Computer Science Concepts Come Alive

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

Haverford College
Philadelphia, PA
April 18, 2010

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
A bit about me, my background...

PhD, 1989
Computer Science

Assistant Prof.
1989-1994

Professor of the Practice
1994-present
About Me - Personally

Spouse       Mother
About Me - These keep me busy...
About Me - Hobby – Baking Shape cakes
How do you make those cakes?
Why find interactive and visual approaches to learning?
Students Ready to learn Automata Theory!
Things start well enough ...
But soon, instead of pictures, there are **WORDS**.
Big words! The type with more than one syllable!
VIOLENCE AMONG STUDENTS AS NERVES FRAY!
We only wanted to learn automata theory! Isn’t there a better way?
Try JFLAP ...
Students Learning Automata with JFLAP
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
• 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)
- Lots of animations and systems on the web!
### 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></td>
</tr>
<tr>
<td>moveRelative c1 60 0</td>
<td>move right</td>
</tr>
<tr>
<td>moveRelative c1 0 50</td>
<td>move down</td>
</tr>
<tr>
<td>changeParam c1 bkgrd blue</td>
<td></td>
</tr>
<tr>
<td>Primitives</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
</tr>
<tr>
<td>circle</td>
<td><img src="circle.png" alt="Image" /></td>
</tr>
<tr>
<td>rectangle</td>
<td><img src="rectangle.png" alt="Image" /></td>
</tr>
<tr>
<td>line</td>
<td><img src="line.png" alt="Image" /></td>
</tr>
<tr>
<td>oval</td>
<td><img src="oval.png" alt="Image" /></td>
</tr>
<tr>
<td>polygon</td>
<td><img src="polygon.png" alt="Image" /></td>
</tr>
<tr>
<td>text</td>
<td><img src="text.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**JAWAA Primitives**
array people 25 25 4.2 Owen running Gail boating Robert toys Susan cakes vert red yellow black
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 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
What is a finite automaton?

• Models a problem – represent a set of strings
• Example:
  – All valid integers {-3, 8, 0, 456, 13, ...}
  – Starting place
    • Start state
  – Ending place
    • Final state
DFA for all valid integers
DFA annotated and w/shortcut

- $q_0$ with an arrow labeled $\lambda$ pointing to $q_2$
- $q_0$ with an arrow labeled 0 pointing to $q_1$
- $q_2$ with an arrow labeled [1-9] pointing to $q_3$
- $q_3$ with an arrow labeled [0-9] pointing back to $q_2$

States:
- $q_0$: negative or not
- $q_1$
- $q_2$: at least one non-zero
- $q_3$
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?

<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>

<table>
<thead>
<tr>
<th>Tabular</th>
<th><img src="tabular.png" alt="" /></th>
</tr>
</thead>
</table>
|                  | \(\begin{array}{c|cc}
| \quad & a & b \\
| \hline
| \(q_0\) & \(q_1\) & \(q_0\) \\
| \(q_1\) &       & \(q_2\) \\
| \(q_2\) &       &       \\
| \end{array}\) |

<table>
<thead>
<tr>
<th>Visual</th>
<th><img src="visual.png" alt="" /></th>
</tr>
</thead>
</table>

| Interactive     | ![](interactive.png) |
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 20 years!
- 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
JFLAP - Regular Languages

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

• Conversions
  – NFA to DFA to minimal DFA
  – NFA ↔ regular expression
  – NFA ↔ 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.
FA Edit & Simulation
Create States

• We create some states ...
FA Edit & Simulation
Create Transitions

- We create some transitions ...
• 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.
• When simulation starts, we have a configuration on the initial state with all input remaining to be processed.
FA Edit & Simulation
After One Step

- 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.
FA Edit & Simulation
After Three Steps

• Yet another step.
FA Edit & Simulation
After Four Steps

• One of the final configurations has been accepted!
FA Edit & Simulation

Traceback

- 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>abc</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 - L-systems

• Create an L-system

• Generate the strings
  B
  [ T – B + + B ]
  [ T g – [ T – B + + B ] + + [ T – B + + B ] ]
  etc

• Render the L-system
JFLAP - 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!
L-Systems

The same stochastic L-system, rendered 3 different times all at the 9th derivation.
Students love 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 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
JFLAP in German

Um das JFLAP-Applet zu starten, [hier] klicken.

<table>
<thead>
<tr>
<th>Lernumgebung Automatentheorie mit JFLAP</th>
<th>Theorie – Themen</th>
<th>Hilfe – Themen</th>
<th>Übungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lernumgebung Automatentheorie mit JFLAP</td>
<td>Theorie – Index</td>
<td>Hilfe – Index</td>
<td>Info – ?</td>
</tr>
</tbody>
</table>

**Was ist JFLAP?**
JFLAP ist ein an der Universität Duke (USA) entwickeltes interaktives Lernprogramm, welches die Automatentheorie mit praktischen Anwendungen ergänzt. JFLAP ermöglicht es dem Anwender, bestehende Beispiele durchzuspielen, sowie eigene Automaten zu konstruieren.

**Was bietet diese Lernumgebung?**
Wir haben JFLAP um zusätzliche Komponenten erweitert, welche zusammen eine einheitliche Lernumgebung bilden. Diese Lernumgebung besteht aus folgenden Komponenten:

<table>
<thead>
<tr>
<th>Theorie</th>
<th>Hilfe</th>
<th>Übungen</th>
<th>JFLAP-Applet</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>eine Einführung in die Automatentheorie mit</td>
<td>eine Hilfedokumentation und Einführung in den Programm JFLAP mit</td>
<td>Übungen zu den einzelnen Automatentypen und zu den JFLAP-Funktionen (mit Lösungen)</td>
<td>das eigentliche Programm JFLAP zum graphischen Konstruieren und Testen von Automaten</td>
<td>(diese) Informationen für den Benutzer</td>
</tr>
<tr>
<td>- Themen (Schritt-für-Schritt)</td>
<td>- Themen (Schritt-für-Schritt)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Index (Stichwortverzeichnis)</td>
<td>- Index (Stichwortverzeichnis)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Einsteigern wird dringend empfohlen, zuerst die Help-Themen anzuschauen, bevor erste Schritte mit JFLAP unternommen werden!
JFLAP in Spanish

Ingeniería Técnica de Informática de Gestión / Sistemas

Asignatura Bases de lenguajes de programación
Curso 2002/03
Práctica opcional nº 1: Introducción a la herramienta JFLAP

Objetivo

El objetivo de la práctica es que el alumno se familiarice con la herramienta JFLAP, orientada a la práctica visual e interactiva de los conceptos sobre lenguajes formales y teoría de automatas. Mediante el uso de esta herramienta se practicarán operaciones relacionadas con gramáticas regulares, automatas finitos y obtención del árbol de derivación en gramáticas independientes del contexto.

Obligatoriedad
La práctica no es obligatoria.

Prerrequisitos

El alumno debe conocer los elementos relacionados con los niveles 2 y 3 de la jerarquía de Chomsky (lenguajes regulares, expresiones regulares, gramáticas regulares, automatas finitos, lenguajes y gramáticas independientes del contexto. Es recomendable un conocimiento elemental de manejo del sistema operativo Windows.

Descripción

A continuación se enuncian diferentes operaciones para experimentar con la herramienta JFLAP.
JFLAP in Chinese
JFLAP Study

• Study of JFLAP’s effectiveness in learning
  – Two year study
  – Fourteen Faculty Adopters
  – Two 2-day faculty Adopter Workshops – June 2005, June 2006
  – Collect data 2005-06 and 2006-07 Academic years
  – Pretest/Posttest
  – Interviews
  – Team of three evaluators
    • Eric Weibe – Education
    • Rocky Ross – Computer Science Theory
    • Joe Bergin – Computer Science Tools
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

• No Conclusive Results from Pretests/Postests

• 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.
JFLAP is free

www.jflap.org

JFLAP tutorial
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
Object Position

- Objects
  - Are positioned in 3D space
  - Have six degrees of freedom
Alice Code is Easy to Learn

Select Code, Drag-and-Drop code in program
Play Alice Animation

- Chicken rises, cow turns head and talks
Computer Science Concepts come alive with Alice - Examples

• Objects are visible
• Variables
• inheritance
• Lists, Arrays
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 - Arrays

• Shuffle, then sort by height
Example with ArrayVisualization

Swapping two elements in an array

- Swap the objects at positions 0 (fanDancer) and 3 (duckPrince)
- Add in an ObjectVisualization, this is like a variable for an object. (same folder where ArrayVisualization is)
Swapping objects at 0 and 3 (cont)

• Only one element at a time can be in a slot in the array. To swap two elements, you have to move one of them out temporarily.
• Move object at index 0 to objectVisualization (this frees up slot 0)
Swapping objects at 0 and 3 (cont)

• Now you can move the item in slot 3 over to slot 0 (note the duckPrince moved over)
• Now slot 3 is empty
Swapping objects at 0 and 3 (cont)

• Now move the object that was originally in slot 0 and was saved temporarily in the ObjectVisualization, over to slot 3
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

TAKE FLIGHT
Sarah Palin’s Seaplane Adventure (cont)

DISTANCE TO TODD 15.0

CONGRATULATIONS, YOU SAVED TODD!
Game: Scarab Beetles take over
Game: Candyland

Select girl and boy to play

Click on red and green buttons to move them.
Game: Frogger – Get frog across road
Game: Dating Game

Questions:
1 2 3 4

Choose Contestant!
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
Interaction in Class – Props
Memory Heap
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
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 usable?

- 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
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
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.
Success - Alice Excites 4th-6th Grade Girls

- Duke Femmes Event, April 07
- 60 girls – 4 groups of 15
- Taught them Alice for an hour
- Handout to take home
- Event again in 2008 and 2009
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 and 2009
  – 1-week and 3-week Teacher workshops
    • Over 130 teachers, mostly middle school, some high school
    • Only a few had ever programmed before
    • Taught them Alice, Developed Lesson Plans
  – 1-week middle school camps
    • Taught Alice
    • Lots of time to build their own Alice worlds
Targetting all subject teachers

• Subject teachers using Alice
  – Language Arts
  – Mathematics
  – Science
  – History
  – Foreign Language
  – Music, Art
  – Media, Technology
  – Business

• Mostly Middle school, some Elementary, and some high school subject teachers (physics, chemistry, etc)
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
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

Over thousands of years, the volcano builds up...
Math Example:
Teacher Lesson Plan on quadrant plane

- Click on lighthouse
- Enter x,y position
- Objects randomly move
Making Alice worlds more real

- Use heads of familiar people on 3D objects
Other Ideas for Projects

• Story from Ancient Egypt
• Spanish Quiz in which you see a word and have to click on the object the word represents
• Animate a scene from a book you have read or a poem you have written
• Create a world about school safety
• Memory game – remember a random color sequence
• Math Quiz – Answer the questions

Alice worlds for these and more are on our website.
Our Free Materials
Introductory Tutorials

1. Simple, Short (15 min) tutorials to try Alice
   – Add an object, use built-in methods
2. One hour starting tutorials
   – Writing methods, simple events, camera
3. Four part tutorials if more time/more detail
   – More detailed on placement of objects, writing methods, events, camera control
   – Animating a skateboarder
   – Adding sound and images
Many short tutorials on CS Topics

- CS Topics
  - Programming – sequential and “at the same time”
  - Methods (teaching characters how to walk)
  - Events (buttons and birds)
  - Looping
  - Conditionals (making a choice)
  - Functions (how tall are you)
  - Lists (objects moving in unison)
  - Variables (timers/scores)
Other “Fun” Topics Blended in

- Storyboards
- Changing camera views
- Scene changes and lighting
- Fading in/out
- Making Billboards
- Making objects invisible and visible
- Sounds
- Glueing objects to others
Web site

- Adventures in Alice Programming
  www.cs.duke.edu/csed/alice/aliceInSchools
Conclusion

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