



**CREATE YOUR WORLD**  
SCRATCH AT MIT 2012

# Making Music with Scratch

a workshop presented at  
**Scratch@MIT 2012**  
Friday, July 27, 2012

**Jesse M. Heines**

Dept. of Computer Science  
[Jesse\\_Heines@uml.edu](mailto:Jesse_Heines@uml.edu)

**Gena R. Greher**

Dept. of Music  
[Gena\\_Greher@uml.edu](mailto:Gena_Greher@uml.edu)

University of Massachusetts Lowell

**[www.performamatics.org](http://www.performamatics.org)**





## Copyright Notice

The materials in this handout, including all text and programs, are copyright © 2012 by Jesse M. Heines, S. Alex Ruthmann, Gena R. Greher, and John Maloney under the GNU General Public License, version 2 (please see <http://www.gnu.org/licenses/gpl-2.0.html>).

All rights are reserved, but permission is hereby granted for these materials to be freely copied or excerpted for educational purposes with credit to the authors.

A full color version of this handout in PDF format may be downloaded from:

<http://bit.ly/scratch2012music>

## Contents

<b>Workshop Description</b> .....	5
Workshop Topics .....	8
Additional Workshop Information and URLs .....	9
About the Workshop Leaders .....	10
Important Note on Turbo Speed .....	11
Acknowledgements .....	11
<b>Performamatics</b>	
Project Website .....	13
Workshop Information .....	14
<b>Playing and Synchronizing MIDI Files</b> .....	17
MP3 Player Scripts .....	18
Control Script .....	19
<b>Example 1 - Frère Jacques</b>	
Version 1: Playing Notes .....	21
Version 2: Using Loops .....	22
Version 3: Separating Phrases.....	23
Version 4: Playing a Round .....	25
<b>Example 2 - Row, Row, Row Your Boat</b>	
Version 1: Playing Notes .....	29

*continued on next page*

## Contents (cont'd)

Version 2: Playing Notes Using Variables .....	30
Version 3: Separating Initialization .....	31
Version 4: Separating Phrases .....	33
Version 5: Looping and Fading .....	36
Version 6: Playing a Round with One Instrument .....	38
Version 7: Playing a Round with Two Instruments .....	40
Version 8: Storing Notes and Rhythms in Lists .....	43
Version 9: Playing a Round Using Lists .....	45
Version 10: Synchronizing Play from Lists .....	47
<b>Extending the Examples</b> .....	<b>51</b>
<b>The IchiBoard</b> .....	<b>55</b>
<b>The PicoBoard</b> .....	<b>57</b>
Serial to USB Setup .....	54
USB to USB Setup .....	59
<b>Computing and Music:</b>	
What Do They Have in Common? .....	61
<b>Computer Science, Math, and Music:</b>	
Concepts Covered in Scratch .....	65
<b>Additional Readings</b> .....	<b>67</b>
<b>Related Websites</b> .....	<b>71</b>

## Workshop Description

“There is nothing like making music and messing with sound to inspire people to learn how to program.”

— Prof. Dan Trueman, co-founder of the Princeton Laptop Orchestra

We couldn't agree more. However, the cost of “getting into the digital music game” can be prohibitive on many fronts. Our work strives to make the excitement of music technology accessible to students of all levels.

This workshop presents a subset of the techniques we've developed to allow students to explore the intersection of computing and music using the sound capabilities built into Scratch. In addition, it gives participants the opportunity to create music-generating programs themselves using a variety of Scratch constructs. The workshop will conclude with a mini concert in which some participants will play the music they created using these techniques.

One of the distinctive characteristics of the workshop is that it is presented in true interdisciplinary fashion: by a CS and a Music professor working together. This is a hallmark of our work. Participants will be exposed to a



## Workshop Description (cont'd)

fresh approach to teaching that they may be able to implement to a greater or lesser degree in their own schools and institutions.

### Background

Music applications are incredibly powerful and engaging tools for getting students interested in learning about technology. They can range from music cataloging applications such as the popular musicbrainz.org site, to music playing applications such as the ubiquitous Apple and Adobe products, to music transcribing and composing applications such as Finale, Sibelius, and Noteflight.

We are interested in harnessing the engaging power of music to stimulate student interest in computing in general and computational thinking in particular. Toward that end we have designed an interdisciplinary, general education course that teaches computing \*and\* music to undergraduates in novel ways and that is open to all students in all majors. Our project is called "Performamatics," and it has been funded by two grants from the National Science

## Workshop Description (cont'd)

Foundation. (Please see [www.performamatics.org](http://www.performamatics.org) for further information on this project.)

The centerpiece of Performamatics is "Sound Thinking," an interdisciplinary course taught with a music and a computer science professor in the room for all class meetings.

(Please see [soundthinking.uml.edu](http://soundthinking.uml.edu) for the course website.)

This approach models the interdisciplinary environments that students will encounter when they graduate and provides valuable lessons for life in the world of work.

The computational thinking component of our approach not only helps arts majors learn to think analytically, but also helps technical majors understand computing concepts at a deeper level through applications that employ the engaging power of music. We take advantage of the "low floor and high ceiling" of Scratch to appeal to students at both ends of the curricular spectrum. We believe that this is one of the real powers of this media-rich visual programming system. We have seen that harnessing its music capabilities is a tremendous "hook" for making programming appealing to a wide range of students, from those in middle school to those in undergraduate programs.

## Workshop Topics

1. Demonstration of Scratch music capabilities
2. Playing MP3 files from Scratch
  - Synching music to animations
  - Manipulation of MP3 files using Audacity
3. Playing MIDI notes from Scratch
  - Creating and playing simple melodies
  - Using loops and broadcasts to structure music
4. Playing MIDI notes using lists
  - Creating and populating lists
  - Working with rhythm and note lists
5. Synchronizing multiple parts
  - Techniques that do not work, and those that do
6. Introduction to external sensor devices
  - The Scratch Board and PicoBoard
  - The IchiBoard
7. Sharing what you've created
  - Live performances by participants 😊



## Additional Workshop Info and URLs

Participants should download and install:

- **Scratch 1.4**  
[scratch.mit.edu/download](http://scratch.mit.edu/download)
- **Audacity 2.0.1**  
[audacity.sourceforge.net/download](http://audacity.sourceforge.net/download)

Participants should also download and install the appropriate sensor board drivers for their systems.

- for **PicoBoards**  
[www.picocricket.com/whichpicoboard.html](http://www.picocricket.com/whichpicoboard.html)
- for **IchiBoards**  
[www.cs.uml.edu/ecg/index.php/IchiBoard](http://www.cs.uml.edu/ecg/index.php/IchiBoard)

**Important Note:** Scratch does not have access to a MIDI synthesizer on systems running Linux, Ubuntu, etc. Scratch **does** synthesize notes on these systems, but you only get one instrument.

## About the Workshop Leaders

**Jesse Heines** is a Professor of Computer Science at UMass Lowell. He has a keen interest in CS education and computer applications in the arts, particularly those in music. He teaches courses in object-oriented and graphical user interface programming. Jesse grew up in a musical household and currently enjoys singing in a barbershop chorus.

**Gena Greher** spent 20 years as a music producer/director in advertising before moving to UMass Lowell. She has published on the influence of multimedia technology in the general music classroom, middle school music curricula, and music teacher education curricula. Gena teaches courses in music methods, world music for the classroom, technology in music ed., and socio-cultural impacts on teaching music.

Jesse's and Gena's work has been supported by two **National Science Foundation** awards: "Performamatics: Connecting Computer Science to the Performing, Fine, and Design Arts" and "Computational Thinking through Computing and Music." Please see [www.performamatics.org](http://www.performamatics.org) for more information on these projects and **the opportunity to attend one of our two-day, NSF-sponsored interdisciplinary workshops.**

## Important Note on Turbo Speed

The timing of virtually all music scripts can be improved by setting Turbo Speed. To do this, select:

**Edit → Set Single Stepping... → Turbo Speed**

## Acknowledgements

Additional contributors to this work include UMass Lowell Profs. Gena Greher, S. Alex Ruthmann, and Fred Martin, graduate student Mark Sherman, recent undergraduates Paul Laidler and Charles Saulters, and current undergraduates Brendan Reilly, Angelo Gamarra, Matt Vaughan, and Nathan Goss.



The materials presented in this workshop are based on work supported by the National Science Foundation under Award Nos. 0722161 and 1118435. Any opinions, findings, and conclusions or recommendations expressed or implied in these materials or the workshop discussion are those of the authors alone and do not necessarily reflect the views of the National Science Foundation.

# SCRATCH



# Performamatics Project Website



UMass Lowell TUES Performamatics Project : Home - Mozilla Firefox

teaching.cs.uml.edu/~heines/TUES/ProjectHome.jsp

University of Massachusetts Lowell  
Depts. of Music and Computer Science

**PERFORMAMATICS**

**UMASS LOWELL**

**Computational Thinking through Computing and Music**  
*an interdisciplinary NSF TUES project*

[Home](#) [Workshop](#) [About Our Project](#) [About Us](#) [Resources](#) [Publications](#)

Our second **two-day interdisciplinary workshop** will take place on:  
**Thursday and Friday, January 17-18, 2013, at UMass Lowell**


Our third **two-day interdisciplinary workshop** will *tentatively* take place on:  
**Thursday and Friday, June 20-21, 2013, at UMass Lowell**

To attend the workshop, please fill out the **workshop application form**.

Our goal is to develop and disseminate ways to enhance students' grasps of computational thinking by engaging them in fundamental concepts that unite computing and music. Our approach leverages students' near universal interest in music as a context and springboard for engaging in rich computational thinking experiences. Prior work in an NSF CPATH project showed this approach to be effective at creating value in both discipline-specific courses for Computer Science and Music majors, as well as General Education courses for all majors. This project will develop additional activities to deepen students' experiences in computing and music, and explore additional techniques for evaluating learning through those activities. The project will also disseminate our work through workshops for pairs of interdisciplinary faculty at 4- and 2-year colleges.




Our materials teach concepts such as modularization by breaking songs down into their components, looping and subroutines by noting where musical phrases are repeated intact and with small variations (requiring parameters), logic flow by creating musical flowcharts, and algorithms by writing programs that generate music. New materials will explore ways to teach more advanced computing concepts such as threads and synchronization by writing programs that play multiple parts simultaneously and use various Application Programmer Interfaces (APIs), allowing us to combine software platforms into systems that to do more than is possible by one alone.



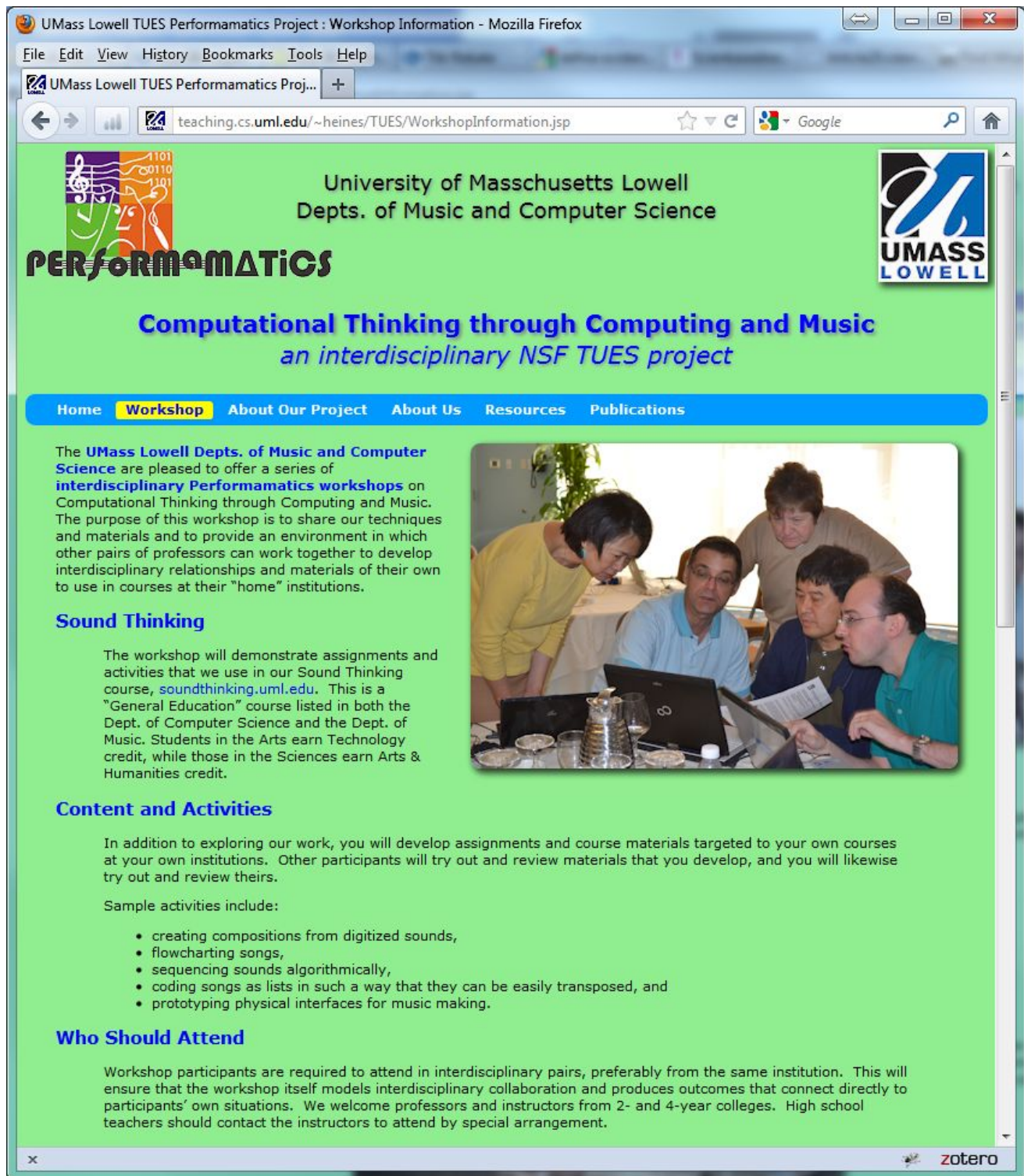
A major component of this project is the sharing of our techniques and materials through sponsored workshops at conferences and on-site at universities where participants attend as a pair: at least one from Computer Science (or another science or engineering department) and one from Music (or another arts department). This will ensure that collaborations begun in the workshops have a foothold on sustainability when the participants return to their own institutions.

This project is supported by Award No. 1118435 from the National Science Foundation (NSF) Division of Undergraduate Education (DUE). It falls under the TUES program: Transforming Undergraduate Education in STEM (Science, Technology, Engineering, and Mathematics). Any opinions, findings, conclusions, or recommendations expressed in our materials are solely those of the authors and do not necessarily reflect the views of the National Science Foundation.



zotero

# Performamatics Workshop Info



UMass Lowell TUES Performamatics Project : Workshop Information - Mozilla Firefox

teaching.cs.uml.edu/~heines/TUES/WorkshopInformation.jsp

University of Massachusetts Lowell  
Depts. of Music and Computer Science

**PERFORMAMATICS**

**UMASS LOWELL**

**Computational Thinking through Computing and Music**  
*an interdisciplinary NSF TUES project*

Home **Workshop** About Our Project About Us Resources Publications

The **UMass Lowell Depts. of Music and Computer Science** are pleased to offer a series of **interdisciplinary Performamatics workshops** on Computational Thinking through Computing and Music. The purpose of this workshop is to share our techniques and materials and to provide an environment in which other pairs of professors can work together to develop interdisciplinary relationships and materials of their own to use in courses at their "home" institutions.

**Sound Thinking**

The workshop will demonstrate assignments and activities that we use in our Sound Thinking course, [soundthinking.uml.edu](http://soundthinking.uml.edu). This is a "General Education" course listed in both the Dept. of Computer Science and the Dept. of Music. Students in the Arts earn Technology credit, while those in the Sciences earn Arts & Humanities credit.

**Content and Activities**

In addition to exploring our work, you will develop assignments and course materials targeted to your own courses at your own institutions. Other participants will try out and review materials that you develop, and you will likewise try out and review theirs.

Sample activities include:

- creating compositions from digitized sounds,
- flowcharting songs,
- sequencing sounds algorithmically,
- coding songs as lists in such a way that they can be easily transposed, and
- prototyping physical interfaces for music making.

**Who Should Attend**

Workshop participants are required to attend in interdisciplinary pairs, preferably from the same institution. This will ensure that the workshop itself models interdisciplinary collaboration and produces outcomes that connect directly to participants' own situations. We welcome professors and instructors from 2- and 4-year colleges. High school teachers should contact the instructors to attend by special arrangement.

## Performamatics Workshop Info (cont'd)

UMass Lowell TUES Performamatics Project : Workshop Information - Mozilla Firefox

File Edit View History Bookmarks Tools Help

teaching.cs.uml.edu/~heines/TUES/WorkshopInformation.jsp



### Venue

Univ. of Massachusetts Lowell  
Inn & Conference Center  
50 Warren Street  
Lowell, MA 01852  
1-978-934-6920

### Upcoming Dates

**Thursday-Friday, January 17-18, 2013**  
*full two-day workshop*  
9:00 AM to 5:00 PM both days

**Thursday-Friday, June 20-21, 2013**  
*full two-day workshop — dates tentative*  
9:00 AM to 5:00 PM both days

### Cost

Thanks to support for the National Science Foundation, there is no charge to participants to attend the workshop. In addition, our project funds cover two nights' lodging and meals on both days of the workshop. Limited funds are also available to help support participant travel to and from the workshop. Prospective participants requiring travel support should contact the instructors for further information.



### How to Apply

Please fill out our [Workshop Application](#) form.

### Instructors

**Jesse Heines** is a Professor of Computer Science with a strong interest in the power of music to engage students and help them learn computing concepts as they explore music applications. He teaches courses on graphical user interfaces, web programming, and C++.

**Gena Greher** is a Professor of Music Education. Her research focuses on creativity and listening skill development in children and examining the influence of integrating multimedia technology into urban music classrooms and in the music teacher education curriculum.

**S. Alex Ruthmann** is an Assistant Professor of Music Education. His research explores social/digital media musicianship and creativity and the development of technologies for music learning, teaching, and engagement in schools and community-based arts+computing programs.





### Further Information

Please contact:

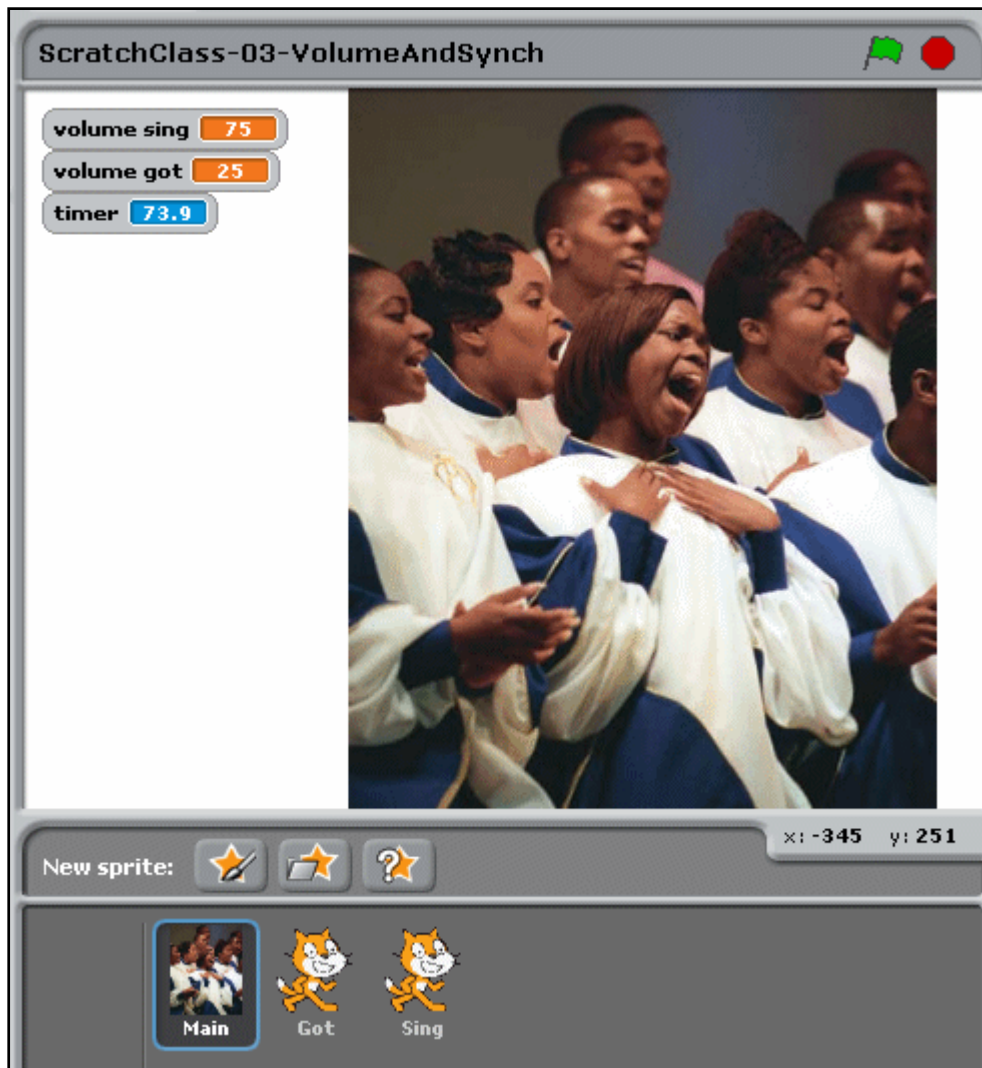
- Jesse Heines — [Jesse\\_Heines@uml.edu](mailto:Jesse_Heines@uml.edu) — 978-934-3634
- Gena Greher — [Gena\\_Greher@uml.edu](mailto:Gena_Greher@uml.edu) — 978-934-3893
- Alex Ruthmann — [Alex\\_Ruthmann@uml.edu](mailto:Alex_Ruthmann@uml.edu) — 978-934-3879

# SCRATCH





## Playing and Synchronizing MIDI Files



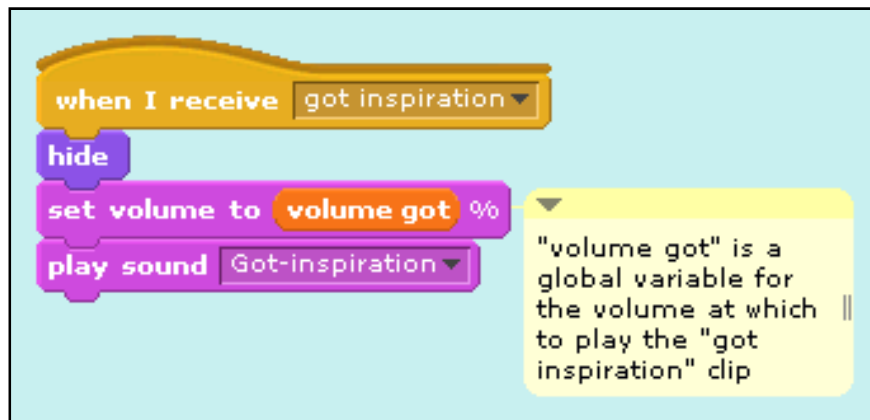
### Volume and Synchronization Concepts

- use of variables when setting the volume
- local vs. global attributes, specifically volume
- use of a control script and broadcasts
- use of the Scratch timer for synchronization

# Playing and Synching MIDI Files (cont'd)

## MP3 Player Scripts

### Script in Sprite "Got"



when I receive `got inspiration`

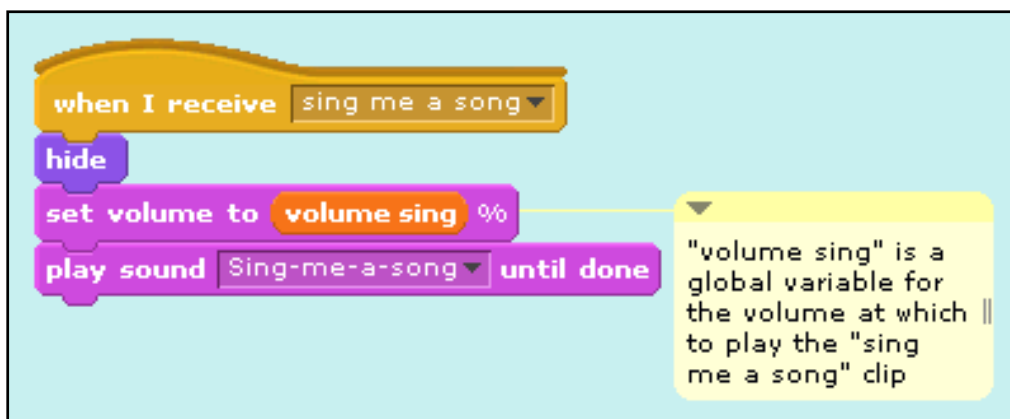
hide

set volume to `volume got` %

play sound `Got-inspiration`

"volume got" is a global variable for the volume at which to play the "got inspiration" clip

### Script in Sprite "Sing"



when I receive `sing me a song`

hide

set volume to `volume sing` %

play sound `Sing-me-a-song` until done

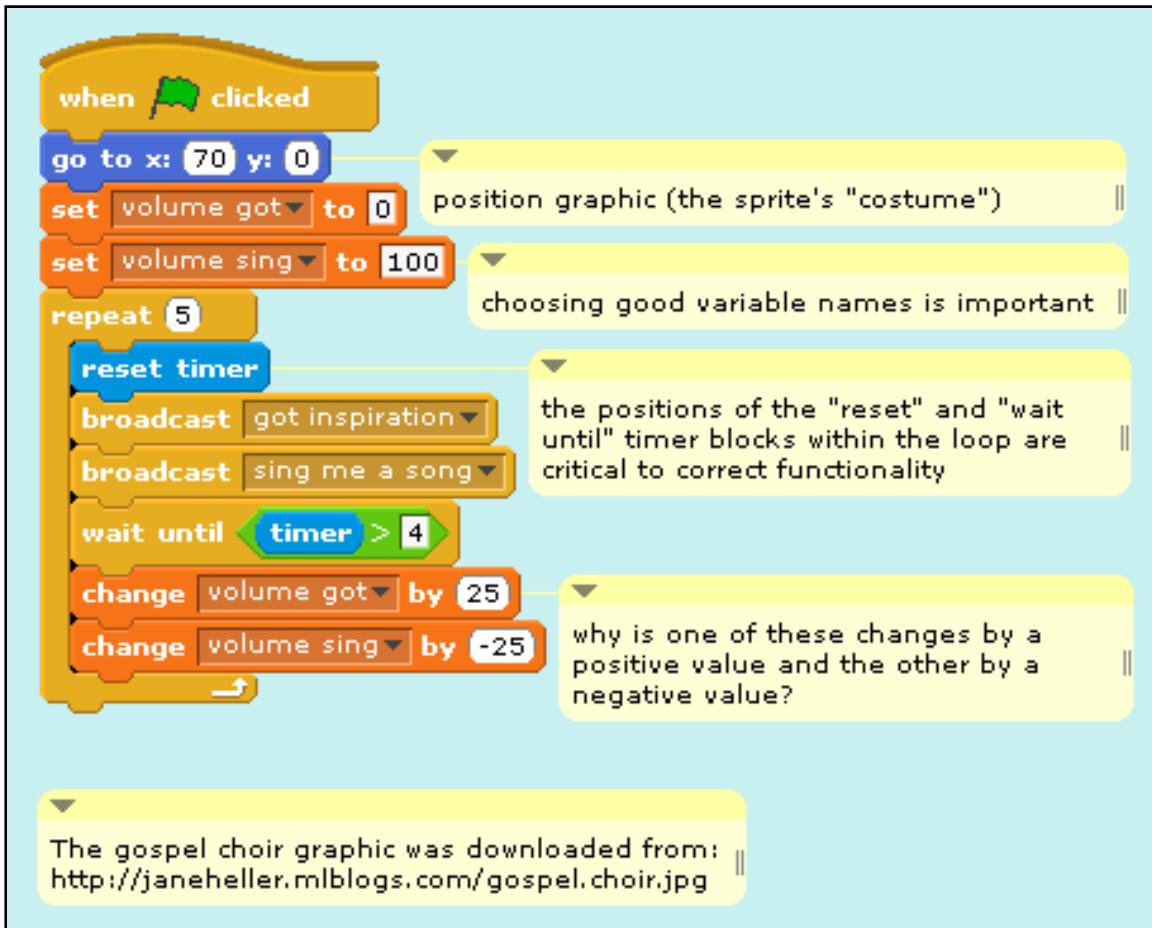
"volume sing" is a global variable for the volume at which to play the "sing me a song" clip

Each script must be in its own sprite to allow volume to be controlled independently.

# Playing and Synching MIDI Files (cont'd)

## Control Script

### Script in Sprite "Main"



The script is as follows:

```

when clicked
  go to x: 70 y: 0
  set volume got to 0
  set volume sing to 100
  repeat 5
    reset timer
    broadcast got inspiration
    broadcast sing me a song
    wait until timer > 4
    change volume got by 25
    change volume sing by -25
  
```

Annotations:

- position graphic (the sprite's "costume")
- choosing good variable names is important
- the positions of the "reset" and "wait until" timer blocks within the loop are critical to correct functionality
- why is one of these changes by a positive value and the other by a negative value?

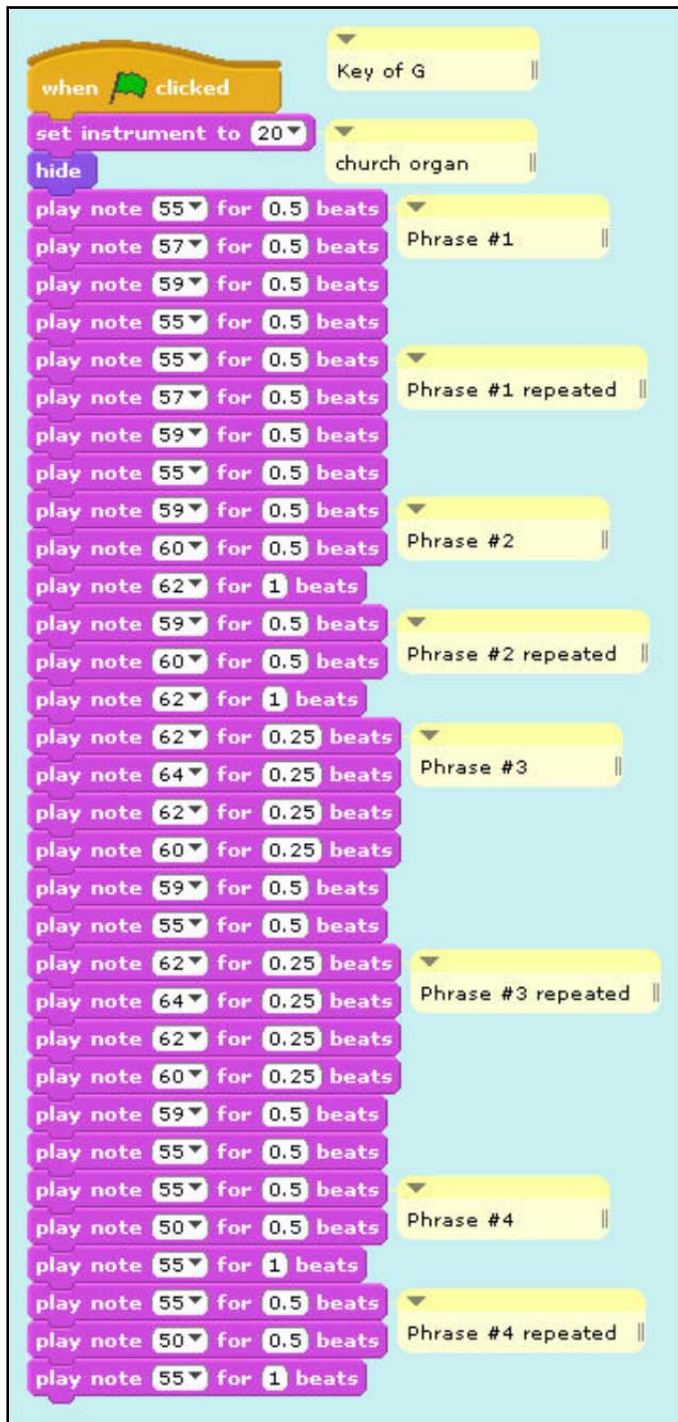
The gospel choir graphic was downloaded from: <http://janeheller.mlblogs.com/gospel.choir.jpg>

Note the order of the blocks and the critical position of the **change** blocks. Changing the volume parameter before the **wait until** block will cause the volume to be changed while the MP3 is playing. Such behavior may be desirable in other programs, but not this one.

# SCRATCH



# Frère Jacques Version 1: Playing Notes



when clicked

set instrument to 20

hide

Key of G

church organ

Phrase #1

Phrase #1 repeated

Phrase #2

Phrase #2 repeated

Phrase #3

Phrase #3 repeated

Phrase #4

Phrase #4 repeated

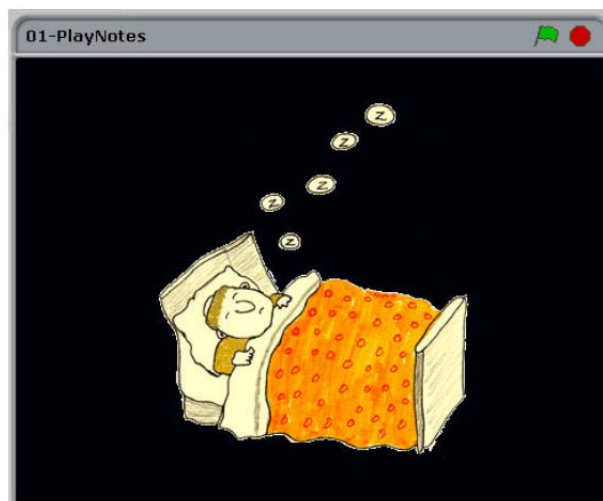


Frè - re Jac - ques Frè - re Jac - ques

Dor - mez - vous? Dor - mez - vous?

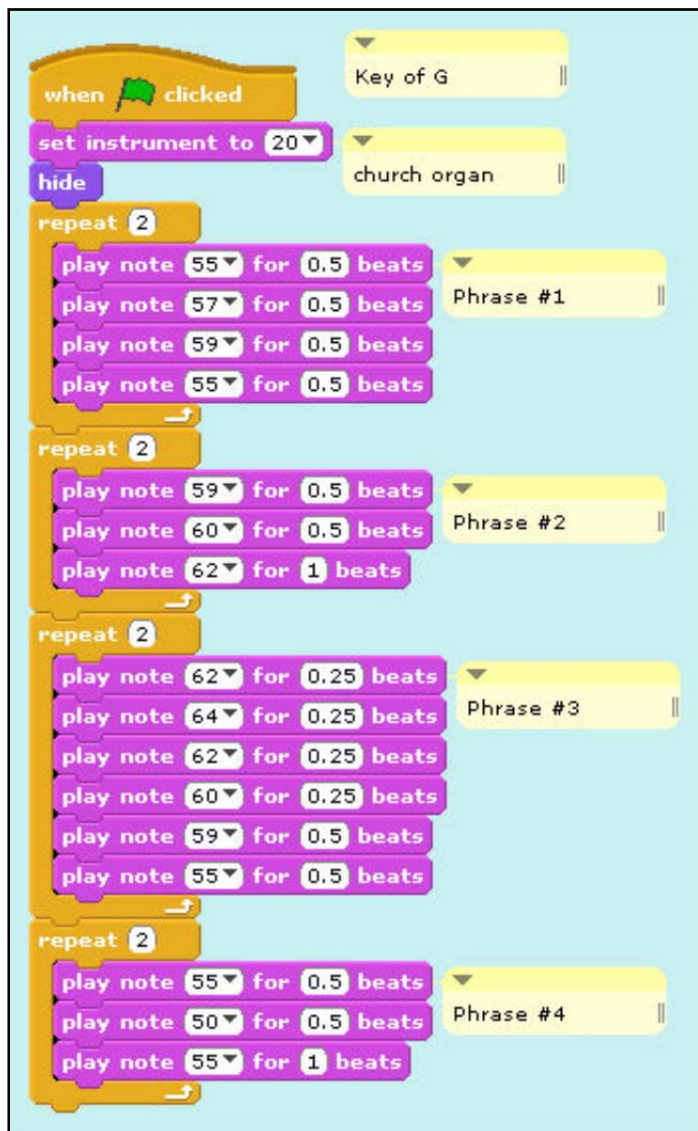
Son - nez les ma - ti - nes Son - nez les ma - ti - nes

Ding dingue dong Ding dingue dong



Remember Turbo Speed!

## Frère Jacques Version 2: Using Loops



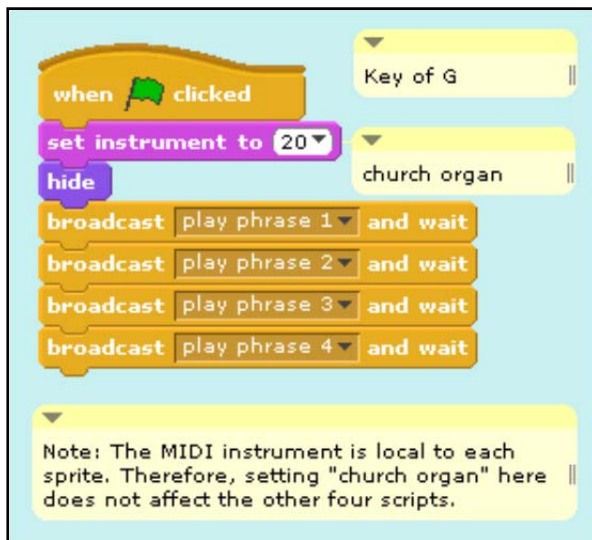

Remember to set Turbo Speed to improve performance.

**Acknowledgement:** The scores on this and the previous page were adapted from [www.csdraveurs.qc.ca/musique/flutalors/images/frere.gif](http://www.csdraveurs.qc.ca/musique/flutalors/images/frere.gif) and [www.mamalisa.com/images/scores/frerejacques.jpg](http://www.mamalisa.com/images/scores/frerejacques.jpg), respectively, but as of July 12, 2012, these URLs no longer appear to be valid.

## Frère Jacques

# Version 3: Separating Phrases

## Main Script



when clicked

set instrument to 20

hide

broadcast play phrase 1 and wait

broadcast play phrase 2 and wait

broadcast play phrase 3 and wait

broadcast play phrase 4 and wait

Note: The MIDI instrument is local to each sprite. Therefore, setting "church organ" here does not affect the other four scripts.



New sprite: [star] [star] [star]

Ver. 03

Phrase1

Phrase2

Phrase3

Phrase4

## Phrases Scripts (4, cont'd on next page)

#1



when I receive play phrase 1

repeat 2

play note 55 for 0.5 beats

play note 57 for 0.5 beats

play note 59 for 0.5 beats

play note 55 for 0.5 beats

Phrase #1

**Thought:** We could set the instrument in each script, but that would contradict the **DRY** programming principle: "Don't Repeat Yourself."

## Frère Jacques

### Version 3: Separating Phrases (cont'd)

#### Phrases Scripts (cont'd)

#2

```

when I receive play phrase 2
repeat 2
  play note 59 for 0.5 beats
  play note 60 for 0.5 beats
  play note 62 for 1 beats
  
```

Phrase #2

#3

```

when I receive play phrase 3
repeat 2
  play note 62 for 0.25 beats
  play note 64 for 0.25 beats
  play note 62 for 0.25 beats
  play note 60 for 0.25 beats
  play note 59 for 0.5 beats
  play note 55 for 0.5 beats
  
```

Phrase #3

#4

```

when I receive play phrase 4
repeat 2
  play note 55 for 0.5 beats
  play note 50 for 0.5 beats
  play note 55 for 1 beats
  
```

Phrase #4

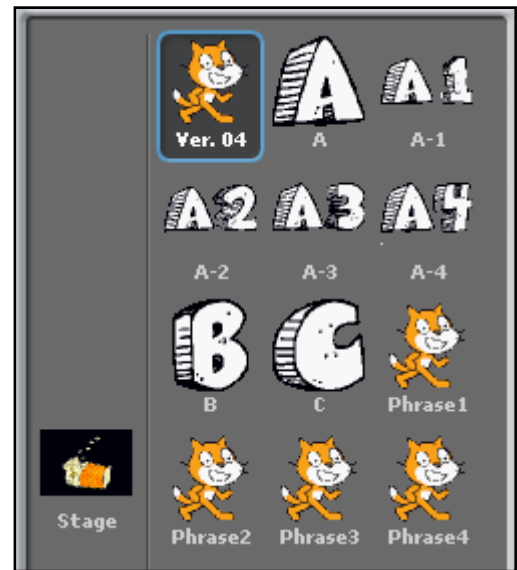
**Challenge:** How can we set the instrument **JUST ONCE** and have that setting apply to all scripts?



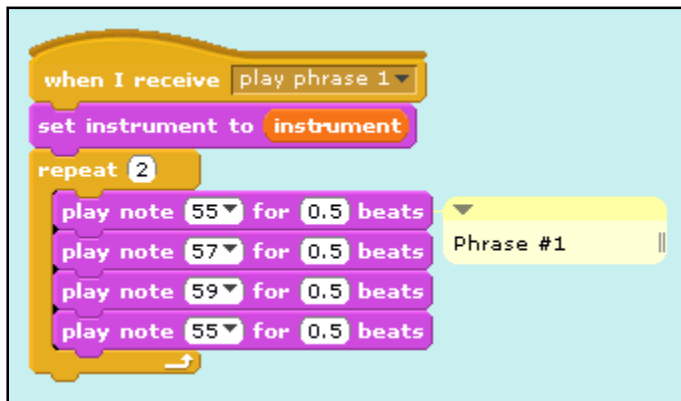
## Frère Jacques

# Version 4: Playing a Round

## Main Script



## Phrases Scripts



Note the addition of the **set instrument** block and the use of the **instrument** variable (set in the Main script) as the value to set. Other phrase scripts similarly contain this one revision.

*continued on next page*

## Frère Jacques

### Version 4: Playing a Round (cont'd)

#### Scripts A-1 through A-4

```

when I receive play part A-1
broadcast play phrase 1
wait 4 secs
broadcast play phrase 2
wait 4 secs
broadcast play phrase 3
wait 4 secs
broadcast play phrase 4
    
```

```

when I receive play part A-2
broadcast play phrase 1
wait 4 secs
broadcast play phrase 2
wait 4 secs
broadcast play phrase 3
wait 4 secs
broadcast play phrase 4
    
```

...

Others scripts are similar, differing only in **when I receive**.

#### Control Script A - single threaded

```

when I receive play version A
hide
broadcast play part A-1
wait 4 secs
broadcast play part A-1
wait 4 secs
broadcast play part A-1
wait 4 secs
broadcast play part A-1
    
```

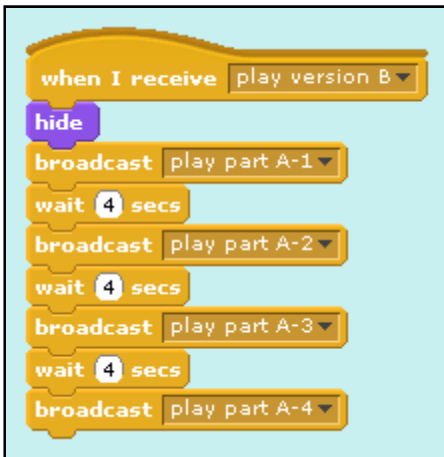
This version does not play as expected.

*continued on next page*

## Frère Jacques

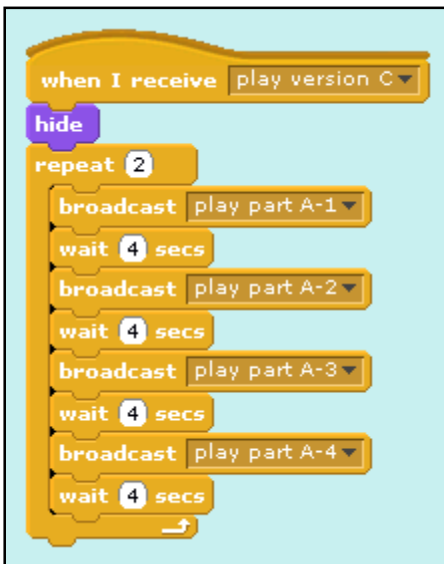
### Version 4: Playing a Round (cont'd)

#### Control Script B - multi-threaded



```
when I receive play version B
hide
broadcast play part A-1
wait 4 secs
broadcast play part A-2
wait 4 secs
broadcast play part A-3
wait 4 secs
broadcast play part A-4
```

#### Control Script C - multi-threaded repeat



```
when I receive play version C
hide
repeat 2
  broadcast play part A-1
  wait 4 secs
  broadcast play part A-2
  wait 4 secs
  broadcast play part A-3
  wait 4 secs
  broadcast play part A-4
  wait 4 secs
```

*end of Version 4*

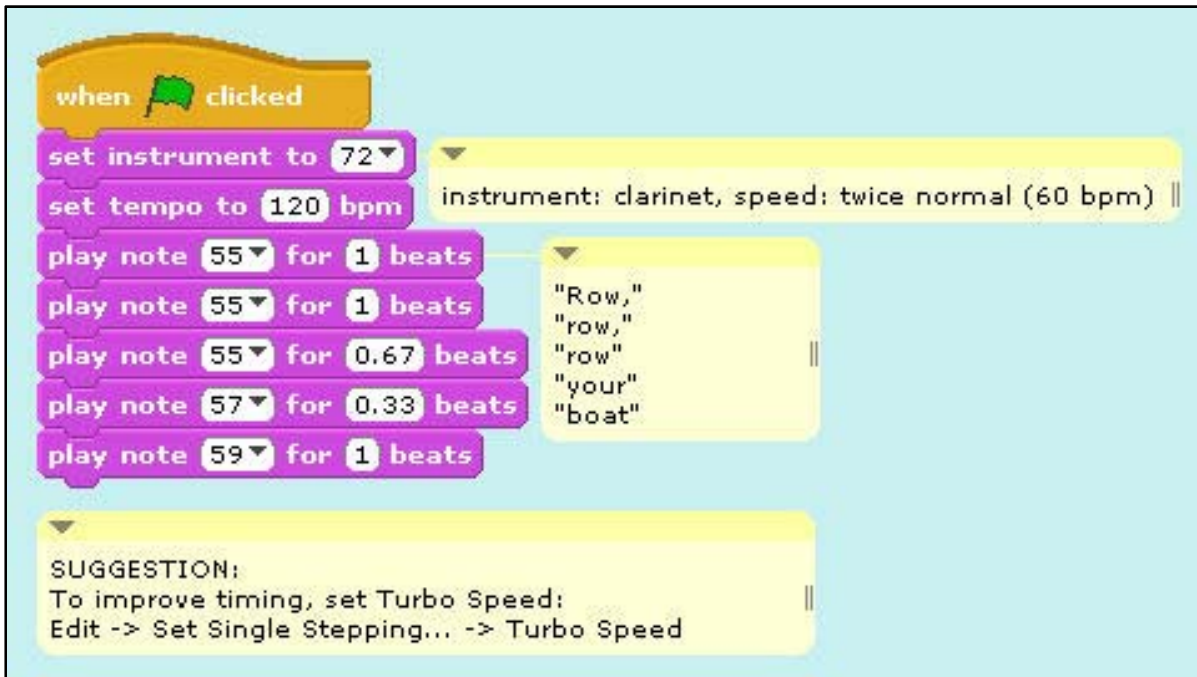
# SCRATCH



# Row, Row, Row Your Boat

## Version 1: Playing Notes

### Single Script



when clicked

- set instrument to 72
- set tempo to 120 bpm
- play note 55 for 1 beats
- play note 55 for 1 beats
- play note 55 for 0.67 beats
- play note 57 for 0.33 beats
- play note 59 for 1 beats

instrument: clarinet, speed: twice normal (60 bpm) ||

"Row,"  
"row,"  
"row"  
"your"  
"boat" ||

SUGGESTION:  
To improve timing, set Turbo Speed:  
Edit -> Set Single Stepping... -> Turbo Speed ||

### Output Window



## Row, Row, Row Your Boat

### Version 2: Playing Notes Using Variables

#### Single Script



The script is as follows:

```

when green flag clicked
  set G to 55
  set A to 57
  set B to 59
  set instrument to 72
  set tempo to 120 bpm
  play note G for 1 beats
  play note G for 1 beats
  play note G for 0.67 beats
  play note A for 0.33 beats
  play note B for 1 beats
  
```

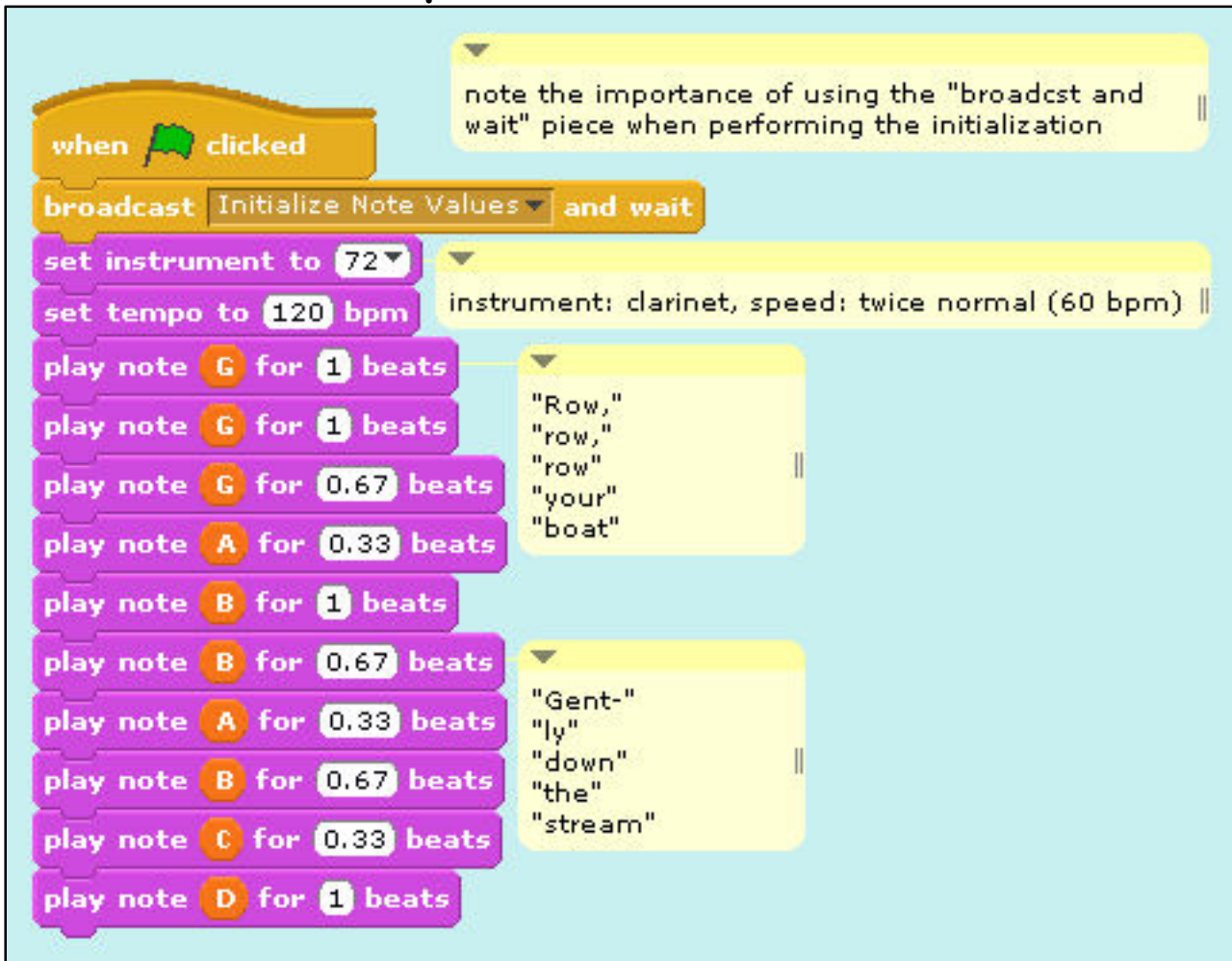
Comments for the script:

- initialize note values ||
- instrument: clarinet, speed: twice normal (60 bpm) ||
- "Row,"  
"row,"  
"row"  
"your"  
"boat" ||

## Row, Row, Row Your Boat Version 3: Separating Initialization

### Two Scripts

#### (3a) Main Script



The script is as follows:

- when clicked
  - broadcast Initialize Note Values and wait
  - set instrument to 72
    - instrument: clarinet, speed: twice normal (60 bpm)
  - play note G for 1 beats
  - play note G for 1 beats
  - play note G for 0.67 beats
  - play note A for 0.33 beats
  - play note B for 1 beats
  - play note B for 0.67 beats
  - play note A for 0.33 beats
  - play note B for 0.67 beats
  - play note C for 0.33 beats
  - play note D for 1 beats

Two text boxes provide additional context:

- note the importance of using the "broadcast and wait" piece when performing the initialization
- "Row,"  
"row,"  
"row"  
"your"  
"boat"
- "Gent-"  
"ly"  
"down"  
"the"  
"stream"

*continued on next page*

# Row, Row, Row Your Boat

## Version 3: Separating Initialization (cont'd)

### (3b) Initialization ("Init") Script

The image shows a Scratch code editor with two scripts. The first script, titled "Initialize Note Values", starts with a "when I receive" trigger, followed by a "hide" block and nine "set" blocks for notes G, A, B, C, D, E, F#, and G' with values 55, 57, 59, 60, 62, 64, 66, and 67 respectively. The second script, titled "Play G Major Scale", starts with a "when I receive" trigger, followed by a "broadcast Initialize Note Values and wait" block, and then nine "play note" blocks for notes G, A, B, C, D, E, F#, and G' each for 0.5 beats.

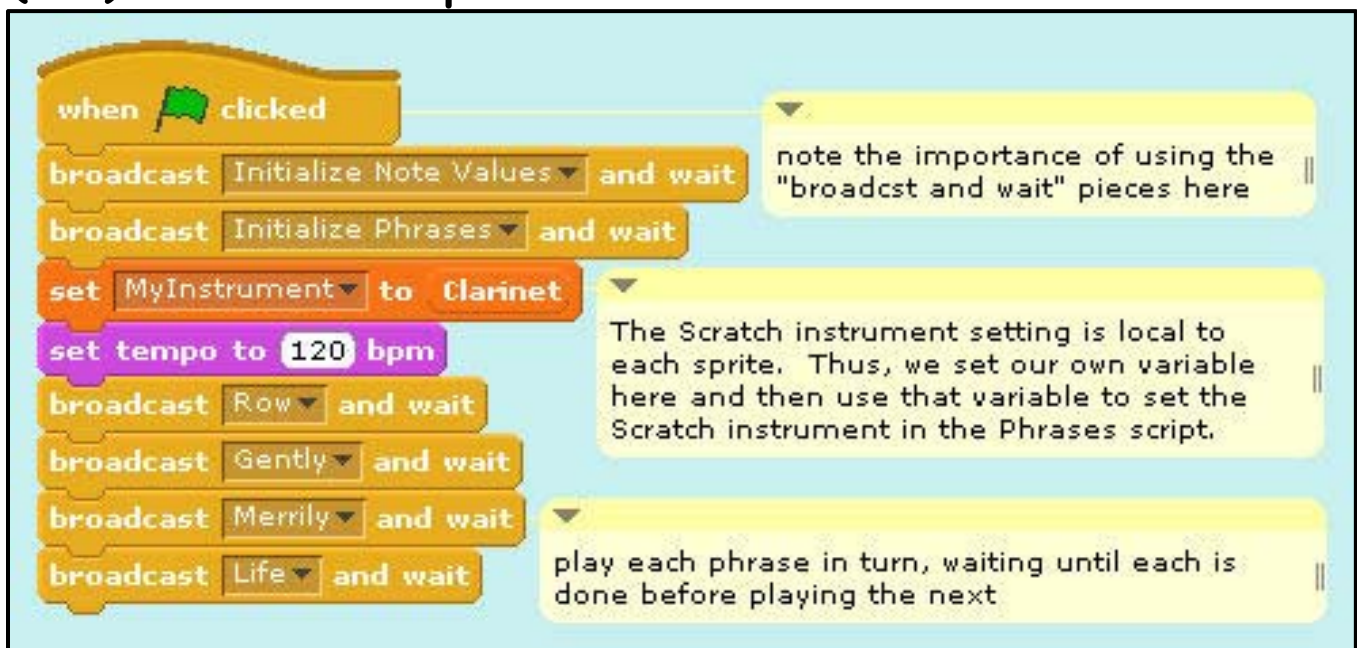
*end of Version 3*



## Row, Row, Row Your Boat Version 4: Separating Phrases

### Three Scripts

#### (4a) Main Script



The image shows a Scratch script for the 'Main Script'. The script starts with a 'when green flag clicked' block, followed by four 'broadcast and wait' blocks for 'Initialize Note Values', 'Initialize Phrases', 'Row', 'Gently', 'Merrily', and 'Life'. A 'set MyInstrument to Clarinet' block is placed after the first 'broadcast and wait' block. A 'set tempo to 120 bpm' block is placed after the 'set MyInstrument' block. The script ends with the 'Life' broadcast block. Three yellow callout boxes provide additional information: the first points to the first 'broadcast and wait' block, the second points to the 'set MyInstrument' block, and the third points to the 'Life' broadcast block.

```
when green flag clicked
broadcast Initialize Note Values and wait
broadcast Initialize Phrases and wait
set MyInstrument to Clarinet
set tempo to 120 bpm
broadcast Row and wait
broadcast Gently and wait
broadcast Merrily and wait
broadcast Life and wait
```

note the importance of using the "broadcast and wait" pieces here

The Scratch instrument setting is local to each sprite. Thus, we set our own variable here and then use that variable to set the Scratch instrument in the Phrases script.

play each phrase in turn, waiting until each is done before playing the next

*continued on next page*

## Row, Row, Row Your Boat Version 4: Separating Phrases (cont'd)

### (4b) Initialization ("Init") Script



The image shows a Scratch script editor with two scripts. The first script, titled "when I receive Initialize Note Values", contains a "hide" block followed by ten "set" blocks for notes G, A, B, C, D, E, F#, G', and Clarinet, each with a numerical value. The second script, titled "when I receive Play G Major Scale", contains a "broadcast Initialize Note Values and wait" block followed by eight "play note" blocks for notes G, A, B, C, D, E, F#, and G', each for 0.5 beats. A yellow tooltip on the right says "Click the set of pieces below to test the variable values by hearing a G major scale". A yellow tooltip at the bottom says "newly added".

```

when I receive Initialize Note Values
  hide
  set G to 55
  set A to 57
  set B to 59
  set C to 60
  set D to 62
  set E to 64
  set F# to 66
  set G' to 67
  set Clarinet to 72

when I receive Play G Major Scale
  broadcast Initialize Note Values and wait
  play note G for 0.5 beats
  play note A for 0.5 beats
  play note B for 0.5 beats
  play note C for 0.5 beats
  play note D for 0.5 beats
  play note E for 0.5 beats
  play note F# for 0.5 beats
  play note G' for 0.5 beats
  
```

*continued on next page*

## Row, Row, Row Your Boat Version 4: Separating Phrases (cont'd)

### (4c) Phrases Script

Note that the instrument value is local to a sprite, so it must be set (or reset) here.

**when I receive Initialize Phrases**

- set instrument to My Instrument
- hide

**when I receive Row**

- play note G for 1 beats
- play note G for 1 beats
- play note G for 0.67 beats
- play note A for 0.33 beats
- play note B for 1 beats

**when I receive Gently**

- play note B for 0.67 beats
- play note A for 0.33 beats
- play note B for 0.67 beats
- play note C for 0.33 beats
- play note D for 2 beats

**when I receive Merrily**

- play note G' for 0.33 beats
- play note G' for 0.33 beats
- play note G' for 0.34 beats
- play note D for 0.33 beats
- play note D for 0.33 beats
- play note D for 0.34 beats
- play note B for 0.33 beats
- play note B for 0.33 beats
- play note B for 0.34 beats
- play note G for 0.33 beats
- play note G for 0.33 beats
- play note G for 0.34 beats

**when I receive Life**

- play note D for 0.67 beats
- play note C for 0.33 beats
- play note B for 0.67 beats
- play note A for 0.33 beats
- play note G for 2 beats

newly added

"Row,"  
"row,"  
"row"  
"your"  
"boat"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Gent-"  
"ly"  
"down"  
"the"  
"stream"

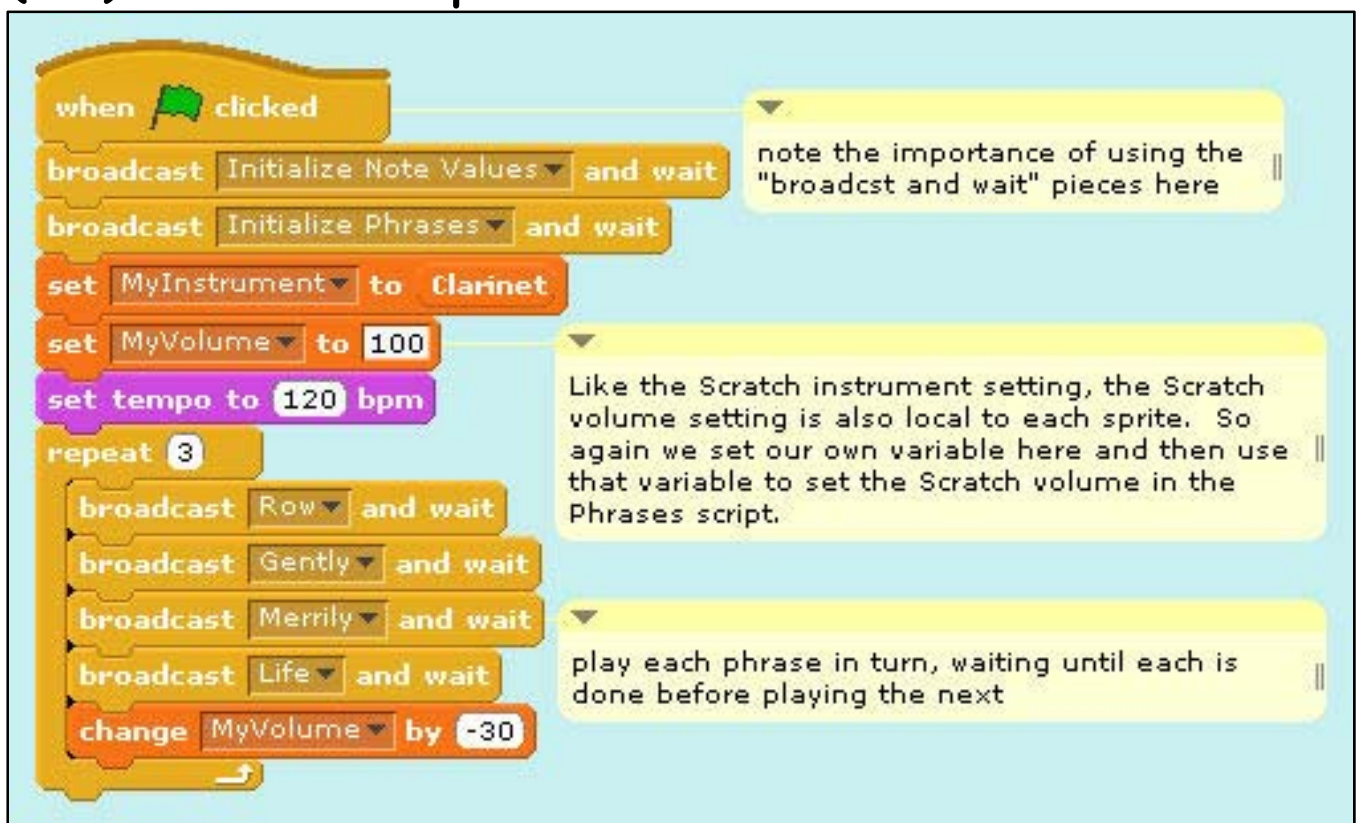
"Life"  
"is"  
"but"  
"a"  
"dream."

*end of Version 4*

## Row, Row, Row Your Boat Version 5: Looping and Fading

### Three Scripts

#### (5a) Main Script



The image shows a Scratch script for the 'Main Script' with three yellow callout boxes providing annotations:

- Callout 1:** points to the first two 'broadcast and wait' blocks. Text: "note the importance of using the 'broadcast and wait' pieces here".
- Callout 2:** points to the 'set MyVolume to 100' block. Text: "Like the Scratch instrument setting, the Scratch volume setting is also local to each sprite. So again we set our own variable here and then use that variable to set the Scratch volume in the Phrases script."
- Callout 3:** points to the 'repeat' loop containing four 'broadcast and wait' blocks and a 'change MyVolume by -30' block. Text: "play each phrase in turn, waiting until each is done before playing the next".

#### (5b) Initialization ("Init") Script

*(same as on page 34)*

*continued on next page*

## Row, Row, Row Your Boat Version 5: Looping and Fading (cont'd)

### (5c) Phrases Script

Note that the instrument value is local to a sprite, so it must be set (or reset) here.

**when I receive Initialize Phrases**

- set instrument to **MyInstrument**
- hide

**when I receive Row**

- set volume to **MyVolume** %
- play note **G** for **1** beats
- play note **G** for **1** beats
- play note **G** for **0.67** beats
- play note **A** for **0.33** beats
- play note **B** for **1** beats

**when I receive Gently**

- play note **B** for **0.67** beats
- play note **A** for **0.33** beats
- play note **B** for **0.67** beats
- play note **C** for **0.33** beats
- play note **D** for **2** beats

**when I receive Merrily**

- play note **G'** for **0.33** beats
- play note **G'** for **0.33** beats
- play note **G'** for **0.34** beats
- play note **D** for **0.33** beats
- play note **D** for **0.33** beats
- play note **D** for **0.34** beats
- play note **B** for **0.33** beats
- play note **B** for **0.33** beats
- play note **B** for **0.34** beats
- play note **G** for **0.33** beats
- play note **G** for **0.33** beats
- play note **G** for **0.34** beats

**when I receive Life**

- play note **D** for **0.67** beats
- play note **C** for **0.33** beats
- play note **B** for **0.67** beats
- play note **A** for **0.33** beats
- play note **G** for **2** beats

newly added

"Row,"  
"row,"  
"row"  
"your"  
"boat"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Gent-"  
"ly"  
"down"  
"the"  
"stream"

"Life"  
"is"  
"but"  
"a"  
"dream."

*end of Version 5*

# Row, Row, Row Your Boat

## Version 6: Playing a Round with One Instrument

### Three Scripts

#### (6a) Main Script

The image shows three Scratch scripts for the 'Main Script' of the 'Row, Row, Row Your Boat' project. The first script is triggered by a 'when green flag clicked' event and includes the following blocks: 'broadcast Initialize Note Values and wait', 'broadcast Initialize Phrases and wait', 'set MyInstrument to Clarinet', 'set MyVolume to 100', 'set NoOfTimesToPlay to 2', 'set tempo to 120 bpm', and 'broadcast Part1'. A yellow callout box notes the importance of using 'broadcast and wait' pieces here. The second script is triggered by 'when I receive Part1' and includes: 'set Counter to 1', a 'repeat until Counter > NoOfTimesToPlay' loop containing 'broadcast Row and wait', an 'if Counter = 1' block containing 'broadcast Part2', and three 'broadcast' blocks for 'Gently', 'Merrily', and 'Life', followed by 'change Counter by 1'. The third script is triggered by 'when I receive Part2' and includes a 'repeat NoOfTimesToPlay' loop containing four 'broadcast' blocks for 'Row', 'Gently', 'Merrily', and 'Life'.

Row, Row, Row Your Boat  
Version 6: Playing a Round with  
One Instrument (cont'd)

(6b) Initialization ("Init") Script



The image shows a Scratch script with two main sections. The first section is triggered by a 'when I receive' event labeled 'Initialize Note Values'. It contains a 'hide' block followed by ten 'set' blocks for notes G, A, B, C, D, E, F#, G', and Clarinet, each assigned a specific frequency value. A 'newly added' tooltip is visible next to the Clarinet block. The second section is triggered by a 'when I receive' event labeled 'Play G Major Scale'. It contains a 'broadcast' block for 'Initialize Note Values' with an 'and wait' block, followed by eight 'play note' blocks for G, A, B, C, D, E, F#, and G', each with a duration of 0.5 beats. A yellow tooltip at the top right of the script area says 'Click the set of pieces below to test the variable values by hearing a G major scale'.

(6c) Phrases Script  
(same as on page 37)

*end of Version 6*

# Row, Row, Row Your Boat

## Version 7: Playing a Round with Two Instruments

### Five Scripts

#### (7a) Main Script

The image shows a Scratch script for a 'Main Script' with two main sections. The first section is triggered by a 'when green flag clicked' event. It contains several 'broadcast and wait' blocks for 'Initialize Note Values', 'Initialize Phrases', and 'Initialize Phrases 2'. It also sets 'MyInstrument' to 'Clarinet', 'MyInstrument2' to 'Trumpet', 'MyVolume' to 100, and 'NoOfTimesToPlay' to 2. The tempo is set to 120 bpm, and 'Part1' is broadcast. The second section is triggered by 'when I receive Part1'. It sets a 'Counter' to 1 and enters a 'repeat until' loop where 'Counter' is greater than 'NoOfTimesToPlay'. Inside the loop, it broadcasts 'Row', and if the counter is 1, it broadcasts 'Part2'. It then broadcasts 'Gently', 'Merrily', and 'Life' before changing the counter by 1.

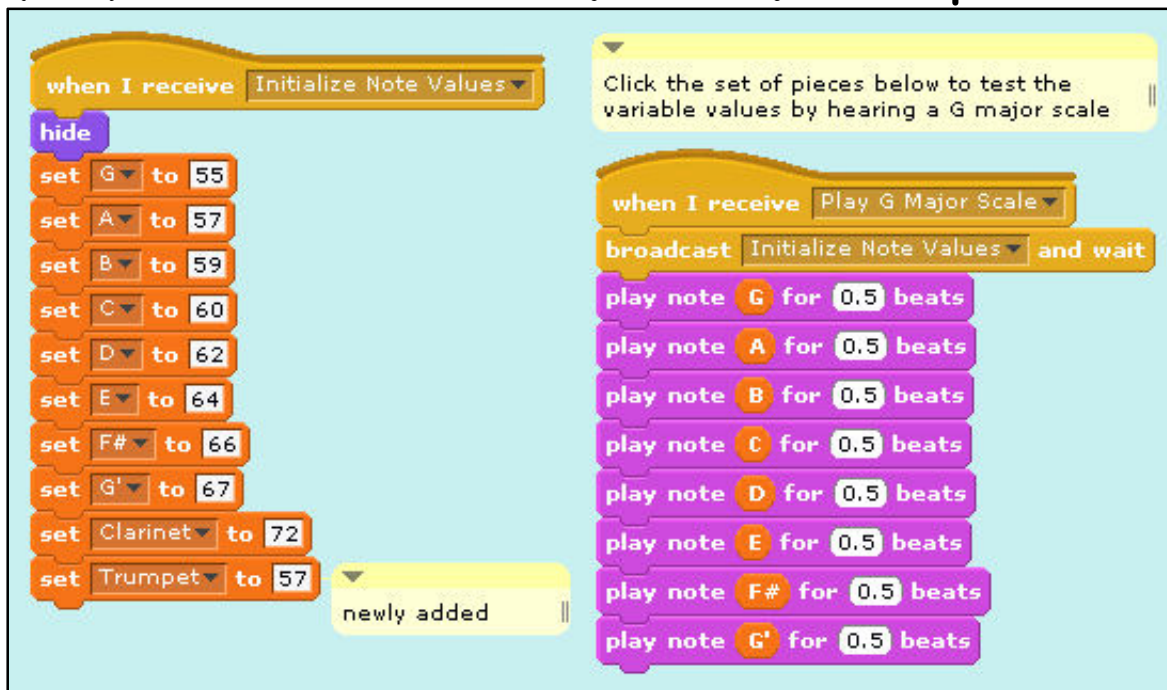
Annotations in yellow boxes provide additional context:

- note the importance of using the "broadcast and wait" pieces here
- Like the Scratch instrument setting, the Scratch volume setting is also local to each sprite. So again we set our own variable here and then use that variable to set the Scratch volume in the Phrases script.
- Part 2 had to be moved to another sprite so that it could be played with another instrument.



## Row, Row, Row Your Boat Version 7: Playing a Round with Two Instruments (cont'd)

### (7b) Initialization ("Init") Script



The script is divided into two parts. The first part, triggered by 'when I receive Initialize Note Values', includes a 'hide' block followed by ten 'set' blocks for variables: G (55), A (57), B (59), C (60), D (62), E (64), F# (66), G' (67), Clarinet (72), and Trumpet (57). A 'newly added' tooltip is visible next to the Trumpet block. The second part, triggered by 'when I receive Play G Major Scale', starts with a 'broadcast Initialize Note Values and wait' block, followed by eight 'play note' blocks for G, A, B, C, D, E, F#, and G', each for 0.5 beats.

### (7c) Phrases Script (same as on page 37)

### (7d) Part2 Script →

*continued on next page*



The script is triggered by 'when I receive Part2'. It starts with a 'hide' block, followed by 'set instrument to MyInstrument2'. A 'repeat' block with 'NoOfTimesToPlay' iterations contains four 'broadcast and wait' blocks for 'Row2', 'Gently2', 'Merrily2', and 'Life2'.

## Row, Row, Row Your Boat Version 7: Playing a Round with Two Instruments (cont'd)

### (7e) Instrument2 ("Instru2") Script

Note that the instrument value is local to a sprite, so it must be set (or reset) here.

```

when I receive Initialize Phrases 2
  set instrument to MyInstrument2
  hide

when I receive Row2
  set volume to MyVolume %
  play note G for 1 beats
  play note G for 1 beats
  play note G for 0.67 beats
  play note A for 0.33 beats
  play note B for 1 beats

when I receive Gently2
  play note B for 0.67 beats
  play note A for 0.33 beats
  play note B for 0.67 beats
  play note C for 0.33 beats
  play note D for 2 beats

when I receive Merrily2
  play note G for 0.33 beats
  play note G for 0.33 beats
  play note G for 0.34 beats
  play note D for 0.33 beats
  play note D for 0.33 beats
  play note D for 0.34 beats
  play note B for 0.33 beats
  play note B for 0.33 beats
  play note B for 0.34 beats
  play note G for 0.33 beats
  play note G for 0.33 beats
  play note G for 0.34 beats

when I receive Life2
  play note D for 0.67 beats
  play note C for 0.33 beats
  play note B for 0.67 beats
  play note A for 0.33 beats
  play note G for 2 beats
  
```

newly added

"Row,"  
"row,"  
"row"  
"your"  
"boat"

"Gent-"  
"ly"  
"down"  
"the"  
"stream"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Mer-"  
"i"  
"ly"

"Life"  
"is"  
"but"  
"a"  
"dream."

*end of Version 7*

## Row, Row, Row Your Boat

### Version 8: Storing Notes and Rhythms in Lists

### Output Window



The image shows a Scratch project window with an illustration of a boy rowing a boat. The boat has the text "Row, Row, Row Your Boat" written on its side. The background features green hills and blue water with several blue fish. The artist's signature "Deborah Cavanaugh" is visible at the bottom of the illustration.

Overlaid on the right side of the window is the "Output Window" showing two lists:

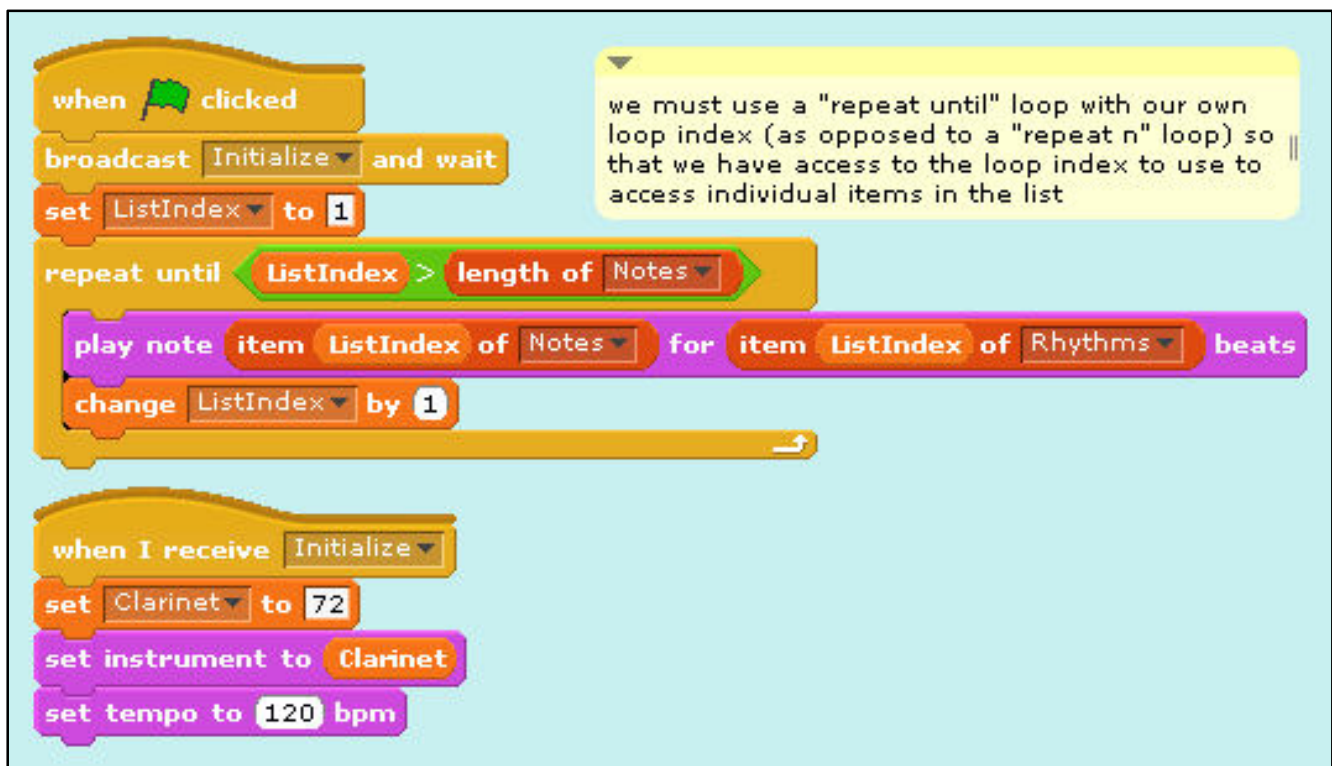
- ListIndex:** 0
- Notes:** A list of 15 notes: 55, 55, 55, 57, 59, 59, 57, 59, 60, 62, 67, 67, 67, 62, 62.
- Rhythms:** A list of 15 rhythms: 1, 1, 0.67, 0.33, 1, 0.67, 0.33, 0.67, 0.33, 2, 0.33, 0.33, 0.34, 0.33, 0.33.

Both lists have a scroll bar and a "+ length: 27" indicator at the bottom.

*continued on next page*

## Row, Row, Row Your Boat Version 8: Storing Notes and Rhythms in Lists (cont'd)

### Single Script



when clicked

broadcast Initialize and wait

set ListIndex to 1

repeat until ListIndex > length of Notes

play note item ListIndex of Notes for item ListIndex of Rhythms beats

change ListIndex by 1

when I receive Initialize

set Clarinet to 72

set instrument to Clarinet

set tempo to 120 bpm

we must use a "repeat until" loop with our own loop index (as opposed to a "repeat n" loop) so that we have access to the loop index to use to access individual items in the list

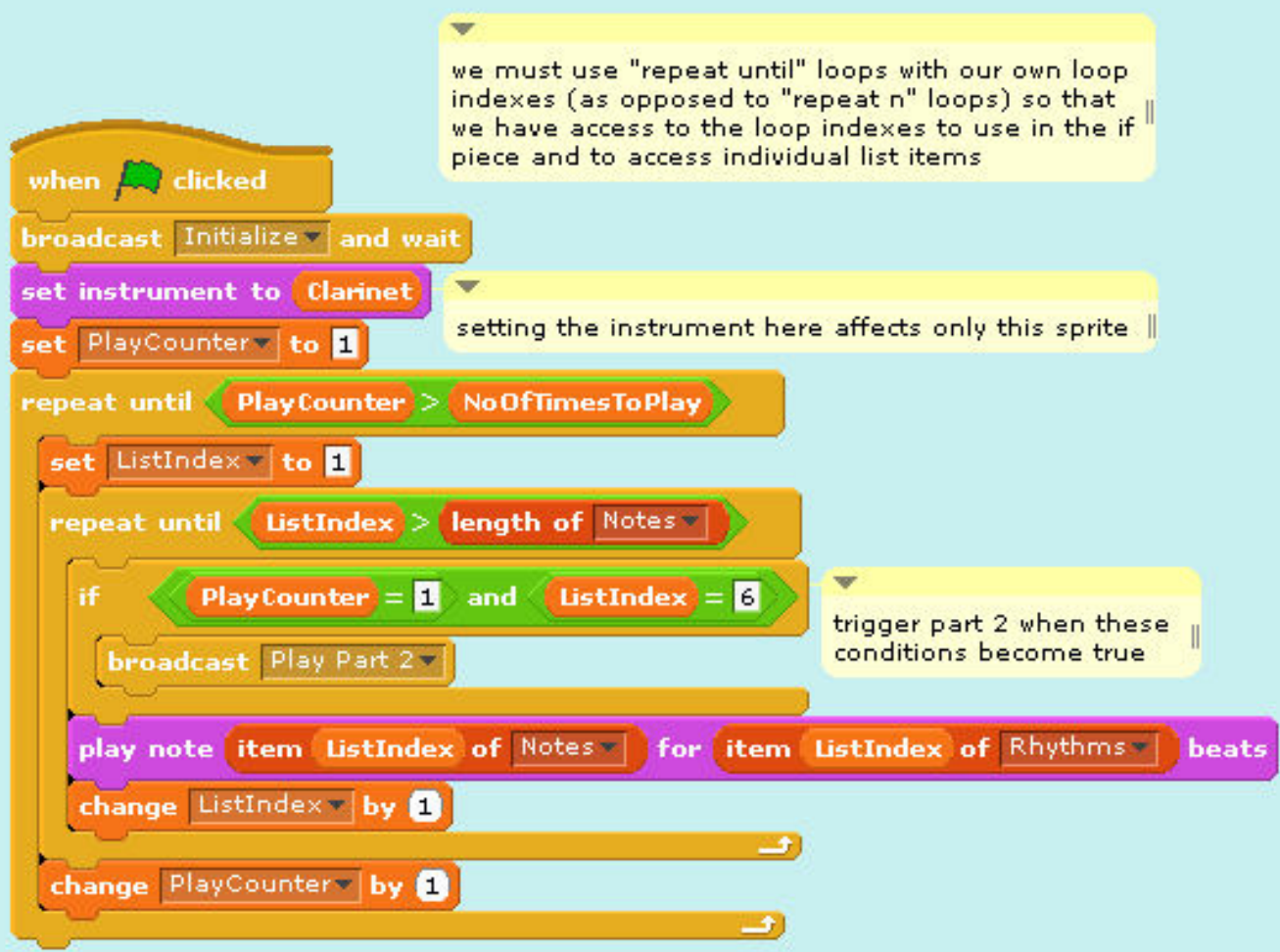
*end of Version 8*

## Row, Row, Row Your Boat

# Version 9: Playing a Round Using Lists

## Three Scripts

### (9a) Main Script



The script is as follows:

```

when green flag clicked
  broadcast Initialize and wait
  set instrument to Clarinet
  set PlayCounter to 1
  repeat until PlayCounter > NoOfTimesToPlay
    set ListIndex to 1
    repeat until ListIndex > length of Notes
      if PlayCounter = 1 and ListIndex = 6
        broadcast Play Part 2
      play note item ListIndex of Notes for item ListIndex of Rhythms beats
      change ListIndex by 1
    change PlayCounter by 1
  
```

Annotations:

- Repeat until loop:** we must use "repeat until" loops with our own loop indexes (as opposed to "repeat n" loops) so that we have access to the loop indexes to use in the if piece and to access individual list items
- Instrument setting:** setting the instrument here affects only this sprite
- Condition:** trigger part 2 when these conditions become true

*continued on next page*

## Row, Row, Row Your Boat Version 9: Playing a Round Using Lists (cont'd)

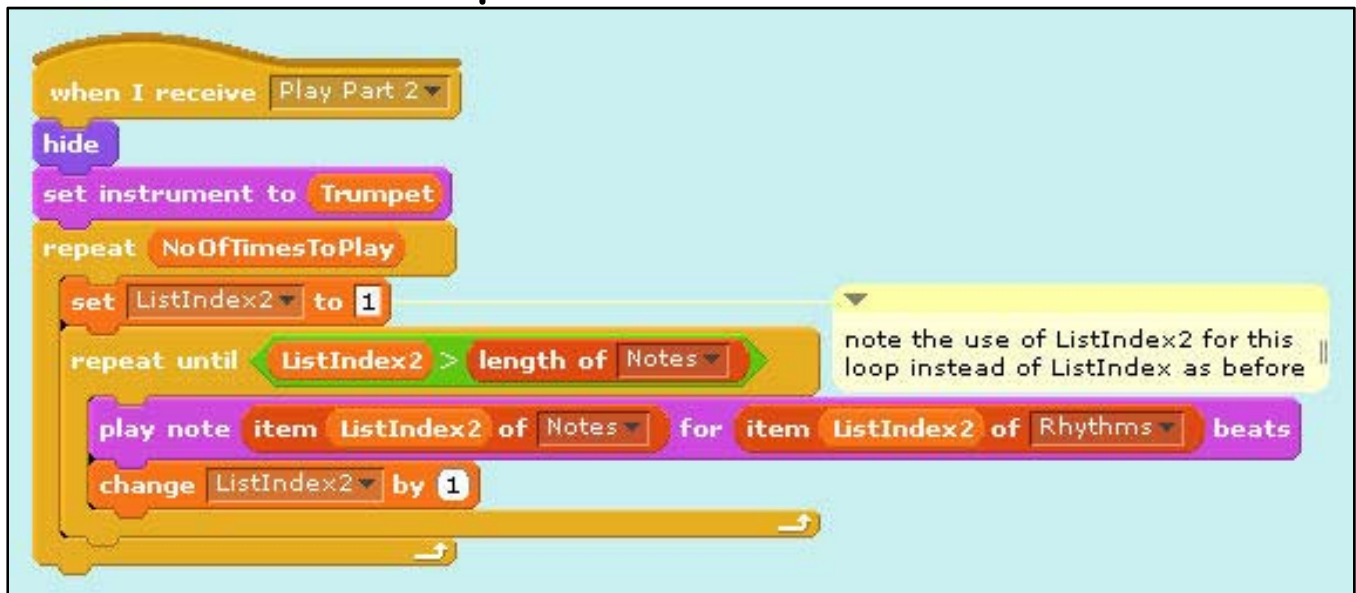
### (9b) Initialization ("Init") Script



```

when I receive Initialize
hide
set Clarinet to 72
set Trumpet to 57
set NoOfTimesToPlay to 2
set tempo to 120 bpm
    
```

### (9c) Part2 Script



```

when I receive Play Part 2
hide
set instrument to Trumpet
repeat NoOfTimesToPlay
  set ListIndex2 to 1
  repeat until ListIndex2 > length of Notes
    play note item ListIndex2 of Notes for item ListIndex2 of Rhythms beats
    change ListIndex2 by 1
    
```

note the use of ListIndex2 for this loop instead of ListIndex as before

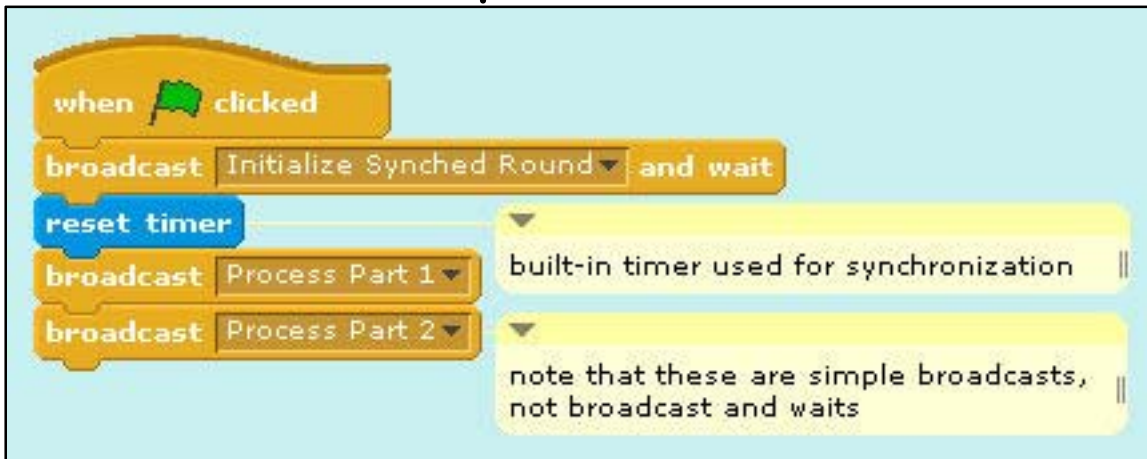
*end of Version 9*

## Row, Row, Row Your Boat

# Version 10: Synchronizing Play from Lists

## Four Scripts

### (10a) Main Script



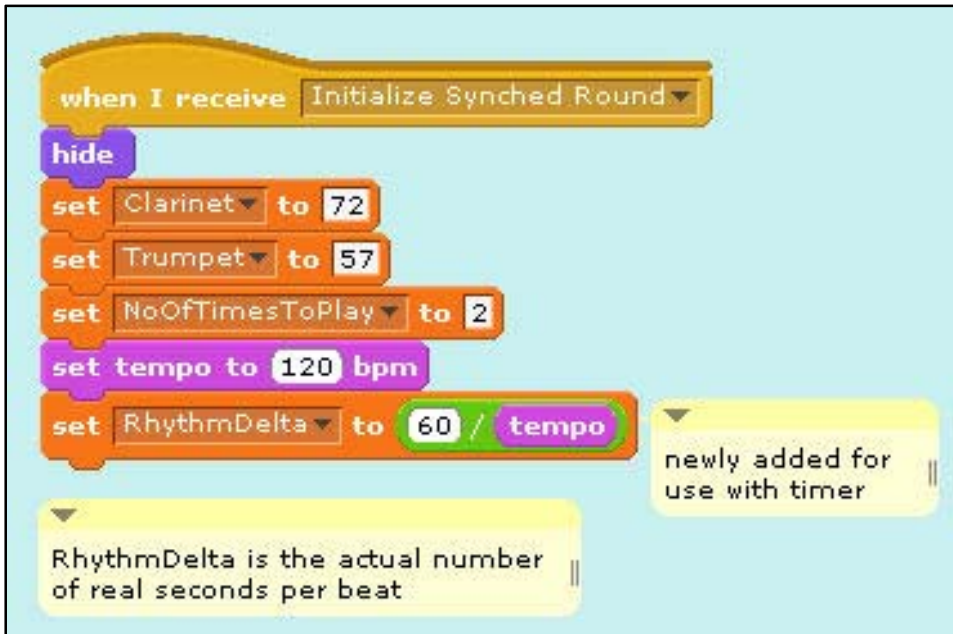
```

when green flag clicked
  broadcast Initialize Synched Round and wait
  reset timer
  broadcast Process Part 1
  broadcast Process Part 2
  
```

built-in timer used for synchronization

note that these are simple broadcasts, not broadcast and waits

### (10b) Initialization (“Init”) Script



```

when I receive Initialize Synched Round
  hide
  set Clarinet to 72
  set Trumpet to 57
  set NoOfTimesToPlay to 2
  set tempo to 120 bpm
  set RhythmDelta to 60 / tempo
  
```

newly added for use with timer

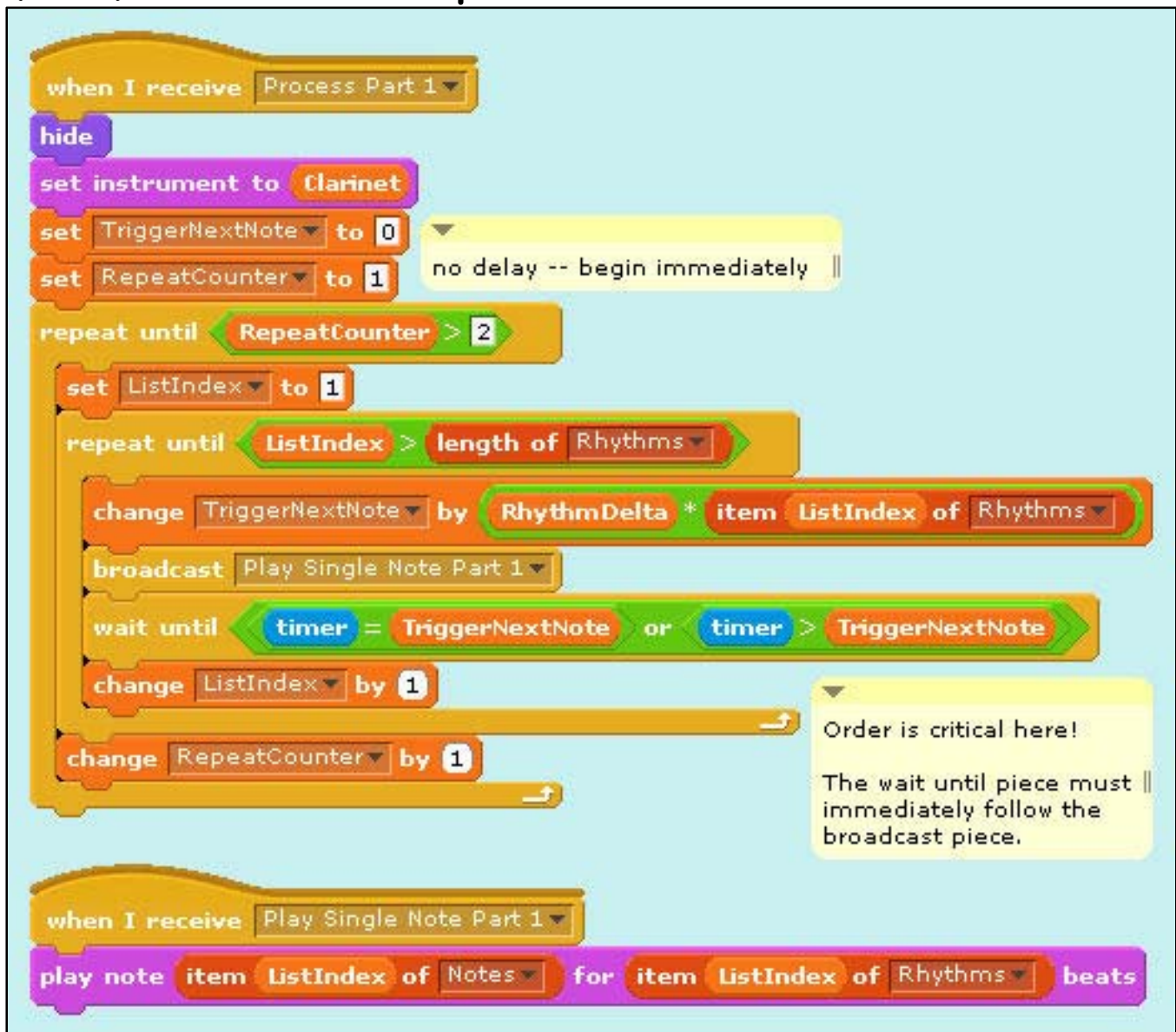
RhythmDelta is the actual number of real seconds per beat

*continued on  
next page*

## Row, Row, Row Your Boat

# Version 10: Synchronizing Play from Lists (cont'd)

### (10c) Part 1 Script



The script is contained within a single script area and consists of the following blocks:

- when I receive** Process Part 1
- hide**
- set instrument to** Clarinet
- set** TriggerNextNote to 0
- set** RepeatCounter to 1
- repeat until** RepeatCounter > 2
  - set** ListIndex to 1
  - repeat until** ListIndex > length of Rhythms
    - change** TriggerNextNote by RhythmDelta \* item ListIndex of Rhythms
    - broadcast** Play Single Note Part 1
    - wait until** timer = TriggerNextNote or timer > TriggerNextNote
    - change** ListIndex by 1
    - change** RepeatCounter by 1
- when I receive** Play Single Note Part 1
- play note** item ListIndex of Notes for item ListIndex of Rhythms beats

**Annotations:**

- A yellow callout box next to the 'set TriggerNextNote to 0' block contains the text: "no delay -- begin immediately ||"
- A yellow callout box next to the 'wait until' block contains the text: "Order is critical here! The wait until piece must immediately follow the broadcast piece."

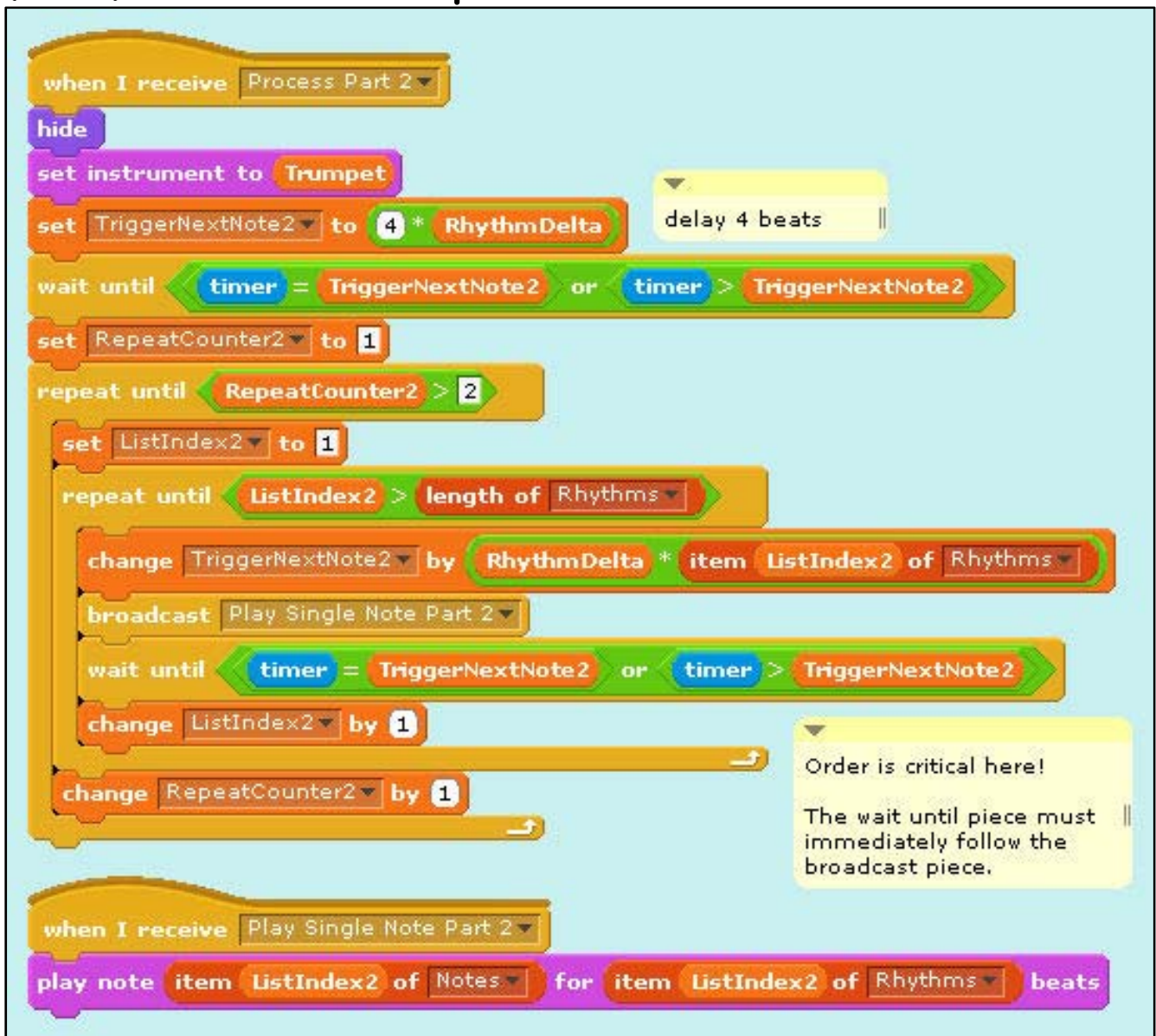
*continued on next page*



Row, Row, Row Your Boat

Version 10: Synchronizing Play from Lists  
(cont'd)

(10d) Part 2 Script



The script is contained within a light blue rectangular area. It begins with a 'when I receive' block for 'Process Part 2'. This is followed by a 'hide' block, a 'set instrument to Trumpet' block, and a 'set TriggerNextNote2 to 4 \* RhythmDelta' block. A 'delay 4 beats' block is attached to the right of the 'set TriggerNextNote2' block. Next is a 'wait until' block with the condition 'timer = TriggerNextNote2 or timer > TriggerNextNote2'. This is followed by a 'set RepeatCounter2 to 1' block. A 'repeat until RepeatCounter2 > 2' block contains a 'set ListIndex2 to 1' block, a 'repeat until ListIndex2 > length of Rhythms' block, and a loop of three blocks: 'change TriggerNextNote2 by RhythmDelta \* item ListIndex2 of Rhythms', 'broadcast Play Single Note Part 2', and 'wait until timer = TriggerNextNote2 or timer > TriggerNextNote2'. This loop is followed by 'change ListIndex2 by 1' and 'change RepeatCounter2 by 1' blocks. A second 'when I receive' block for 'Play Single Note Part 2' is followed by a 'play note' block: 'item ListIndex2 of Notes for item ListIndex2 of Rhythms beats'. A yellow callout box on the right side of the script contains the text: 'Order is critical here! The wait until piece must immediately follow the broadcast piece.'

*end of Version 10*

# SCRATCH



## Extending the Examples

- 1. Use a variable to set the tempo.**
  - Add a slider to the variable so that you can change the tempo in real time.
  - Find all the places you need to use the variable to reset the tempo when you change it in real time.
  - Which version of playing the round best stays synchronized when you change the tempo?
- 2. Transpose the melody to another key.**
  - Create a variable to hold a pitch offset.
  - Find all the places you need to use that variable to play the melody in the new key.
- 3. Increase the number of times that the round repeats.**
  - Do the parts stay in synch?
- 4. Increase the number of parts that play simultaneously.** (Be sure to set Turbo Speed to do this!)
  - When should each part "come in"?
  - How much should the first beat of each part be offset?

## Extending the Examples (cont'd)

5. **Play the melody backwards.**
  - Can you play multiple parts backwards, too?
6. **Increase the number of times that the round repeats.**
  - Do the parts stay in synch?
7. **Increase the number of parts that play simultaneously.** (Be sure to set Turbo Speed before you try this!)
  - When should each part "come in"?
  - How much should the first beat of each part be offset?
8. **Make a round using the G-major scale.**
  - Put the note values for a G-major scale into a list. See page 32 for code that initializes and plays a G-major scale, but remember that you must use the integer values, not the variable names, to play notes from a list.
  - Start Part 2 when Part 1 plays its third note (B, MIDI note #59).
  - Add Part 3, starting when Part 1 plays its fifth note (D, #62).

## Extending the Examples (cont'd)

9. **Play random notes in the G-major scale.**
  - Start with the list created for the previous exercise.
  - Use the “pick random” piece in the Operators group to pick a random note from the list.
  - Play each note for 0.25, 0.50, 0.75, or 1.00 beats, also selected randomly.
  - Does the result sound musical?
  
10. **Create a program that can play any major scale given any starting note.**
  - Store the starting note in a variable.
  - For a major scale, the number of half-tones between each note is:  
2, 2, 1, 2, 2, 2, 1
  - Another way to think about this is:  
Do + 2 → Re + 2 → Mi + 1 → Fa + 2 →  
Sol + 2 → La + 2 → Ti + 1 → Do
  - Create a list containing the changes between the notes, and then use a loop to process the list and play the scale.

## Extending the Examples (cont'd)

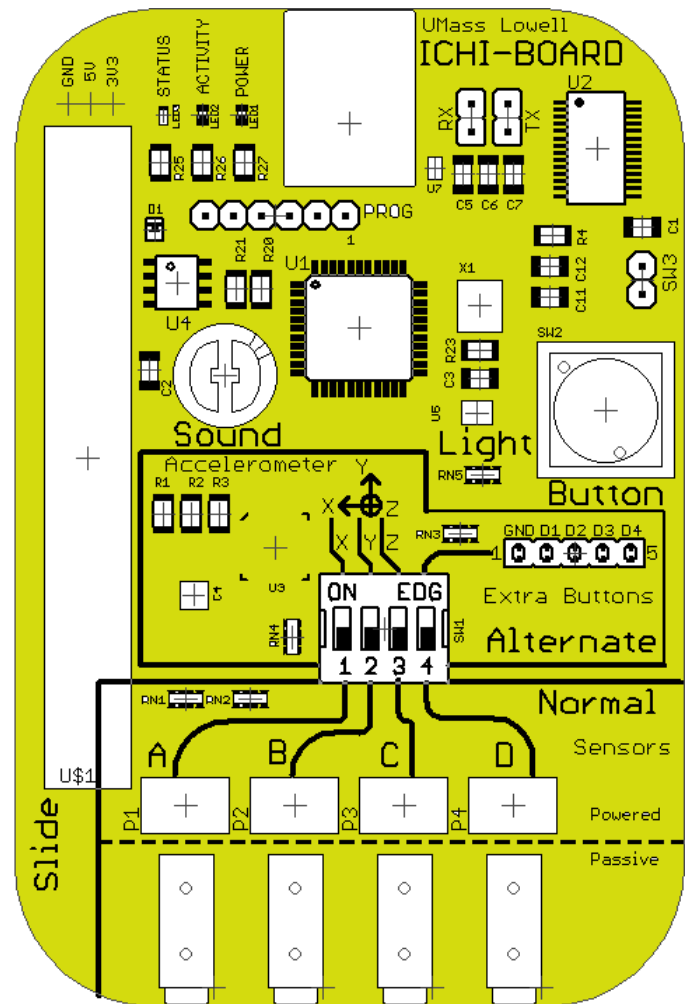
11. Create a program that can play any harmonic minor scale given any starting note.
  - For a harmonic minor scale, the number of half-tones between each note is:
 

2, 1, 2, 2, 1, 3, 1
  - Create a new list containing these changes, but use the same loop that you created for the previous exercise to play this scale.
  
12. Create a program to play a major chord.
  - A major chord is the 1st, 3rd, and 5th notes of the scale, usually complemented by the octave above the 1st note. Thus, a G-major scale has notes G (#55), B (#59), D (#62), and G' (#67).
  - Another way to think about this is to compute the half-tone difference from the starting note: 0, 4, 7, 12.
  - Set a starting note and then use a "broadcast" to play the four notes simultaneously.

## The IchiBoard

### Board Layout

(courtesy of Mark Sherman,  
UMass Lowell  
Computer Science  
Engaging Computing Group)



### Scratch Code for an IchiBoard Musical Instrument

(courtesy of Alex Ruthmann)

```


when clicked
  forever
    if button pressed
      set Note to round (slider sensor value mod 2)
      play note Note + (slider sensor value) for 0.01 beats
  
```

# SCRATCH





# The PicoBoard



**CRICKET**

With PicoCrickets, you can create musical sculptures, interactive jewelry, dancing creatures, and other playful inventions.

**MAKE YOUR CREATIONS COME TO LIFE!**

What Is It?
Order
Ideas
Support
Educators
About PICO
News

## PICOBOARD SET UP

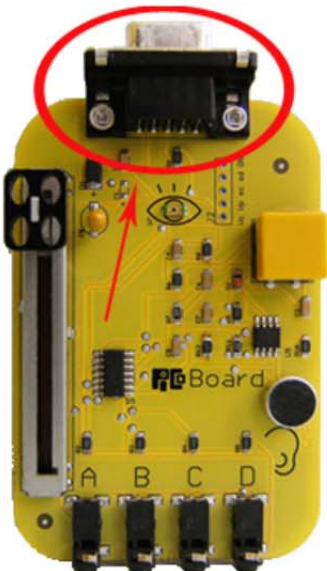
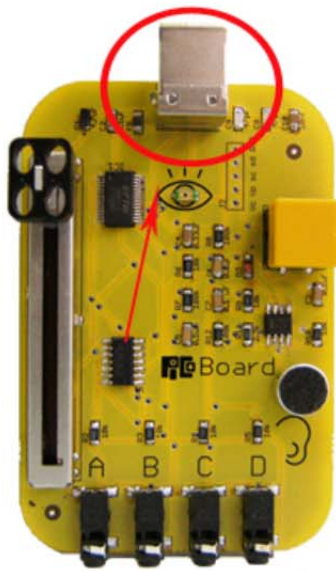
**Which version of the PicoBoard do you have?**

**Serial**

Does your PicoBoard connect to your computer using a Serial to USB Cable?  
All PicoBoards purchased before June 2009 are Serial PicoBoards.

**USB**

Does your PicoBoard connect to your computer using a USB to USB Cable?  
All PicoBoards purchased after June 2009 are USB PicoBoards.

[info@playfulinvention.com](mailto:info@playfulinvention.com)  
© 2006-2010 The Playful Invention Company

**CREATE · PROGRAM · PLAY**

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See <http://scratch.mit.edu>. Performamatics is an interdisciplinary project at UMass Lowell funded by the National Science Foundation. See <http://performamatics.org>.

**57**

## PicoBoard Serial to USB Setup

### Windows Instructions

**Windows XP (and older) users:** Download the PicoBoard [Windows Driver \(1471 KB\)](#)

**Windows Vista/7 users** should just plug the USB to Serial cable in their computer and let the operating system choose the correct driver. Then skip to step #3 below.

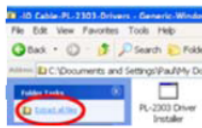
If you do not have internet access on your computer, download the PicoBoard [Windows Driver \(1471 KB\)](#).

### Mac Instructions

[Mac OS X Driver \(61 KB\)](#)



**1** First, open the file that you just downloaded. Click the link that says "Extract all files".



**1** First, open the file that you just downloaded by clicking the magnifying glass in the Download panel.



**2** Then double-click on the file that you just extracted, and follow the on-screen instructions.

**2** Then double-click on the file that you just extracted, and follow the on-screen instructions.



### Both Operating Systems:

**3** Connect the USB part of your USB-Serial Cable to a USB port on your computer.



**4** Connect the serial part of your USB-Serial Cable to the serial port on the PicoBoard.



**5** Read [Getting Started with PicoBoards](#) to start making projects in [Scratch](#) with your PicoBoard.



# PicoBoard USB to USB Setup

## Windows Instructions

**Windows XP users:** Download the PicoBoard [Windows Driver \(1.71 MB\)](#)

**Windows Vista/Windows 7 users** should just plug the USB cable in their computer and let the operating system choose the correct driver. Then skip to step #3 below.

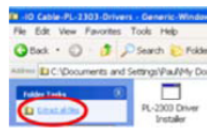
If you do not have internet access on your computer, download the PicoBoard [Windows Driver \(1.71 MB\)](#).

## Mac Instructions

[Mac OS X Driver \(419 KB\)](#)



**1** First, open the file that you just downloaded. Click the link that says "Extract all files".



**1** First, open the file that you just downloaded by clicking the magnifying glass in the Download panel.



**2** Then double-click on the file that you just extracted, and follow the on-screen instructions.

**2** Then double-click on the file that you just extracted, and follow the on-screen instructions.

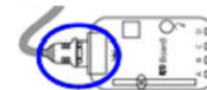


## Both Operating Systems:

**3** Connect the USB Cable to a USB port on your computer.



**4** Connect the other part of the USB Cable to the USB port of the PicoBoard.



**5** Read [Getting Started with PicoBoards](#) to start making projects in [Scratch](#) with your PicoBoard.

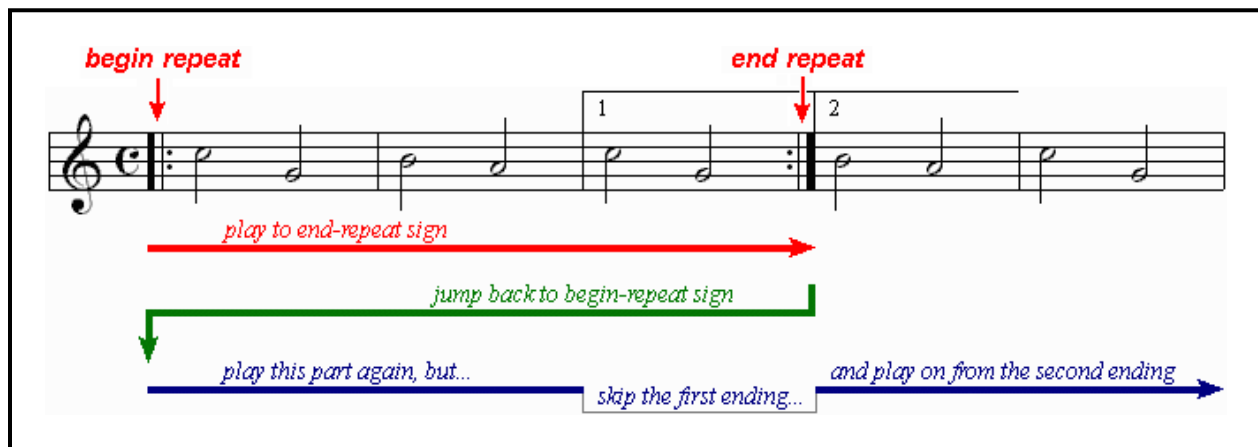


# SCRATCH



## Computing and Music: What Do They Have in Common?

Computing and music share deep structural similarities. For starters, both rely on notational symbol systems. Programming loops are typically delineated with opening and closing curly brackets { }, parentheses, or levels of indentation. Music loops are delineated with begin and end repeat signs { } or initiated by "D.S." (Italian: *dal segno*), which instructs musicians to "repeat back to the sign," typically designated as  $\text{\textcircled{S}}$ . As in programming, musical iteration can also make use of loop control variables. For example, Figure 1 shows a loop in which the music changes the second time through.



*Figure 1. Musical iteration with a loop control variable. [6]*

Both computing and music have logic and flow. Figure 2 shows the logic one student saw in The Beatles' All You Need Is Love. If one were to turn this flowchart into a computer program, it would not only contains loops, but if and switch statements as well.

One can also go the other way, converting musical concepts into computer programs. For example, the Scratch [2, 3] program in Figure 3a plays Jimmy Page's famous guitar riff from Led Zeppelin's Kashmir. This code works properly, but consider the many computational thinking (CT) concepts learned by transforming the code in Figure 3a to 3b, which plays exactly the same riff.

## Computing and Music (cont'd)

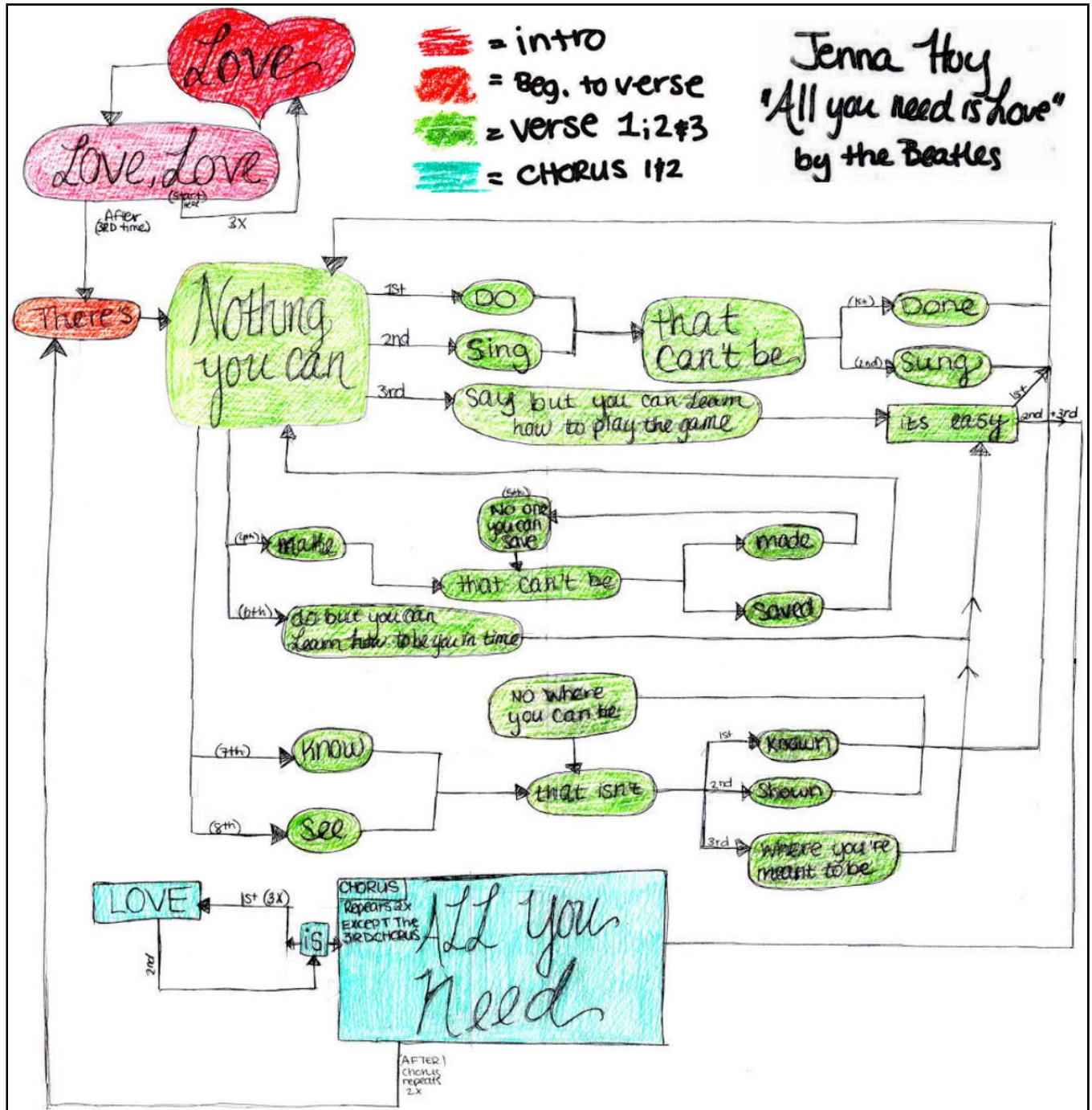
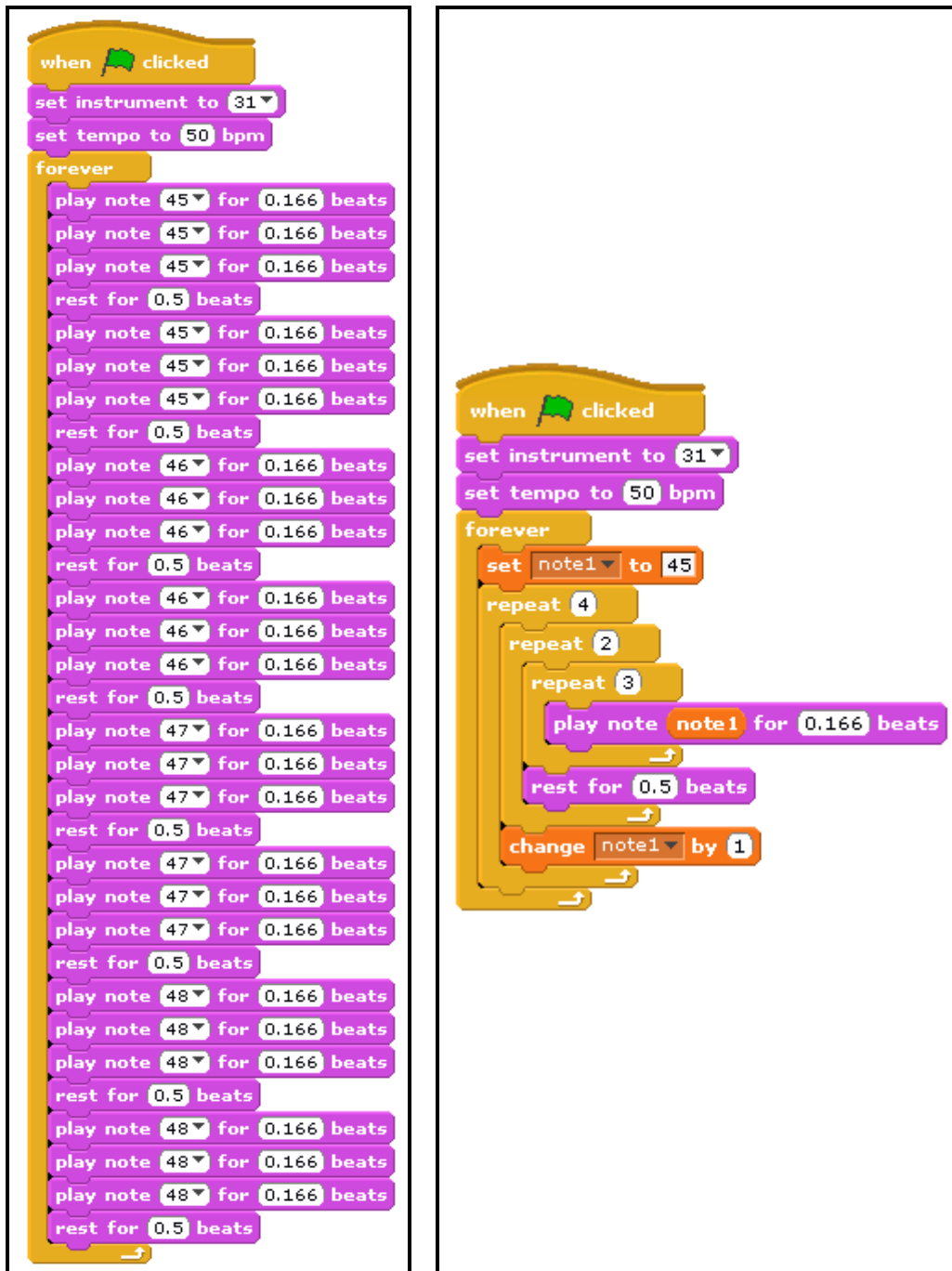


Figure 2. A song flowchart. [1]

## Computing and Music (cont'd)



3a

3b

Figure 3. Two versions of Jimmy Page's *Kashmir* riff programmed in Scratch. [4]

## Computing and Music (cont'd)

List and array data structures can be used to represent pitches and durations. Figure 4 shows an array (or indexed list) of MIDI note values paired with an array of note durations (in fractions of beats) that plays part of Row, Row, Row Your Boat. Using such structures, one can explore synchronization when the values are read by multiple threads with entrances staggered in time, resulting in the performance of a canon (or round).

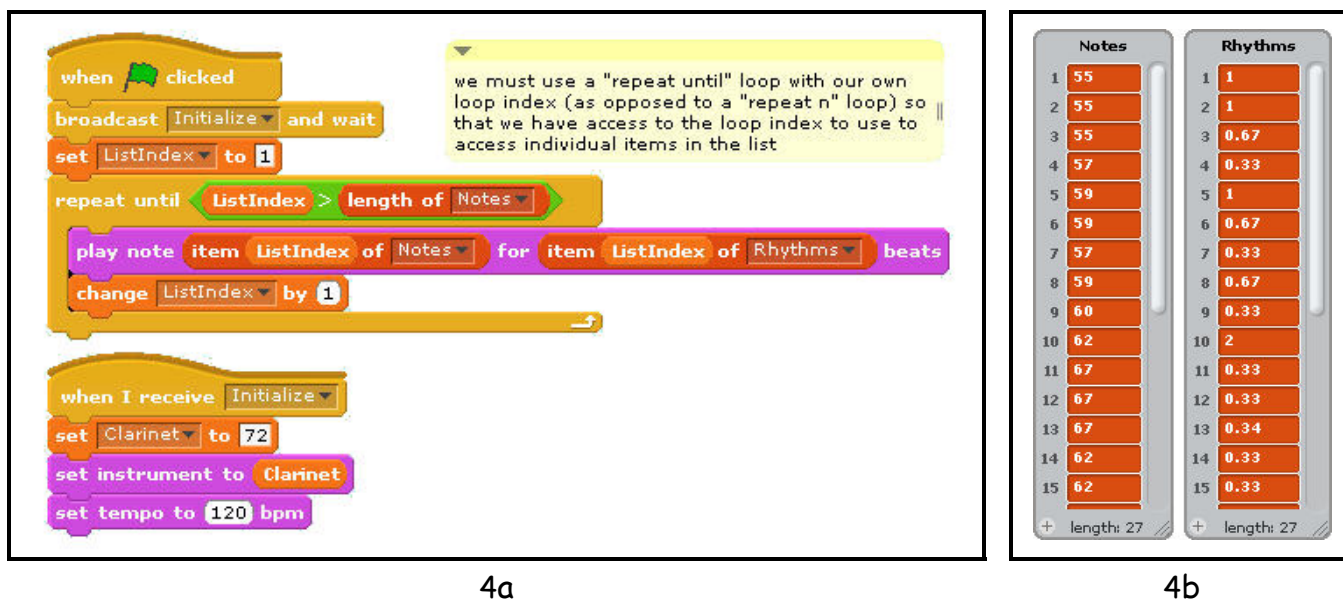


Figure 4. Processing Scratch lists of notes and rhythms for Row, Row, Row Your Boat. [5]

### References Cited

- [1] Hoy, J. (2010). *Song flowchart for The Beatles' "All You Need is Love."* Created for a course assignment in "Sound Thinking."
- [2] MIT Scratch Team (2009). *Scratch*. [scratch.mit.edu](http://scratch.mit.edu) accessed Dec. 21, 2009.
- [3] Resnick, M., Maloney, J., Monroyhernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). *Scratch Programming for All*. *Comm. of the ACM* 52(11):60-67.
- [4] Ruthmann, S.A. (2009). *Computational Zeppelin*. [scratch.mit.edu/projects/alexruthmann/736779](http://scratch.mit.edu/projects/alexruthmann/736779) accessed Jan. 5, 2010.
- [5] Ruthmann, S.A., & Heines, J.M. (2010). *Exploring Musical and Computational Thinking Through Musical Live Coding in Scratch*. Scratch@MIT. Cambridge, MA.
- [6] Smith, D.E. (1997). *Repeats, Second Endings, and Codas*. [www.scenicnewengland.net/uitar/notate/repeat.htm](http://www.scenicnewengland.net/uitar/notate/repeat.htm) accessed Dec. 25, 2009.



# Computer Science, Math, and Music: Concepts Covered in Scratch

## Computer Science

- statements
- sequential control flow
- iteration
- conditional execution
- arithmetic operators
- Boolean operators
- objects
- concurrency
- variables
- lists
- event handling
- user interaction
- optimization

## Math

- positive and negative numbers
- real numbers
- decimal notation
- built-in functions with inputs
- angles
- Cartesian coordinates
- trigonometric operators
- random numbers

## Music

- pitch
- rhythm (as duration)
- melodic fragments
- modes and scales
- polyphony
- synchronization
- harmony
- composing
- performing
- transposition
- balance and dynamics
- digital audio (as sound files)
- MIDI notes and timbres
- tempo
- form and structural analysis

# SCRATCH



## Additional Readings

Heines, J.M., Greher, G.R., Ruthmann, S.A., & Reilly, B. (2011). **Two Approaches to Interdisciplinary Computing+Music Courses**. *IEEE Computer* 44(12):25-32, December 2011.

<http://teaching.cs.uml.edu/~heines/academic/papers/2011ieee/ieee2011paper-v33-forWebsite.pdf>

The intersection of computing and music can enrich pedagogy in numerous ways, from low-level courses that use music to illustrate practical applications of computing concepts to high-level ones that use sophisticated computer algorithms to process audio signals. This paper explores the ground between these extremes by describing our experiences with two types of interdisciplinary courses. In the first, arts and computing students worked together to tackle a joint project even though they were taking independent courses. In the second, all students enrolled in the same course, but every class was taught by two professors: one from music and the other from computer science. This course was designed to teach computing and music together, rather than one in service to the other. This paper presents the philosophy and motivation behind these courses, describes some of the assignments students do in them, and shows examples of student work.

Ruthmann, S.A., Heines, J.M., Greher, G.R., Laidler, P., & Saulters, C. (2010). **Teaching Computational Thinking through Musical Live Coding in Scratch**. *41st ACM SIGCSE Technical Symposium on CS Education*. Milwaukee, WI, March 12, 2010.

<http://teaching.cs.uml.edu/~heines/academic/papers/2010sigcse/SoundThinking-SIGCSE-2010.pdf>

This paper discusses our ongoing experiences in developing an interdisciplinary general education course called Sound Thinking that is offered jointly by our Dept. of Computer Science and Dept. of Music. It focuses on the student outcomes we are trying to achieve and the projects we are using to help students realize those outcomes. It explains why we are moving from a web-based environment using HTML and JavaScript to Scratch and discusses the potential for Scratch's "musical live coding" capability to reinforce those concepts even more strongly.

## Additional Readings (cont'd)

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). **The Scratch Programming Language and Environment**. *ACM Transactions on Computing Education* 10(4). Article 16.

<http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf>

Scratch is a visual programming environment that allows users (primarily ages 8 to 16) to learn computer programming while working on personally meaningful projects such as animated stories and games. A key design goal of Scratch is to support self-directed learning through tinkering and collaboration with peers. This article explores how the Scratch programming language and environment support this goal.

Martin, F., Greher, G.R., Heines, J.M., Jeffers, J., Kim, H.J., Kuhn, S., Roehr, K., Selleck, N., Silka, L., and Yanco, H. (2009). **Joining Computing and the Arts at a Mid-Size University**. *2009 Conference of the Consortium for Computing Sciences in Colleges – Northeastern Region (CCSCNE 2009)*. Plattsburgh, NY, April 24, 2009.

<http://teaching.cs.uml.edu/~heines/academic/papers/2009ccscne/JoiningComputingAndArts.pdf>

This paper describes two NSF-funded collaborations among faculty members in the Computer Science, Art, Music, and English departments at a public university in the Northeast USA. Our goal has been to create undergraduate learning opportunities across the university, focusing on connecting computer science to creative and expressive domains. In past publications, we have focused on student learning outcomes. This paper reports on the motivations, opportunities, and challenges for the faculty members involved.

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

<http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf>

"Digital fluency" should mean designing, creating, and remixing, not just browsing, chatting, and interacting. In this article we discuss the design principles that guided our development of Scratch and our strategies for making programming accessible and engaging for everyone.

## Additional Readings (cont'd)

Heines, J.M., Greher, G.R., & Kuhn, S. (2009). **Music Performamatics: Interdisciplinary Interaction.** *40th ACM SIGCSE Technical Symposium on CS Education.* Chattanooga, TN, March 7, 2009.

<http://teaching.cs.uml.edu/~heines/academic/papers/2009sigcse/fp119-heines.pdf>

This paper describes how a graphical user interface (GUI) programming course offered by the Dept. of Computer Science (CS) was paired with a general teaching methods course offered by the Dept. of Music in an attempt to revitalize undergraduate CS education and to enrich the experiences of both sets of students. The paper provides details on the joint project done in these classes and the evaluation that assessed its effect on the curriculum, students, and professors.

Urban, J. (organizer), Heines, J.M., Fox, E.A., & Taylor, H.G. (2009). **Panel on Revitalized Undergraduate Computing Education.** *40th ACM SIGCSE Technical Symposium on CS Education.* Chattanooga, TN, March 5, 2009.

<http://teaching.cs.uml.edu/~heines/academic/papers/2009sigcse/sigcse2009panel-JMH-accepted.pdf>

There is an imbalance in the supply and demand for computing professionals that has generated shortages in meeting personnel needs within industry. A major program was developed by the U.S. National Science Foundation to encourage innovations in undergraduate computing education. There are a variety of new projects that are revitalizing undergraduate computing education. One approach to such revitalization is the introduction of interdisciplinary courses to expand the scope of computing education. The basic idea is to have students from various disciplines work together on computing projects to expand their educational horizons and make computing courses more appealing. This panel brings together research managers with educators who have developed and taught interdisciplinary courses with these goals in mind.

## Additional Readings (cont'd)

Heines, J.M., Jeffers, J., & Kuhn, S. (2008). **Performamatics: Experiences With Connecting a Computer Science Course to a Design Arts Course.** *The International Journal of Learning* 15(2):9-16.

<http://teaching.cs.uml.edu/~heines/academic/papers/2008learning/AsPublished-IntlJrnlLearning.pdf>

This paper describes our efforts to stem the tide of declining CS enrollments by introducing innovations into our curriculum to give students more flexibility in course selection, especially in the freshman and sophomore years. Our approach is based on a partnership between the CS and Art, Music, and English departments in the area of exhibition and performance technologies.

In addition to describing our work, this paper provides the results of an evaluation conducted by an independent research. It reports on the impact this work has had on the CS and Art students and their respective projects, as well as on the professors and the way they teach their courses. It also describes steps that are being taken to improve the courses in the future.



## Related Websites

### Performamatics Website and Scratch Gallery and YouTube Channel

<http://www.performamatics.org> → <http://teaching.cs.uml.edu/~heines/TUES/>  
<http://www.scratchmusic.org> → <http://scratch.mit.edu/galleries/view/90913>  
<http://www.youtube.com/performamatics>

### Scratch Projects by Performamatics People

<http://scratch.mit.edu/users/alexruthmann> (Music Prof. Alex Ruthmann)  
<http://scratch.mit.edu/users/drjay> (CS Prof. Jesse Heines)  
<http://scratch.mit.edu/users/performamatics> (additional collections)

### Scratch Software

<http://scratch.mit.edu> (home page)  
<http://scratch.mit.edu/download> (download page)  
<http://scratch.mit.edu/forums> (discussion forums)

### Scratch Resources for Teaching and Teachers

<http://scratched.media.mit.edu> (learn - share - connect for educators)

### Scratch Project Galleries

<http://scratch.mit.edu/channel/featured> (featured projects)  
<http://scratch.mit.edu/galleries/browse/newest> (members' personal galleries)

### Scratch Information and Support

[http://info.scratch.mit.edu/Support/Get\\_Started](http://info.scratch.mit.edu/Support/Get_Started) (getting started instructions)  
<http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/files/file/ScratchGettingStartedv14.pdf> (Getting Started Guide)  
[http://info.scratch.mit.edu/Support/Reference\\_Guide\\_1.4](http://info.scratch.mit.edu/Support/Reference_Guide_1.4) (Reference Guide)  
<http://info.scratch.mit.edu/Support> (support page)  
[http://info.scratch.mit.edu/Video\\_Tutorials](http://info.scratch.mit.edu/Video_Tutorials) (video tutorials)  
[http://info.scratch.mit.edu/Support/Scratch\\_Cards](http://info.scratch.mit.edu/Support/Scratch_Cards) (single-topic lessons)

### Lifelong Kindergarten Group and Collaborators' Websites

<http://llk.media.mit.edu> (John Maloney and Mitchel Resnick)  
<http://teaching.cs.uml.edu> (Jesse Heines)  
<http://www.alexruthmann.com> (Alex Ruthmann)