Minimum Time, Maximum Effect

Introducing Parallel Computing
in CS0 and STEM Outreach Activities using Scratch

Russell Feldhausen
russfeld@ksu.edu

Scott Bell
rsbell@ksu.edu

Daniel Andresen
dan@ksu.edu

Kansas State University

The Problem

- Growing demand for HPC professionals
- Lack of CS and HPC awareness among target groups (middle school, females, new students)
- Lack of understanding of HPC at all levels
- Not many CS students interested in HPC
- HPC introduced very late in CS curricula

Our Goal

- Build interest in CS and HPC
  - Within target groups
  - With easy to understand examples and activities
  - For students with a wide range of age and skill

KAWSE GROW Program

- K-State Office for the Advancement of Women in Science and Engineering (KAWSE)
- Girls Researching our World (GROW) Summer Workshop
  - Three day camp for 6th through 8th grade girls
  - Encourages students to pursue science and technology related fields
  - We presented a 45 minute session on HPC

IBM
IBM
GROW

CIS 115 (CS 0)

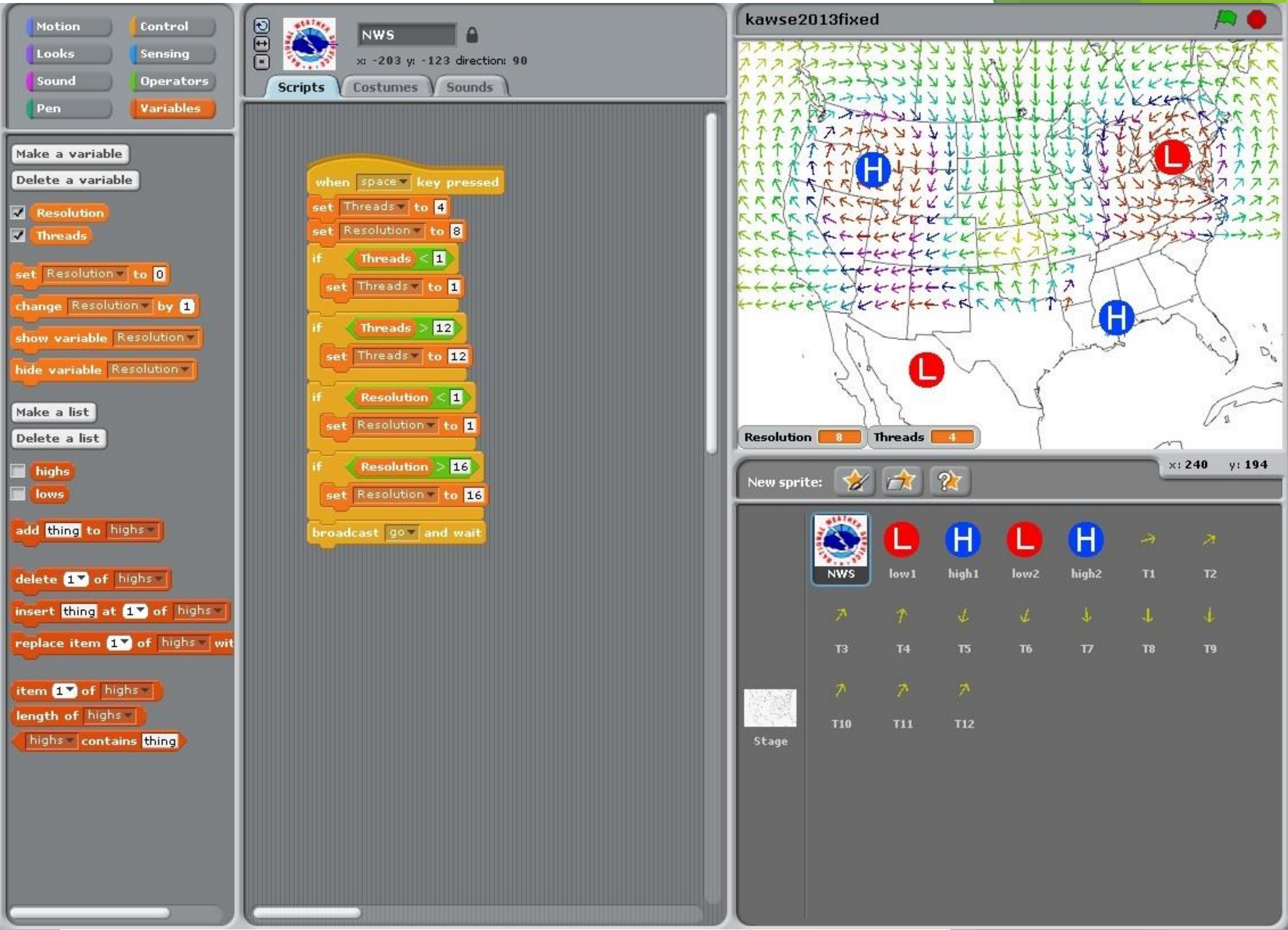
- CIS 115 – "Introduction to Computing Science"
- 3 major course objectives
  - Review of Computing Science history
  - Overview of basic programming concepts and various related topics (e.g. parallelism)
  - Information and examples based on major Computing Science sub-topics and research areas, including HPC

Scratch

- Developed by the Lifelong Kindergarten Group at the MIT Media Lab
- Drag & Drop programming environment
- Facilitates exploration of programming concepts without worrying about compilation errors or illogical statements

Activity 1: Wind Model

- Scratch program that models wind speed and direction based on pressure centers
- Each map arrow is calculated independently
- The program model can be adjusted to use different resolutions and numbers of threads
- Students record the time elapsed for various settings, then graph the results



kawse2013fixed

NWS

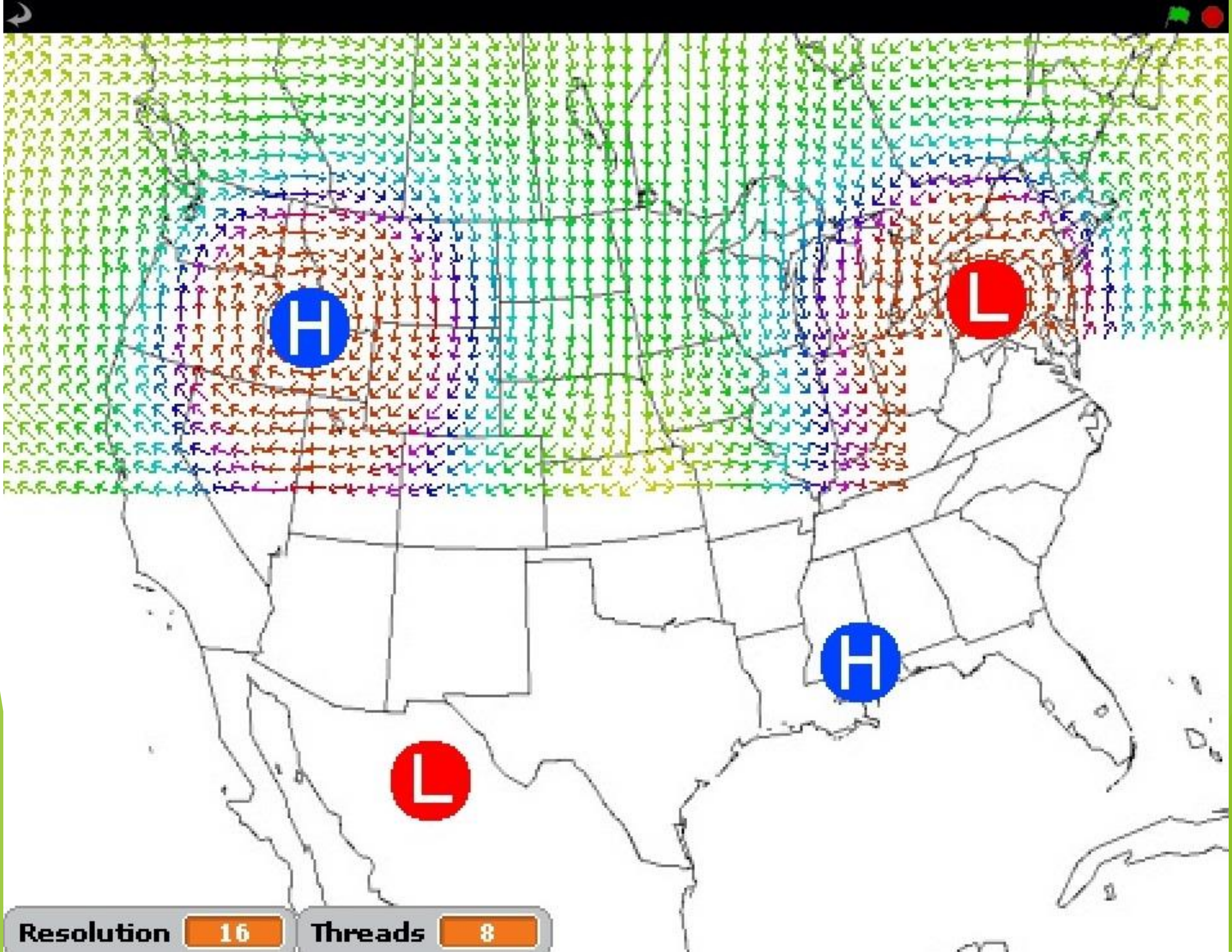
x: -203 y: -123 direction: 90

```
when space key pressed
set Threads to 4
set Resolution to 8
if Threads < 1
    set Threads to 1
if Threads > 12
    set Threads to 12
if Resolution < 1
    set Resolution to 1
if Resolution > 16
    set Resolution to 16
broadcast go and wait
```

Resolution 8

Threads 4

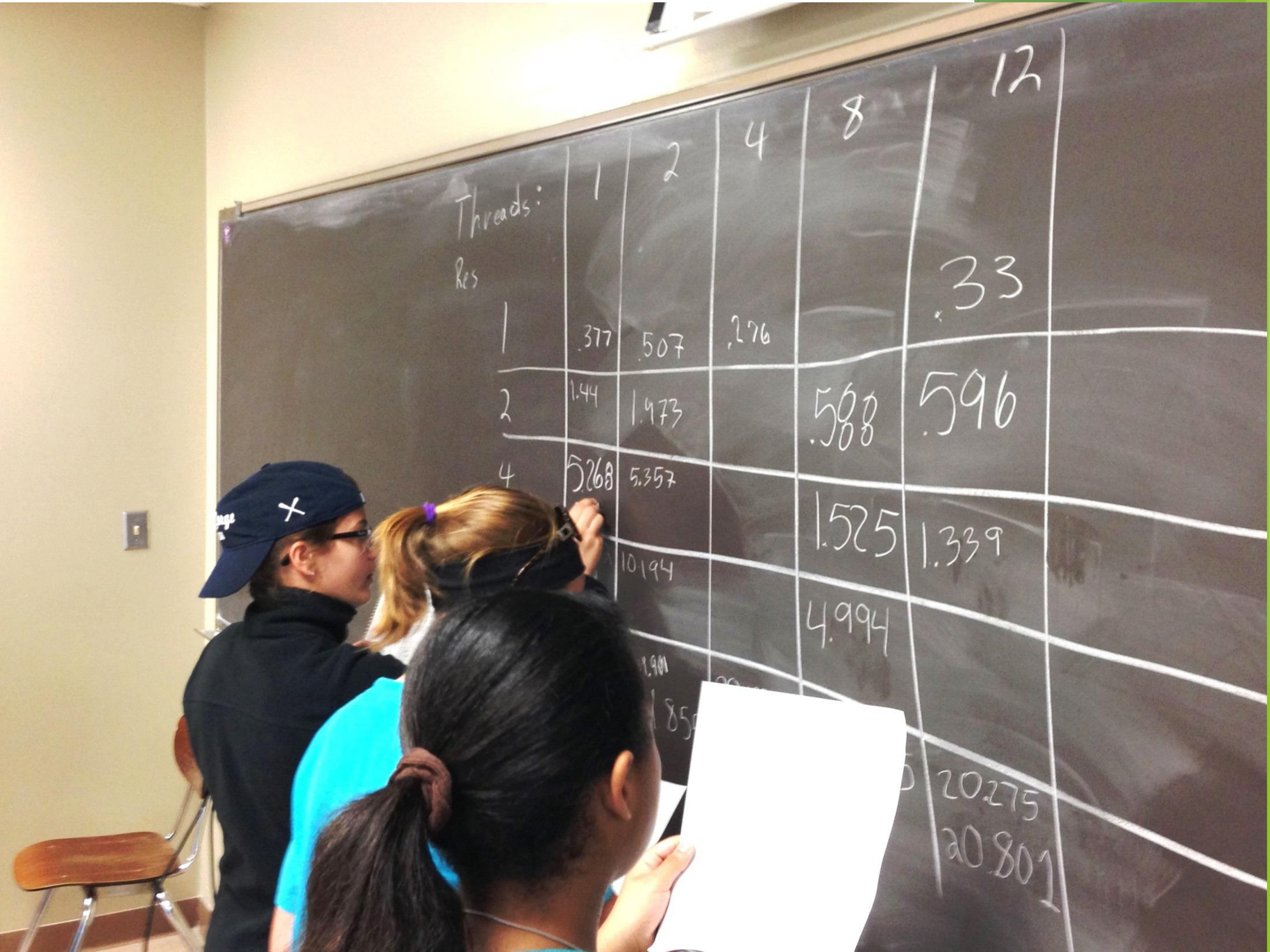
x: 240 y: 194



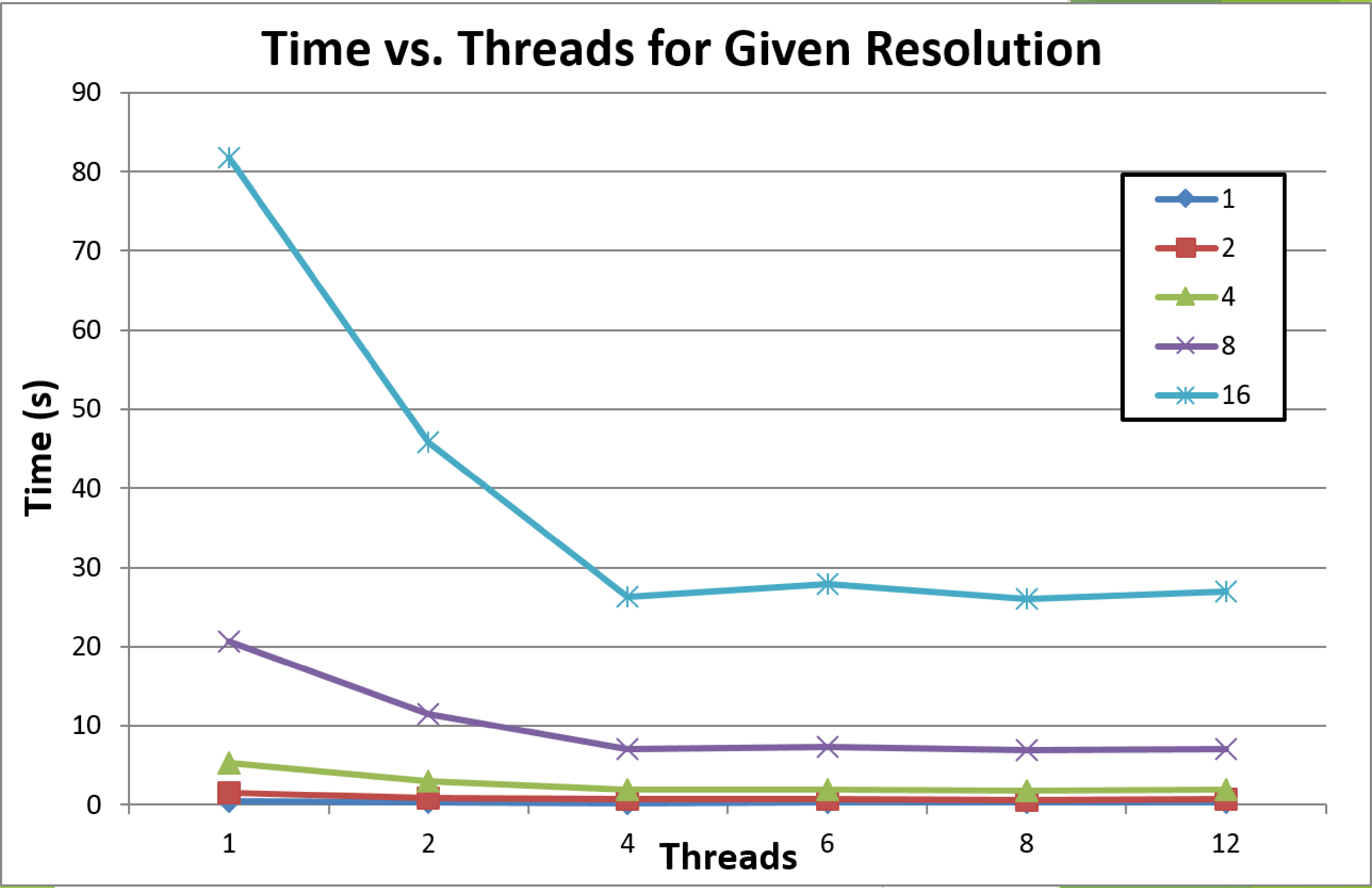
Resolution 16

Threads 8

Kansas State University
Computing & Information Sciences



Threads:	1	2	4	8	12
Res					
1	.377	.507	.276		.33
2	1.44	1.473		.588	.596
4	5.268	5.357		1.525	1.339
[illegible]	[illegible]	10.194		4.994	
[illegible]		[illegible].901 [illegible].85[illegible]	20.[illegible]		
[illegible]				[illegible]5	20.275 20.801

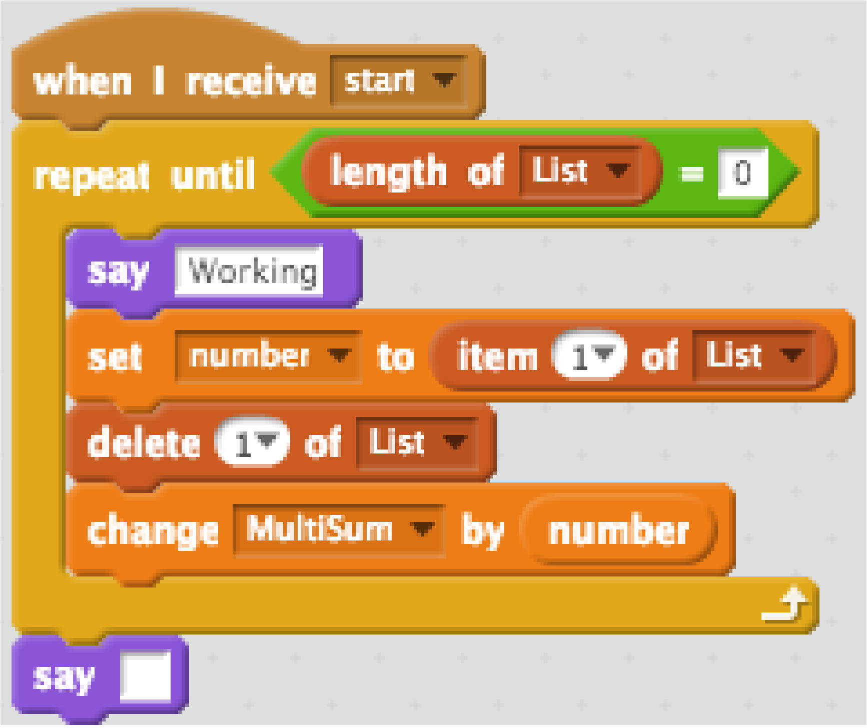


Time vs. Threads for Given Resolution

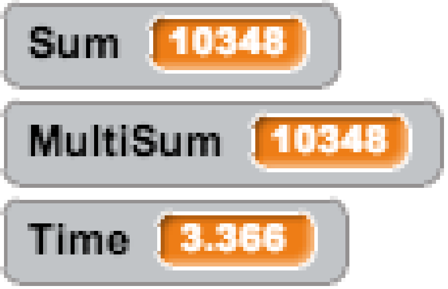
Activity 2: Summing Algorithm

- Introduce parallelism by building a summing algorithm in Scratch
- Goal is to sum a list of numbers as quickly as possible using multiple threads
- We modify the program to experience real-world concurrency problems (e.g. race conditions)

No Race Conditions (Why?)



```
when I receive start
repeat until length of List = 0
    say Working
    set number to item 1 of List
    delete 1 of List
    change MultiSum by number
say
```

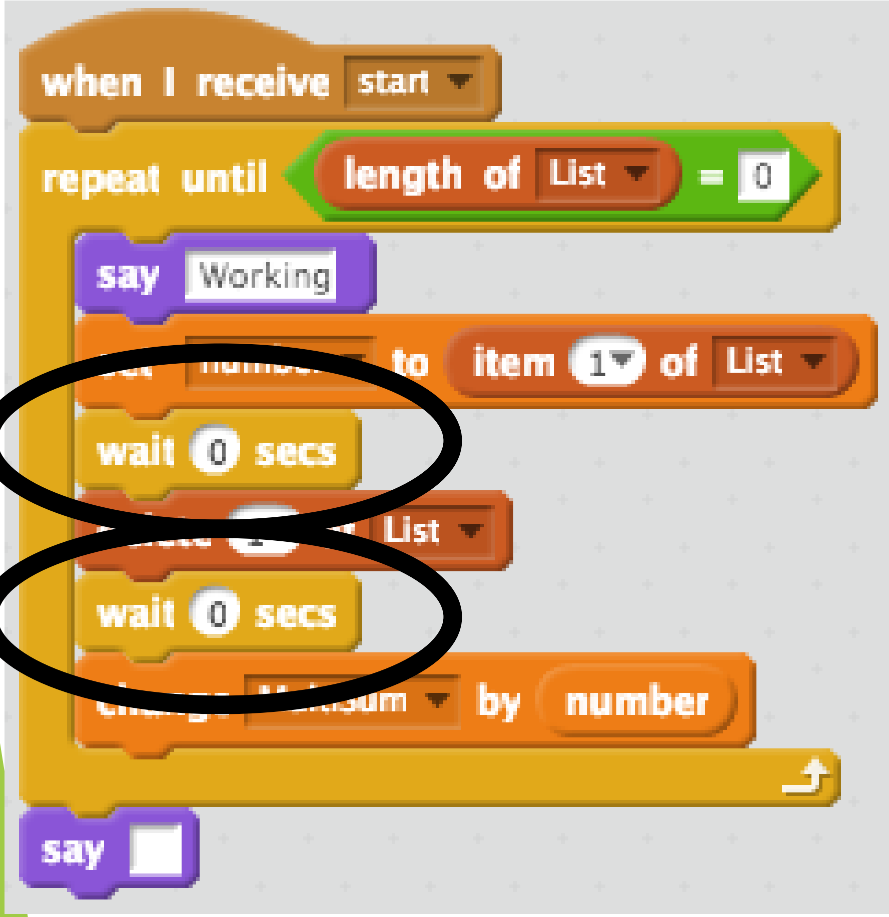


Sum 10348

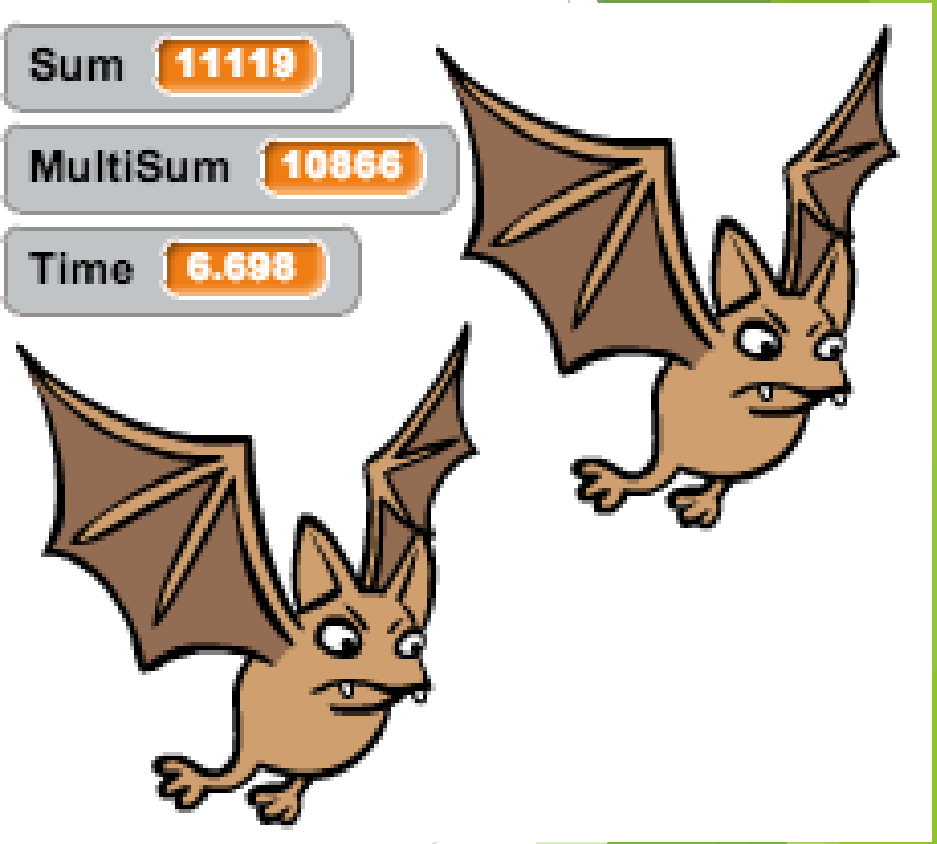
MultiSum 10348

Time 3.366

Race Condition

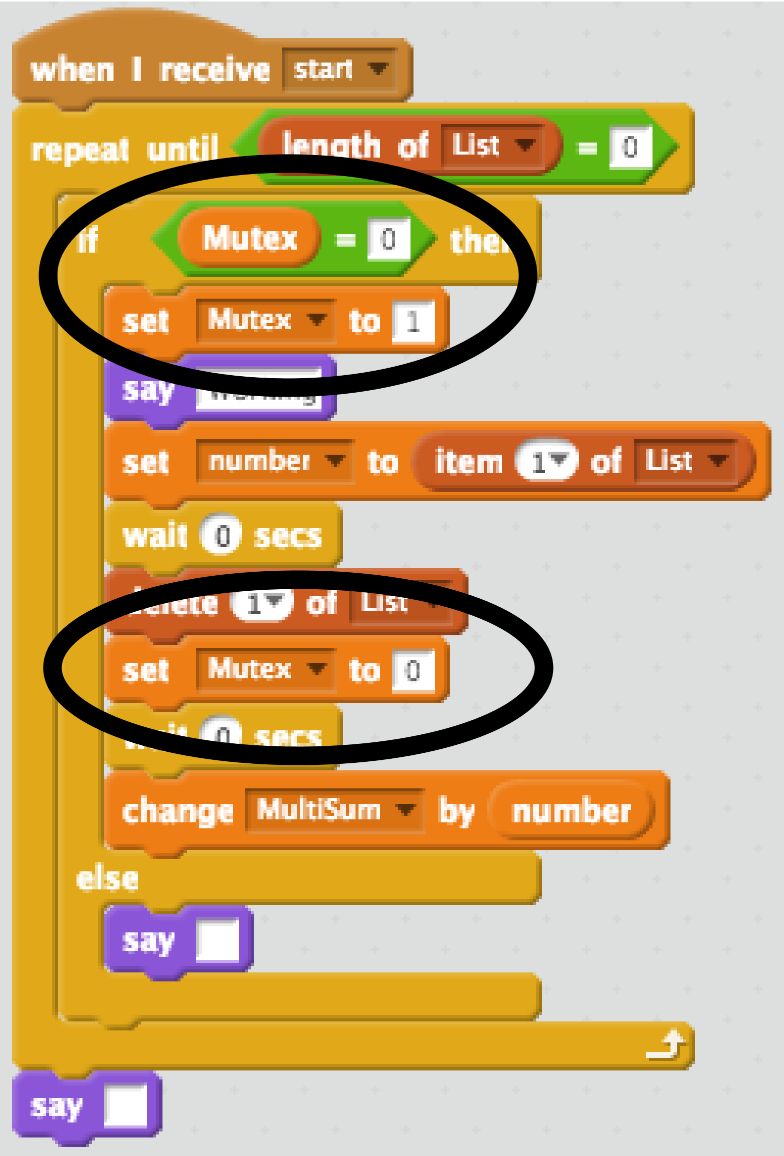


```
when I receive start
repeat until < (length of List) = 0 >
    say Working
    set number to item 1 of List
    wait 0 secs
    delete 1 of List
    wait 0 secs
    change MultiSum by number
say
```

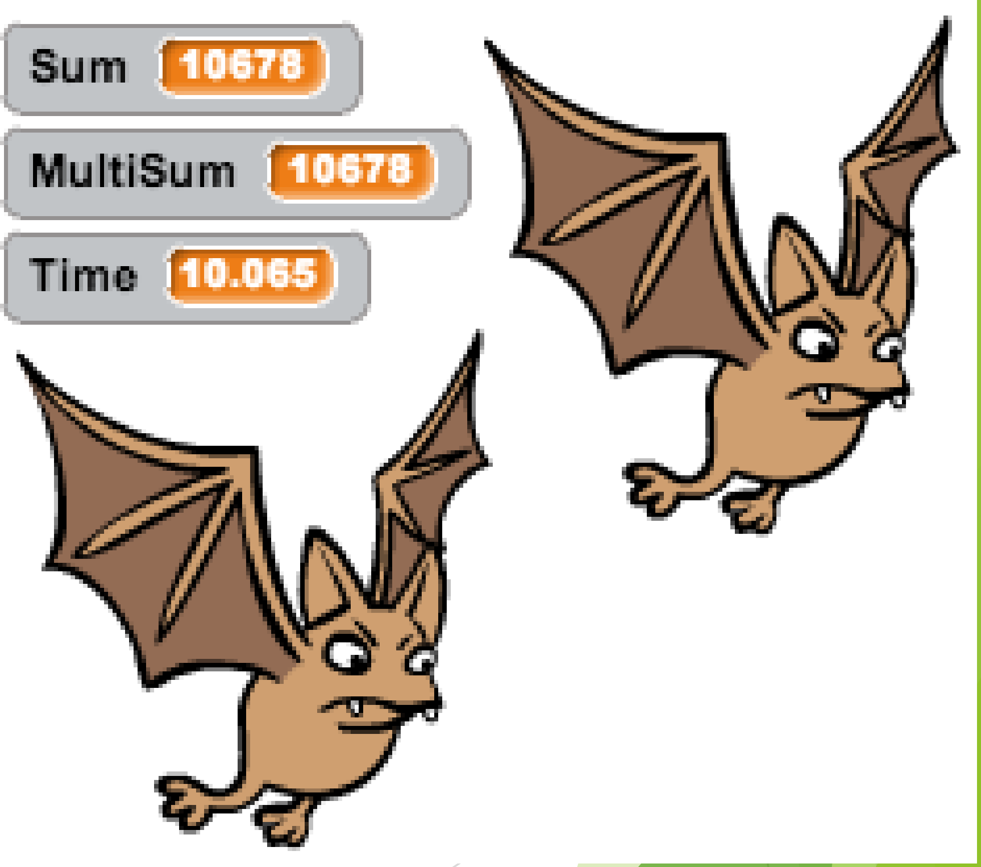


Sum	11119
MultiSum	10866
Time	6.698

Race Condition Fixed!



```
when I receive [start ▾]
repeat until <(length of [List ▾]) = [0]>
    if <(Mutex) = [0]> then
        set [Mutex ▾] to [1]
        say [Working]
        set [number ▾] to (item (1 ▾) of [List ▾])
        wait (0) secs
        delete (1 ▾) of [List ▾]
        set [Mutex ▾] to [0]
        wait (0) secs
        change [MultiSum ▾] by (number)
    else
        say []
    end
end
say []
```



```
Sum       10678
MultiSum  10678
Time      10.065
```

Results from GROW Program

- Students surveyed after session:
  - 22 of 41 had an interest in a job using HPC to solve problems
  - 27 of 37 felt they were capable of learning to write computer programs

- We were glad to see that this was a promising approach for introducing HPC

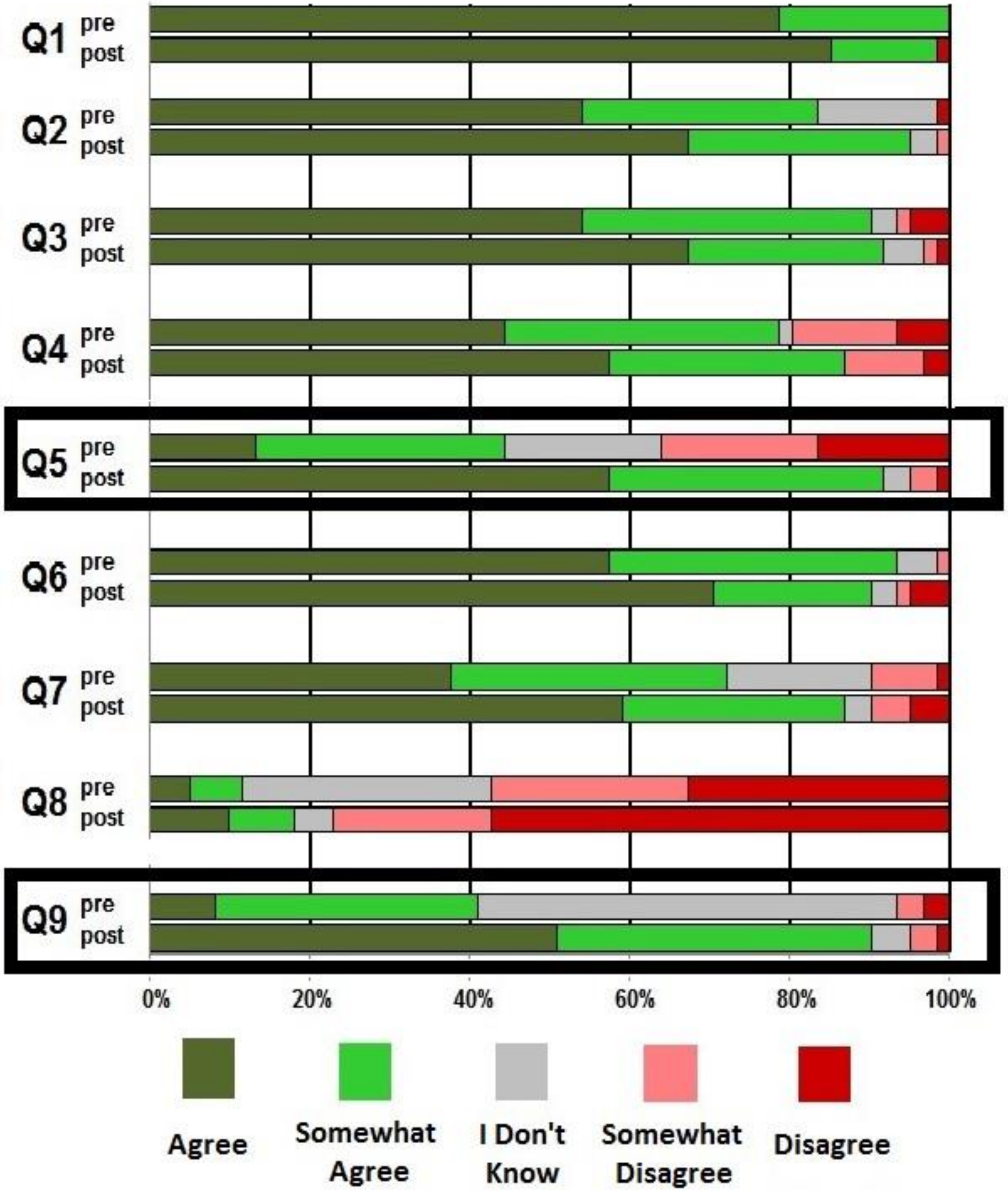
"The girls had a great time. They talked about it for the rest of the week!" *GROW'14*

Results from CS 0 Class (Spring 2014)

- Students completed a survey before and after HPC sessions
- 9 questions rated on a 5 pt. scale
  - Agree
  - Somewhat agree
  - I don’t know
  - Somewhat disagree
  - Disagree

Results from CS 0 Class (Spring 2014)

- The results were outstanding!
- Many students showed an increase in knowledge of and interest in HPC
- Two questions in particular showed a significant increase in agreement:
  - Q5: I understand what parallel programming means
  - Q9: I could apply parallel programming to other problems in Scratch


Q1 pre post
Q2 pre post
Q3 pre post
Q4 pre post
Q5 pre post
Q6 pre post
Q7 pre post
Q8 pre post
Q9 pre post
0%
20%
40%
60%
80%
100%
Agree
Somewhat Agree
I Don't Know
Somewhat Disagree
Disagree

What We Learned

- We can use Scratch to introduce HPC concepts to students with little or no programming background
- Students from a wide range of ages can comprehend HPC ideas through our activities
- Students show an increased interest in and knowledge of HPC after our activities

Future Work

- Expand the scope of simulations to cover other subject areas in computer science (e.g. bio-informatics)
- Enhance the activities in our CS 0 class to include more HPC concepts such as work queuing and producer/consumer relationships
- Evaluate these activities in upper level architecture and operating systems courses

More Information Online:

http://bit.ly/russfeld_hpc

Russell
Feldhausen
russfeld@ksu.edu

Scott Bell
rsbell@ksu.edu

Daniel Andresen
dan@ksu.edu