Computer Science Concepts Come Alive

Susan H. Rodger
Duke University
February 20, 2007

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

About Me - Education

PhD, 1989
Computer Science
Assistant Prof.
1989-1994

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
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

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

What is JAWAA?

• 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
• Students: Pierson, Patel, Finley, Akingbade, Jackson, Gibson, Gartland
Related Work

- Samba, Jsamba - Stasko (Georgia Tech)
- AnimalScript – Roessling (Darmstadt Univ of Tech, SIGCSE 2001)

JAWAA Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle cl 30 20 60 blue red</td>
<td>circle</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>Primitive</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td><img src="circle.png" alt="Circle" /></td>
</tr>
<tr>
<td>rectangle</td>
<td><img src="rectangle.png" alt="Rectangle" /></td>
</tr>
<tr>
<td>line</td>
<td><img src="line.png" alt="Line" /></td>
</tr>
<tr>
<td>oval</td>
<td><img src="oval.png" alt="Oval" /></td>
</tr>
<tr>
<td>polygon</td>
<td><img src="polygon.png" alt="Polygon" /></td>
</tr>
<tr>
<td>text</td>
<td><img src="text.png" alt="Text" /></td>
</tr>
</tbody>
</table>

JAWAA Data Structures

Array

```
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] bigrd white
changeParam people[0].1 text bubblesort
moveRelative people[2] 30 0
changeParam people[2] swap people[0]
```
JAWAA Data Structures

• Stack
  
  stack s1 200 200 4 Pop The Top Off black red
  pop s1
  pop s1

• Queue
  
  queue q1 200 200 6 A B 2 C 3 red blue
  dequeue q1
  dequeue q2
  3C2B1A 3C2B1 3C2B

JAWAA Editor

• Easily create animations
• Graphically layout primitives
• Modify across time
• No knowledge of JAWAA
• Export to JAWAA file
• Start with JAWAA editor, finish with JAWAA output from program

Instructor Use of JAWAA in CS 1/2

• Use JAWAA Editor to make quick animations for lecture
  • Fast - 4-8 minutes each animation, Fall 2002 CS 2 Course
• Create quick animation of data structure in an existing program, add JAWAA commands as output
• Show web pages with JAWAA animations in lecture
• Students replay animations later

JAWAA Data Structures

• Linked List
  
  
  
• Trees
  
  
  

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
- Josephus Problem

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

JAWAA w/o Editor vs Editor

Nonmajors course

Spring 2001
No JAWAA Editor

Fall 2002
Using JAWAA Editor

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
What is a finite automaton?

- Models a problem – represent a set of strings
- Example:
  - All valid integers \{-3, 8, 0, 456, 13, \ldots\}
  - Starting place
    - Start state
  - Ending place
    - Final state

Why study finite automata?

- Application: Compiler
- Compiler identifies your syntax errors
- Can write a big DFA to identify all words in a Java program
  - integers, doubles, boolean
  - keywords, variable names
  - arithmetic operators, punctuation symbols

Why Develop Tools for Automata?

| Textual | \[ \{(q_0, q_0), (a, b), \delta, q_0, (q_2)\} \] 
| Tabular | | 
| Visual | | 
| Interactive | | 

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
Thanks to Students - Worked on JFLAP and Automata Theory Tools

• NPDA - 1990, C++, Dan Caugherty
• FLAP - 1991, C++, Mark LoSacco, Greg Badros
• 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 – 2005-2006 Stephen Reading, Bart Bressler, Jinghui Lim

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

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

JFLAP - Lsystems

- Create an L-system

- Render the L-system

Finite Automata Editing and Simulation

- The most basic feature of JFLAP has always been the creation of automata, and simulation of input on automata.
- Here we demonstrate the creation and simulation on a simple NFA.

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.

FA Edit & Simulation
Initial and Final State

• We set an initial and final state.
• Now we can simulate input on this automaton!

FA Edit & Simulation
Input to Simulate...

• When we say we want to simulate input on this automaton, a dialog asks us for the input.

FA Edit & Simulation
Start Simulation!

• When simulation starts, we have a configuration on the initial state with all input remaining to be processed.
After Four Steps

- One of the final configurations has been accepted!

Traceback

- One can then see a traceback to see the succession of configurations that led to the accepting configuration.

Multiple Run

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

FA Multiple Run

<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>ab</td>
<td>Reject</td>
</tr>
<tr>
<td>abbb</td>
<td>Reject</td>
</tr>
<tr>
<td>abbc</td>
<td>Accept</td>
</tr>
<tr>
<td>abcb</td>
<td>Reject</td>
</tr>
<tr>
<td>abc</td>
<td>Reject</td>
</tr>
<tr>
<td>bb</td>
<td>Reject</td>
</tr>
</tbody>
</table>

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.
L-Systems

• 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!

Students like L-systems
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 110,000 hits since 1996
• Google Search
  – JFLAP appears on over 20,000 web pages
  – Note: search only public web pages
• JFLAP been downloaded in over 160 countries

More on JFLAP

• JFLAP is free!

• www.jflap.org

• JFLAP book (Jones & Bartlett, 2006)
  – Use as supplement to a textbook
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

What Is Alice?

• A modern programming tool
  – 3-D graphics
  – 3-D models of objects
• Animation
  – Objects can be made to move around virtual world (a simulation or video game)
• Developed at Carnegie Mellon University
• At Duke – Use Alice in CompSci 4

The Power of Alice

• Automatically keeps track of 3-D objects
  – What objects are in the virtual world
  – Types of objects
  – Positions of objects in the world

Classes and Objects

• Classes
  – In Alice, classes are predefined as 3D models
• Objects
  – An object is an instance of a class
    • Class: Chicken
    • Objects: Chicken, Chicken2, Chicken3
Objects in Alice

- Objects already exist
- Objects have parts

Galleries of 3D Objects

- Sources of 3D objects
  - Local gallery – comes with Alice
  - Alice web gallery

Object Position

- Objects
  - Are positioned in 3D space
  - Have six degrees of freedom

Program an Object - Demo
Can Teach Computer Science Concepts with Alice

- Conditional and looping structures
- Methods, functions
- Events
- Inheritance
- Recursion
- Lists, Arrays

Methods

- Built-in methods
- Write class methods
- Write world methods

Inheritance

- Dragon
- FlyingDragon

Example – parameters and events

- People are trapped in a burning building
- Select which person will be rescued
Parameters

- Types and number of parameters must match with arguments.

Three Events

- The argument sent to parameters depends on which person is mouse clicked.

Example – Lists - WacAMole

- List of Moles
- Randomly moves one of them up and down
- Counter to keep track of score
- Event: when click on object, search through list to see if object is a mole

Events

- Default event
  
- Other events

- Note - we positioned fire truck so distance from floor X is X meters (to floor 3 is 3 meters)
Example – Binary guessing Game

- Three switches represent 3 digit binary number
- Random number generated to guess
- Click on switch to move it (up = 1, down = 0)

Example - Arrays

- Shuffle, then sort by height

Function to return object with the tallest height from an Array

Alice Software – is free!

- Runs on Mac and PC
- CompSci 4 web site
  www.cs.duke.edu/courses/fall06/cps004/rodger/
- Textbook available
  – Learning to Program with Alice by Dann, Cooper, and Pausch

- Download from web
  www.alice.org
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

Teaching Strategies
Activities Without a Computer

• Get creative in bringing hands-on activities into the classroom

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
Interaction in Class – Props

Memory Heap

Interaction with Class
Binary Tree and Recursion

• Build a binary tree
  – Pick a root
  – Root picks two children – point at them
  – Repeat until everyone is part of the tree
• Recursively calculate height of tree
  – Start at root
  – Ask children their height
  – Leaf notes know their height is 0


Students building DFA with cookies and icing

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
Example: Sorting Over 100 Words

- An envelope with over 100 words, each word on one slip of paper
- Sort the words
- Write down the algorithm
- Early assignment, before sorting is covered

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

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

Suggest Appropriate Uses of Tool

- Order HW questions so they are incremental in the usage of JFLAP
  1. Load a DFA. What is the language?
     Students only enter input strings.
  2. Load a DFA that is not correct. What is wrong? Fix it.
     Students only modifying a small part.
  3. Build a DFA for a specific language.
     Last, students build from scratch.

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

That’s all!

- JFLAP and ALICE are free
  - www.jflap.org
  - www.alice.org

- My Home page:
  - www.cs.duke.edu/~rodger

Questions?