
COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN
Broadening Participation in Computing
by Moving Away from Computer Science
Mark Guzdial
1

1959

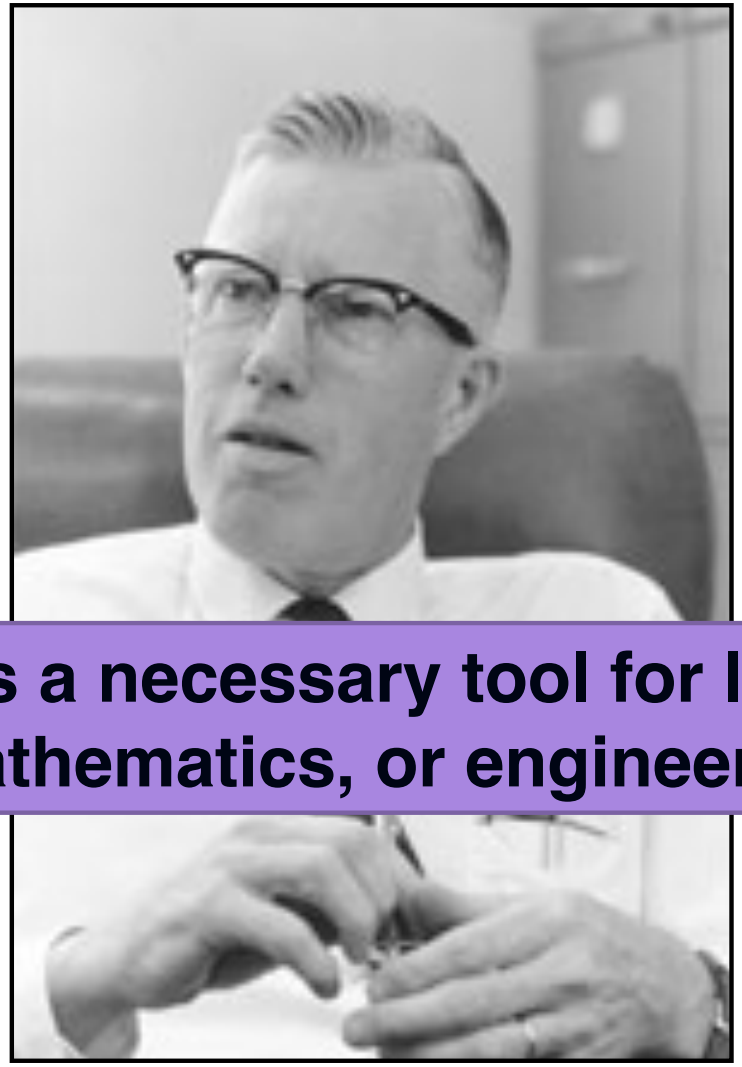
Today's Story

- Computing was created to be taught to *everyone*.
- Computer science has become more narrow.

- Who we reach with computing education today

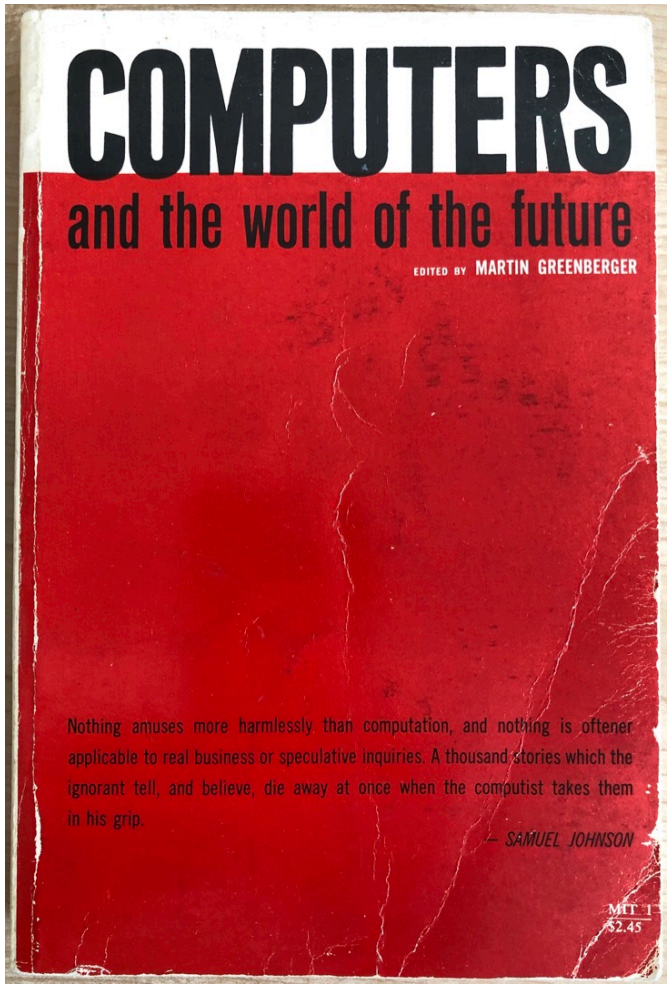
- What should we teach to *everyone* and *how*?
- Broadening Participation in Computing *outside* of computer science classes

1961

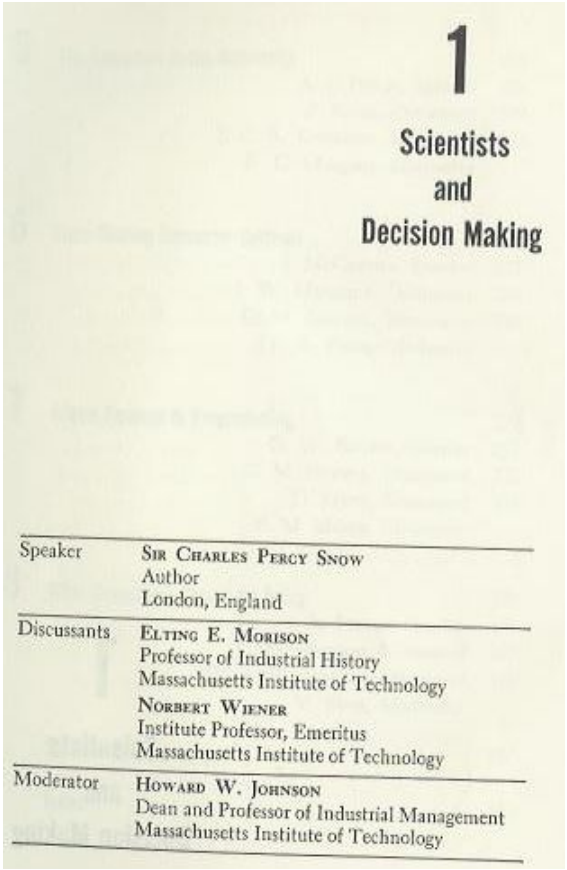


The computer is a necessary tool for learning science, mathematics, or engineering

1968



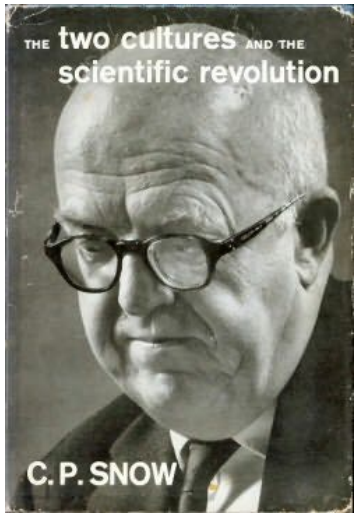
1961



1

Scientists and Decision Making

Speaker	Sir Charles Percy Snow Author London, England
Discussants	Elting E. Morison Professor of Industrial History Massachusetts Institute of Technology Norbert Wiener Institute Professor, Emeritus Massachusetts Institute of Technology
Moderator	Howard W. Johnson Dean and Professor of Industrial Management Massachusetts Institute of Technology



"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."

Peter Naur (1928-2016)

Turing Laureate (2005)

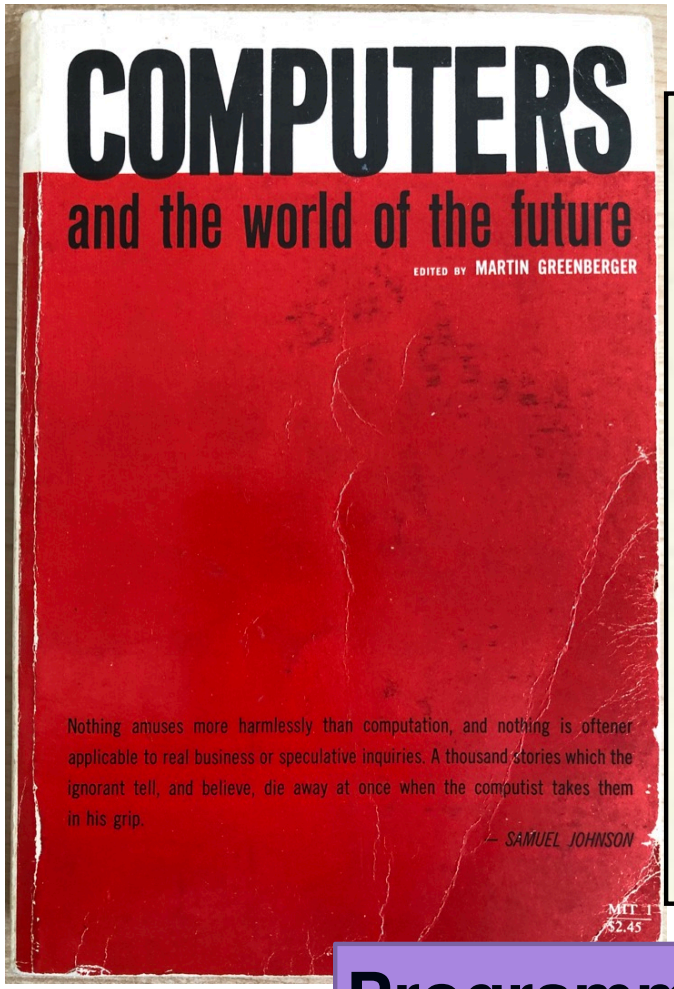
1966: Danmarks Radios Rosenkjærforelæsninger

Datalogi – læren om data

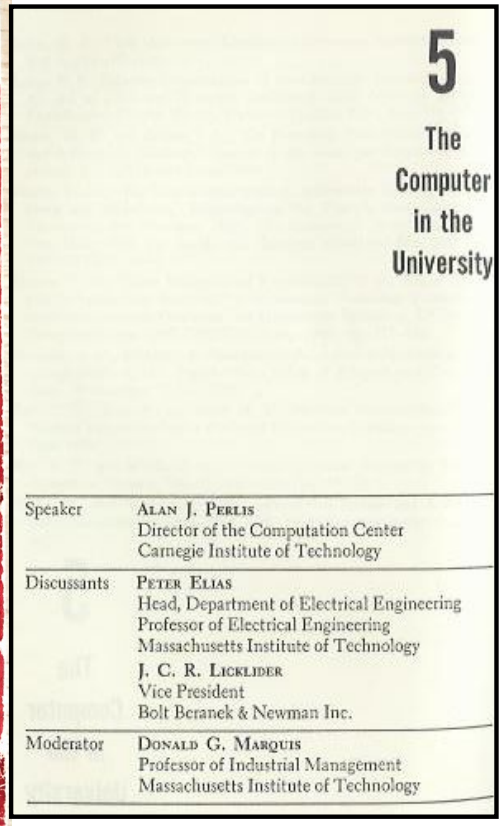
"Once informatics has become well established in general education, the mystery surrounding computers in many people's perceptions will vanish. This must be regarded as perhaps the most important reason for promoting the understanding of informatics. This is a necessary condition for humankind's supremacy over computers and for ensuring that their use do not become a matter for a small group of experts, but become a usual democratic matter, and thus through the democratic system will lie where it should, with all of us."

fag.

Thanks to Michael Caspersen



1961



5

The Computer in the University

Speaker	ALAN J. PERLIS Director of the Computation Center Carnegie Institute of Technology
Discussants	PETER ELIAS Head, Department of Electrical Engineering Professor of Electrical Engineering Massachusetts Institute of Technology J. C. R. LICKLIDER Vice President Bolt Beranek & Newman Inc.
Moderator	DONALD G. MARQUIS Professor of Industrial Management Massachusetts Institute of Technology

Alan Perlis

Programming changes how we understand

Photo: CMU

First published definition of Computer Science

"The study of computers and all the phenomena surrounding them."

***Science*, 1967**

Alan Perlis

Herb Simon

Alan Newell

This is broader than how most people define computer science today.
We might call this *Computing*

Definitions of Computer Science

- Computer Science is the study of computers and computational systems. (U. Maryland CS)

- Computer science is the study of computers and algorithmic processes, including their principles, design, implementation, and impact on society (Tucker, 2006 - K-12 CS Framework)

- Computer science is the foundational discipline with an emphasis on discovery related to programming, algorithms, and data structures. (ACM/IEEE Computing Curriculum 2021)

President Obama “CS for All”

2016

Computer science (CS) is a “new basic” skill necessary for economic opportunity and social mobility.

Photo: ObamaWhitehouse.archives.gov

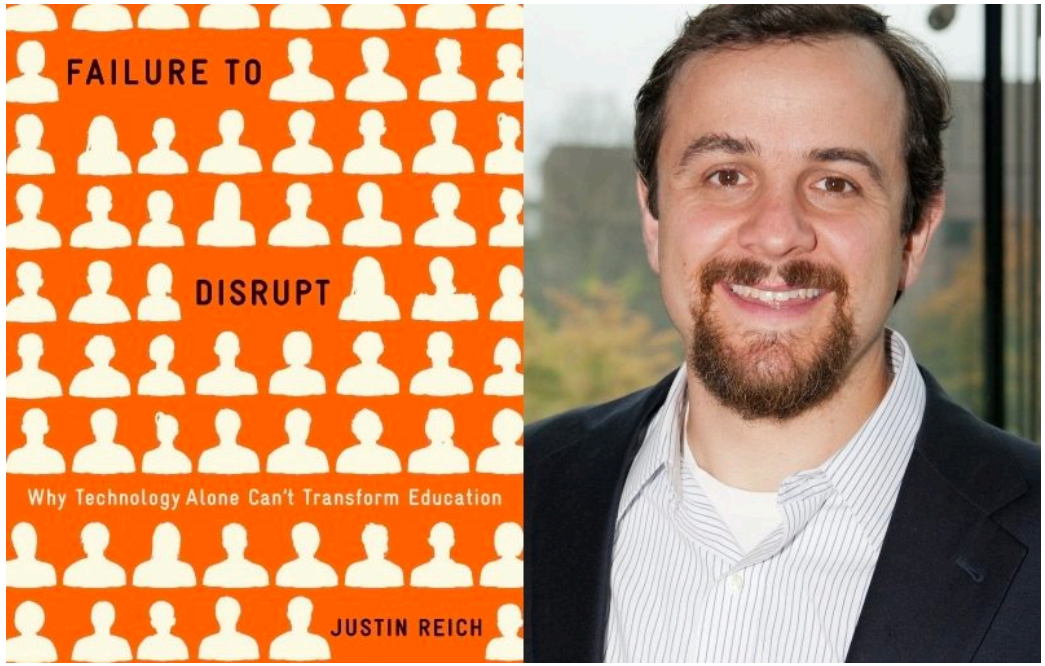
When did this become about “economic opportunity”?

- Forsythe, Perlis, Snow, Naur, Simon, and Newell were all arguing for computing education ***decades*** before Silicon Valley.
  - They weren’t preparing students for software development jobs.
- **What is the goal of Computing Education, if not job skills?**

Who gets access to these powerful ideas?

WHO GETS COMPUTING EDUCATION TODAY

Failure to Disrupt



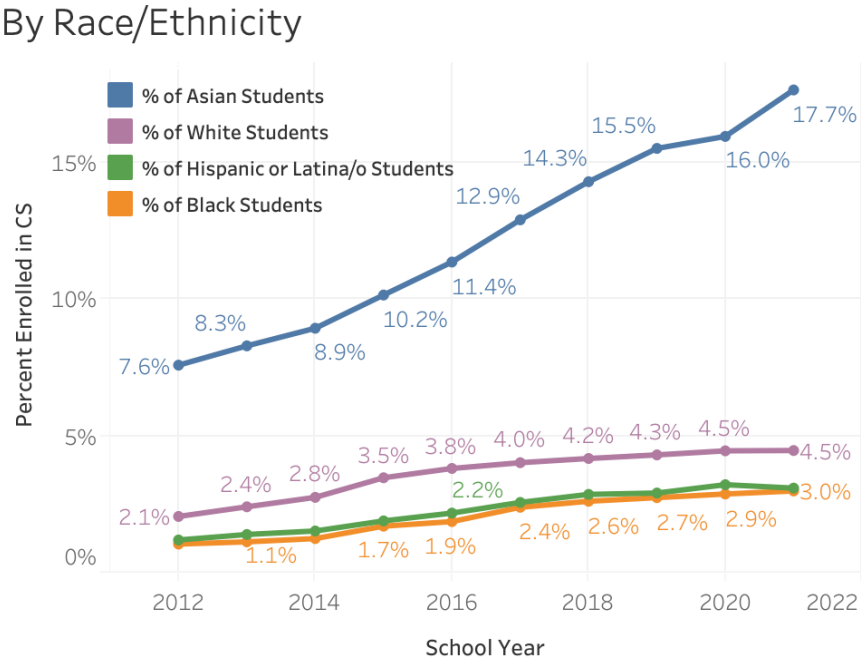
Average number of Scratch projects written by each of the 59+ million Scratch programmers:

< 1

5.6%

5.6%

Texas dashboard accessed through the ECEP State Dashboards page



IN THE 2021-22 SCHOOL YEAR:

4.4% OF ALL HIGH SCHOOL STUDENTS TOOK A CS COURSE

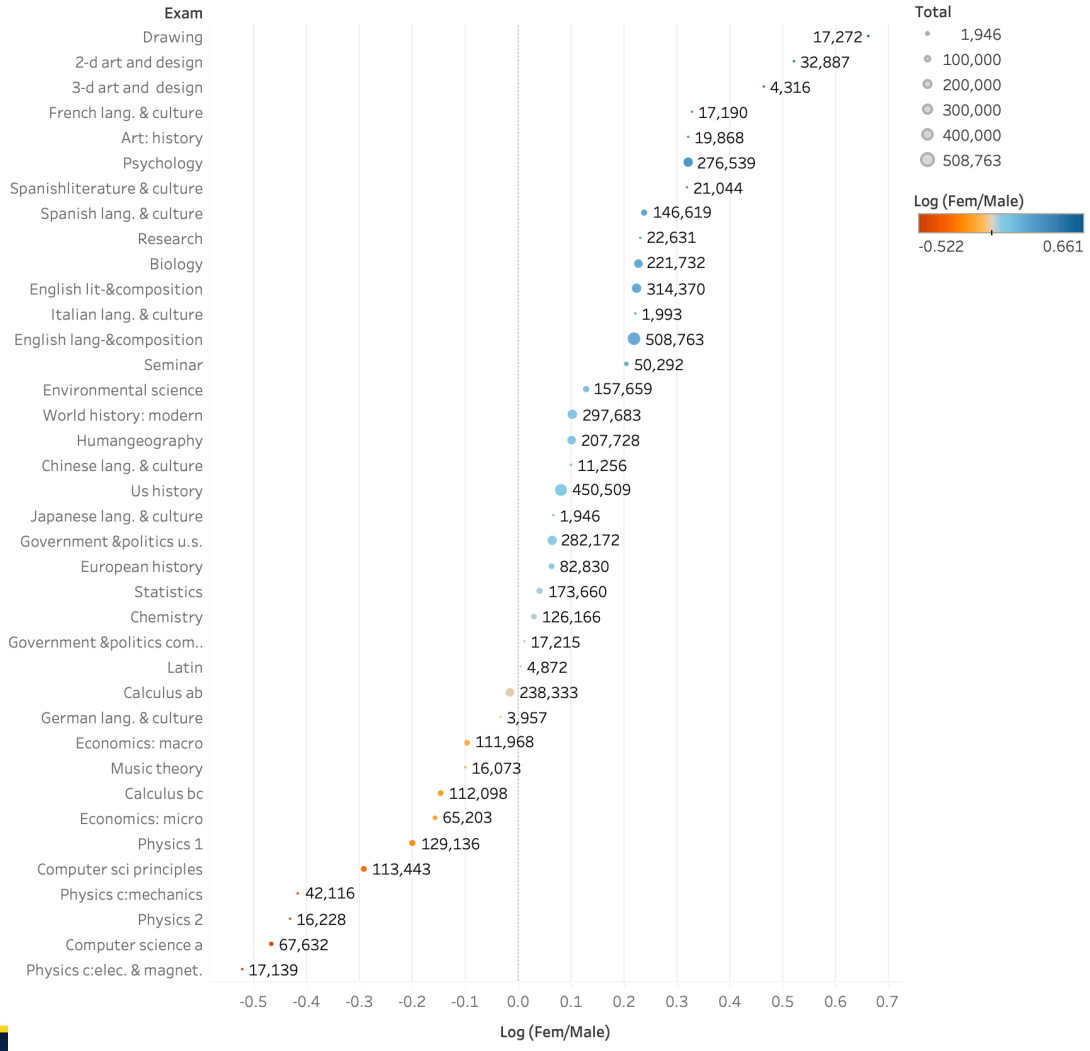
5.6% OF STUDENTS AT SCHOOLS OFFERING CS TOOK A CS COURSE

There is no* computer science in US schools

almost

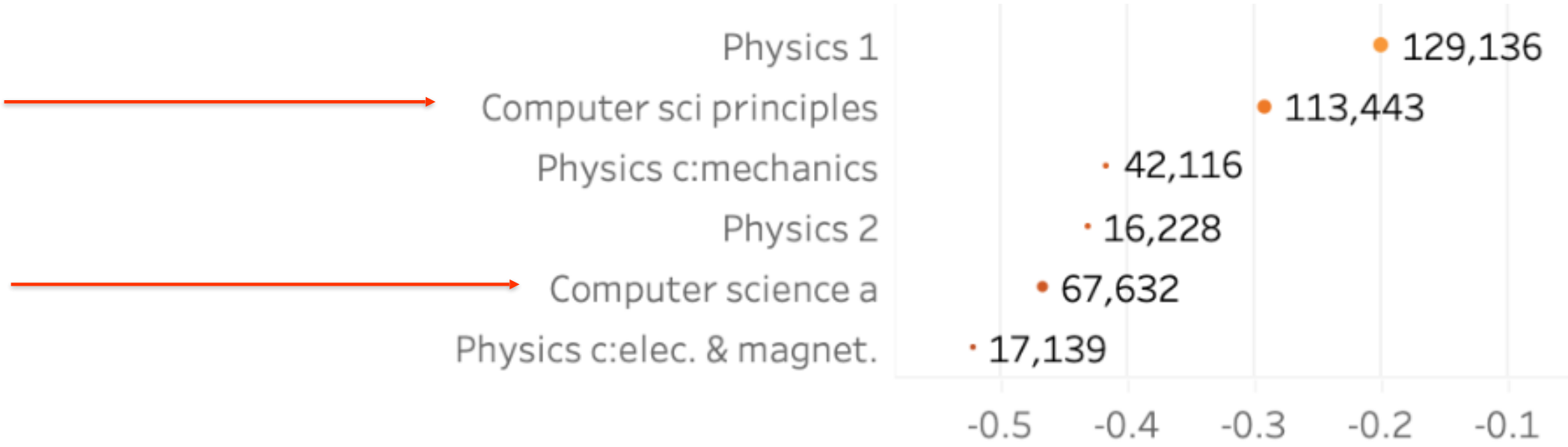
* Over 90% of US high school students never take a CS class

Log (# Female / # Male) for Advanced Placement Exams in 2021



Sum of Log (Fem/Male) for each Exam. Color shows sum of Log (Fem/Male). Size shows sum of Total. The marks are labeled by sum of Total.

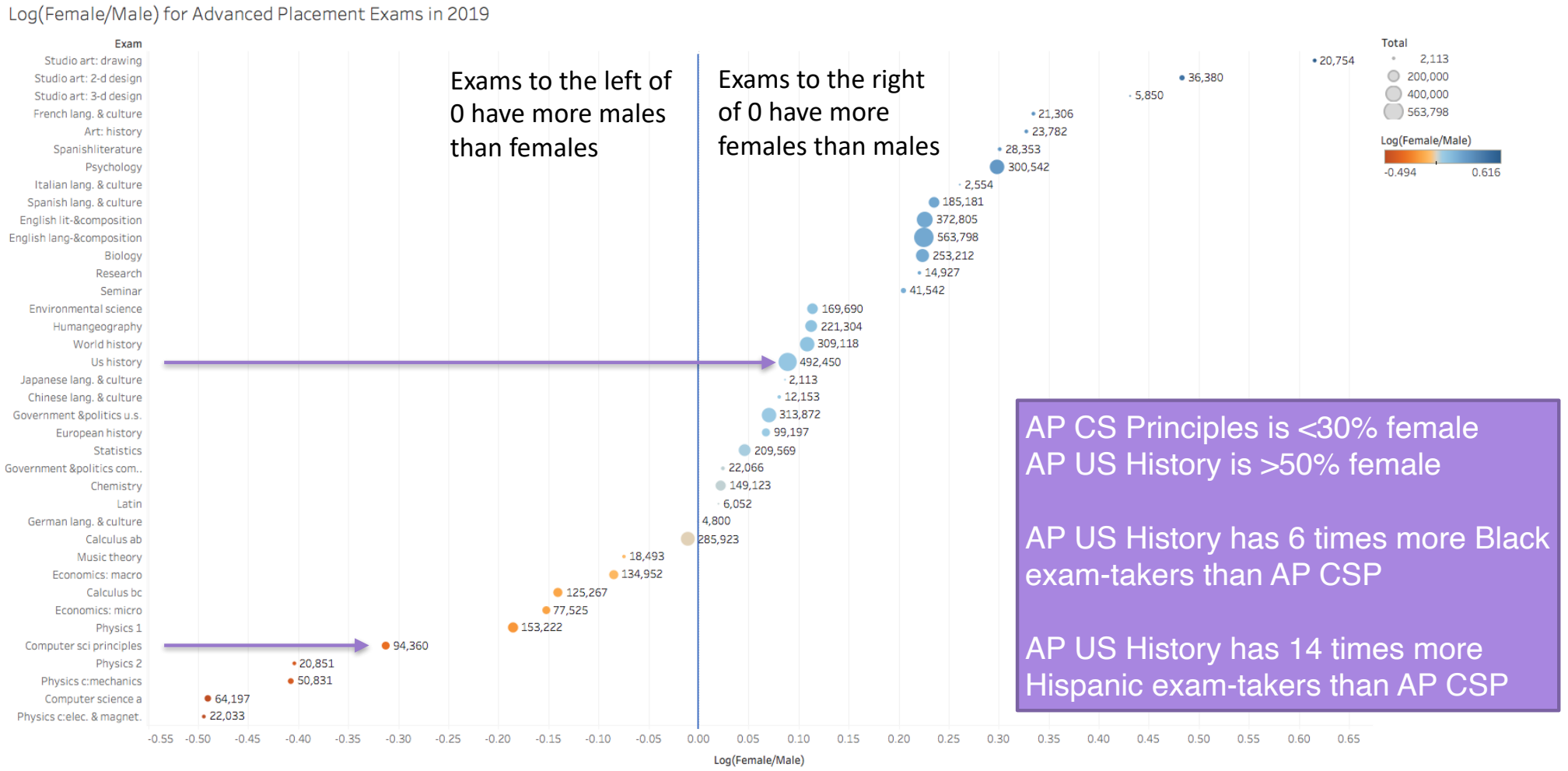
Data and Visualization from Barbara Ericson and Willa Hua



If we want to teach **Computer Science for All**, we have to teach where "All" are.

And that's not CS classes.

Data and Visualization from Barbara Ericson and Willa Hua



AP CS Principles is <30% female
AP US History is >50% female

AP US History has 6 times more Black exam-takers than AP CSP

AP US History has 14 times more Hispanic exam-takers than AP CSP

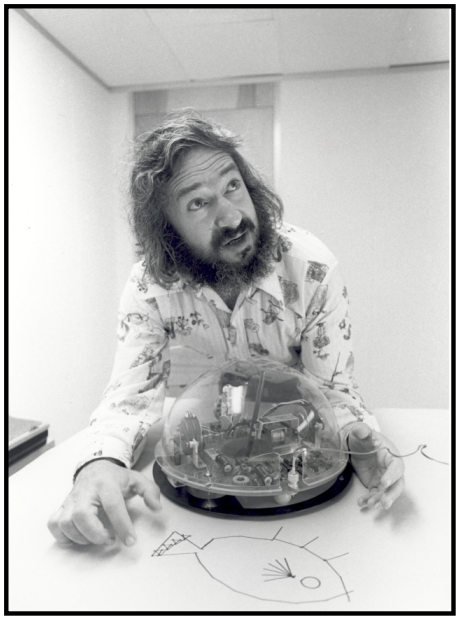
Sum of Log(Female/Male) for each Exam. Color shows sum of Log(Female/Male). Size shows sum of Total. The marks are labeled by sum of Total.

Data and Visualization from Barbara Ericson and Willa Hua

What parts of computing are *really* ***broadly*** useful?

WHAT SHOULD WE TEACH TO *EVERYONE*?

Seymour Papert



```
TO NOUN
OUTPUT PICK [BIRDS DOGS WORMS DONKEYS GEESE CATS [GUINEA PIGS]]
END

TO VERB
OUTPUT PICK [HATE TRIP BITE LOVE]
END

TO ADJECTIVE
OUTPUT PICK [RED PECULIAR JUMPING FAT FUZZY [FUZZY WUZZY]]
END

TO SENGEN
PRINT (SENTENCE ADJECTIVE NOUN VERB ADJECTIVE NOUN)
SENGEN
END
```

When SENGEN is invoked,[1] this code produces sentences such as

```
RED GUINEA PIGS TRIP FUZZY WUZZY DONKEYS
PECULIAR BIRDS HATE JUMPING DOGS
FAT WORMS HATE PECULIAR WORMS
FAT GEESE BITE JUMPING CATS
```

1968

Seymour Papert claimed “that children can learn to program and learning to program can affect the way that they learn everything else.”

Rich, Strickland, Binkowski, Moran, and Franklin (ICER 2017) asked the question:

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

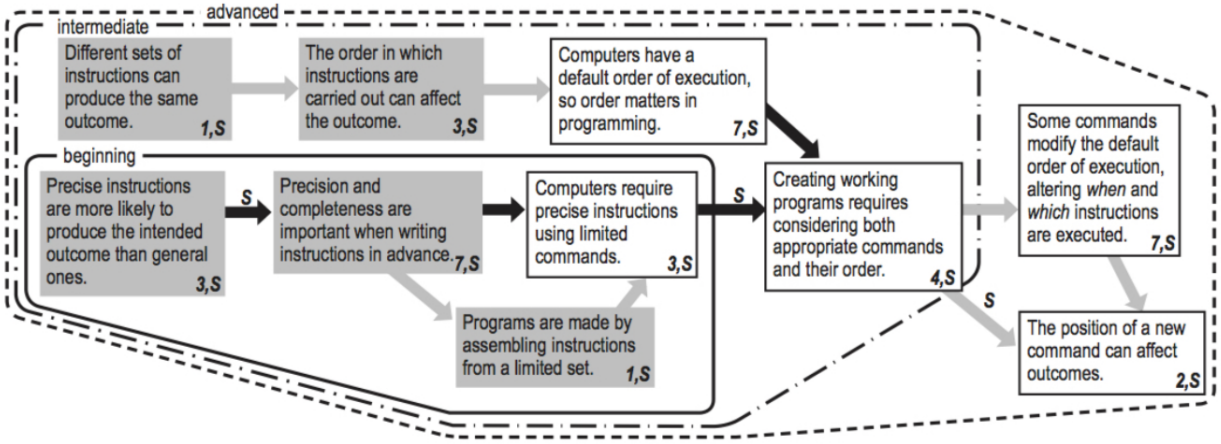


Figure 3: Sequence learning trajectory.

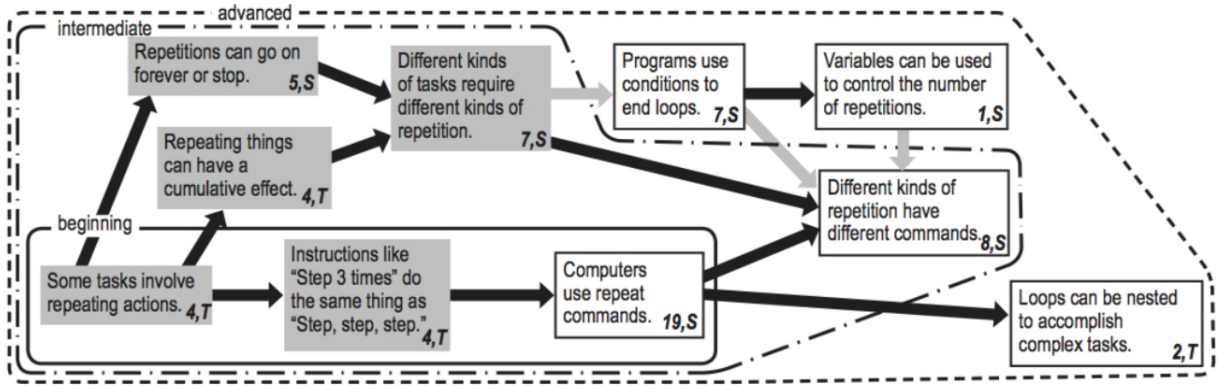


Figure 4: Repetition learning trajectory.

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.

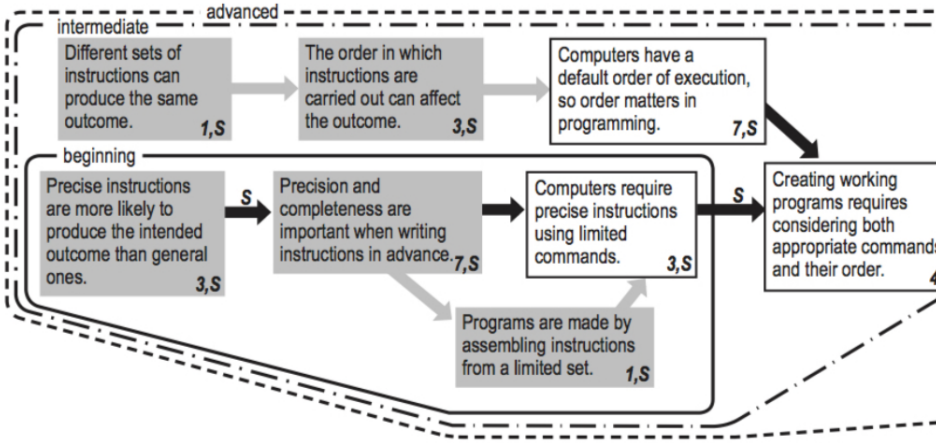


Figure 3: Sequence learning trajectory.

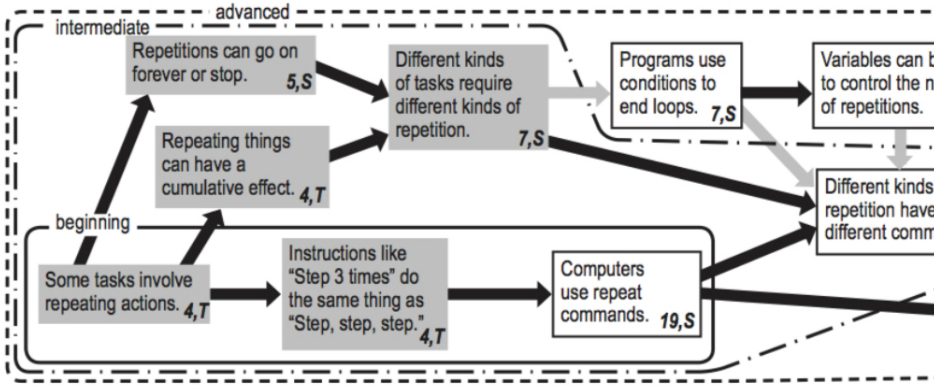
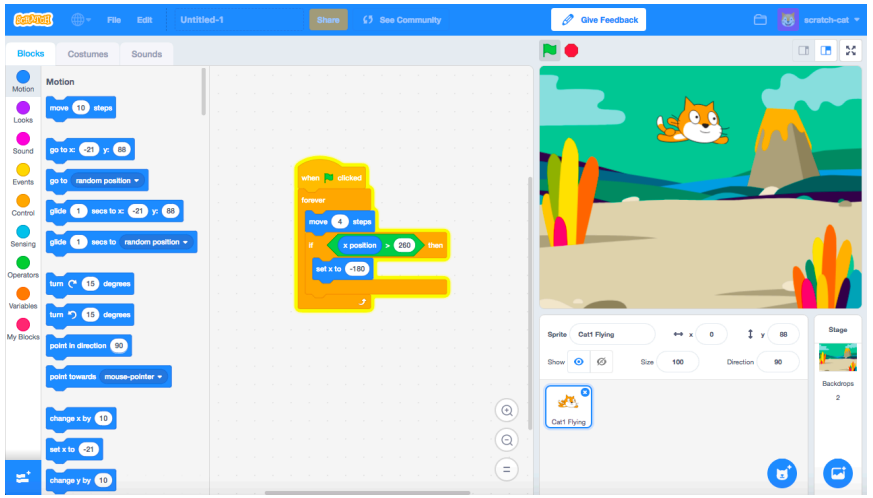


Figure 4: Repetition learning trajectory.

Scratch fluency doesn't need that whole list

- Over 59 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

BOOTSTRAP
Equity • Scale • Rigor

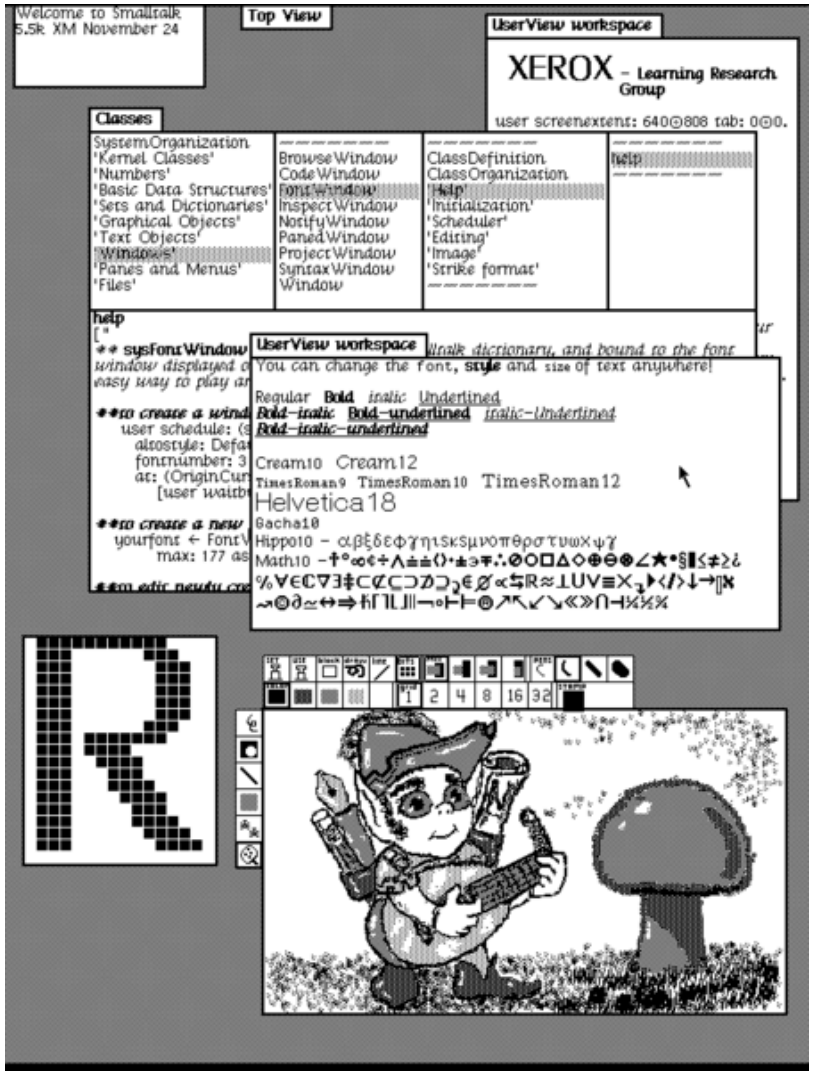
There is *learning* power in even a subset of CS.

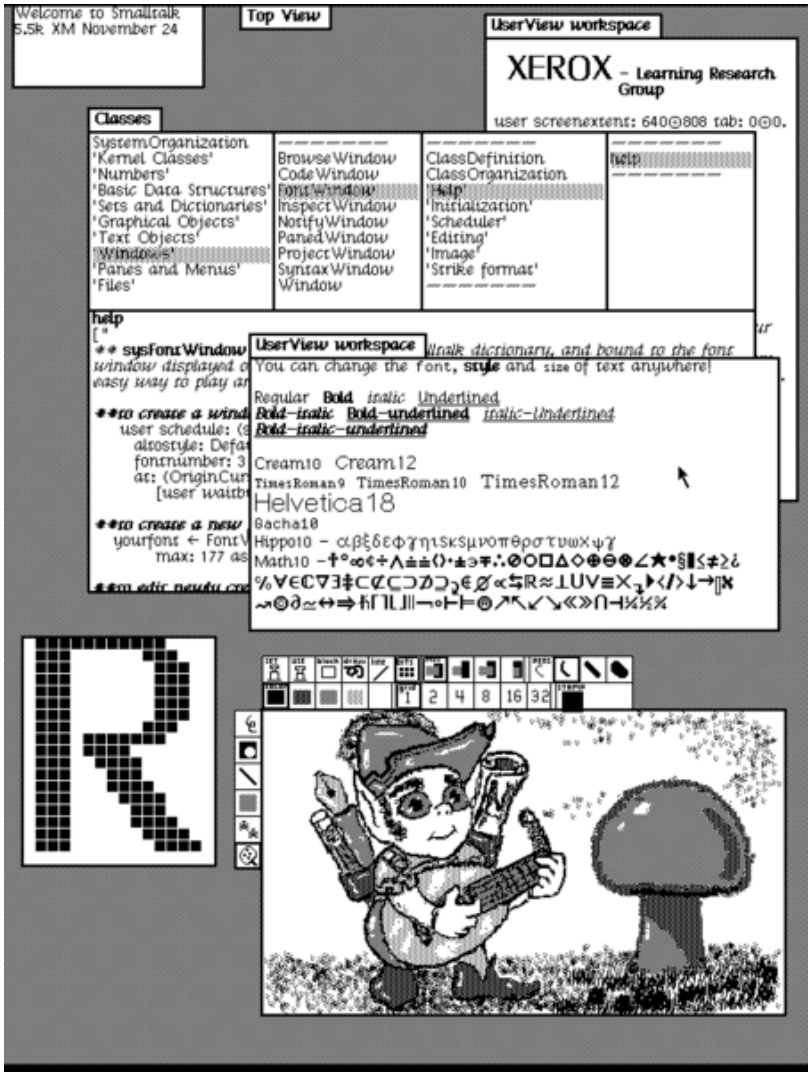
Unit	Game Feature	Programming Concept	Math Concept
1	locating elements on screen	expressions, Circles of Evaluation	coordinates
2	creating text and images	string and image operations	domain, range, kinds of data
3–5	making moving images	defining functions, examples	multiple function representations: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

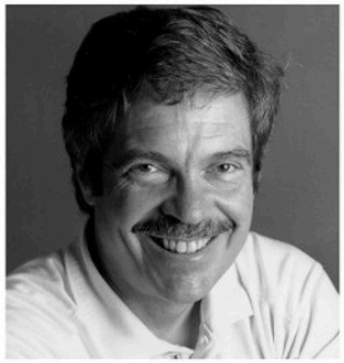
Figure 1: Curriculum structure: each unit introduces game, programming, and math concepts in parallel.

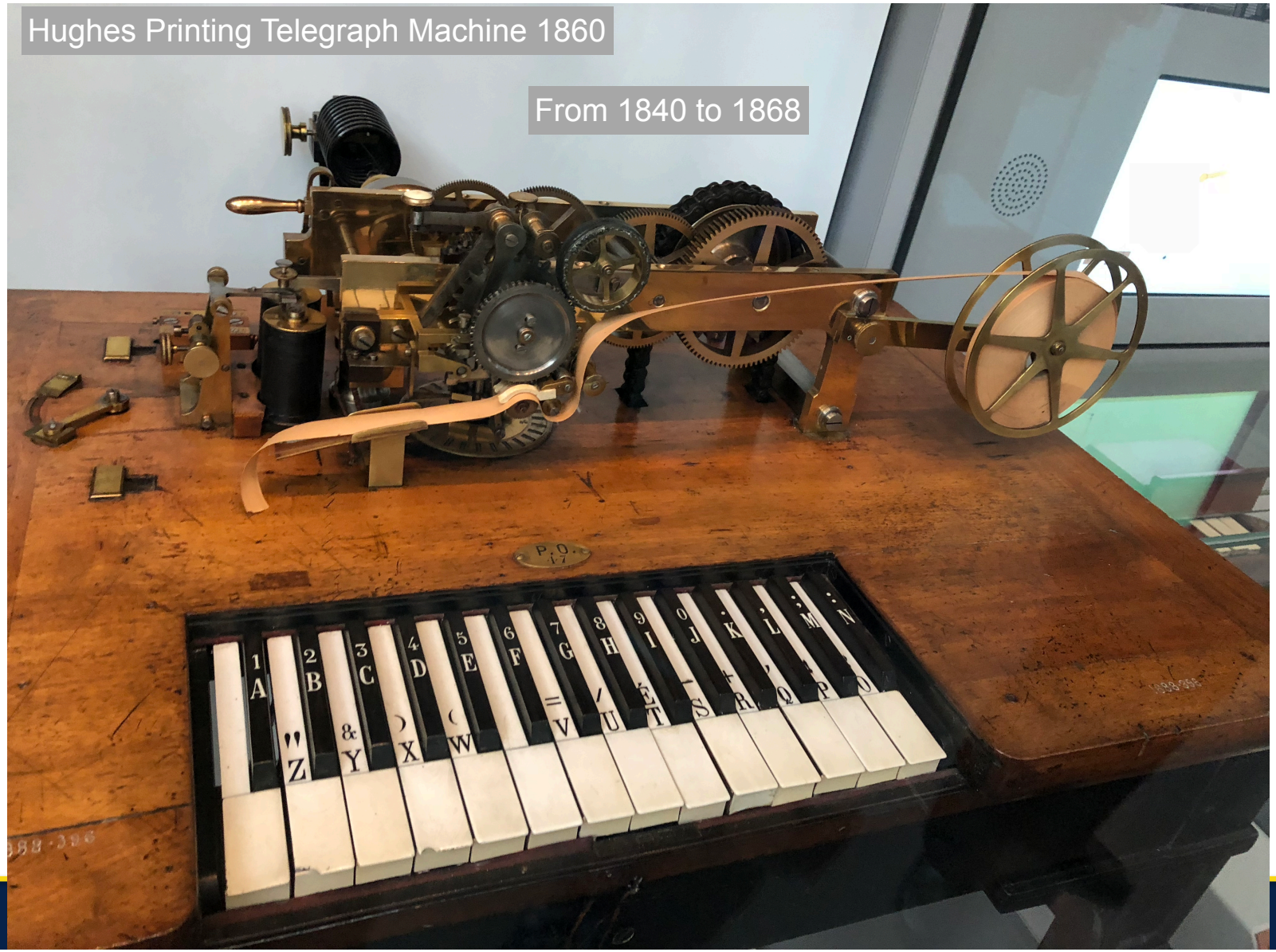
The parts of computing are ***specifically*** useful

CHANGING COMPUTING EDUCATION TO REACH EVERYONE


Welcome to Smalltalk
5.5k XM November 24
Top View
UserView workspace
XEROX - Learning Research Group
user screenextent: 640@808 tab: 0@0.
Classes
SystemOrganization
'Kernel Classes'
'Numbers'
'Basic Data Structures'
'Sets and Dictionaries'
'Graphical Objects'
'Text Objects'
'Windows'
'Panes and Menus'
'Files'
BrowseWindow
CodeWindow
FontWindow
InspectWindow
NotifyWindow
PanedWindow
ProjectWindow
SyntaxWindow
Window
ClassDefinition
ClassOrganization
'Help'
'Initialization'
'Scheduler'
'Editing'
'Image'
'Strike format'
help
help
UserView workspace
You can change the font, style and size of text anywhere!
Regular Bold italic Underlined
Bold-italic Bold-underlined italic-Underlined
Bold-italic-underlined
Cream10 Cream12
TimesRoman9 TimesRoman10 TimesRoman12
Helvetica18
Gacha10
Hippo10 - αβξδεφγηιsκsμνοπθρστυωxψγ
Math10 -†°∞¢÷∧≐≙()·±∋∓∴∅○□Δ◇⊕⊖⊗∠★•§∥≤≠≥¿
%∀∈ℂ∇∃♯⊂⊄⊆⊃∌ϑ∈∅∝⇆ℝ≈⊥UV≡X↓▸‹/›↓→▯✖
⇝◎∂≃↔⇒ℏΓ℩⅃∥¬▫⊢⊨◎↗↖↙↘«»∩⊣¼½¾


Welcome to Smalltalk
5.5k XM November 24
Top View
UserView workspace
XEROX - Learning Research Group
user screenextent: 640@808 tab: 0@0.
Classes
SystemOrganization
'Kernel Classes'
'Numbers'
'Basic Data Structures'
'Sets and Dictionaries'
'Graphical Objects'
'Text Objects'
'Windows'
'Panes and Menus'
'Files'
BrowseWindow
CodeWindow
FontWindow
InspectWindow
NotifyWindow
PanedWindow
ProjectWindow
SyntaxWindow
Window
ClassDefinition
ClassOrganization
'Help'
'Initialization'
'Scheduler'
'Editing'
'Image'
'Strike format'
help
UserView workspace
You can change the font, style and size of text anywhere!
Regular Bold italic Underlined
Bold-italic Bold-underlined italic-Underlined
Bold-italic-underlined
Cream10 Cream12
TimesRoman9 TimesRoman10 TimesRoman12
Helvetica18
Gacha10
Hippo10 - αβξδεΦγηιϗκσμνοπθρστυωχψζ
Math10 -
1 2 4 8 16 32




Hughes Printing Telegraph Machine 1860
From 1840 to 1868

For 30 years, this was the common keyboard

We may still be waiting for our QWERTY keyboard.

We need to find what makes great ideas of computing accessible.

Our Research Focus: Task-Specific Programming

Goal: Integrate programming* to enhance learning in high school and university non-CS classes — where the students are.

- Using participatory design with teachers to result in *adoptable* programming.

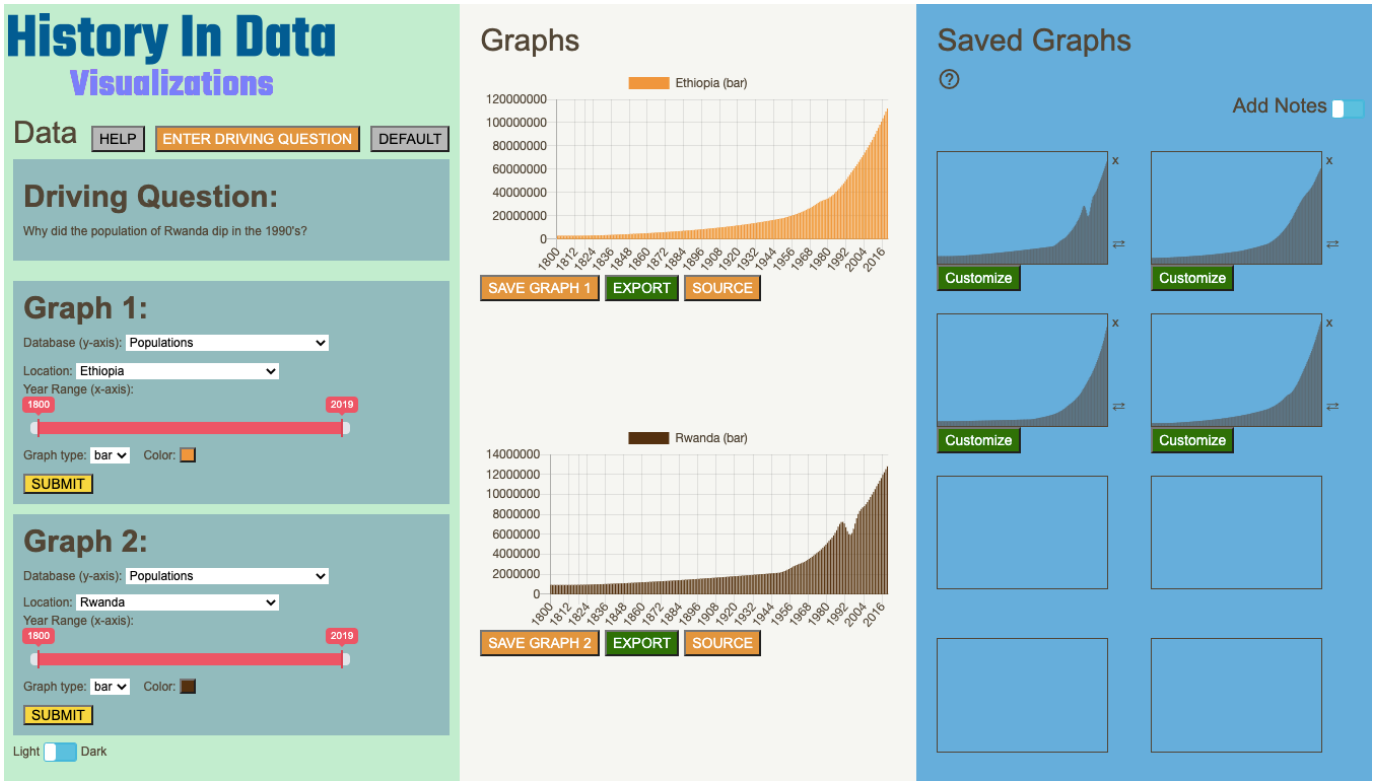
- Building task-specific programming environments to be *highly-usable*.

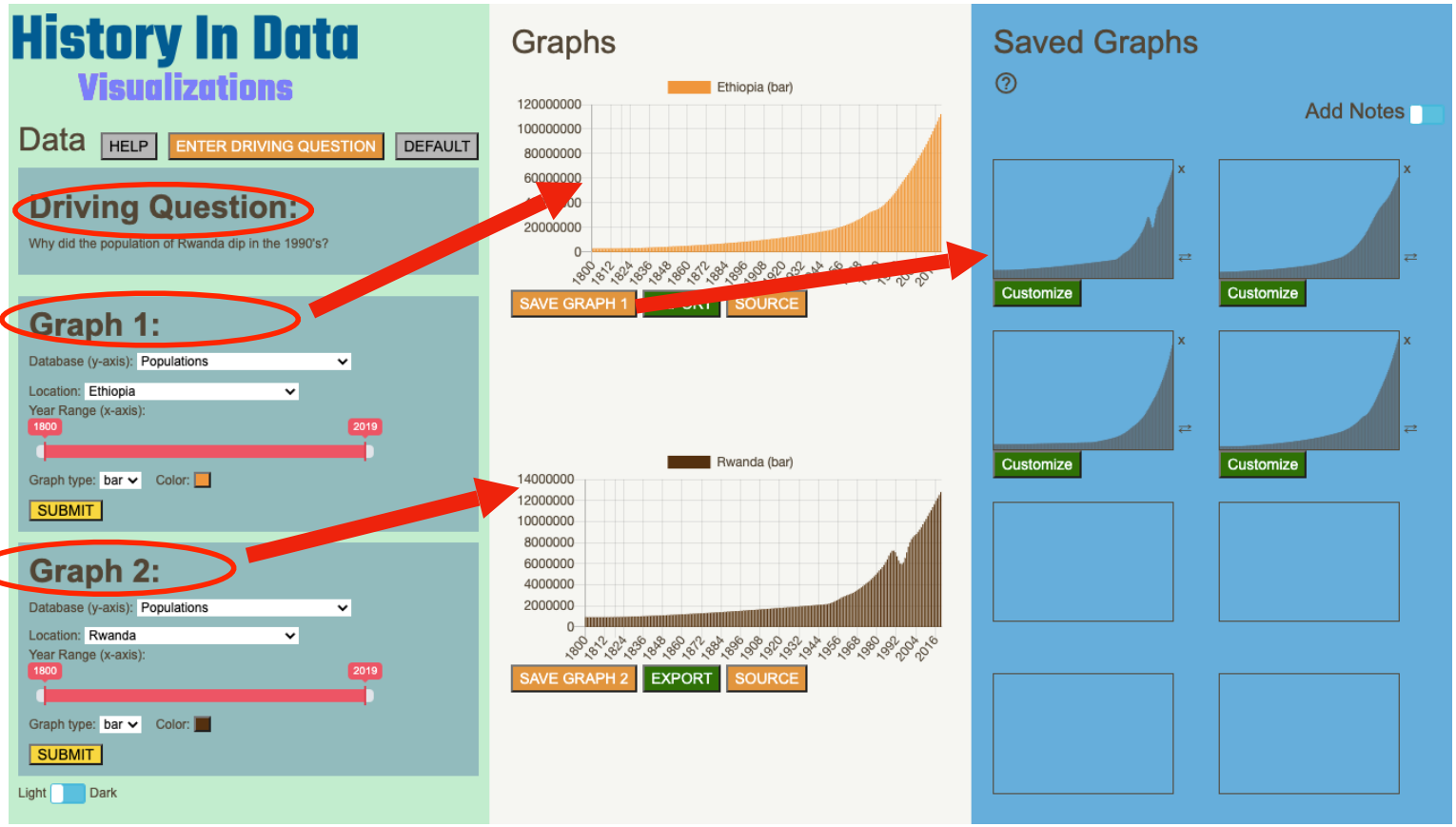
- *TSP* Languages => Teaspoon Languages
  Putting a Teaspoon of Computing in other subjects

DV4L: Data Visualization for Learning

For History Courses

Collaboration with Tammy Shreiner




History In Data
Visualizations
Data
HELP
ENTER DRIVING QUESTION
DEFAULT
Driving Question:
Why did the population of Rwanda dip in the 1990's?
Graph 1:
Database (y-axis): Populations
Location: Ethiopia
Year Range (x-axis):
1800
2019
Graph type: bar
Color:
SUBMIT
Graph 2:
Database (y-axis): Populations
Location: Rwanda
Year Range (x-axis):
1800
2019
Graph type: bar
Color:
SUBMIT
Light
Dark
Graphs
Ethiopia (bar)
120000000
100000000
80000000
60000000
20000000
0
SAVE GRAPH 1
SOURCE
Rwanda (bar)
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
SAVE GRAPH 2
EXPORT
SOURCE
Saved Graphs
Add Notes
Customize
Customize
Customize
Customize

Collaboration with Tammy Shreiner

History In Data

Visualizations

Data

UPLOAD SCRIPT | DEFAULT | CLEAR | HELP

Graph 1:

Database (DB): Populations

Y axis: Rwanda

Year Range: 1800 – 2019

Graph type: bar Color:

SUBMIT

```
{
  "DB": "Populations",
  "Yaxis": "Rwanda",
  "lowDate": 1800,
  "highDate": 2019,
  "gtype": "bar",
  "color": "orange"
}
```

Graph 2:

Database (DB): Populations

Y axis: Congo

Year Range: 1800 – 2019

Graph type: bar Color:

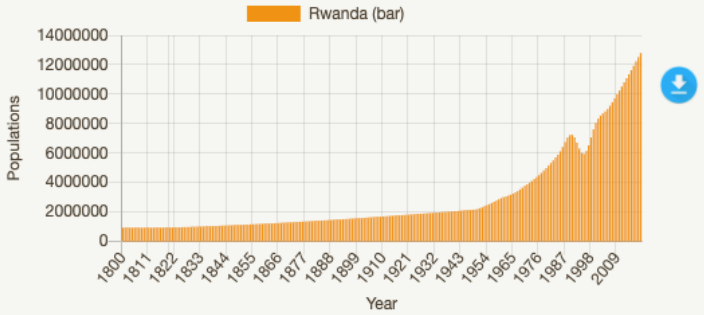
SUBMIT

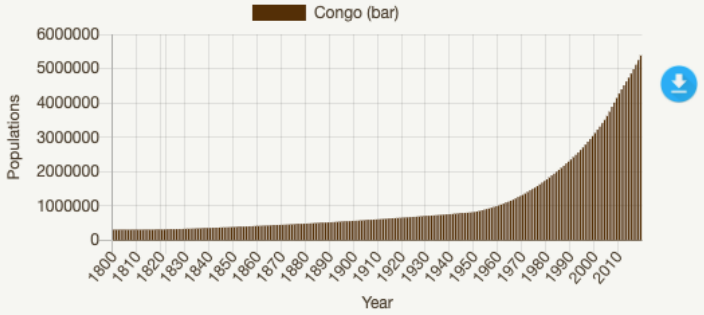
```
{
  "DB": "Populations",
  "Yaxis": "Congo",
  "lowDate": 1800,
  "highDate": 2019,
  "gtype": "bar",
  "color": "darkBrown"
}
```

Light Dark

Graphs

Are there any noticeable differences in the trend of population growth in the following countries? Why?





Collaboration with Tammy Shreiner

Learning challenges that our teachers face

Our students, too.

- ***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.

Pixel Equations

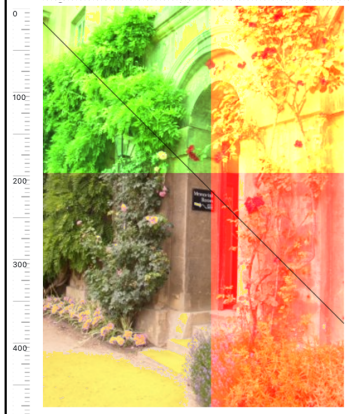
Designed For High School Math and Engineering classes

If this is true Si esto es cierto	Set Red Asignar Rojo	Set Green Asignar Verde	Set Blue Asignar Azul
x > 200	255		
y < 200		2 * green	
blue > 200			blue / 2
x = y - 20	0	0	0

Step 3: Run Equations

Result Picture Appears Here:

Show Result



Pixel Equations

Select your preferred language

◉English

Idioma/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`)).

If this is true Si esto es cierto	Set Red Asignar Rojo	Set Green Asignar Verde	Set Blue Asignar Azul
x > 200	255		

Step 3: Run Equations

Result Picture Appears Here:

Show Result

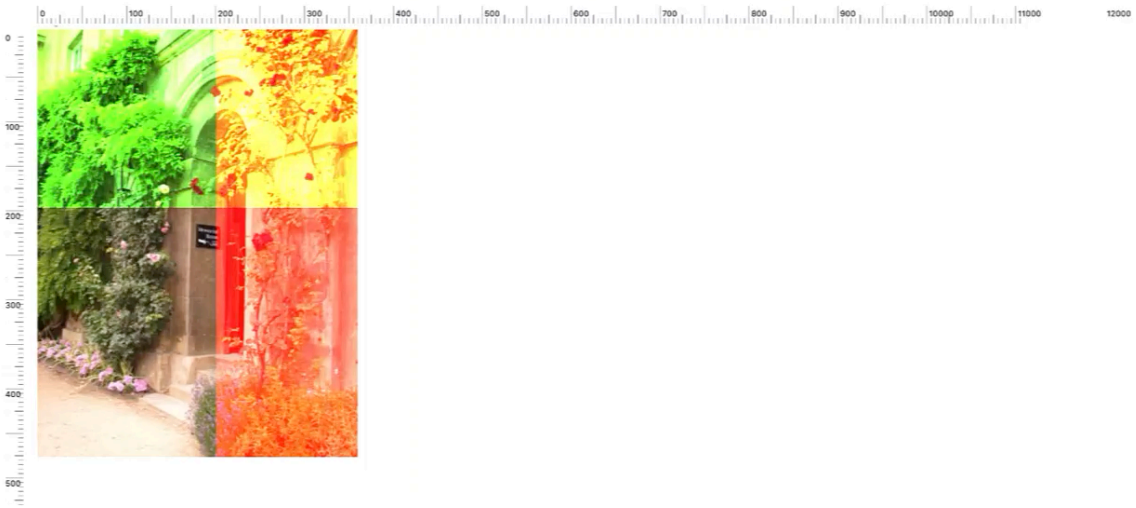
set red/rojo to 255-red (o 255-rojo)).

If this is true Si esto es cierto	Set Red Asignar Rojo	Set Green Asignar Verde	Set Blue Asignar Azul
x > 200	255		
y < 200		2 * green	

Step 3: Run Equations

Result Picture Appears Here:

Show Result



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`)).

If this is true Si esto es cierto	Set Red Asignar Rojo	Set Green Asignar Verde	Set Blue Asignar Azul
x > 200	255		
y < 200		2 * green	
blue > 200			blue / 2

Step 3: Run Equations

Result Picture Appears Here:

Show Result

Defining a Teaspoon Language

- They can be used by students for a task that is useful to a teacher.
  - Integrating into formal, mandatory schooling through teachers is the best way to get broad access and participation.
- They are programming languages, i.e., a notation for defining a computational process.
- They can be learned within 10 minutes, so students can learn and use them within a single class session.

Broadening access and participation in computing

WHAT WE'RE TRYING AT MICHIGAN

Program in Computing for the Arts and Sciences

Mark Guzdial, mjguz@umich.edu

PCAS Story

- University of Michigan is creating a new **Program in Computing for the Arts and Sciences** in our College of Literature, Science, and the Arts (LSA).
  - No connection to our Computer Science & Engineering Division nor our School of Information.

Key Points:

1. A process of self-study: What do LSA students need in computing education? What do we offer?
2. A participatory design process for our first two courses
3. A scaffolded series of computing activities

LSA Computing Education Task Force

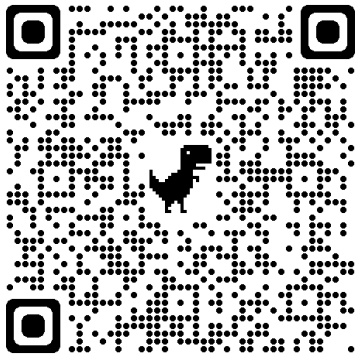
What does LSA need in Computing Education?

Charged the Computing Education Task Force 2020-2021

- What do LSA students need to know about computing?
- What classes and programs already exist?
- Where should we be going?

- Conducted dozens of interviews, reviewed hundreds of courses, surveyed over 100 LSA faculty.

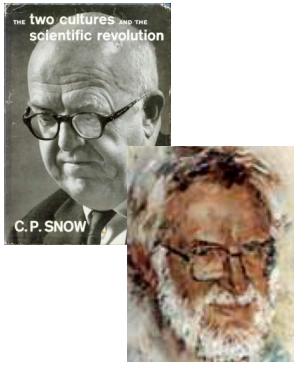
- Final report is available:



3 Themes for Computing Education in LSA

- ***Computing for Discovery***: Computational science enables new discoveries across natural and physical sciences.

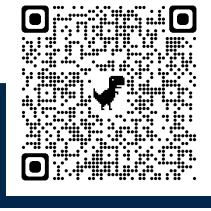
- ***Computing for Expression***: Computing has changed how we communicate and engage with others, from social media to Pixar to AR/VR.



- ***Critical Computing,* or *Computing for Justice*:** Computers and applications are pervasive in our daily lives, and thus have immense cultural, social, and political influence. Who is supported by computing, who is oppressed, and how can we create better models?

Some Findings from the Task Force

- Roughly half of LSA students take one of the ”Big Three” courses that include programming in Statistics and Computer Science. Only 20% take more.
  - Most LSA computing classes are in Discovery.
    Computing-intensive courses in Expression and Justice are electives.
- Faculty want an LSA perspective on computing education:
  - “I think **our goal should be for LSA students to receive more of their computing education within LSA**, **with a liberal arts/basic science perspective**. To me this means not just writing code, but interpreting and critiquing the products of the computation.”
- We found no institution in the US that is providing computing education across all three themes.



Creating PCAS

Program in Computing for the Arts and Sciences

Launched Summer 2022 - me and Gus Evrard, a first-generation computational cosmologist

- Goals:
  - To **meet the needs of all LSA students** to learn about computing, especially programming.
  - To create **new computing courses** around the themes of justice, expression, and discovery.
  - To **create new programs and credentials** to provide computing-centric majors and minors in all divisions

Developing Courses

Course code for PCAS: **COMPFOR** – COMPuting FOR…

First courses are being taught this year:

- **COMPFOR 111 "Computing's Impact on Justice: From Text to the Web"**
- **COMPFOR 121 "Computing for Creative Expression"**

Courses being introduced in Fall 2023:

- COMPFOR 131 "Introduction to Python for the Sciences." Worked with faculty across the Natural Sciences to develop the new course.
- COMPFOR 101 "The Transistor Disruption: How a Tiny Tool Transforms Society and Science"

What should be in these classes?

Formed advisory groups of faculty who self-identified as being about Computing for Expression or Computing for Justice.

Created two sets of shared whiteboards of:

1. Learning objectives that were identified by the computing education task force
2. Student activities to support reaching those learning objectives.

- Instructions: “Please move to the right those that you think are important for the course, and move the left those that you think are less or unimportant for the course.”

Mark Guzdial · 1m

Endstate 2/2 - LSA Computing Learning Goals for Justice

What do we want students to know/do? Put important ones to the right, and less useful to the left.

- Write a program to algorithmically generate a sentence, a picture, or a sound.
- Explain the difference between MIDI and MP3
- Know the difference between Twine and Unreal Game Engines
- Create a model of some phenomena, run the simulation of the model, and compare the data to another dataset.
- Compute statistics on a spreadsheet and make a graph
- Write secure, safe, and robust code.
- Build a game in Gamemaker.
- Use a Web service API in JavaScript
- Write the program "Hello World!" in a textual language (like C++ or Python)
- Build a web page in HTML and CSS.
- Build an iOS or Android app.
- Know the difference between C++, Python, and Snap!
- Build an image filter in some programming language
- Write the program "Hello World!" in a block-based language (like Snap! or Scratch)
- Explain the difference between digital and analogue, using the example of Spotify and vinyl records.
- Be able to talk to programmers about their processes, including references to Github and IDEs.
- Explain the impact of bitcoin mining on the environment.
- Know what a GPU is and what it has to do with making virtual reality work.
- Explain what blockchain is and how it's related to NFTs
- Use a Jupyter Notebook
- Explain why programming languages are a barrier to non-English language speakers.
- Explain what an API is for a website or library
- Explain how the Internet works at the level of servers, domain names, and IP addresses.
- Explain how a database is used to generate HTML pages through a template
- Understand how user behavior data can be analyzed and inferences made
- Explain why social engineering is a great cybersecurity risk.
- Critique a website for its accessibility.
- Explain how and why facial recognition syste can be biased.
- Describe how sounds and pictures can be represented in numbers or bits.
- Download Facebook or Twitter data to analyze it for keyword trends or sentiment.
- Scrape a website for data and put the data into a spreadsheet for analysis.

Be able to talk to programmers about their processes, including references to Github and IDEs.

Use a Jupyter Notebook

Explain why programming languages are a barrier to non-English language speakers.

Explain what an API is for a website or library

Explain how the Internet works at the level of servers, domain names, and IP addresses.

Explain how a database is used to generate HTML pages through a template

Understand how user behavior data can be analyzed and inferences made

Explain why social engineering is a great cybersecurity risk.

Critique a website for its accessibility.

Explain the impact of bitcoin mining on the environment.

Know what a GPU is and what it has to do with making virtual reality work.

Explain what blockchain is and

Explain how and why facial

Describe how sounds and pictures

dial • 1m

te 2/2 - LSA Computing Learning Goals for Justice

want students to know/do? Put important ones to the right, and less useful to the left.

rogram to
nically
a sentence,
, or a sound.

e
between
MP3

between
Unreal
nes

model of
enomena,
imulation of
el, and
the data to
dataset.

ute statistics
preadsheet
ake a graph

Write secure, safe, and robust code.

Write the program "Hello World!" in a textual language (like C++ or Python)

Build a game in Gamemaker.

Build a web page in HTML and CSS.

Use a Web service API in JavaScript

Build an iOS or Android app.

Be able to talk to programmers about their processes, including references to Github and IDEs.

Know the difference between C++, Python, and Snap!

Build an image filter in some programming language

Write the program "Hello World!" in a block-based language (like Snap!

Explain the impact of bitcoin mining on the environment.

Know what a GPU is and what it has to do with making virtual reality work.

Explain what blockchain is and

Mark Guzdial · 1m

Endstate 2/4 - LSA Computing Learning Goals for Expression

What do we want students to know/do? Put important ones to the right, and less useful to the left.

Explain why social engineering is a great cybersecurity risk.

Write secure, safe, and robust code.

Explain how the Internet works at the level of servers, domain names, and IP addresses.

Build an image filter in some programming language

Build a web page in HTML and CSS.

Explain the difference between MIDI and MP3

Know what a GPU is and what it has to do with making virtual reality work.

Know the difference between C++, Python, and Snap!

Write the program "Hello World!" in a textual language (like C++ or Python)

Write the program "Hello World!" in a block-based language (like Snap! or Scratch)

Critique a website for its accessibility.

Describe how sounds and pictures can be represented in numbers or bits.

Explain what an API is for a website or library

Create a model of some phenomena, run the simulation of the model, and compare the data to another dataset.

Download Facebook or Twitter data to analyze it for keyword trends or sentiment.

Use a Jupyter Notebook

Explain what blockchain is and how it's related to NFTs

Explain why programming languages are a barrier to non-English language speakers.

Explain how and why facial recognition systems can be biased.

Compute statistics on a spreadsheet and make a graph

Scrape a website for data and put the data into a spreadsheet for analysis.

Know the difference between Twine and Unreal Game Engines

Write a program to algorithmically generate a sentence, a picture, or a sound.

Talk to programmers about their processes, including references to Github and IDEs.

Explain the impact of bitcoin mining on the environment.

Build an iOS or Android app.

Designing an interface

Build a game in Gamemaker.

Explain the difference between digital and analogue, using the example of Spotify and vinyl records.

Use a Web service API in JavaScript

Explain how a database is used to generate HTML pages through a template

Common learning goals for the courses

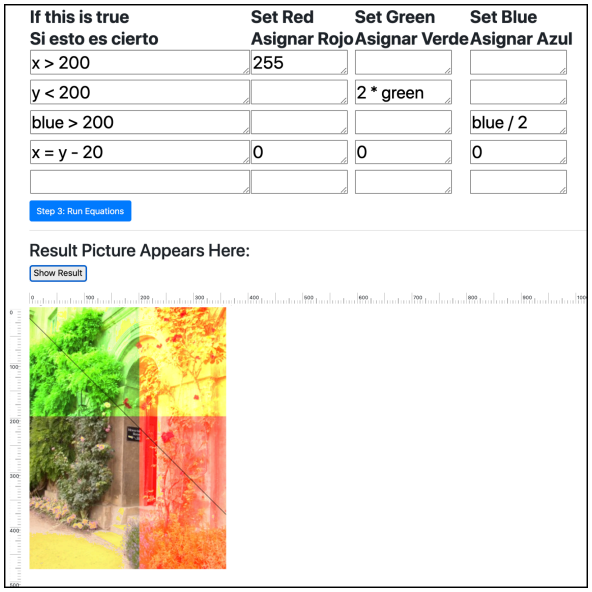
"Don't scare them off!" and "They don't want math"

- To become "conversational programmers": To know the processes and language of software development, to be effective in managing and working with programmers.
- To develop self-efficacy, confidence that they *can* use programming, that they have *seen* programming, and that they could *learn more* programming.
- To use programming to learn a set of concepts (e.g., about computational art, about computation's impact on society).

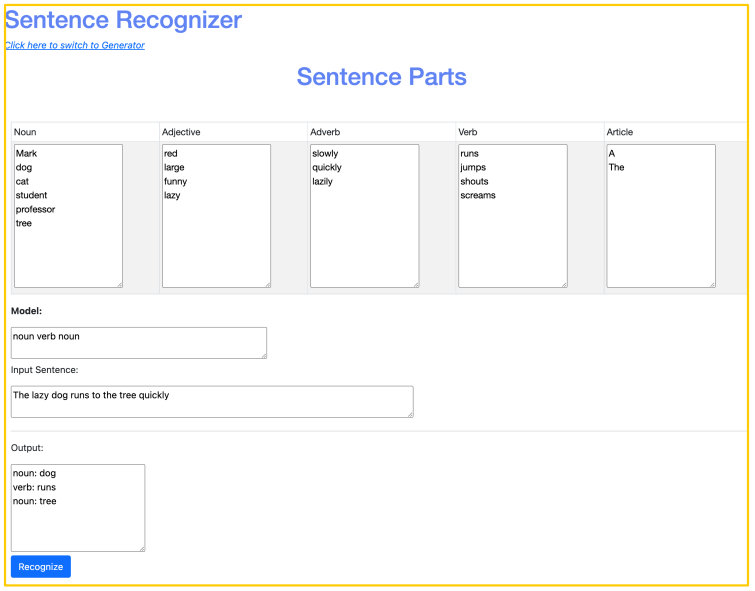
Scaffolded Structure in COMPFOR Courses

Learning Programming in Context, Step 1

Students start learning computational concepts using task-specific programming languages, “teaspoon languages.”

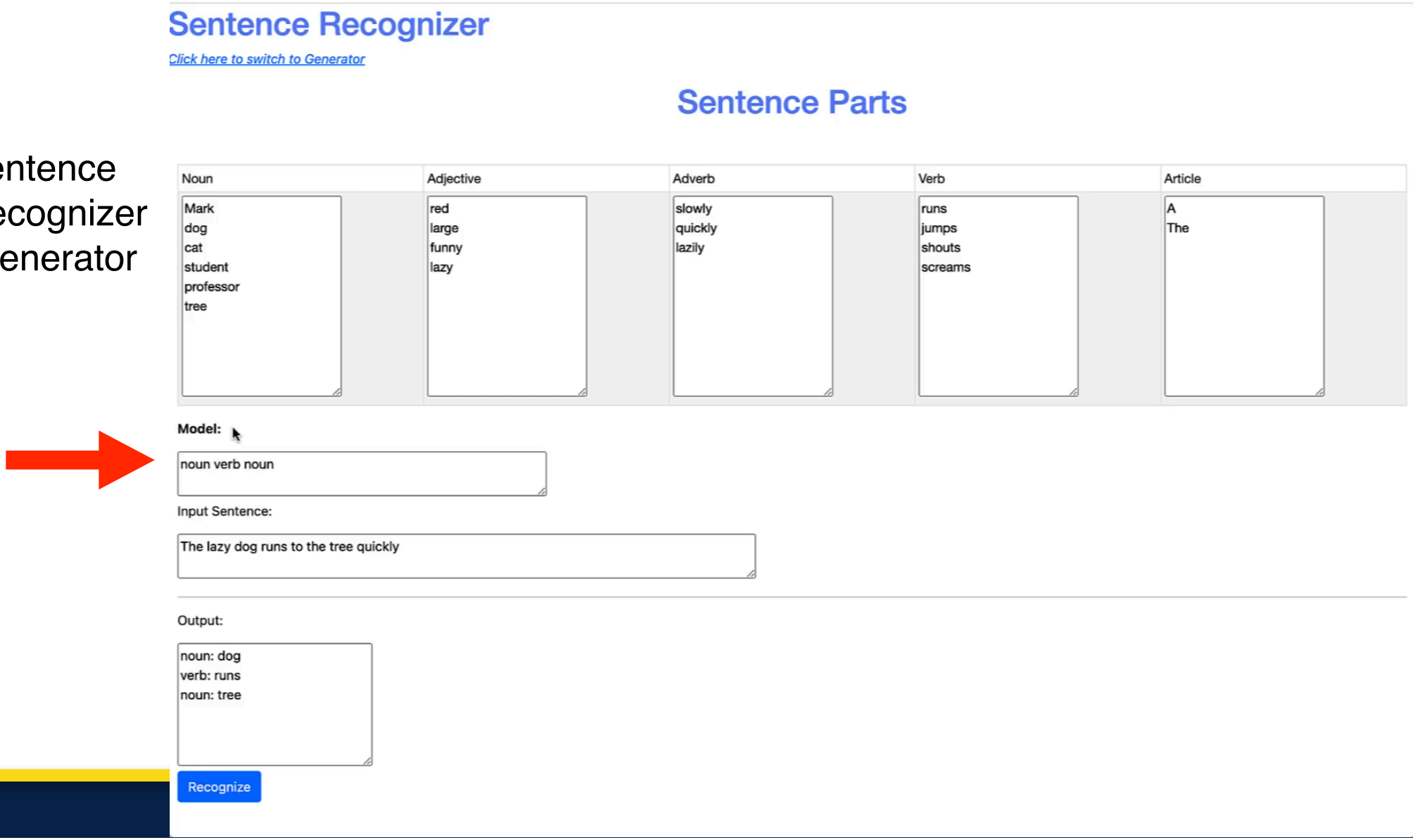


For *Creative Expression*, learning how image filters work.



For *Computing’s Impact on Justice,* learning how computers recognize sentence parts.

Sentence Recognizer /Generator



What we ask students to do with teaspoon languages

- Here is a filter for negation. What happens with numbers other than 255?

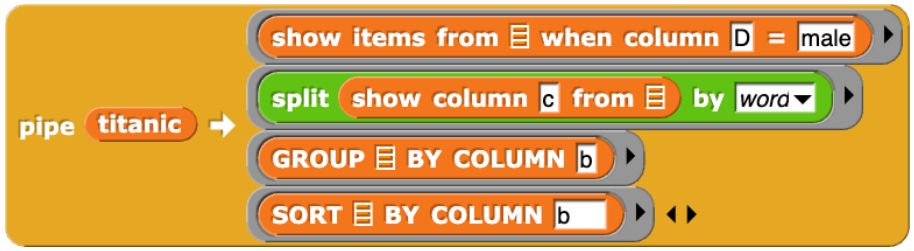
If this is true Si esto es cierto	Set Red Asignar Rojo	Set Green Asignar Verde	Set Blue Asignar Azul
x > 0	255 - red	255-green	255-blue

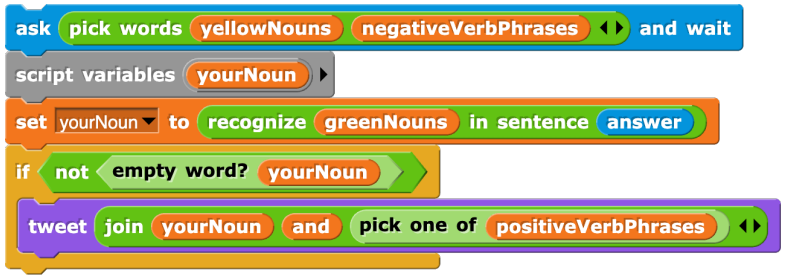
- Can you build a sentence model that understands both sentences?
  - The boys get lunch at the Mall *and* The boys grab lonche at the Mall
  - I am relieved that the test was rescheduled *and* Estoy relieved that the test was rescheduled.

Learning Programming in Context, Step 2

Students build their own programs using *Snap*, a block-based language developed at the U. California-Berkeley. Examples:

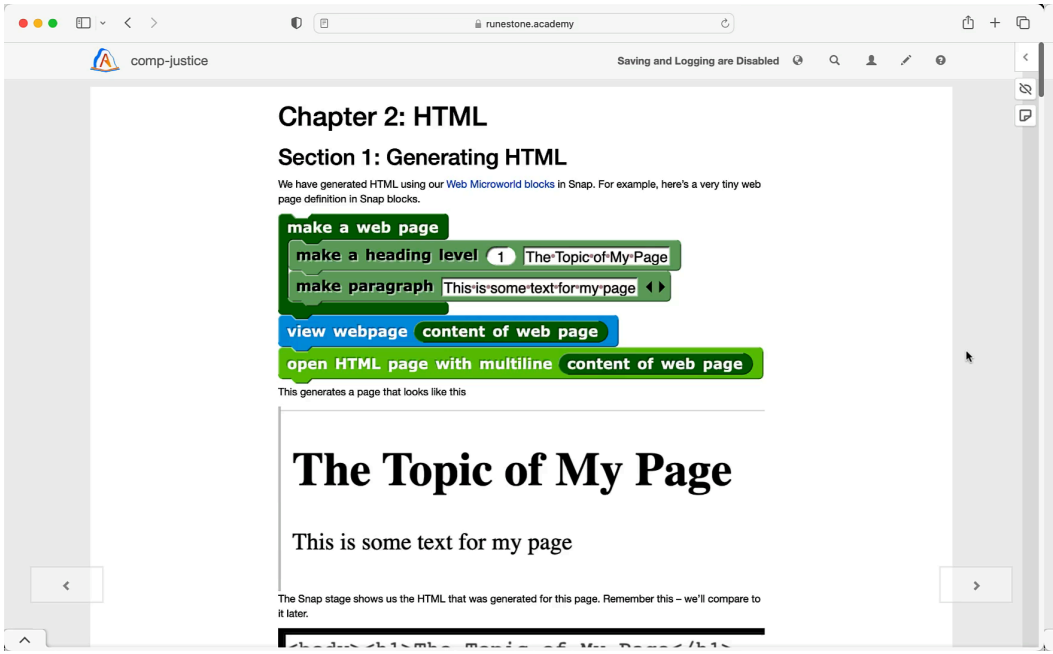
- For *Creative Expression*, building their own image and sound filters.
- For *Computing's Impact on Justice,* building biased chatbots to understand how misinformation spreads, and building database queries.

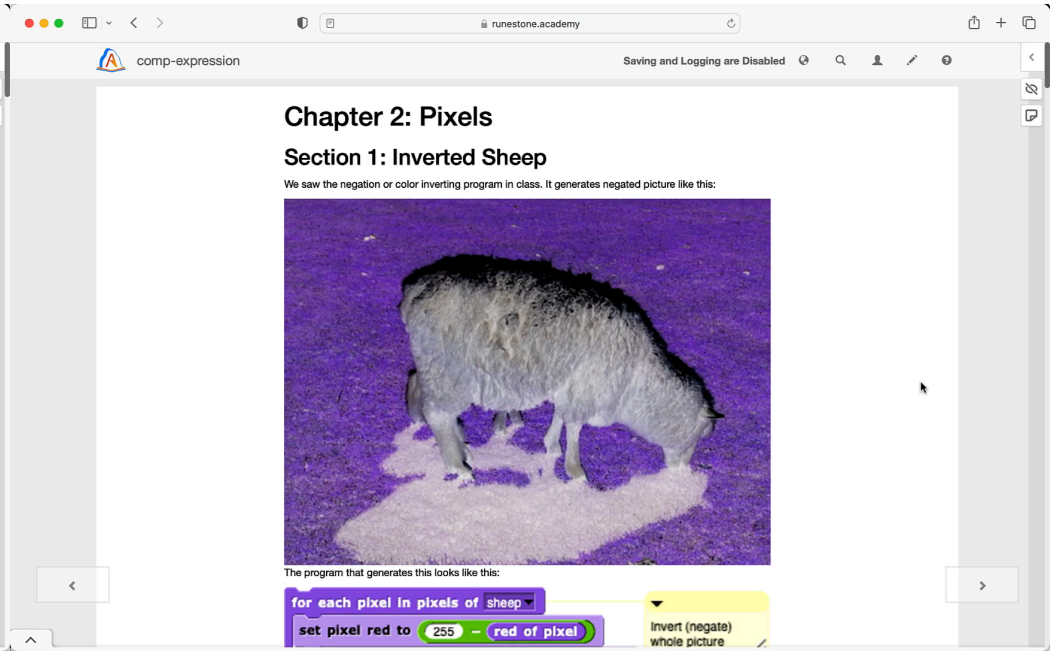




Learning Programming in Context, Step 3

At the end of each unit, students complete activities in an ebook where they see Snap code they've used before, then Python or Processing programs that do the same things, then answer questions about the textual code, which sometimes invites them to change the code.



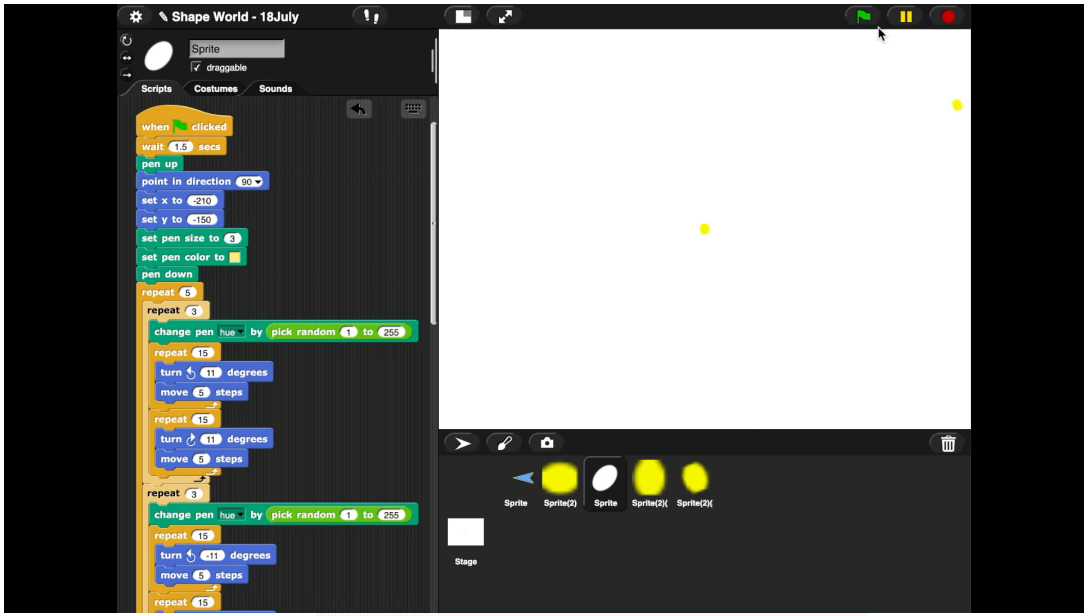


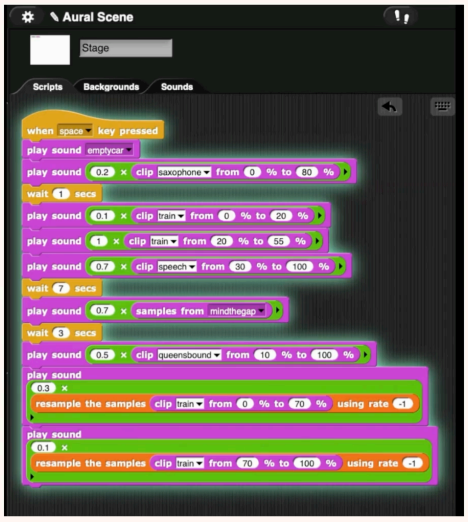
COMPFOR 111 *Computing's Impact on Justice: From Text to the Web*

- How computers understand text and language.
  - It's English-first and mostly English-only
  - How info bots are made. **Build a haiku generator. Build a Chatbot (in multiple languages)**
- How the Internet works: From Text to the Web
  - HTML. **Generating Web pages: From blocks, from Twine, from databases**.
- From the Web to Data
  - How search engines work. Security, privacy, and GDPR.
  - Analyzing what the Web records about you. **Analyzing server log files.**
  - How to Visualize data. **Making arguments with visualizations.**
- Artificial Intelligence and Machine Learning
  - Kinds of ML. **Build a gesture recognizer**
- Limitations of technology: Security, Blockchain and NFTs, and DALL-E

COMPFOR 121 “Computing for Creative Expression”

- Building art from shapes
- How image filters work
- How to manipulate digital sounds
- Interactivity
  - Building a drawing program that uses the microphone or camera
- Side-scroller video game
- Text:
  - Build a chatbot with graphical and audio representations
  - Build webpages with styles and embedded interaction





Open Research Questions

Research Questions We're Exploring

- Are arts and humanities students and faculty getting what they need from these classes?

- What is the process that students follow when programming in these classes, and how does that interact with the unusual structure (e.g., multiple languages, worked examples)?

- Is there transfer or increased cognitive load from teaspoon languages to Snap to textual languages? What is the cost and benefit of trilingual classes?

Research Questions We're NOT Exploring

- Do these students major in computer science or information?
- How difficult is it for these students to learn C++?
- Are they taking jobs in the computing industry?
- Are they learning how to write safe, secure, and robust code?

Summary

- The founders of computing proposed programming as a tool for learning for ***everyone***.
  - But somewhere along the line, it narrowed to become about getting jobs
- It's easier to argue that *no one* is getting computing than ***everyone*** is.
- Computing education for ***everyone*** is both broader and smaller than for software development.

Maybe need just a **Teaspoon**

- Who should be teaching computing education for everyone?
  - At Michigan, we are developing PCAS for the liberal arts and sciences

Programming can be a Tool for Learning Anything

WE NEED TO MAKE COMPUTING ACCESSIBLE TO EVERYONE

Some of the Collaborators on This Work

- Barbara Ericson, Gus Evrard, Kelly Campbell, Miranda Parker, Kathryn Cunningham, Amber Solomon, Bahare Naimipour, Tamara Nelson-Fromm, Emma Dodoo, Tammy Shreiner, Elise Lockwood, Adaline de Chenne.
- Undergraduate researchers: Aryan Bannerjee, Alexandra Rostkowycz, Erin Shi, Brandon Geng, Jessica Zhang, Ben Steinig, Fuchun Yang, Aoife Harte, Chloe Nguyen, Kashmira Reddy, Kristen Taurence, Angela Li, Derrick White, Jessie Houghton.

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

Thank you!