Computer Science Concepts Come Alive

Susan H. Rodger
Duke University
rodger@cs.duke.edu

Virginia Tech
Blacksburg, VA
April 1, 2011

Supported by NSF Grants DUE 0442513 and ESI 0624642, CRA, and IBM Faculty Awards
Outline

• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
About Me - Research Interests

• Computer Science Education
• Visualization and Interaction
  – Instructional Tools for Theoretical concepts
    • Automata theory and formal languages
  – Teaching Introductory Computer Science
• Algorithm Animation
Why find interactive and visual approaches to learning?
Intro - Why Use Interaction and Visualization?

- Learning Styles
  - Visual Learners
    - Learn through seeing
    - Learn best from visual displays
  - Auditory Learners
    - Learn through listening
    - Learn best through verbal lectures, discussions
  - Kinesthetic Learners
    - Learn through moving, doing and touching
    - Learn best through hands-on approach
How do you reach all three types?

• You must do all three!
  – Provide pictures, diagrams
  – Discuss what you are doing
  – Provide activities for trying it
• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
JAWAA
Java and Web-based Algorithm Animation

• Scripting Language for Animation
• Easily create, modify and move objects
• Runs over the web, no need to install
• More Advanced Students
  • Output JAWAA Command from Program
  • Animate Data Structures Easily
• SIGCSE 2003 and SIGCSE 1998
• www.cs.duke.edu/~rodger/tools/
• Students: Pierson, Patel, Finley, Akingbade, Jackson, Gibson, Gartland
Related Work

• Samba, Jsamba - Stasko (Georgia Tech)
• AnimalScript – Roessling (Darmstadt Univ of Tech, SIGCSE 2001)
• JHAVE – Naps (U. Wisc. Oshkosh, SIGCSE 2000)
• AlgoViz portal – lots of animations!
• Lots of animations and systems on the web!
<table>
<thead>
<tr>
<th>JAWAA Commands</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>circle cl 30 20 60 blue red</td>
<td></td>
</tr>
<tr>
<td>moveRelative cl 60 0</td>
<td>move right</td>
</tr>
<tr>
<td>moveRelative cl 0 50</td>
<td>move down</td>
</tr>
<tr>
<td>changeParam cl bkgrd blue</td>
<td></td>
</tr>
</tbody>
</table>
## JAWAA Primitives

<table>
<thead>
<tr>
<th>Shape</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td><img src="image1" alt="Circle" /></td>
</tr>
<tr>
<td>rectangle</td>
<td><img src="image2" alt="Rectangle" /></td>
</tr>
<tr>
<td>line</td>
<td><img src="image3" alt="Line" /></td>
</tr>
<tr>
<td>oval</td>
<td><img src="image4" alt="Oval" /></td>
</tr>
<tr>
<td>polygon</td>
<td><img src="image5" alt="Polygon" /></td>
</tr>
<tr>
<td>text</td>
<td><img src="image6" alt="Text" /></td>
</tr>
</tbody>
</table>
array people 25 25 4.2 Owen running Gail boating
   Robert toys Susan cakes vert red yellow black
changeParam people index on
changeParam people[1] bkgrd white
changeParam people[0].1 text bubblesort
moveRelative people[2] 30 0
changeParam people[2] swap people[0]
JAWAA Data Structures

• Stack

```plaintext
stack s1 200 200 4 Pop The Top Off black red
pop s1
pop s1
```

• Queue

```plaintext
queue q1 200 200 6 A 1 B 2 C 3 red blue
dequeue q1
dequeue q2
```

```
3C2B1A
3C2B1
3C2B
```
JAWAA Data Structures

- Linked List

- Trees
Use of JAWAA in CS 1/2

• Instructor
  – Use JAWAA Editor to make quick animations for lecture
  – Show web pages with JAWAA animations in lecture
  – Students replay animations later

• Student
  – Create animation of data structure in an existing program, add JAWAA commands as output
Instructor Animations for CS 2 Lecture

• How Pointers Work in Memory
• Recursion
• Shellsort
• Linked List - Insert at the Front
• Quadratic Collision Resolution
• Build Heap and Heapsort
Outline

• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Formal Languages and Automata Theory

• Traditionally taught
  – Pencil and paper exercises
  – No immediate feedback

  – More mathematical than most CS courses
  – Less hands-on than most CS courses
# Why Develop Tools for Automata?

<table>
<thead>
<tr>
<th>Textual</th>
<th>( {q_0, q_1, q_2}, {a, b}, \delta, q_0, {q_2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \delta = {(q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_0), (q_1, b, q_2), (q_2, a, q_1)} )</td>
</tr>
</tbody>
</table>

| Tabular       | \[\begin{array}{c|cc}
| q_0 & a & b \\
| q_0 & q_1 & q_0 \\
| q_1 & q_0 & q_2 \\
| q_2 & & \\
| \end{array}\] |

| Visual        | ![Diagram](image) |

| Interactive   | ![Diagram](image) |
Overview of JFLAP

• **Java Formal Languages and Automata Package**

• Instructional tool to learn concepts of Formal Languages and Automata Theory

• Topics:
  – Regular Languages
  – Context-Free Languages
  – Recursively Enumerable Languages
  – Lsystems

• With JFLAP your creations come to life!
Thanks to Students - Worked on JFLAP and Automata Theory Tools

- NPDA - 1990, C++, Dan Caugherty
- JFLAP - 1996-1999, Java version
  Eric Gramond, Ted Hung, Magda and Octavian Procopiuc
- Pâté, JeLLRap, Lsys
  Anna Bilska, Jason Salemme, Lenore Ramm, Alex Karweit, Robyn Geer
- JFLAP 4.0 – 2003, Thomas Finley, Ryan Cavalcante
- JFLAP 6.0 – 2005-2008 Stephen Reading, Bart Bressler, Jinghui Lim, Chris Morgan, Jason Lee
- JFLAP 7.0 - 2009 Henry Qin, Jonathan Su

Over 20 years!
JFLAP - Regular Languages

• Create
  – DFA and NFA
  – Moore and Mealy
  – regular grammar
  – regular expression

• Conversions
  – NFA to DFA to minimal DFA
  – NFA $\leftrightarrow$ regular expression
  – NFA $\leftrightarrow$ regular grammar
JFLAP – Regular languages (more)

- Simulate DFA and NFA
  - Step with Closure or Step by State
  - Fast Run
  - Multiple Run
- Combine two DFA
- Compare Equivalence
- Brute Force Parser
- Pumping Lemma
FA Edit & Simulation
Start up JFLAP

• When we start up JFLAP we have a choice of structures.
• The first of these is the Finite Automata!
FA Edit & Simulation
Start Editing!

• We start with an empty automaton editor window.
• We create some states ...
FA Edit & Simulation
Create Transitions

• We create some transitions ...
FA Edit & Simulation
Initial and Final State

• We set an initial and final state.
• Now we can simulate input on this automaton!
• When we say we want to simulate input on this automaton, a dialog asks us for the input.

![Image of a dialog box with the text "Input? aabc"]
When simulation starts, we have a configuration on the initial state with all input remaining to be processed.
This is a nondeterministic FA, and on this input we have multiple configurations after we “Step.”
FA Edit & Simulation
After Two Steps

• The previous configurations on $q_1$ and $q_2$ are rejected, and are shown in red.
• The remaining uncolored configurations paths are not rejected, and are still open.
• Yet another step.
One of the final configurations has been accepted!
One can then see a traceback to see the succession of configurations that led to the accepting configuration.
FA Multiple Run

- Select Multiple Run
- One can then enter many strings and receive acceptance info.

<table>
<thead>
<tr>
<th>Input</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Accept</td>
</tr>
<tr>
<td>aa</td>
<td>Accept</td>
</tr>
<tr>
<td>aab</td>
<td>Accept</td>
</tr>
<tr>
<td>aabb</td>
<td>Accept</td>
</tr>
<tr>
<td>acb</td>
<td>Reject</td>
</tr>
<tr>
<td>abcbb</td>
<td>Accept</td>
</tr>
<tr>
<td>abbcc</td>
<td>Accept</td>
</tr>
<tr>
<td>abcab</td>
<td>Reject</td>
</tr>
<tr>
<td>bc</td>
<td>Reject</td>
</tr>
</tbody>
</table>
JFLAP – Context-free Languages

• Create
  – Nondeterministic PDA
  – Context-free grammar
  – Pumping Lemma

• Transform
  – PDA $\rightarrow$ CFG
  – CFG $\rightarrow$ PDA (LL & SLR parser)
  – CFG $\rightarrow$ CNF
  – CFG $\rightarrow$ Parse table (LL and SLR)
  – CFG $\rightarrow$ Brute Force Parser
JFLAP – Recursively Enumerable Languages

• Create
  – Turing Machine (1-Tape)
  – Turing Machine (multi-tape)
  – Building Blocks
  – Unrestricted grammar

• Parsing
  – Unrestricted grammar with brute force parser
L-Systems

• L-Systems may be used to model biological systems and create fractals.
• Similar to Chomsky grammars, except all variables are replaced in each derivation step, not just one!
• Commonly, strings from successive derivations are interpreted as strings of render commands and are displayed graphically.
JFLAP Demos

• Finite state automaton for integers (both versions)
• NPDA for palindromes of even length
• Brute force parser compare two grammars
• Unrestricted grammar for $a^n b^n c^n$
• L-system for trees
DFA for all valid integers
DFA annotated and w/shortcut

- State q0 transitions to q1 on input 0.
- State q0 transitions to q2 on input λ.
- State q2 transitions to q3 on input [1-9], with a condition of "negative or not".
- State q3 transitions to itself on input [0-9], with a condition of "at least one non-zero".

Note: The diagram shows a transition from q1 to q0, which is not explicitly mentioned in the text.
Example: JFLAP during Lecture

• Ask students to write on paper an NPDA for palindromes of even length
• Build one of their solutions using JFLAP
  – Shows students how to use JFLAP
Example 1: JFLAP during Lecture (cont)

• Run input strings on the NPDA
  – Shows the nondeterminism
Example: JFLAP during Lecture

- Brute Force Parser
  - Give a grammar with a lambda-production and unit production
  - Run it in JFLAP, see how long it takes (LONG)
    - Is aabbab in L?
  - Transform the grammar to remove the lambda and unit-productions
  - Run new grammar in JFLAP, runs much faster!
Example 2 (cont)
Parse Tree Results

• First Grammar – 1863 nodes generated
• Second Grammar – 40 nodes generated
• Parse tree is the same.
Unrestricted Grammar - `anbncn`

<table>
<thead>
<tr>
<th>S</th>
<th>→</th>
<th>AX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>→</td>
<td>aAbc</td>
</tr>
<tr>
<td>A</td>
<td>→</td>
<td>aBbc</td>
</tr>
<tr>
<td>Bb</td>
<td>→</td>
<td>bB</td>
</tr>
<tr>
<td>Bc</td>
<td>→</td>
<td>D</td>
</tr>
<tr>
<td>Dc</td>
<td>→</td>
<td>cD</td>
</tr>
<tr>
<td>Db</td>
<td>→</td>
<td>bD</td>
</tr>
<tr>
<td>DX</td>
<td>→</td>
<td>EXc</td>
</tr>
<tr>
<td>BX</td>
<td>→</td>
<td>λ</td>
</tr>
<tr>
<td>cE</td>
<td>→</td>
<td>Ec</td>
</tr>
<tr>
<td>bE</td>
<td>→</td>
<td>Eb</td>
</tr>
<tr>
<td>aE</td>
<td>→</td>
<td>aB</td>
</tr>
</tbody>
</table>
Trace aabbcc
This L-System renders as a tree that grows larger with each successive derivation step.
L-Systems

- L-systems may also be stochastic.
- The $T \rightarrow Tg$ rule adds $g$ to the derivation, which draws a line segment.
- We add another rewriting rule for $T$, $T \rightarrow T$.
- With two rewriting rules for $T$, the rule chosen is random, leading to uneven growth!
L-Systems

The same stochastic L-system, rendered 3 different times all at the 9th derivation.
Using JFLAP during Lecture

• Use JFLAP to build examples of automata or grammars
• Use JFLAP to demo proofs
• Load a JFLAP example and students work in pairs to determine what it does, or fix it if it is not correct.
JFLAP’s use Outside of Class

• Homework problems
  – Turn in JFLAP files
  – OR turn in on paper, check answers in JFLAP

• Recreate examples from class

• Work additional problems
  – Receive immediate feedback
JFLAP’s Use Around the World

• JFLAP web page has over 250,000 hits since 1996

• Google Search
  – JFLAP appears on over 39,000 web pages
  – Note: search only public web pages

• JFLAP now used in several textbooks – JFLAP exercises

• JFLAP been downloaded in over 160 countries
Two-year JFLAP Study
2005-2007

Fourteen Faculty Adopter Participants

- small, large
- public, private
- includes minority institutions

• Duke
• UNC-Chapel Hill
• Emory
• Winston-Salem State University
• United States Naval Academy
• Rensselaer Polytechnic Institute
• UC Davis
• Virginia State University
• Norfolk State University
• University of Houston
• Fayetteville State University
• University of Richmond
• San Jose State University
• Rochester Institute of Technology
Conclusions From Study

• Results of Study showed
  – All the faculty used JFLAP in their courses, mostly for homework, some in lecture
  – Students had a high opinion of JFLAP
  – Majority of students felt access to JFLAP
    • Made learning course concepts easier
    • Made them feel more engaged
    • Made the course more enjoyable
  – Over half the students used JFLAP to study for exams
  – Over half the student thought time and effort using JFLAP helped them get a better grade.
Outline

• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Alice Programming Language

• Create interactive stories or games
• Learn programming in an easy way, drag-and-drop your code
• Problem solving with visual feedback
  – Objects are visual!
• Alice is free: www.alice.org
• Developed by Randy Pausch
**Alice Programming Language**

- Has libraries of 3D objects
- Keeps Track of objects you select
Objects Have Multiple Parts that are moveable
Alice Code is Easy to Learn

Select Code, Drag-and-Drop code in program
Computer Science Concepts come alive with Alice - Examples

• Objects are visible
• Variables
• inheritance
• List
• Array
Variables – Scores/Timers

Game: Eragon

4 tasks to win the game
Example - Inheritance

• Start with a chicken object
• Rename it to TalentedChicken
  – Change its color
  – Resize it larger
  – Add new methods (jump, fly, scurry)
  – Add events for this chicken
• Save this new class TalentedChicken that inherits from the Chicken class
Example - List

The Alice Team Summer 2008
Example – Arrays
Shuffle, then Selection Sort

Sort by height
Some projects from the Duke Alice course...
Game: Sarah Palin’s Seaplane Adventure

TODD’S SNOW MACHINE HAS BROKEN DOWN... AND IT'S UP TO YOU TO SAVE HIM!

SARAH PALIN’S SEAPLANE ADVENTURE

INSTRUCTIONS     PLAY     CREDITS

TAKING FLIGHT
Sarah Palin’s Seaplane Adventure (cont)
Game: Scarab Beetles take over
Game: Frogger – Get frog across road
Outline

• Introduction

• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice

• CS Concepts Come Alive with Props

• Challenges in Designing Educational Software

• Integrating Computer Science into K-12
Interaction in Class – Props
Passing “Parameters” in Class

• Pass by reference – throw frisbee

• Pass by value – throw copy of frisbee

• Pass by const reference – throw “protected” frisbee
Interaction in Class – Props
Linked List and Memory Heaps

ITiCSE 98 – Astrachan – “Concrete Teaching: Hooks and Props as Instructional Technology
Example: Be a Robot

• 4 People
  – Controller (head)
  – Sensors (eyes)
  – Manipulators (2 hands)

• Blindfolded except eyes

• Controller knows what to build

• Limited communication

SIGCSE 96, Rodger,Walker
Interaction in Class – Props

Edible Turing Machine

- TM for \( f(x)=2x \) where \( x \) is unary

- TM is not correct, can you fix it? Then eat it!

- States are blueberry muffins
Students building DFA with cookies and icing
Red-Black Tree (cookies)
Outline

• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Make your tool as interactive as possible – but not too tedious!

- User shouldn’t type everything
- Sometimes select
Allow user to proceed on if they got it

• Complete the rest for them

• Complete parts for them
Avoid Too Many Pop up windows

- OLD JFLAP LR PARSE TOOL
Add Checkpoint questions

• Pop up a quiz question to see if the user understands what he/she just did

• JHAVE tool does this
What can make the tool more useable?

- Annotations on states
- Multiple run window
  - Develop test data
  - Easier for grading
- General definitions
  - FA – recognize one or more symbols
  - NPDA – pop or push 0 or more symbols
- Batch processing
Naming your software

What is a “good” name for your tool?
JAWAA name is not unique

How popular is JAWAA?
JFLAP name is unique
• Introduction
• CS Concepts Come Alive with Software
  – CS 1/CS 2 with JAWAA
  – Automata Theory with JFLAP
  – Pre-CS 1 with Alice
• CS Concepts Come Alive with Props
• Challenges in Designing Educational Software
• Integrating Computer Science into K-12
Engaging Middle School Teachers and Students with Alice in a Diverse Set of Subjects

Supported by the National Science Foundation Collaborative Grant ESI-0624642, 0624654, 0624528, NSF Supplement DRL-0826661, two CRA distributed mentor awards, and two Faculty Awards from International Business Machines.
Problem – Few students major in CS

- Students come to college with their mind made up on their career! This choice is based on what they know.
- Students don’t know what computer science is when in middle and high school
- They like Alice, but not staying with computer science
Where could Alice help in decisions?

• Students in middle school are starting to form decisions on careers
• They have exposure to Teachers, Doctors, Astronauts, etc.
• They learn about Biology, Physics, Chemistry
  – BUT DON’T KNOW WHAT COMPUTER SCIENCE IS
  – THEIR EXPOSURE is SPREAD SHEETS, POWERPOINT, etc.
Bring on Alice Virtual Worlds!

• Alice is
  – Hands-on!
  – Interactive!
  – Visual!
  – Less Error prone
  – Exciting Results right away!

• Alice has the potential to excite kids about computer science in the same way that experiments excite kids about chemistry, physics and biology!
Adventures in Alice Programming

• Integrate Alice into high school and middle schools by training teachers

• Six sites:
  Durham, NC    Charleston, SC    Virginia Beach, VA
  Denver, CO    Oxford, MS       San Jose, CA

• Durham site focuses on Middle Schools in NC
  www.cs.duke.edu/csed/alice/aliceInSchools
Duke: Adventures in Alice site

- Summer 2008-2011
- 1-week and 3-week Teacher workshops
  - Over 130 teachers, mostly middle school, some high school
  - All disciplines
  - Taught them Alice, Developed Lesson Plans
- 1-week middle school camps
  - Taught Alice
  - Lots of time to build their own Alice worlds
How to Use Alice in Middle/High Schools

• Teachers
  – Examples in lecture
  – Make interactive quizzes
  – Make worlds on concepts for students to view

• Students
  – Projects (in place of a poster, a model)
  – To take or build quizzes
  – To view and answer questions about a world
  – Older students can do more with Alice.
Examples of integrating Alice into K-12
Let's look at your x-ray Kitty.

Sometimes Her mom takes her to the Doctor so that she can check out her knee. Sometimes that hurts a bit and sometimes it doesn’t.
Science Example
How a volcano is formed
Math Example: Teacher Lesson Plan on quadrant plane

- Click on lighthouse
- Enter x,y position
- Objects randomly move
Our Free Materials
Over 40 Tutorials

1. Getting started tutorials
   – 1-4 hours

2. Tutorials on CS topics
   – Methods, conditionals, lists, etc
   – Variables (timers/scores).

3. Animation tutorials
   – Lights, camera, scene change, billboards, invisible objects,
• Adventures in Alice Programming
www.cs.duke.edu/csed/alice/aliceInSchools

Adventures in Alice Programming
Durham, North Carolina Region

Adventures in Alice Programming
Durham, NC

This website describes a National Science Foundation project for integrating the programming language Alice into middle schools and high schools in the Durham, NC region. The target schools are the Durham public schools, Vance County, Person...
Conclusion

• We have shown several ways to visualize CS concepts and make them interactive
  – Software: JAWAA, JFLAP, Alice
  – Props: Food, frisbees, etc.
Questions?