Introducing Computer Science using Block-Based Programming

Z.D.R. Copic

University of Minnesota, Morris

April 30, 2016

Outline

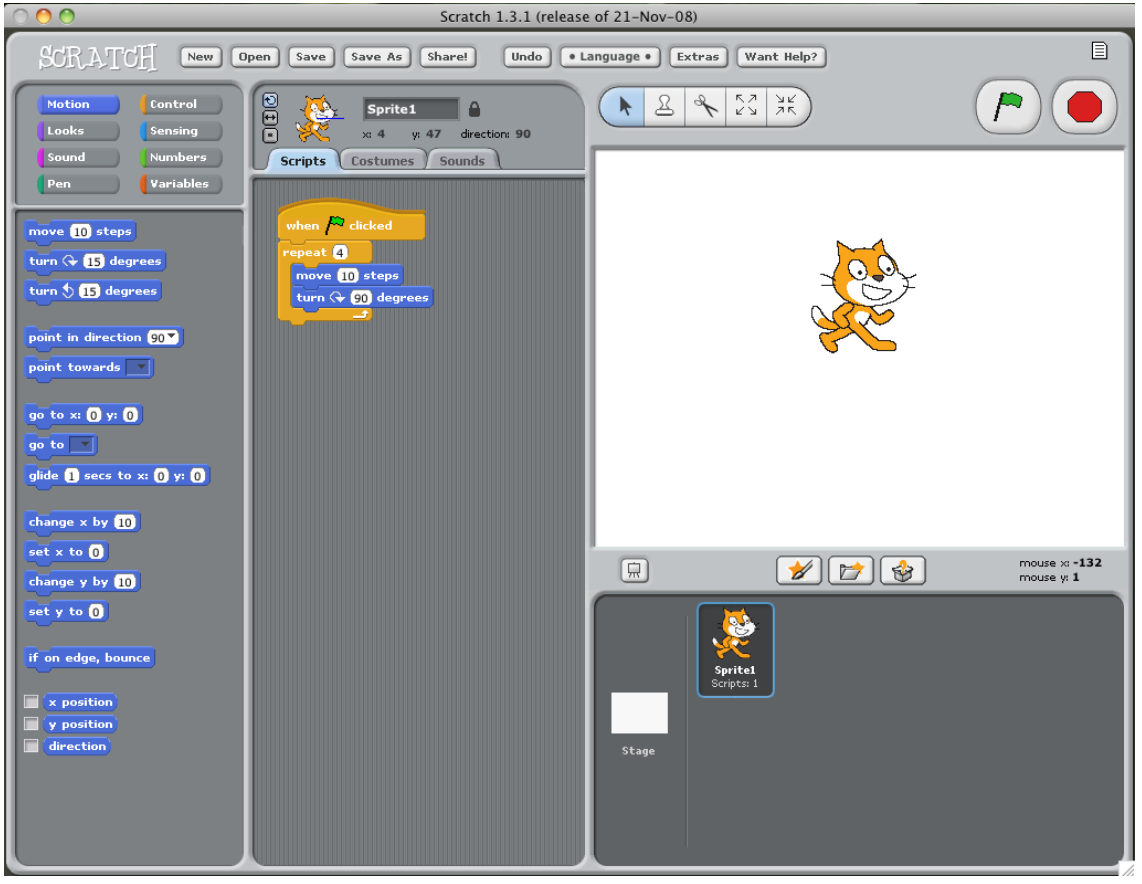
Background Information

- Block-based programming languages
  - Designed to be a simple way to code for kids
  - Most of the code is abstracted into blocks
  - Blocks are stacked upon one another to form a script
- Scratch was developed by Mitchel Resnik, and it was released in 2003.
  - Used by millions

Scratching the Surface

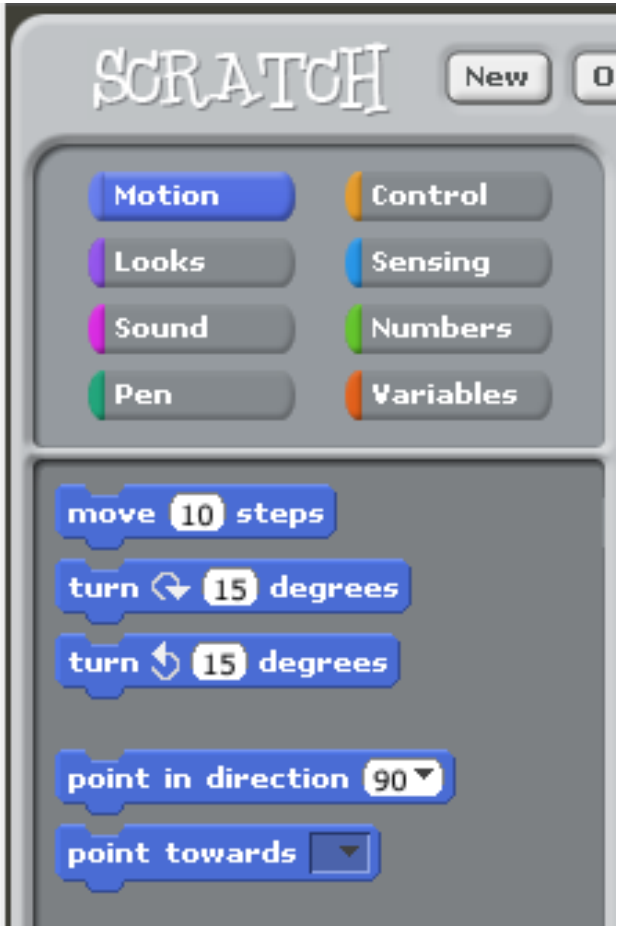
- It is read top-down, left to right

- Graphical User Interface (GUI)



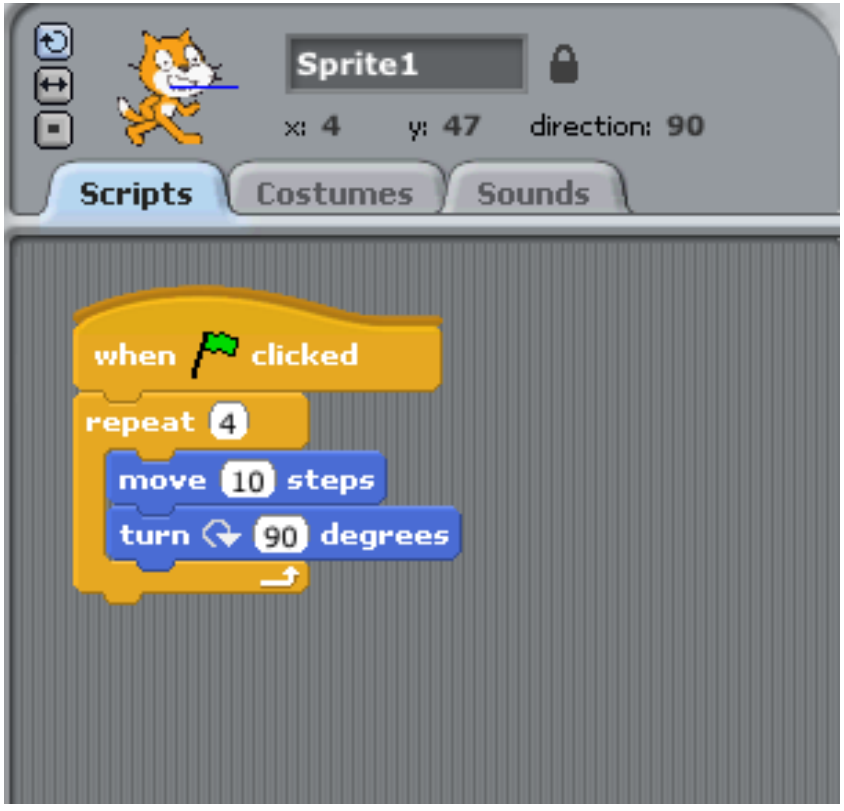
Scratch GUI: The First Column

- The first column has some pre-made code to play around with
- Several categories to choose from
  - Motion
  - Controls
  - Looks
- Click and drag
- Colors of the blocks represent certain categories
- They can be modified in the white spaces



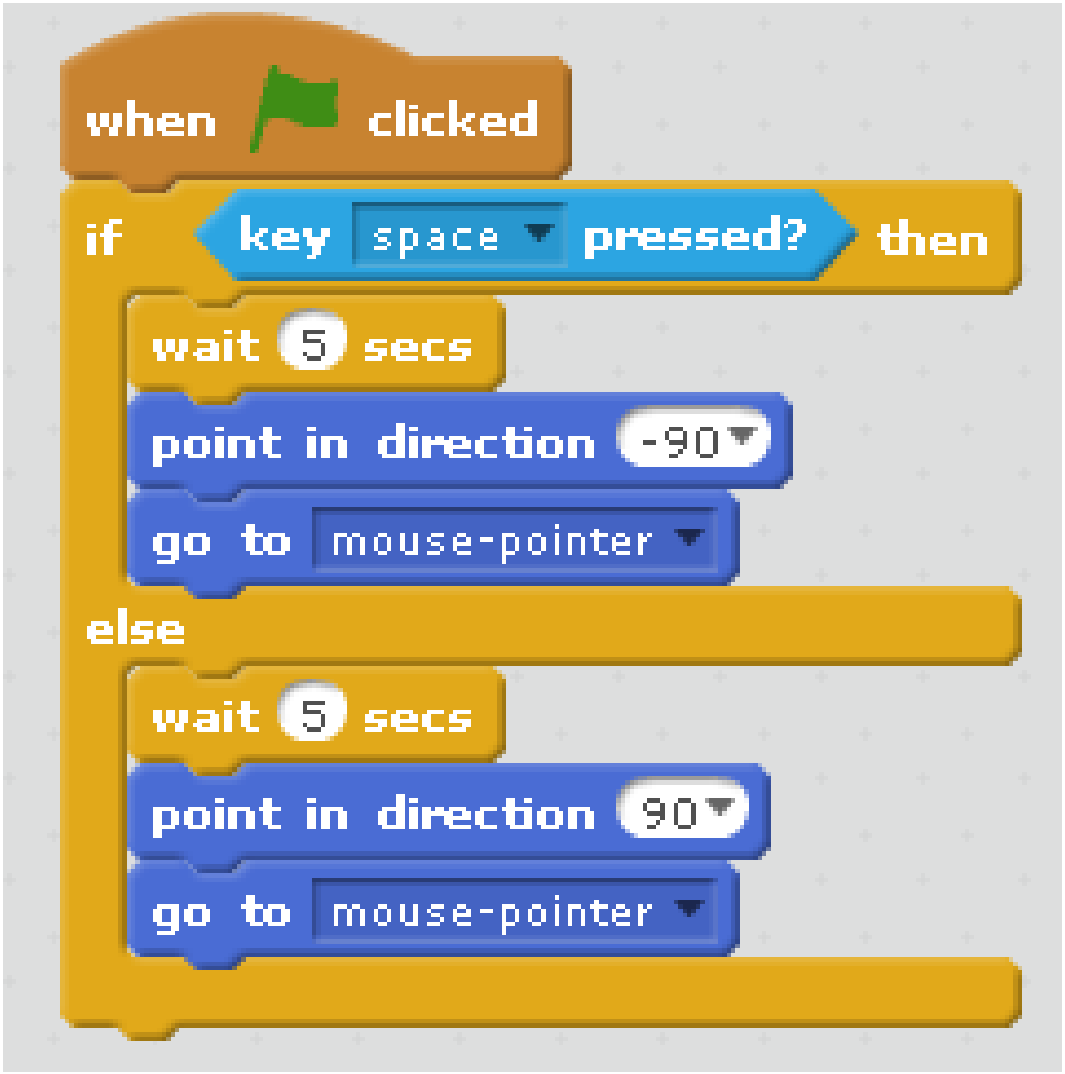
Scratch GUI: The Second Column

- This is where you lay out the code
- The code can also be run here by double-clicking anywhere on the block coding.
- This is also where you can modify the sprite
- It tells the current position of the sprite in the third column



Scratch GUI: The Third Column

- This is where the magic happens
- The green flag in the upper right will execute the code
- You can manually drag the sprite anywhere in the white box
- You can also edit, save, or get help by clicking on the buttons in the middle of the column
- the bottom dark grey box shows the sprite and the number of scripts it has associated with it


when clicked
if key space pressed? then
wait 5 secs
point in direction -90
go to mouse-pointer
else
wait 5 secs
point in direction 90
go to mouse-pointer

Overview

- Middle School Studies
- Weekly
- Summer
- College Level

Middle School Study 1

- Group of middle school aged children were taught the basics of Scratch
- They later had these same students take a course in high school to compare to other students of the same age who had not taken a course previously in computer science
- The high school course used Java

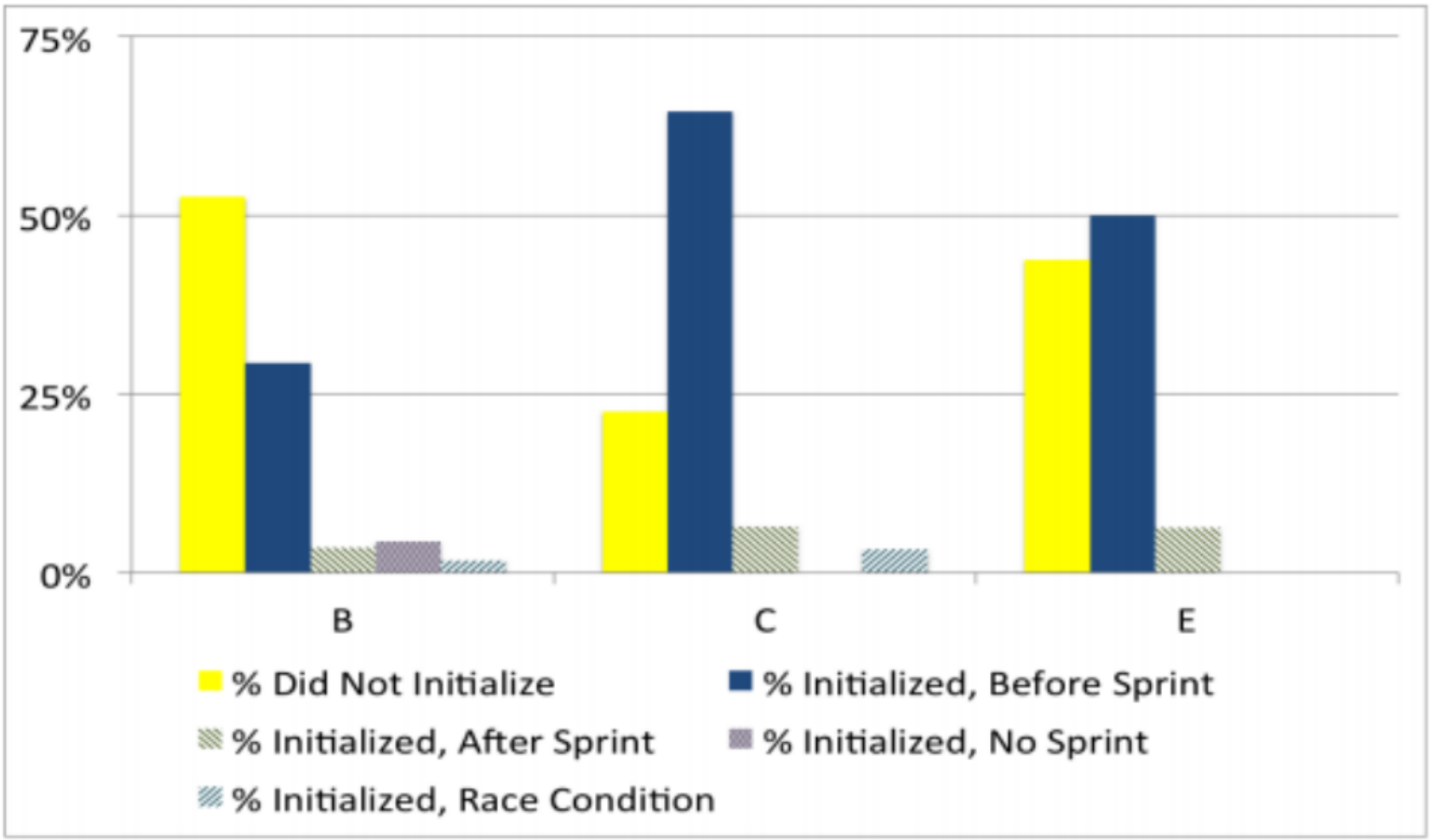
Results of Middle School Study 1

- Not surprisingly, the ones who had taken the Scratch course learned the concepts faster
- Overall, the teachers agreed that they could cover material at a "significantly" higher rate
- Concepts learned in Scratch were easier to teach in Java to the WHOLE class
- Entire class scored about the same on the final at the end of the high school
- Conditionals pop up as being difficult to understand for both groups of students in this class

Middle School Study 2

- Involved 10 classrooms across five schools in California
- Schools were called School A, School B, etc. etc.
- Schools A and D only had their final projects looked at and will not be discussed here
- Schools B, C, and E had extensive field notes taken, and the students were periodically interviewed
- Goal: to assess students in their ability to initialize objects in Scratch through identifying features in their final projects
  - Basically wanted a cat sprite to run across the screen and then go back to its original position, while changing its size at least once and then returning its size to back 100%

The correct script initialized 2 of the cats attributes, size and position


75%
50%
25%
0%
B
C
E
% Did Not Initialize
% Initialized, Before Sprint
% Initialized, After Sprint
% Initialized, No Sprint
% Initialized, Race Condition

Results of Middle School Study 2

- School C was best because they focused specifically on requirements for the final project
- The other 2 schools did fine but because they only received general training, they only some of the projects did exactly as they were supposed to

Weekly

- Aged 12-15
- Done over the course of 8 weeks
- Final project culminated into a “Hi-low Tech board-game”
- 8 sessions total, students met once per week for an hour
  - Scratch Basics
  - Debug’ems
  - Documentation
  - Testing the Games

Results of Weekly

- 3 "Successful" Games
  - Were logical
  - Code functioned properly
  - They were fun to play and met user requirements

2 Weeks

Study done in the Summer

- Aged 13-15
- More time for students
- "Fun" Environments
  - No "tests" (Except for end of project quiz)
  - No grades
- Concepts learned
  - Event-driven Programming
  - Initial State
  - Broadcast/Receive
  - Say/Sound Synchronization
  - Visibility
  - Variables
  - Conditionals
- End of study quiz

Question	Answer	Results
If You want the cat rather than a panda, what would you change?	Sprite	100%
If you want the cat to go into a cave rather than a temple, what do you change?	Scene	82%
How could you tum the monkey red?	Costume	91%
How do you make the bees keep going up and down?	Repeat	73%
How can you track how many times the bat has been hit by a banana?	Variable	73%
How do you make the bat disappear on the third hit but not the first hit?	If	28%

College Class

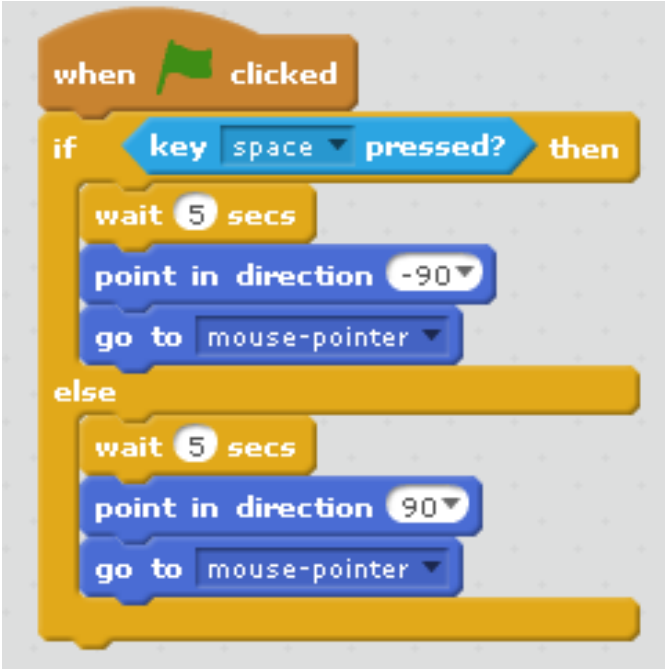
- Aged 18-21
- Took place over the course of a semester
- Students placed in Alice/Java course vs. Java only

Results of College Class

- Students who took the combined course overall scored 20% better on tests, quizzes, and homework than those who did not
- One drawback that came up in this course was that block-based coding lacked support for parallelism
- The students in the combined course learned the concepts faster but both classes struggled with conditionals
  - Both classes scored low on the first quiz given to assess their grasp of conditionals

Conditionals

- If then
- Else



Acknowledgments

In no Particular Order

- Nic Mcphee
- Peter Dolan
- Elena Machkasova
- KK Lamberty
- Stephen Adams
- Mom

Sources for Graphs/Pictures

- D. Franklin, C. Hill, H. A. Dwyer, A. K. Hansen, A. Iveland, and D. B. Harlow. Initialization in scratch: Seeking knowledge transfer. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE 16, pages 217222, New York, NY, USA, 2016. ACM.
- `https://scratch.mit.edu/projects/editor/?tip_bar=home`