Adding a Teaspoon of Computing to History and Mathematics Classes
Mark Guzdial
Today’s Story

- Computing was created to be taught to everyone.

- Are we reaching everyone now? (Hint: “No.”)

- Teaspoon languages as a way to change computing to reach everyone.
  - For History: DV4L
  - For Mathematics/Engineering: Pixel Equations
  - For Mathematics/Combinatorics: Counting Sheet

- Big question: What are students learning?
"A handful of people, having no relation to the will of society, having no communication with the rest of society, will be taking decisions in secret which are going to affect our lives in the deepest sense."
Programming changes how we understand
First published definition of Computer Science

“The study of computers and all the phenomena surrounding them.”

*Science, 1967*

This is broader than how most people define computer science today. Let’s call this *Computing*
4.7%

Percentage of US high school students enrolled in a CS course
4.7%
Computer science in high schools is growing very slowly

- In England (from Roehampton Report 2018):
  - 53% of schools offer CS GCSE, 12% of students take it.
    - < 20% female
  - 36% offer A Level CS, under 3% take it.
    - < 10% female

Data from Peter Kemp
BOTTOMLINE:

THE MAJORITY OF SECONDARY SCHOOL STUDENTS IN THE US AND ENGLAND HAVE NEVER SEEN COMPUTER SCIENCE
AP US History vs. AP CS Principles

399K vs 114K

>50% female vs. <30% female

6x more Black

14x more Hispanic
Teaspoon Languages

- A Teaspoon language is a task-specific programming (TSP) language — specification of process to be executed by a computational agent.

Adding a teaspoon of computing to other subjects.

- **USEFUL**: Supports a task (learning activity) that an other-than-CS teacher wants to achieve.

- **USABLE**: Can be learned in less than 10 minutes
#1: DV4L: Data Visualization for Learning

For History Courses

Collaboration with Tammy Shreiner
Modify the json code in the scripting version of DV4L
Graph 1: Imported From DV4L

Database (DB): Populations
Y axis: Rwanda
Year Range: 1860 to 2019
Graph type: bar
Color: orange

Graph 2:

Database (DB): Populations
Y axis: Algeria
Year Range: 1800 to 2019
Graph type: bar
Color: darkBrown

Are there any noticeable differences in the trend of population growth in the following countries? Why?
#2: Pixel Equations

For Math and Engineering classes

<table>
<thead>
<tr>
<th>If this is true</th>
<th>Set Red</th>
<th>Set Green</th>
<th>Set Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si esto es cierto</td>
<td>Asignar Rojo</td>
<td>Asignar Verde</td>
<td>Asignar Azul</td>
</tr>
<tr>
<td>x &gt; 200</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y &lt; 200</td>
<td></td>
<td>2 * green</td>
<td></td>
</tr>
<tr>
<td>blue &gt; 200</td>
<td></td>
<td></td>
<td>blue / 2</td>
</tr>
<tr>
<td>x = y - 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 3: Run Equations

Result Picture Appears Here:

[Show Result]
Pixel Equations
Select your preferred language

Step 1: Pick your input picture
Which picture would you like to use?

- File named: arch.jpg
- File named: Bayamon.jpeg
- File named: beach.jpg
- File named: dog.png
- File named: san-juan.jpeg
- File named: TSM-Map.png
- File named: detroit.jpg
- File named: DetroitSkyline.jpg
which will select all pixels where the \( x \) coordinate is greater than the \( y \) coordinate.

Then write equations for how to change red, green, and blue (rojo, verde, y azul) for the selected pixels. You can invert each color by subtracting from 255 (e.g., set red/rojo to 255-red (o 255-rojo)).

<table>
<thead>
<tr>
<th>If this is true</th>
<th>Set Red</th>
<th>Set Green</th>
<th>Set Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si esto es cierto</td>
<td>Asignar Rojo</td>
<td>Asignar Verde</td>
<td>Asignar Azul</td>
</tr>
<tr>
<td>( x &gt; 200 )</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 3: Run Equations

Result Picture Appears Here:
<table>
<thead>
<tr>
<th>If this is true</th>
<th>Set Red</th>
<th>Set Green</th>
<th>Set Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si esto es cierto</td>
<td>Asignar Rojo</td>
<td>Asignar Verde</td>
<td>Asignar Azul</td>
</tr>
<tr>
<td>x &gt; 200</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y &lt; 200</td>
<td></td>
<td>2 * green</td>
<td></td>
</tr>
</tbody>
</table>

Step 3: Run Equations

Result Picture Appears Here:
Then write equations for how to change red, green, and blue (rojo, verde, y azul) for the selected pixels. You can invert each color by subtracting from 255 (e.g., set red/rojo to 255–red (o 255–rojo)).

<table>
<thead>
<tr>
<th>If this is true</th>
<th>Set Red</th>
<th>Set Green</th>
<th>Set Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si esto es cierto</td>
<td>Asignar Rojo</td>
<td>Asignar Verde</td>
<td>Asignar Azul</td>
</tr>
<tr>
<td>x &gt; 200</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y &lt; 200</td>
<td></td>
<td>2 * green</td>
<td></td>
</tr>
<tr>
<td>blue &gt; 200</td>
<td></td>
<td></td>
<td>blue / 2</td>
</tr>
</tbody>
</table>

Step 3: Run Equations

Result Picture Appears Here:
#3: Counting Sheets

Elise Lockwood: Teaching combinatorics with Python

**Student challenge:** How many arrangements do you expect to get from the letters in the word ROCKET?

- No repeated letters
- Order matters

Why do you think that? How will the list of outcomes be structured?

```
arrangements = 0
People = ['R', 'O', 'C', 'K', 'E', 'T']

for p1 in People:
    for p2 in People:
        if p2 != p1:
            for p3 in People:
                if p3 != p1 and p3 != p2:
                    for p4 in People:
                        if p4 != p1 and p4 != p2 and p4 != p3:
                            for p5 in People:
                                if p5 != p1 and p5 != p2 and p5 != p3 and p5 != p4:
                                    for p6 in People:
                                        if p6 != p1 and p6 != p2 and p6 != p3 and p6 != p4 and p6 != p5:
                                            arrangements = arrangements + 1
                                            print(p1, p2, p3, p4, p5, p6)

print(arrangements)
```

"Reinforcing key combinatorial ideas in a computational setting: A case of encoding outcomes in computer programming," 2021, Journal of Mathematical Behavior, Lockwood and De Chenne
### Counting Sheet Interactive Tool

#### Try a code snippet:

2: Shirts and pants

#### Counting Sheet:

<table>
<thead>
<tr>
<th>col1</th>
<th>col2</th>
<th>col3</th>
<th>col4</th>
<th>col5</th>
<th>col6</th>
</tr>
</thead>
<tbody>
<tr>
<td>tee, polo, sweater</td>
<td>jeans, khaki</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Generate

- [ ] Chart Output

#### Results:

tee
jeans
tee khaki
polo jeans
polo khaki
sweater jeans
sweater khaki

Collaboration with Elise Lockwood
Is this computing education?

WHAT ARE STUDENTS LEARNING HERE?
Rich, Strickland, Binkowski, Moran, and Franklin (ICER 2017) asked the question:

What’s the starting place for K-8 CS learners?

Figure 3: Sequence learning trajectory.

Figure 4: Repetition learning trajectory.
Proposed:
What comes first when learning programming?

1. Precision and completeness are important when writing instructions in advance.

2. Different sets of instructions can produce the same outcome.

3. Programs are made by assembling instructions from a limited set.

4. Some tasks involve repeating actions.

5. Programs use conditions to end loops.
Scratch fluency doesn’t need that whole list

- Over 60 million users.
- Most Scratch projects are stories that use...
  - Only Forever loops
  - No booleans
  - Just movement and sequence.

There is expressive power in even a subset of CS.
Bootstrap: Algebra doesn’t use all of that list

- Improves learning in algebra
- Students do not code repetition.
- Functional

<table>
<thead>
<tr>
<th>Unit</th>
<th>Game Feature</th>
<th>Programming Concept</th>
<th>Math Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>locating elements on screen</td>
<td>expressions, Circles of Evaluation</td>
<td>coordinates</td>
</tr>
<tr>
<td>2</td>
<td>creating text and images</td>
<td>string and image operations</td>
<td>domain, range, kinds of data</td>
</tr>
<tr>
<td>3-5</td>
<td>making moving images</td>
<td>defining functions, examples</td>
<td>multiple function representations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>as formulas and as tables</td>
</tr>
<tr>
<td>6</td>
<td>determine when game elements are</td>
<td>Booleans and Boolean operators</td>
<td>inequalities</td>
</tr>
<tr>
<td></td>
<td>off-screen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>responding to key-presses</td>
<td>conditional</td>
<td>piecewise function</td>
</tr>
<tr>
<td>8</td>
<td>collision detection</td>
<td>(nothing new)</td>
<td>Pythagorean Theorem</td>
</tr>
<tr>
<td>9</td>
<td>polishing games for presentation</td>
<td>code reviews</td>
<td>explaining math concepts to others</td>
</tr>
</tbody>
</table>

Figure 1: Curriculum structure: each unit introduces game, programming, and math concepts in parallel.

Schanzer, Fisler, Krishnamurthi, Felleisen, 2015
Learning challenges that our teachers face

- **Intermediate representations:**
  - Much of computing involves use of a notation (HTML, programs) that is interpreted by a computer for a final result (web page, program execution).

- **Debugging:**
  - The computer only interprets your notation — it does not know your intention. When the interpretation does not match what you intended for the result, you will have to debug.
REPEATING THE BOTTOMLINE:

THE MAJORITY OF SECONDARY SCHOOL STUDENTS IN THE US AND ENGLAND HAVE NEVER SEEN COMPUTER SCIENCE

We don’t know much about teaching all students about computing
This too is Programming

A place to learn about intermediate representations and debugging. Useful tools in social studies, mathematics, and engineering.

We are developing Teaspoon languages with English and Spanish keywords.

2/3 of the world does not speak English.

Reaching everyone requires new languages and tools.
Teaspoon Languages as a CS for All Strategy

- **Hour of Code**: One hour of a Turing-complete programming language every year.

- **Teaspoon Languages**: One to three little languages in every social studies, mathematics, and language arts class.

- Which results in more retained and transferrable CS learning? Which creates more of a school culture about using programming across disciplines?
Time to Play with **PROTOTYPES**

- For history: DV4L

- For Mathematics/Engineering: Pixel Equations

- For Mathematics/Counting: Counting Sheets
Questions to Think About

- What would it take to get other-than-CS teachers in your schools to try a Teaspoon language?

- Do you see students struggling with the fundamental issues of intermediate representations and the left side of the learning trajectories?

- How would you improve Teaspoon languages? For what tasks should we be developing new Teaspoon languages?
Collaborators on This Work


- Undergraduate researchers: Aryan Bannerjee, Alexandra Rostkowycz, Erin Shi, Brandon Geng, Jessica Zhang, Ben Steinig, Kashmira Reddy, Kristen Taurence, Angela Li, Derrick White, Jessie Houghton.

- http://computinged.wordpress.com
- http://guzdial.engin.umich.edu

Some of this material is based upon work supported by the National Science Foundation under Grant No. 2030919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Thank you!
SPARE SLIDES