Comparing Textual and Block Interfaces in a Novice Programming Environment

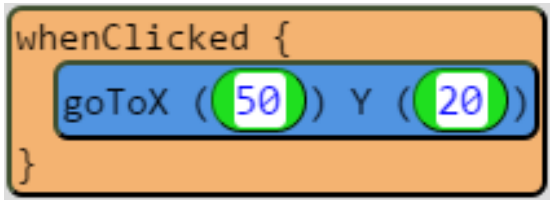
Thomas Price
Tiffany Barnes

North Carolina State University

ICER 2015

Block-Based Programming Environments

"Environments that allow users to construct and execute computer programs by composing atomic blocks of code together to produce program structure."



Motivation

- Block-based environments are becoming popular for teaching novices
- These environments are successful
  - Known for being *accessible* and *engaging*
  - E.g. Scratch, Alice, Snap, MIT App Inventor, LEGO Mindstorms
- They include (at least) two important features:
  - They use visual, drag-and-drop block programming
  - They are media-rich, connect students with interests
- Which features are important for this success?
  - Specifically, does the block interface make a difference?

Example - *Scratch*

- Designed to be more tinkerable, meaningful and social than past environments (Resnick et al. 2009)
- Graphical output centers around programmable sprites
- Used to make games, animations, music videos
- 25th most popular programming language (TIOBE Index, Jun. 2015)

Example - *Scratch*

Evaluations:

- A semester-long course with Scratch significantly improved 9th graders' test scores on most CS concepts (Meerbaum-Salant et al. 2013)
  - Students struggled with initialization, variables and concurrency
- Scratch was a popular choice in an urban after-school center (Maloney et al. 2008)
  - Students used Scratch voluntarily, without instruction
  - 50% used loops and user interaction
  - 25% used conditionals and concurrency
- Video game making with Scratch can "provide a rich context for programming" (Peppler & Kafai 2007)

Comparing Interfaces

- Students learning Scratch and Logo had similar, but not identical outcomes (Lewis 2010)
  - Logo users reported higher confidence afterwards
  - Scratch users did better on conditional test questions
  - Both groups gave similar difficulty ratings
- Comparing Modkit and Java users learning to program Arduino, Modkit users completed more activities (Booth & Stumpf 2013)
  - Modkit users reported lower perceived workload and more positive user experience

Comparing Interfaces

- From an HCI perspective, block and textual languages support different programming tasks better (McKay & Kölling 2013)
  - Block languages had differing strengths
- Students can transfer skills learned in a block language to a textual language (Wagner et al. 2013; Dann et al. 2012)
  - Facilitated by matching APIs
  - Students bridging from Alice to Java performed an average of 1 letter grade higher on a Java test than students learning only Java

Research Questions

When compared to a textual interface,

how will a block interface:

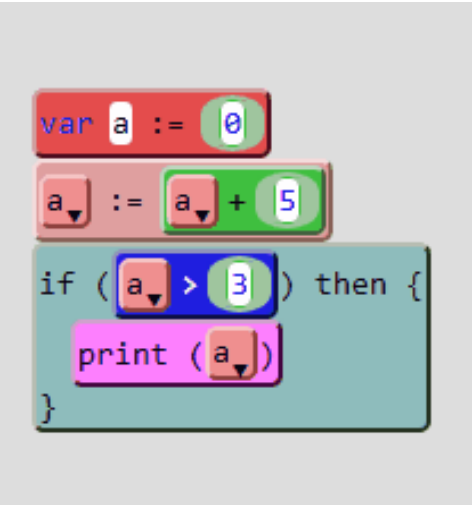
1. Affect students' attitudes towards computing?
2. Affect their perceived difficulty of programming?
3. Affect their performance on a programming activity?

Procedure Overview

- Modified an environment to directly compare block and textual interfaces
- Adapted an "Hour of Code" activity
- Collected data from two groups of students as they completed the activity, one with each interface
  - Pre-survey
  - Programming activity lasted 45 minutes
  - Post-survey
- Data collected and analyzed

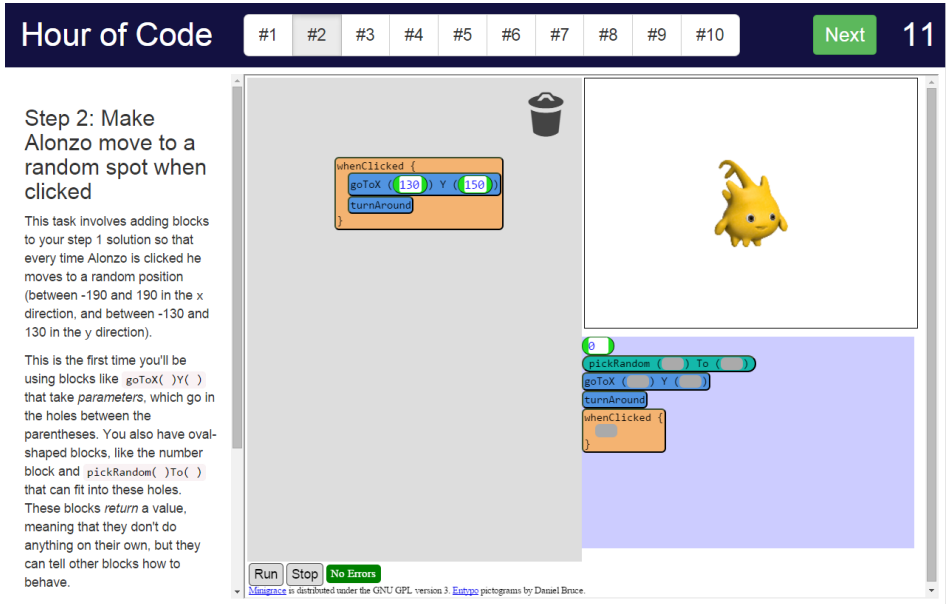
Tiled Grace

- Supports both "tiled" (block) and textual interfaces (Homer & Noble 2014)
  - Participants were locked into one interface or the other
- Original language designed for novice programmers
- Block interface very similar to Scratch



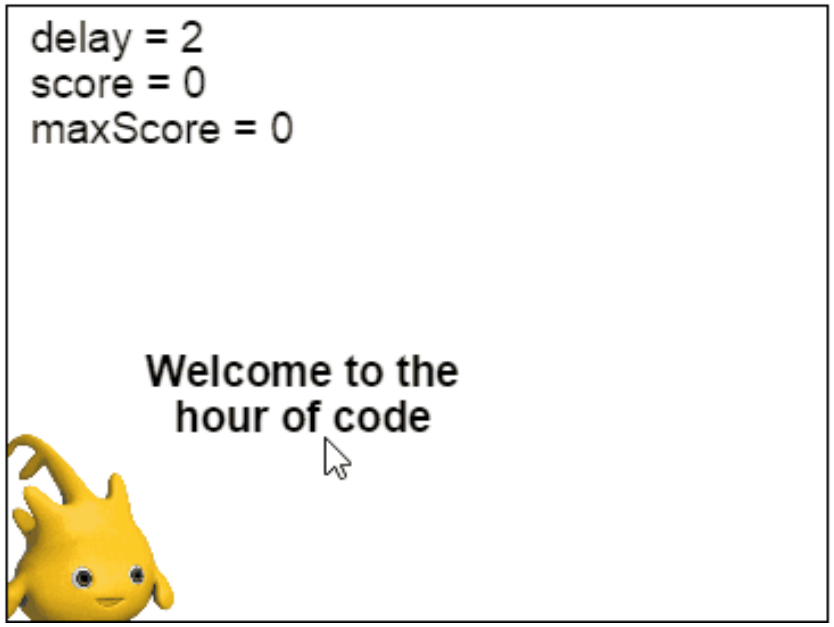
The Environment

- Created two versions of Tiled Grace, locked into one interface
- Embedded in a tutorial environment



The Activity

```
1  dialect "hoc"
2
                        Goals:  1 2 3 4 5 6 7 8 9
   var delay := 2       Colors:
   var score := 0
   var maxScore := 0
   whenClicked {
       say ""
       delay := delay - 0.1
       score := score + 10
       if (maxScore < score) then {
           maxScore := score
           clear
       }
       forever {
           goToX( pickRandom (-190) To (190) )
                Y ( pickRandom (-130) To (130) )
           turnAround
           if (score > 0) then {
               score := score - 1
           }
           wait (delay)
       }
   }
24
25 goToX (-190) Y (130)
   say ("Welcome to the hour of code")
   penDown
```



Participants

- Two classes from SPARCS, a middle school CS outreach program (Cateté et al. 2014)
  - No students from previous years
- 6th grade assigned to block interface
  - N=17: 12 male, 5 female
- 7th grade assigned to textual interface
  - N=14: 11 male, 3 female
- Condition assignments were random and groups were found to be similar populations
  - Block group had higher interest ratings on pre-survey

Data Collected

Pre-survey

- 4 Likert items to assess **Efficacy** w.r.t. CS
- 3 Likert items to assess **Interest** in computing
- 3 code evaluation (**Knowledge**) questions

Logs

- Complete code snapshots were saved at regular intervals and at each run

Post-survey

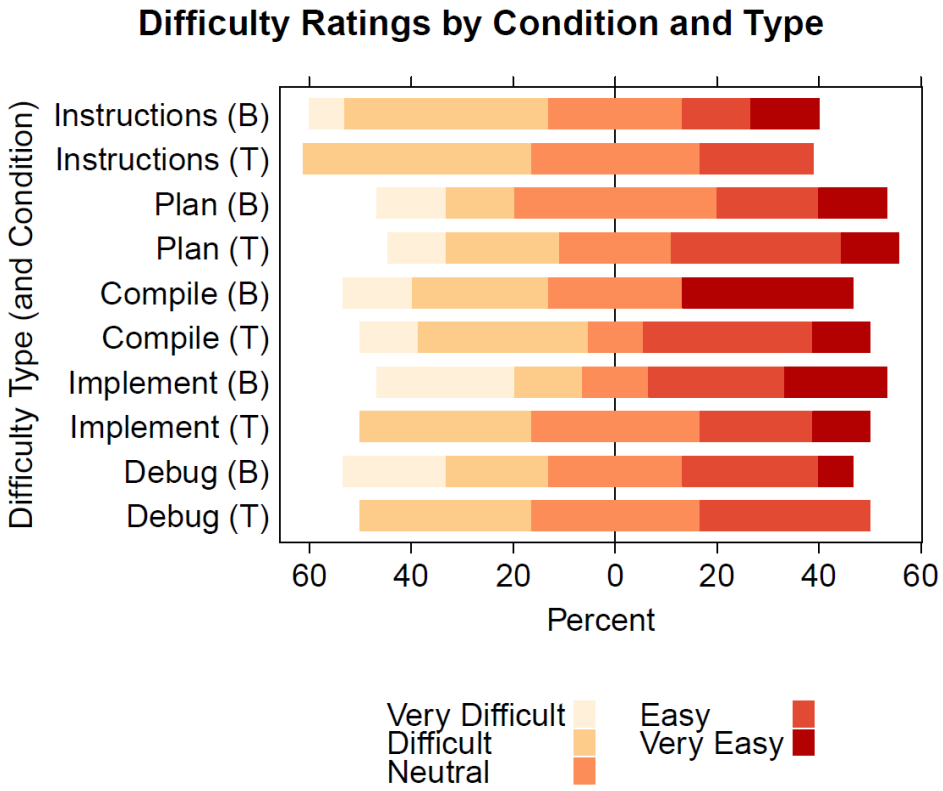
- Repeated pre-survey questions
- Users rated the difficulty of the activity

Survey - Attitudes

- **Efficacy** ratings significantly improved after the activity
  - The individual Likert items had contradictory results
- This effect was not significantly different between conditions
- There was no significant change in **Interest** ratings or **Knowledge** scores

Survey - Difficulty

- Students reported very similar difficulty across conditions, for each category



Survey - Dropout

- Some students in both groups dropped out of the post-survey
  - These students were omitted in pre/post survey comparisons
- These students may have been among the least engaged, possibly covering up a difference between conditions

	Pre-survey	Difficulty	Efficacy/Interest	All
Block	17	15	13	10
Text	14	9	9	7

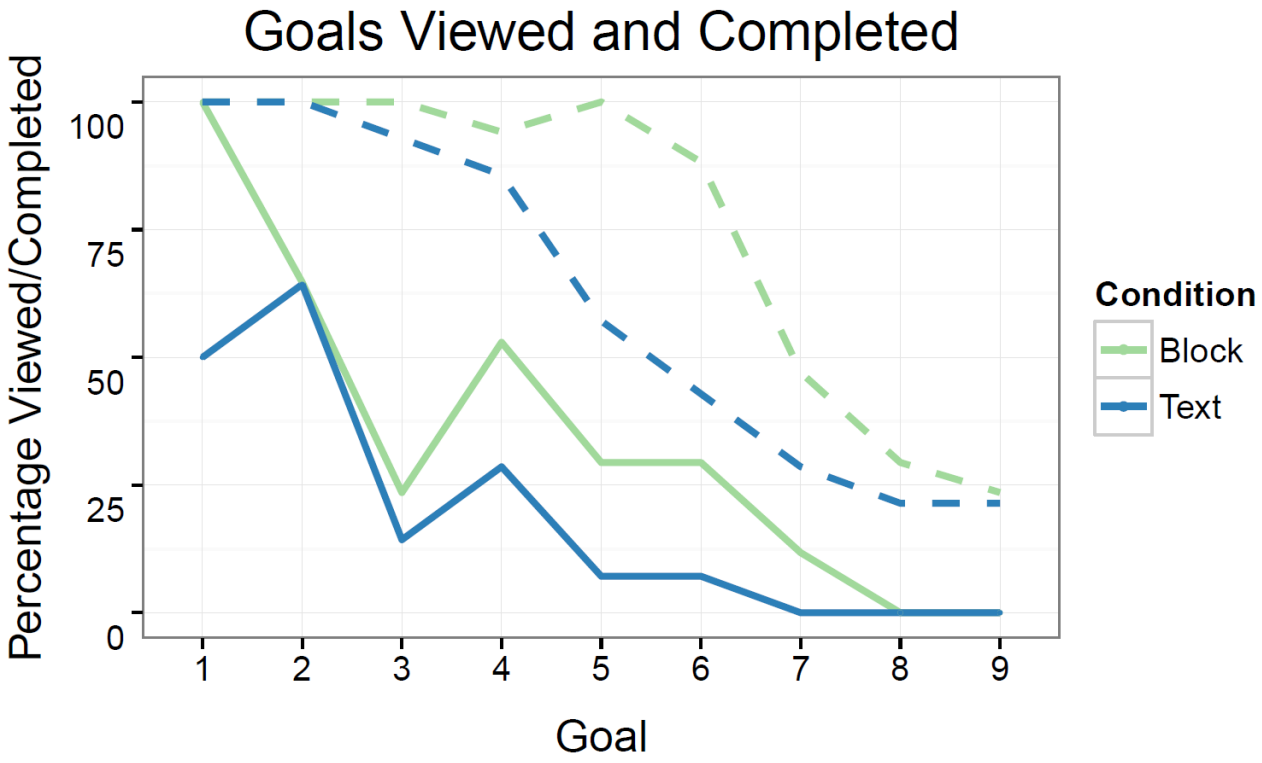
Performance - On-Task Behavior

- Total, Idle and on-task time were calculated
  - Idle means the student made no action for 60s
- Idle time was significantly less in the Block condition, and on-task time was significantly greater

Value	Block	Text	p	d
Total	2273.9 (596.4)	2208.0 (427.1)	0.851	–
Idle	407.2 (238.9)	793.5 (368.3)	0.002	1.27
Active	1866.8 (617.4)	1414.5 (463.1)	0.014	0.82

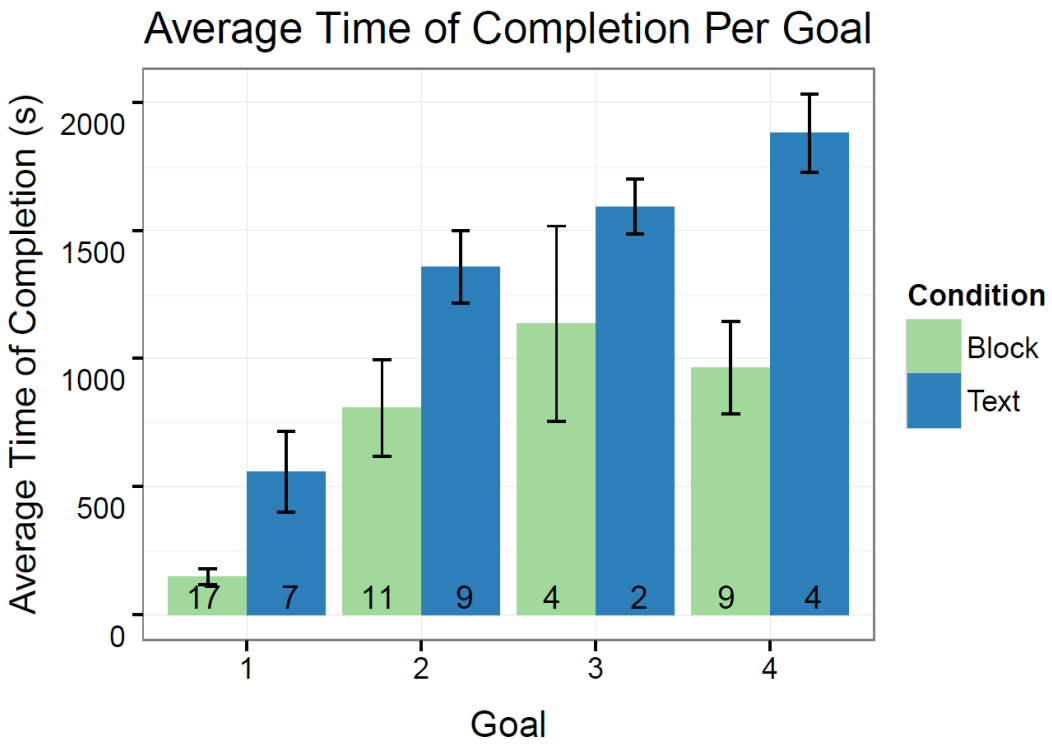
Performance - Achieving Goals

- A larger or equal percent of the Block condition completed each goal



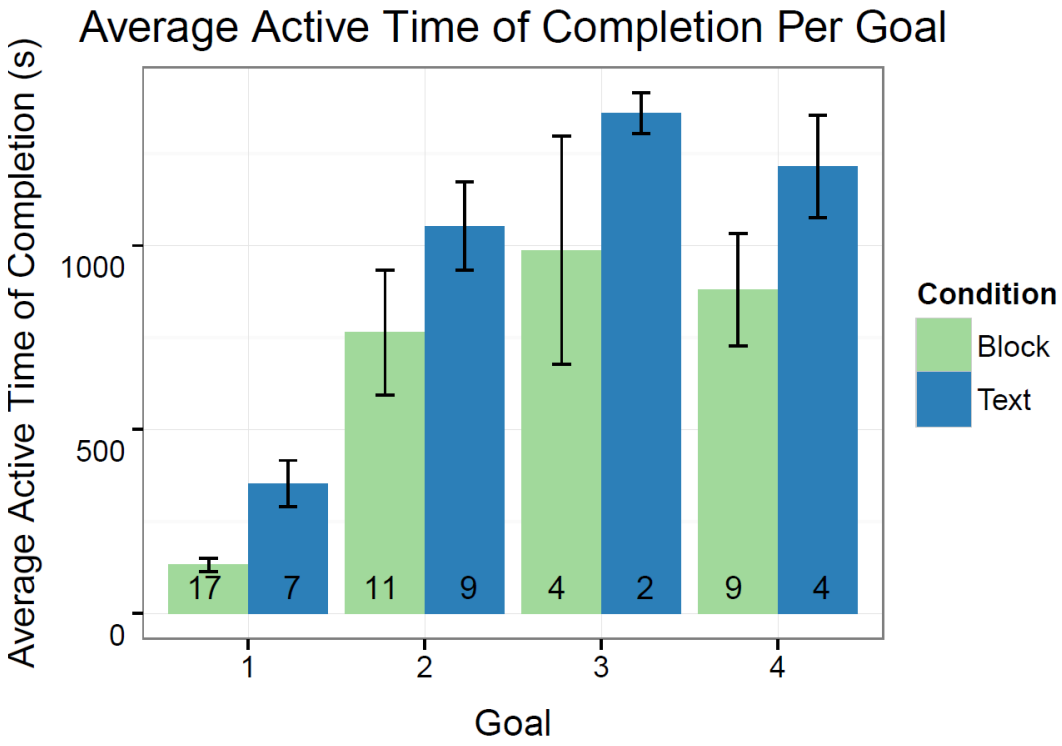
Performance - Achieving Goals

- Students in the Block condition completed Goals 1, 2 and 4 after significantly less time had passed



Performance - Achieving Goals

- Students in the Block condition completed Goals 1, 2 and 4 after significantly less time had passed



Discussion - RQ1

How did the interface affect users' attitudes towards computing?

- The activity did significantly improve students' perceived efficacy
  - This was not significantly different between groups
- No other attitudinal effects were observed in either condition
- We can offer no evidence to support the claim that the interface affects attitudes.
  - It is possible there was insufficient sample size after dropout to see an effect

Discussion - RQ2

How did the interface affect users' perceived difficulty of the activity?

- There were almost identical distributions of perceived difficulty
- This agrees with previous results (Lewis 2010)
- Perhaps this is because students proceed until they encounter something difficult
  - The block interface allows students to surpass the difficulties of syntax, and grapple with logic
  - This would suggest the categories of difficulty should still see different ratings

Discussion - RQ3

How did the interface affect users' performance on the activity?

- By almost any measure, the Block interface improved performance
  - Students spent more of their time on task
  - They completed more goals in less time

Limitations

- Results about a single Block-based programming environment may not generalize (McKay & Kölling 2013)
- The activity was designed for a block interface, which may have biased results
- The survey was not validated and had high dropout on the post-survey
- Populations were not identical
  - 6th vs 7th grade
  - Block group had higher initial interest scores

Future Work

- At what level of experience do the benefits of a block interface deteriorate?
- What mechanisms lead to increased, faster goal completion?
  - Is this simply a function of increased time on task?
  - Could it be an effect of biased program structure?
- With an improved survey and increased sample size, will we see an effect of the interface on student attitudes?
- (Stick around for some possible answers!)

Conclusions

- A block interface improves novice performance on an open-ended programming task
  - Faster completion of goals, and more goals completed
  - Less idle time and more time spent on-task
- We have no evidence to support a claim that the interface significantly affects novices attitudes towards computing or their perceived difficulty on the programming task

Questions?

References

M. Resnick, J. Maloney, H. Andres, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60-67, 2009.

O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. Learning computer science concepts with scratch. *Computer Science Education*, 23 (3):239-264, 2013.

J. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk. Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, 40(1):367-371, 2008.

K. Peppler, and Y. Kafai. What Videogame Making Can Teach Us about Literacy and Learning: Alternative Pathways into Participatory Culture. Situated Play. In *Proceedings of the Digital Games Research Association (DiGRA) Conference*, 2007.

B. Moskal, D. Lurie, and S. Cooper. Evaluating the effectiveness of a new instructional approach. *ACM SIGCSE Bulletin*, 36(1):75-79, 2004.

S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. *ACM SIGCSE Bulletin*, 2003.

C. Kelleher, R. Pausch, and S. Kiesler. Storytelling alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI conference on Human Computer Interaction*, 2007.

C. Lewis. How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 346-350, 2010.

T. Booth and S. Stumpf. End-user experiences of visual and textual programming environments for Arduino. *End-User Development*, pages 25-39, 2013.

F. McKay and M. K olling. Predictive modelling for HCI problems in novice program editors. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*, pages 35-41, 2013.

A. Wagner, J. Gray, J. Corley, and D. Wolber. Using app inventor in a K-12 summer camp. In *Proceeding of the 44th ACM technical symposium on Computer Science Education*, 2013.

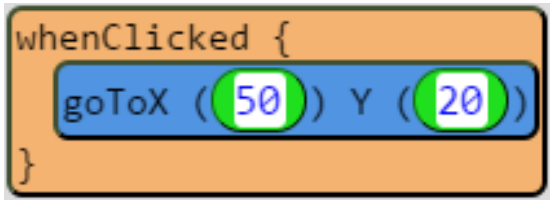
W. Dann, D. Cosgrove, and D. Slater. Mediated transfer: Alice 3 to java. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 141-146, 2012.

V. Catete, K. Wassell, and T. Barnes. Use and development of entertainment technologies in after school STEM program. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 163-168, 2014.

(Bonus Slides…)

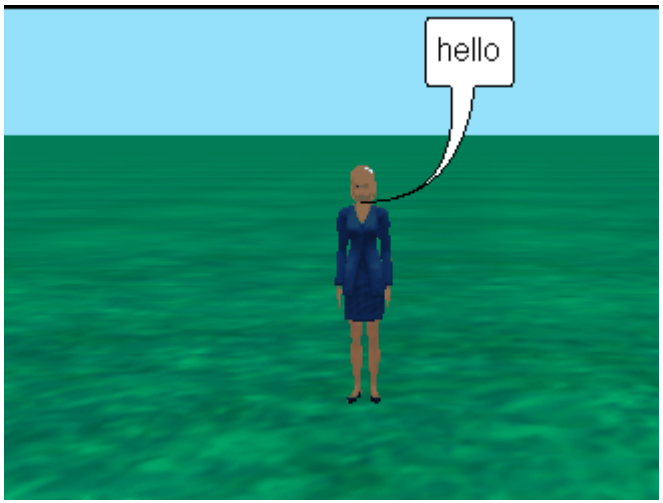
Block-Based Programming Environments

- Blocks can represent control structures, function calls, operators, expressions, etc.
- Blocks have slots which can have other nested blocks
- Generally, blocks are dragged-and-dropped
- For our purposes, they are procedural



Example - *Alice*

- One of the first drag-and-drop novice environments
  - Combines this with a menu interface
- Allows users to program objects in a 3D scene
- Object-oriented and event-driven paradigms
  - Users manipulate objects' properties and call their methods



Example - *Alice*

Evaluation:

- An Alice-based CS0 course, taken before or with a traditional CS1 course, significantly improves students' grades (Moskal et al. 2004)
  - These trends are more apparent in "high-risk" students, with no CS and less math experience
  - Also improves retention and attitudes
- Alice contextualizes Object-oriented programming and teaches good program design (Cooper et al. 2003)

Example - *Alice*

Evaluation:

- Modifying Alice with an increased emphasis on storytelling (e.g. easier animations) increased its appeal to girls (Kelleher et al. 2007)
  - Participants indicated increased interest in using Storytelling Alice and taking it home to use later

Novelty and Importance

- First study to *directly* compare block and textual programming interfaces
  - All other aspects of the environment are controlled
- Many resources go into the design of novice programming environments
  - It is important that we focus on the aspects that help students learn

Contributions

1. Strong evidence that a block interface has a positive impact on novice programmers
2. Support for previous findings that a block interface does not change perceived difficulty
3. A clear direction for future research into the mechanism by which block interfaces improve performance