-home programming projects</td>
<td>PC</td>
<td>passive - teacher as facilitator to answer Qs</td>
</tr>
<tr>
<td>Stan Kurkovsky</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSF award DUE-0941348</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POGIL</td>
<td><a href="http://pogil.org">http://pogil.org</a> <a href="http://cspogil.org">http://cspogil.org</a></td>
<td>in-class activity</td>
<td>paper or web</td>
<td>passive - teacher as facilitator to answer Qs</td>
</tr>
<tr>
<td>Clif Kussmaul</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSF award TUES 1044679</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Course(s)</td>
<td>Language(s)</td>
<td>Focus</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Program by Design</strong></td>
<td>Middle school, pre-AP CS in HS, CS0, CS1, CS2 in college</td>
<td>Usually Scheme-like teaching languages leading into Java; has also been done in Python, ML, Java, Scala, ...</td>
<td>problem-solving process, particularly test-driven development and use of data types to guide coding &amp; testing</td>
<td></td>
</tr>
<tr>
<td><strong>Problets</strong></td>
<td>AP-CS, CS I, CS 2. also as refresher or to switch languages in other courses</td>
<td>C, C++, Java, C#</td>
<td>code tracing, debugging, expression evaluation, predicting program state</td>
<td></td>
</tr>
<tr>
<td><strong>Mobile Game Development</strong></td>
<td>AP-CS, CS1, CS2</td>
<td>Java</td>
<td>core OO programming; intro to advanced subjects such as AI, networks, security</td>
<td></td>
</tr>
<tr>
<td><strong>POGIL</strong></td>
<td>CS1, CS2, SE, <em>etc.</em> CS Principles (HS) (used across STEM)</td>
<td>often concept-based and language-independent; CS1 in Java &amp; Python</td>
<td>knowledge construction, process skills</td>
<td></td>
</tr>
</tbody>
</table>

*Stephen Bloch*

*Amruth Kumar*

*Stan Kurkovsky*

*Clif Kussmaul*
<table>
<thead>
<tr>
<th>Project</th>
<th>Prep before class</th>
<th>During class</th>
<th>After class</th>
<th>Teacher</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program by Design Stephen</td>
<td>select examples</td>
<td>model problem-solving process; answer questions</td>
<td>feedback to students: how well did they follow the process?</td>
<td>solo or small team</td>
<td>mostly programming</td>
</tr>
<tr>
<td>Problets Amruth</td>
<td>sign up to get URL; specify which problet to use when</td>
<td>none; not even in supervised mode</td>
<td>use report to select concepts to review in class</td>
<td>solo</td>
<td>solving problems on programming</td>
</tr>
<tr>
<td>Mobile Game Development Stan</td>
<td>become familiar with the technical scaffolding provided by each project and with sample solution</td>
<td>explain objectives, demonstrate sample solution, help students with scaffolding</td>
<td>review completed programming projects</td>
<td>teams of 2</td>
<td>working on programming project</td>
</tr>
<tr>
<td>POGIL Clif</td>
<td>make copies or post. (writing activities is time intensive)</td>
<td>facilitate teams, share conclusions</td>
<td>review team reports</td>
<td>teams of 2-4</td>
<td>working through activity, summary report (optional)</td>
</tr>
</tbody>
</table>
Program By Design

Stephen Bloch, Adelphi University

with Eli Barzilay, John Clements, Matthias Felleisen, Robby Findler, Kathi Fisler, Matthew Flatt, Kathy Gray, Shriram Krishnamurthi, Viera Proulx, Emmanuel Schanzer, ...
Curricular ideas

- Start in simple, beginner-friendly language
  - need beginner-friendly IDE & compiler
- Teach fundamental, transferable principles & habits
- Test-driven development from the beginning
  - need beginner-friendly testing harness
- Graphics, animation, GUI, music as motivators
  - need beginner-friendly libraries
- Then revisit same ideas in “mainstream” language (next semester or next year)
Pedagogical ideas

● Concrete design recipe
  ○ Identify input & output data types
  ○ Write test cases (guided by data types)
  ○ Write function skeleton (guided by data types)
  ○ Fill in gaps (guided by test cases)
  ○ Run test cases

● Each step is explicit & worth partial credit

● Writing test cases is much easier for functional than imperative code, so start in functional paradigm
  ○ even for graphics & interaction

● Functional GUI programming teaches model/view separation early
Technical ideas

● Start in language subset…
  ○ enforced by compiler
  ○ Several concentric languages matching stages of curriculum

● Read-eval-print loop to encourage experimentation
  ○ like DrJava, BlueJ CodePad, irb, python, ghci, etc.

● “Image” is a data type, just like “integer” or “string”
  ○ even in the REPL
  ○ Can enter an image as a literal, interactively
  ○ Can see images as expression values, interactively

● Demo: http://screencast.com/t/12O3RGxFH
Versions of the curriculum

- Bootstrap (middle school)
  - http://bootstrapworld.org

- *Picturing Programs* (high school pre-AP, college CS0)
  - http://picturingprograms.org

- *How to Design Programs* 2ed (college CS1)
  - http://www.ccs.neu.edu/home/matthias/HtDP2e/
  - or search “htdp2e”
Software support

- WeScheme (IDE in a browser, used with Bootstrap)
  - [http://wescheme.org](http://wescheme.org)
- DrRacket (IDE, used with PP and HtDP)
  - [http://racket-lang.org](http://racket-lang.org)
- JavaLibWorld and JavaLibTester (support libraries for Java-based course)
  - search on GitHub
Who likes this approach?

- Grants from Exxon, DoEd, NSF, Google
- ACM SIGCSE “Outstanding Contribution to Computer Science Education” award (2011)
- ACM Karlstrom award (2009)
Who uses this approach?

Bootstrap:
- Park Elementary School
- Ballou High School
- Edison Middle School
- Yanbu International School
- Crossroads School
- Sedro-Woolley High School
- Albuquerque Academy

NYOS Charter School
- Boston Latin Academy
- United for Success Academy
- Barnard Saturday Science Seminar
- St. Andrew's-Sewanee School
- Academy for Science and Design
- International School Ho Chi Minh City

124 more omitted
Who uses this approach?

Picturing Programs:

- University of Toronto
- University of California, Irvine
- Vassar College
- Adelphi University
- Georgia Regents University
- Indian Institute of Information Technology and Management-Kerala, Trivandrum, India
- St Francis Borgia HS
- Whitney Young HS
- The Fay School
- Lakehill Preparatory School
- Aberdeen HS
- Holy Name HS

- Owatonna HS
- Bancroft School
- Dighton-Rehoboth HS
- Augusta Preparatory Day School
- Nashoba Regional High School
- St Luke’s School
- The Webb Schools
- oxfordcomputerscience.wikispaces.org (HS level)
- DuPont Manual HS (in Scala?)
- Evergreen Middle School
- at least one 4th-grade teacher (!)
- various others omitted
Who uses this approach?

How to Design Programs

- University of Chicago
- Northeastern University
- University of Delaware
- Westmont College
- Worcester Polytechnic Institute
- University of Notre Dame
- University of Waterloo
- Istanbul Bilgi University
- Seton Hall University
- Berry College
- Brown University
- Monmouth College
- University of Minnesota Morris
- Northwestern University
- Suffolk County Community College
- University of British Columbia (both traditional course and MOOC)

- Zefat Academic College
- UNAM
- Manhattanville College
- Rhode Island College
- University of Tübingen
- University of Freiburg
- University of Dallas
- South Carolina State University
- Pacific Union College
- Humboldt College
- University of Chile (in Python)
- Ochanomizu University (in OCaML)
- Carnegie-Mellon (in ML)
- various others omitted
Problets

Amruth Kumar,
amruth@computer.org
problets.org
Curricular Goals

• Learn programming concepts by solving problems
• Supplement classroom instruction
• Complement programming projects
What Problets do:

- Present problems
- Grade student’s answer
- Provide instant feedback
- Record student performance
- Provide summary to the instructor
Types of problems

- Identify the output of a program
- Debug a program
- Resolve the state of program variables
- Evaluate expressions

::

- Step-by-step
- *Not* multiple-choice problems
Identifying the output

// The Java program
public class Problem {
    public static void main( String[] args ) {
        public static void main( String args[] ) {
            short depth = 111;
            short amount;
            if (depth <= 28 )
                amount = 1;
            else
                if (depth <= 59 )
                    amount = 2;
                else
                    if (depth <= 85 )
                        amount = 3;
                    else
                        amount = 4;
                // End of else clause
            // End of else clause
            System.out.println(amount);
            // End of method main
        } // End of class Problem
    }
} // The Java program

// The Java program
public class Problem {
    public static void main( String args[] ) {
        short measure = 3;
        System.out.println( measure );
        short quantity = 7;
        System.out.println( quantity );
        System.out.println( quantity - measure );
    }
} // The Java program

If the code does not produce any output, click on the No Output button.

Enter outputs one at a time, in the correct order:

1st: 4  23
2nd: 
3rd: 
4th: 
5th: 

Submit
Debugging
State of a variable
Expression Evaluation

The left-hand side of the `&&` operator is always evaluated before its right-hand side. Since the left side evaluates to false, the right side of the `&&` is short circuited, i.e., not evaluated.
Topics (17 modules)

• Expression evaluation
  o Arithmetic, Relational, Logical, Assignment, Bitwise

• Selection
  o if, if-else, switch, nested statements

• Loops
  o while, for, do-while, nested loops, infinite loops

• Functions - behavior, bugs, recursion

• Arrays, Access in Classes, C++ pointers
### Topics and Problems

<table>
<thead>
<tr>
<th>Topic</th>
<th>Sub-Topic</th>
<th>Used Since</th>
<th>No. Problems</th>
<th>Learning Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions</td>
<td>Arithmetic</td>
<td>Fall 2004</td>
<td>192</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Relational</td>
<td>Fall 2004</td>
<td>268</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Logical</td>
<td>Fall 2006</td>
<td>280</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Assignment</td>
<td>Fall 2008</td>
<td>255</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Bitwise</td>
<td>Fall 2010</td>
<td>303</td>
<td>28</td>
</tr>
<tr>
<td>Selection</td>
<td>If/If-else</td>
<td>Spring 2005</td>
<td>165</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>switch</td>
<td>Spring 2010</td>
<td>147</td>
<td>12</td>
</tr>
<tr>
<td>Loops</td>
<td>while</td>
<td>Fall 2004</td>
<td>201</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>for</td>
<td>Fall 2004</td>
<td>213</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>do-while</td>
<td>Fall 2010</td>
<td>125</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Advanced</td>
<td>Spring 2010</td>
<td>139</td>
<td>13</td>
</tr>
<tr>
<td>Functions</td>
<td>Debugging</td>
<td>Fall 2009</td>
<td>117</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Tracing</td>
<td>Fall 2009</td>
<td>95</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Recursion</td>
<td>Spring 2013</td>
<td>68</td>
<td>10</td>
</tr>
<tr>
<td>Arrays</td>
<td>1-D</td>
<td>Fall 2010</td>
<td>172</td>
<td>14</td>
</tr>
<tr>
<td>Classes</td>
<td>Access</td>
<td>Spring 2013</td>
<td>128</td>
<td>18</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>**</td>
<td><strong>2868</strong></td>
<td><strong>249</strong></td>
</tr>
</tbody>
</table>
Target

• Languages:
  o Java, C, C++, C#, some Visual Basic

• Audience:
  o CS I, CS II, AP-CS
  o Refresher for advanced courses/language change

• Institutions:
  o High school, 2-year, 4-year colleges
Pedagogy

• Learn by solving problems
  o Mastery learning
• Step-by-step explanation of correct solution
• Adaptive problem-sequence
• Randomized problem set
• Learning at one’s pace on one’s time
  o Any time, as often as necessary
• Extensively evaluated over 14 years
Visualization
Usage

• Closed-Lab exercises
• After-class assignments (24 x 7)
• Language refreshers
  o As many as necessary
  o When necessary
  o As often as necessary
• Continuous third-party use since fall 2004
  o 60+ schools in spring 2014
Adoption

• No software installation necessary - Web-based
• No supervision necessary - self-administering
• Report available on demand
  o By problems, learning objectives
• Free for educational use
## Snapshot of a report

<table>
<thead>
<tr>
<th>Name</th>
<th>Problems</th>
<th>PreTest Score</th>
<th>Ave Score</th>
<th>Practice Score</th>
<th>Ave Score</th>
<th>Post-Test Score</th>
<th>Ave Score</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joel</td>
<td>3</td>
<td>7</td>
<td>0.64</td>
<td>11</td>
<td>27</td>
<td>24.3</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>1</td>
<td>6</td>
<td>0.33</td>
<td>3</td>
<td>10</td>
<td>12.5</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>2</td>
<td>2</td>
<td>0.29</td>
<td>7</td>
<td>21</td>
<td>11</td>
<td>0.62</td>
<td>1 l</td>
</tr>
<tr>
<td>Rob</td>
<td>11</td>
<td>8.93</td>
<td>0.9</td>
<td>11</td>
<td>16</td>
<td>13</td>
<td>0.87</td>
<td>11</td>
</tr>
<tr>
<td>Jill</td>
<td>8</td>
<td>8</td>
<td>2.83</td>
<td>3</td>
<td>23</td>
<td>23</td>
<td>1.9</td>
<td>10</td>
</tr>
<tr>
<td>Jim</td>
<td>11</td>
<td>9.33</td>
<td>0.65</td>
<td>23</td>
<td>20.9</td>
<td>20.9</td>
<td>0.91</td>
<td>11</td>
</tr>
<tr>
<td>Tim</td>
<td>11</td>
<td>11</td>
<td>9.67</td>
<td>9.67</td>
<td>9.67</td>
<td>9.67</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>11</td>
<td>11</td>
<td>9.33</td>
<td>9.67</td>
<td>9.67</td>
<td>9.67</td>
<td>0.88</td>
<td></td>
</tr>
</tbody>
</table>

### Average

- **Average: 8.957**
- **Std Dev: 4.425**
- **p-value: 0.0003202**
Contact Information

Additional information at:

www.problets.org

If interested, please contact:

amruth@ramapo.edu

Acknowledgements: NSF CCLI DUE 0088864
Mobile Game Development

Stan Kurkovsky
Central Connecticut State University
http://www.mgdcs.com/

with Archana Chidanandan and Delvin Defoe
Overarching Goals

• Improve student engagement and motivation
• Decrease attrition in introductory CS courses

• Method: use a relevant learning context
Curricular Objectives

• Expose students to advanced topics
• Strengthen student mastery of the core concepts
• *CS is more than just coding!*

• Method: project-based learning modules
Learning Modules

• Context
  o A well-known game (arcade, board, etc.)
  o Casual games

• Learning objectives
  o Introduce an advanced topic (e.g. networking)
  o Reinforce a core topic (e.g. for loops)

• Game implementation
  o Working demo
  o Technical scaffolding
Target

• Language
  o Java: J2ME, Android

• Audience
  o AP-CS, CS I, CS II
  o Also: advanced topical courses

• Institutions
  o High school, 2- and 4-year colleges
Pedagogy

- Context-based learning
- Relevance to everyday life
- Hands-on experiences
- Teamwork
- Instant gratification
Sample Modules

- Battleship - computer networking
- Connect Four - artificial intelligence
- Frogger - software engineering
- Space Bears - human-computer interaction
- Craps - security
- Text Twister - algorithms
Process Oriented Guided Inquiry Learning (POGIL)

Clif Kussmaul, Muhlenberg College

http://cspogil.org  http://pogil.org
POGIL - Curricular Goals

- Across CS (& other STEM) disciplines, we should help our students learn to:
  - analyze, design, synthesize ideas
  - read, write, & debug code & docs
  - communicate (oral & written)
  - work in teams, manage time
  - learn or create ideas on their own
POGIL - Pedagogy

- Research shows that learning is improved when people:
  - work in teams with other people
  - construct knowledge through a learning cycle (explore, invent, apply)
  - receive prompt constructive feedback
  - reflect on learning process & outcomes
Process Oriented Guided Inquiry Learning

• Students work in teams with assigned roles (e.g. manager, recorder, speaker)
• Teams work on classroom activities that present a model followed by questions.
• Instructor is a facilitator, not a lecturer.
• Activities are designed to guide students to:
  o construct understanding of key ideas
  o develop key process skills
POGIL Example: 1st Day of CS1

Hi-Lo: Guessing Game

- Two players – A and B.
- A picks a number 1-100.
- B guesses a number.
- A responds “too high”, “too low”, or “you win”.
- Continue to play until B wins (or gives up).

Questions

1. Play the game 3 times.
2. Identify 4-5 strategies B could use to play.
3. (Discuss with class.)
4. Rank by # of guesses.
5. Rank by difficulty.
6. Plot rankings & describe.
POGIL Example: 1st Day of CS1

<table>
<thead>
<tr>
<th>Strategy</th>
<th>S</th>
<th>D</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>4</td>
<td>1</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Count up by 1.</td>
<td>3</td>
<td>2</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Count up by 10, down by 1.</td>
<td>2</td>
<td>3</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Split range in ½ each time.</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

Questions
1. Max # of guesses?
2. Avg # of guesses?
3. (Discuss with class.)
4. Repeat for 1-1000.
5. Repeat for any N.
7. (Discuss with class.)
CS-POGIL

cspogil.org

• NSF TUES project (2011-2014) to develop POGIL materials for CS
  o CS2, Data Structures, Software Engineering
  o sci comp, CS1 (Java, Python), theory, AI, ...

• Numerous CS collaborators, including:
  o Helen Hu, Lisa Olivieri, Matt Lang, Chris Mayfield, Heidi Ellis, Stoney Jackson, Tammy Pirmann

• $$$ available to attend POGIL workshops
The POGIL Project
pogil.org

• Non-profit to support use of POGIL & related approaches
• Long history of NSF funding (15+ years)
• 3-day regional summer workshops
• Review POGIL activities, support classroom implementation
POGIL - Implementations

• Students: 10-200; HS, undergrad, grad
• Models: UML, Code, API doc
• Media
  o Paper copies for each team or student
  o Google Doc for each team
  o Presentation slides
• Team structure
  o Teams of 4, split for pair programming
DISCUSSION

• What might the approach not accomplish or do well?
• When would you not use it as opposed to when would you use it?
• How would your approach combine well with another of the approaches outlined here?