

# Making Music with Scratch

a workshop at presented at

**CS4HS @ UML**

June 28, 2011

**Jesse M. Heines**

Dept. of Computer Science  
University of Massachusetts Lowell  
Jesse\_Heines@uml.edu

**S. Alex Ruthmann**

Dept. of Music  
University of Massachusetts Lowell  
Alex\_Ruthmann@cs.uml.edu





## Copyright Notice

The materials in this handout, including all text and programs, are copyright © 2011 by Jesse Heines, Alex Ruthmann, Gena 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.

## Contents

Workshop Description .....	5
Workshop Leaders .....	6
Workshop Agenda .....	7
Additional Workshop Info and URLs .....	8
Important Note on Turbo Speed .....	9
Acknowledgements .....	9
<b>Playing and Synchronizing MIDI Files</b> .....	<b>11</b>
MP3 Player Scripts .....	12
Control Script .....	13
<b>Example 1 - Frère Jacques</b>	
Version 1: Playing Notes .....	15
Version 2: Using Loops .....	16
Version 3: Separating Phrases.....	17
Version 4: Playing a Round .....	19
<b>Example 2 - Row, Row, Row Your Boat</b>	
Version 1: Playing Notes .....	23
Version 2: Playing Notes Using Variables .....	24
Version 3: Separating Initialization .....	25
Version 4: Separating Phrases .....	27

*continued on next page*

## Contents (cont'd)

Version 5: Looping and Fading .....	30
Version 6: Playing a Round with One Instrument .....	32
Version 7: Playing a Round with Two Instruments .....	34
Version 8: Storing Notes and Rhythms in Lists .....	37
Version 9: Playing a Round Using Lists .....	39
Version 10: Synchronizing Play from Lists .....	41
<b>Extending the Examples</b> .....	<b>45</b>
<b>The IchiBoard</b> .....	<b>49</b>
<b>Computer Science, Math, and Music:</b>	
Concepts Covered in Scratch .....	50
<b>Computing and Music:</b>	
What Do They Have in Common? .....	51
<b>Additional Readings</b> .....	<b>55</b>
<b>Related Websites</b> .....	<b>58</b>

## Workshop Description

This workshop introduces the music playing and generation abilities of Scratch, a media-rich visual programming system ([scratch.mit.edu](http://scratch.mit.edu)). It is based on experiences gained using Scratch to teach both music and computer science in an interdisciplinary, college-level GenEd course. As students write programs that make music, they learn CS concepts such as control flow, user interaction, synchronization, real-time programming, and data structures.

Workshop participants will explore progressively complex Scratch programs that incorporate music in a variety of ways (see [www.scratchmusic.org](http://www.scratchmusic.org)), including the use of external sensor devices to make custom musical instruments. The workshop culminates in a concert of participant-created music.

This handout provides the sample programs presented in the workshop and suggestions for extending them. These programs are available for download from:

<http://teaching.cs.uml.edu/~heines/academic/papers/2011cs4hs/ScratchMusicWorkshopMaterials/>

## Workshop Leaders

**Jesse Heines** is a Professor of Computer Science at the University of Massachusetts Lowell. He has a keen interest in CS education and computer applications in the arts, particularly those in music. This interest was recently supported by NSF award 0722161, "Performamatics: Connecting Computer Science to the Performing, Fine, and Design Arts" ([www.nsf.gov/awardsearch/showAward.do?AwardNumber=0722161](http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0722161) and [www.performamatics.org](http://www.performamatics.org)). Jesse grew up in a musical household and currently enjoys singing in a barbershop chorus.

**S. Alex Ruthmann** is an Assistant Professor of Music Education at the University of Massachusetts Lowell, where he teaches courses at the intersection of music, education, and technology. He is currently working on developing musical algorithms for interactive audio games in Scratch. In his free time, Alex composes and hacks new electronic interfaces for musical expression and performance.

## Workshop Agenda

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. Linking to the IchiBoard, an external sensor device
7. Sharing what you've created
  - Using the Scratch website
  - Concert of live performances by participants 😊

## Additional Information on the Software Used in This Workshop

To replicate the workshop activities and examples on their own systems, participants should download and install:

- **Scratch**  
[scratch.mit.edu/download](http://scratch.mit.edu/download)
- **Audacity**  
[audacity.sourceforge.net/download](http://audacity.sourceforge.net/download)
  - download the **1.3 Series (Beta)**
- **IchiBoard Drivers**  
[www.cs.uml.edu/ecg/index.php/IchiBoard](http://www.cs.uml.edu/ecg/index.php/IchiBoard)
  - drivers are available for both Windows and Macintosh systems

**Please 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.



## 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

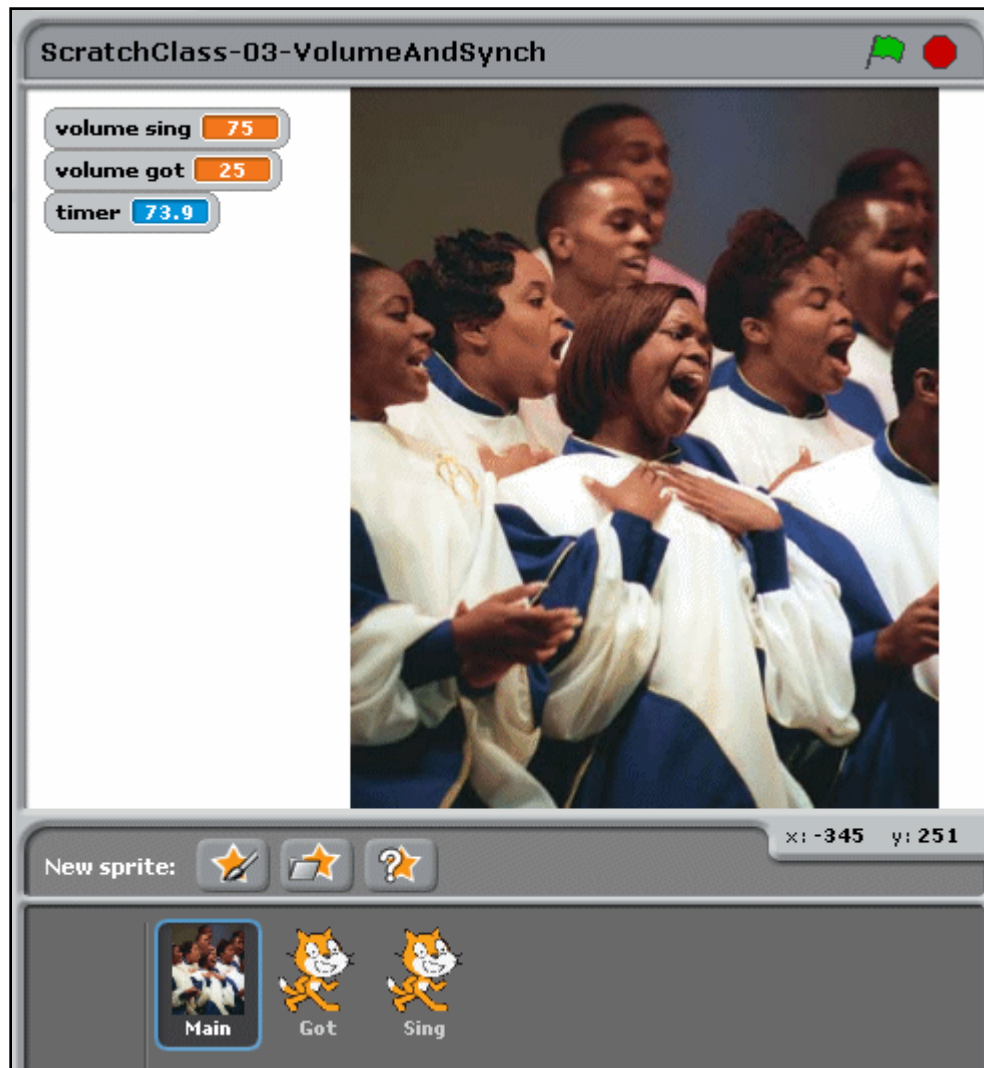
Additional contributors to this work include UMass Lowell Profs. Gena Greher and Fred Martin, graduate student Mark Sherman, recent baccalaureate graduates Paul Laidler and Charles Saulters, and John Maloney of the MIT Media Laboratory Lifelong Kindergarten Group.

The materials presented in this workshop is based in part upon work supported by the National Science Foundation under Grant No. 0722161, "CPATH CB: Performamatics: Connecting Computer Science to the Performing, Fine, and Design Arts" and complementary Research Experience for Undergraduates (REU) supplements. 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



## 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)

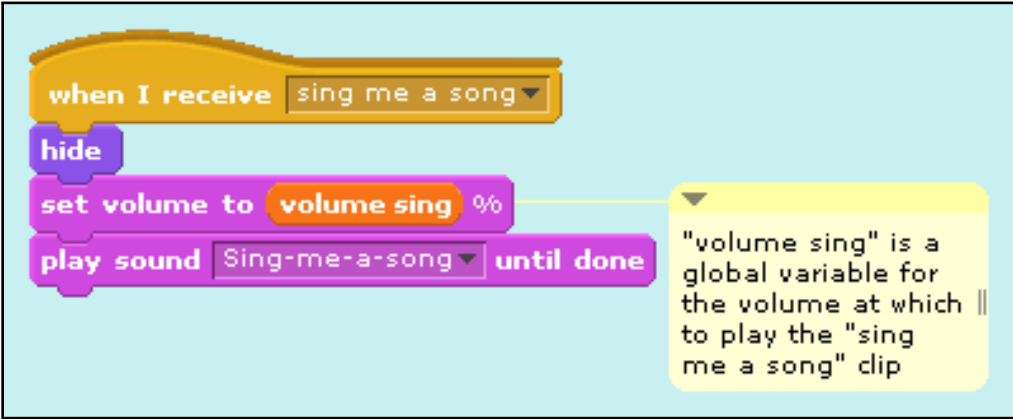
### MP3 Player Scripts

#### Script in Sprite "Got"



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

#### Script in Sprite "Sing"



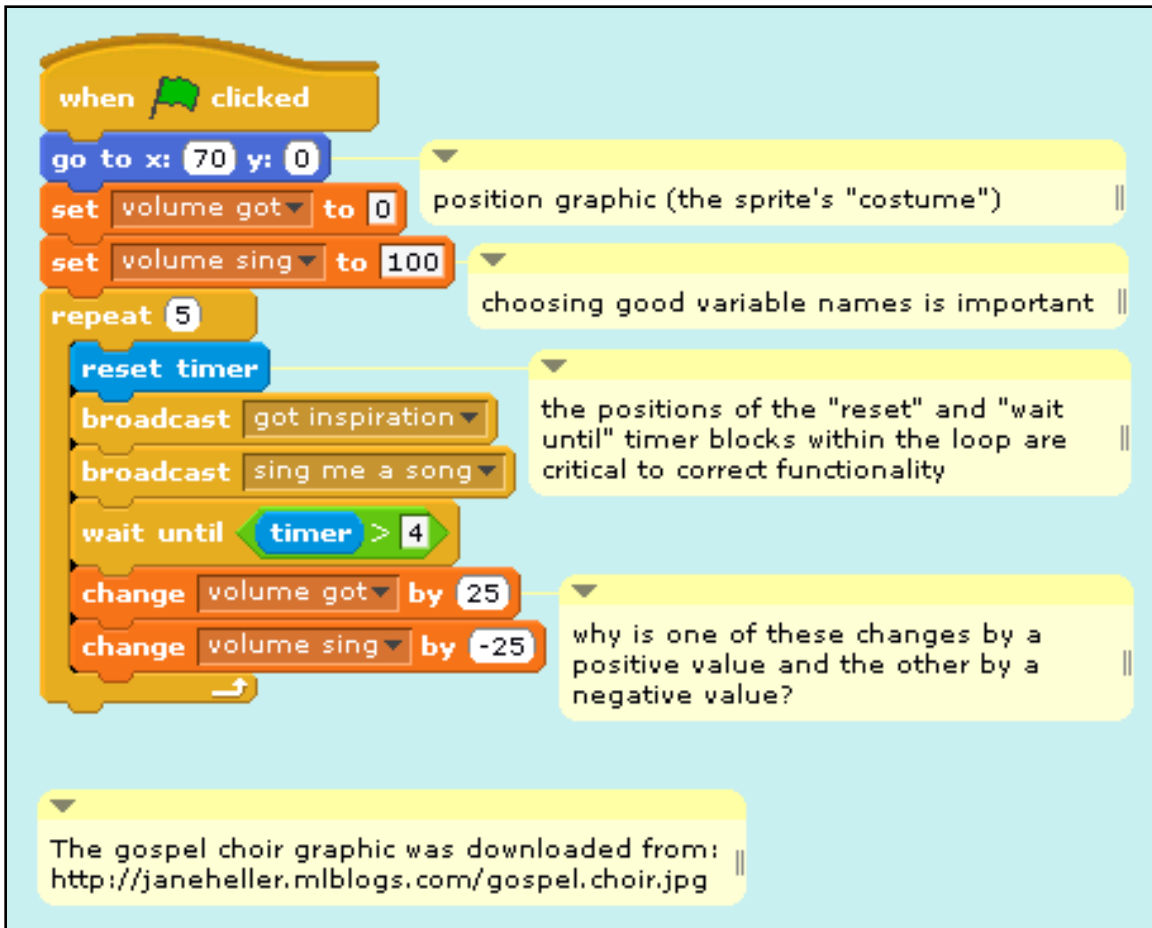
"volume sing" is a global variable for the volume at which to play the "sing me a song" clip

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 image shows a Scratch script for a sprite named "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 in the image explain key parts of the script:

- position graphic (the sprite's "costume")**: Points to the `go to x: 70 y: 0` block.
- choosing good variable names is important**: Points to the `set volume got to 0` and `set volume sing to 100` blocks.
- the positions of the "reset" and "wait until" timer blocks within the loop are critical to correct functionality**: Points to the `reset timer` and `wait until timer > 4` blocks.
- why is one of these changes by a positive value and the other by a negative value?**: Points to the `change volume got by 25` and `change volume sing by -25` blocks.

A note at the bottom states: "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



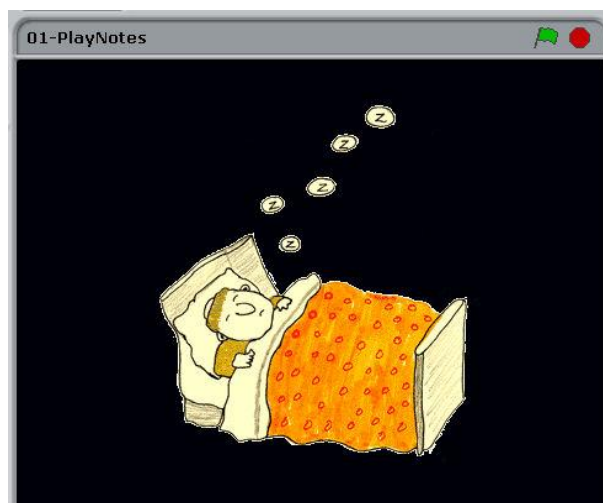
The script starts with a 'when clicked' event block. It then sets the instrument to 'church organ' and hides the sprite. The main sequence consists of 28 'play note' blocks, each followed by a label in a yellow box:

- play note 55 for 0.5 beats → Phrase #1
- play note 57 for 0.5 beats → Phrase #1
- play note 59 for 0.5 beats → Phrase #1
- play note 55 for 0.5 beats → Phrase #1
- play note 55 for 0.5 beats → Phrase #1 repeated
- play note 57 for 0.5 beats → Phrase #1 repeated
- play note 59 for 0.5 beats → Phrase #1 repeated
- play note 55 for 0.5 beats → Phrase #1 repeated
- play note 59 for 0.5 beats → Phrase #2
- play note 60 for 0.5 beats → Phrase #2
- play note 62 for 1 beats → Phrase #2
- play note 59 for 0.5 beats → Phrase #2 repeated
- play note 60 for 0.5 beats → Phrase #2 repeated
- play note 62 for 1 beats → Phrase #2 repeated
- play note 62 for 0.25 beats → Phrase #3
- play note 64 for 0.25 beats → Phrase #3
- play note 62 for 0.25 beats → Phrase #3
- play note 60 for 0.25 beats → Phrase #3
- play note 59 for 0.5 beats → Phrase #3
- play note 55 for 0.5 beats → Phrase #3
- play note 62 for 0.25 beats → Phrase #3 repeated
- play note 64 for 0.25 beats → Phrase #3 repeated
- play note 62 for 0.25 beats → Phrase #3 repeated
- play note 60 for 0.25 beats → Phrase #3 repeated
- play note 59 for 0.5 beats → Phrase #4
- play note 55 for 0.5 beats → Phrase #4
- play note 50 for 0.5 beats → Phrase #4
- play note 55 for 1 beats → Phrase #4
- play note 55 for 0.5 beats → Phrase #4 repeated
- play note 50 for 0.5 beats → Phrase #4 repeated
- play note 55 for 1 beats → Phrase #4 repeated



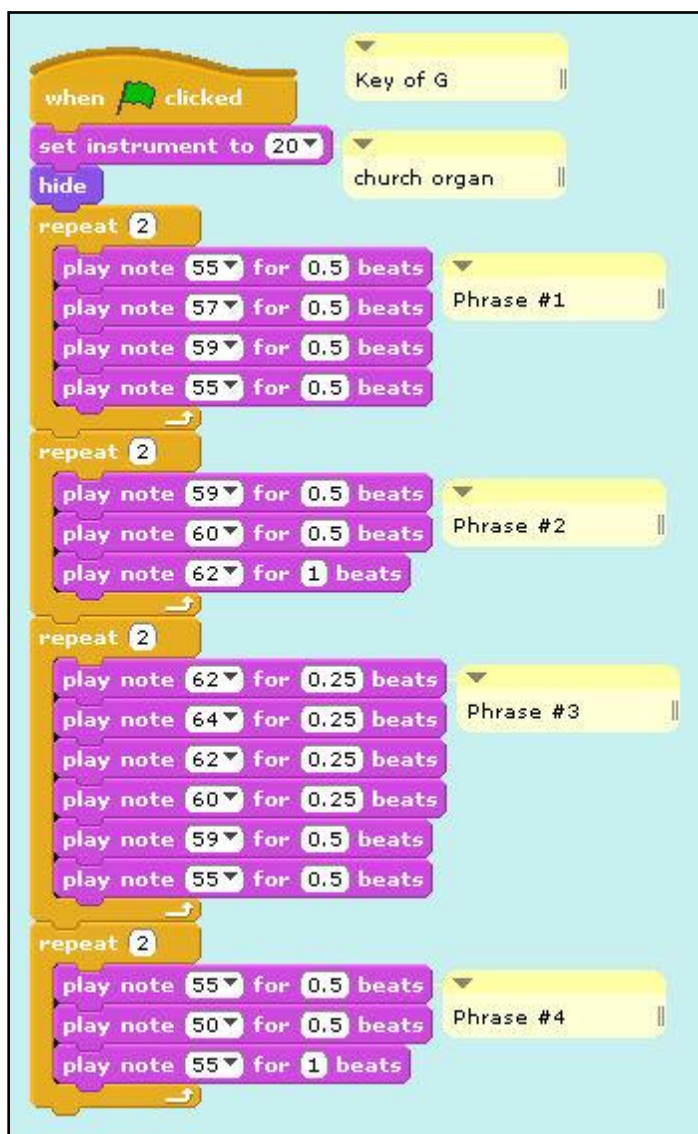
Frère Jacques musical notation with lyrics:

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



The Scratch code is as follows:

```

when clicked
  set instrument to 20
  hide
  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
  repeat 2
    play note 59 for 0.5 beats
    play note 60 for 0.5 beats
    play note 62 for 1 beats
  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
  repeat 2
    play note 55 for 0.5 beats
    play note 50 for 0.5 beats
    play note 55 for 1 beats
  
```



The musical score consists of four staves of music in G major, 2/4 time. The lyrics are: Frère Jacques, Dormez-vous? Sonnez les matines, Ding dingue dong.

Remember to set Turbo Speed to improve performance.

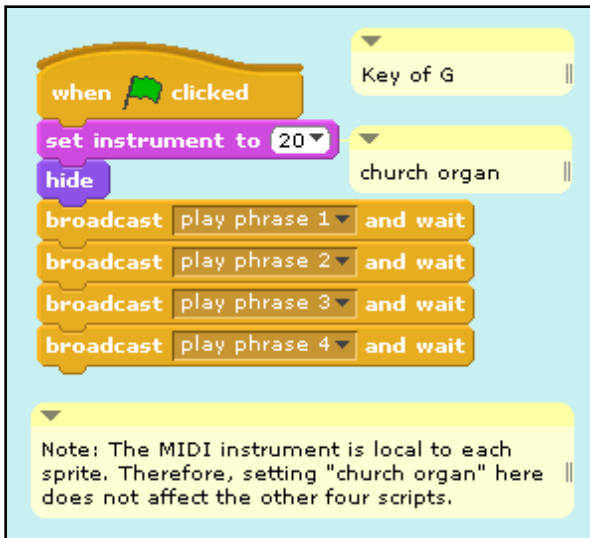
**Acknowledgement:** The scores on this and the previous page were adapted from [www.csdraveurs.qc.ca/musique/flutalors/images/frere.gif](http://www.csdraveurs.qc.ca/musique/flutalors/images/frere.gif) and [www.mamalisa.com/images/scores/frerejacques.jpg](http://www.mamalisa.com/images/scores/frerejacques.jpg), respectively.



## 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

Key of G

church organ

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?

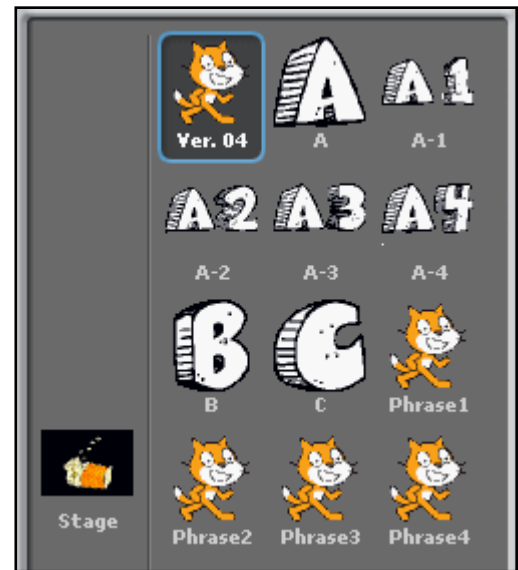
## Frère Jacques

# 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
    
```

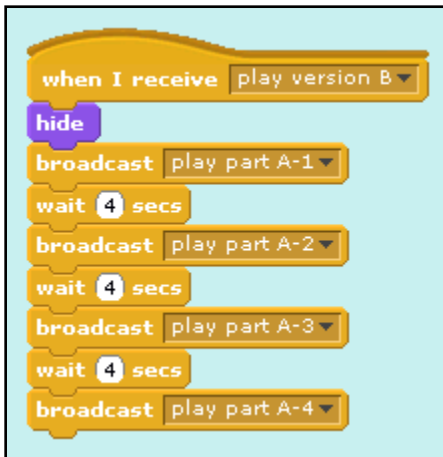
This version does not play as expected.

*continued on next page*

## Frère Jacques

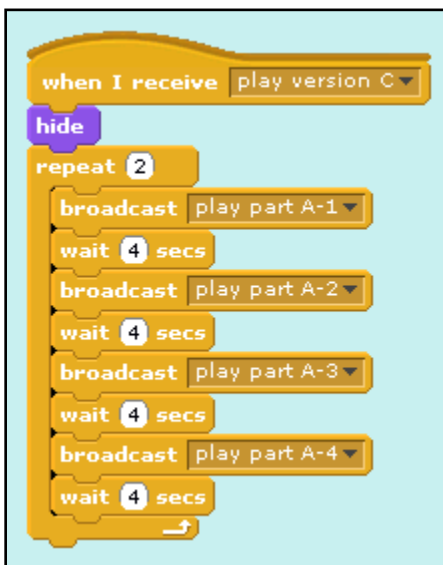
### Version 4: Playing a Round (cont'd)

#### 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
```

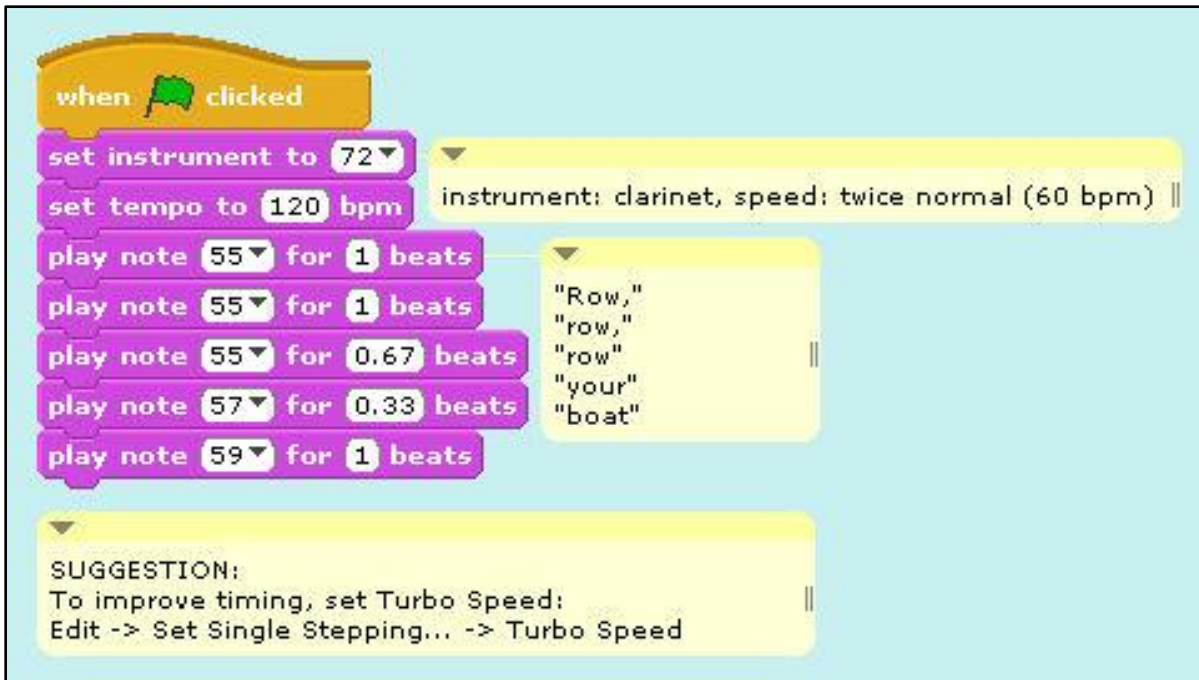
*end of Version 4*

# SCRATCH



# Row, Row, Row Your Boat Version 1: Playing Notes

## Single Script



when clicked

set instrument to 72

set tempo to 120 bpm

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

play note 55 for 1 beats

play note 55 for 1 beats

play note 55 for 0.67 beats

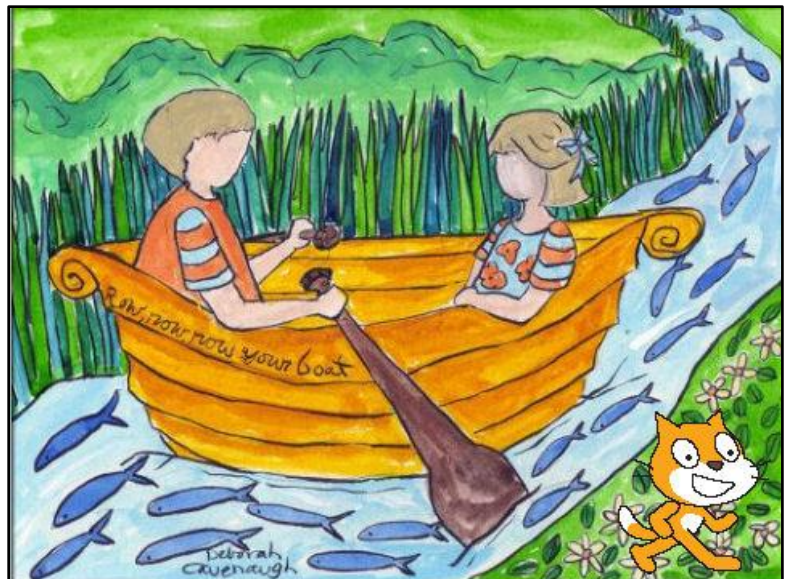
play note 57 for 0.33 beats

play note 59 for 1 beats

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

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

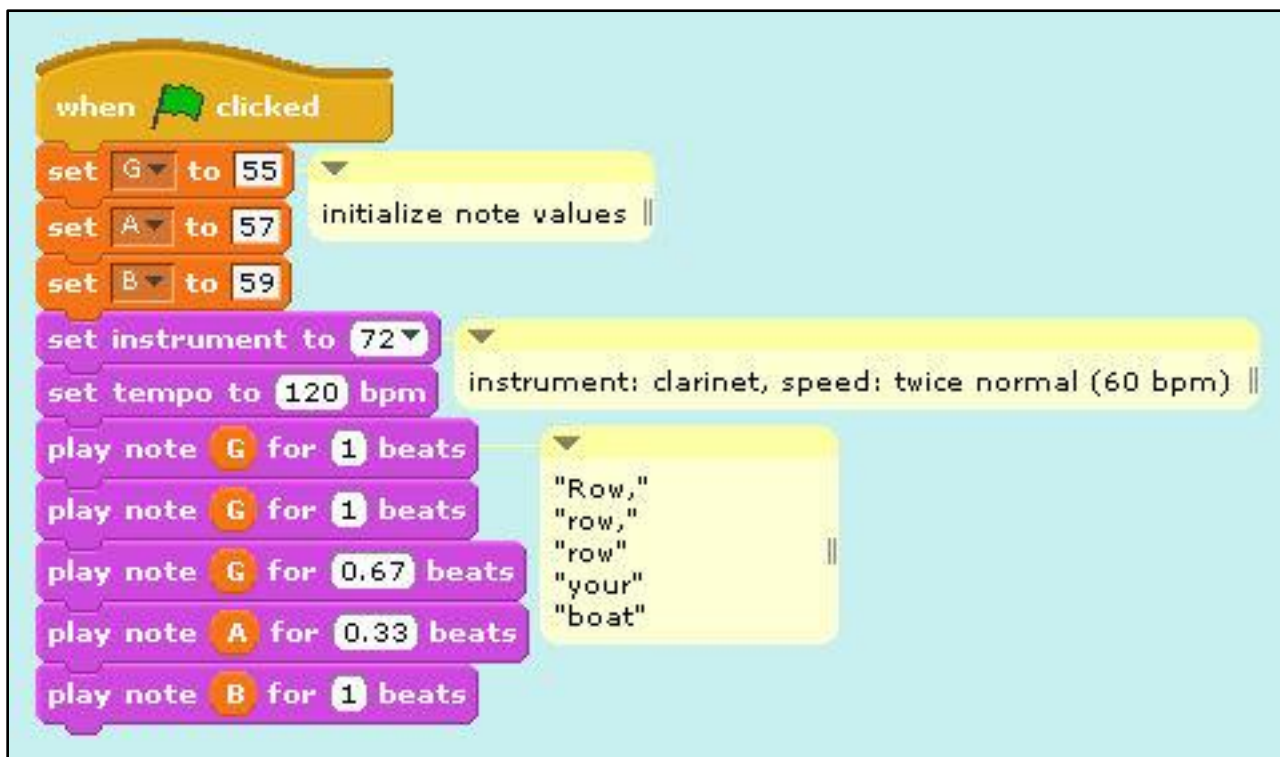
## Output Window



## 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:

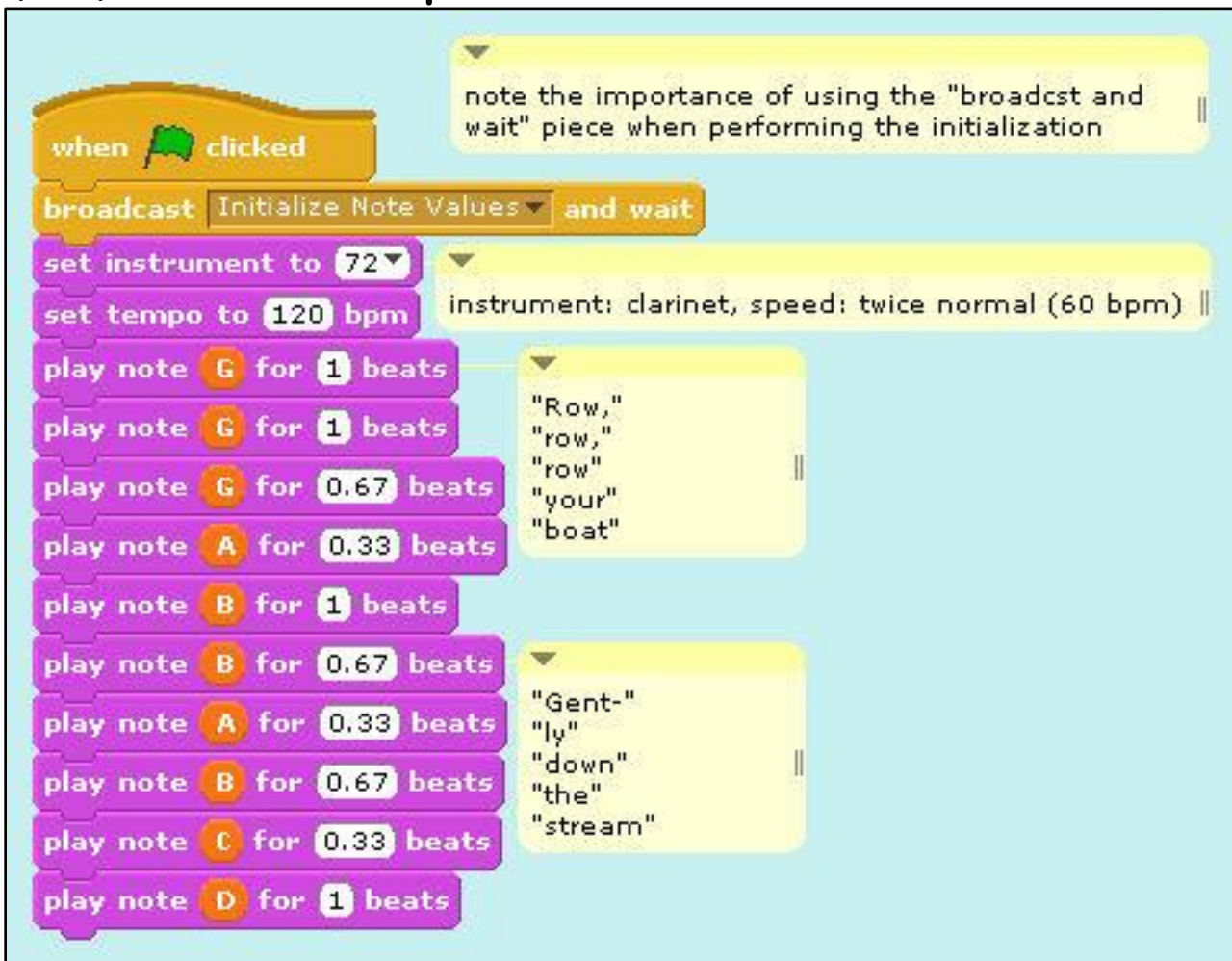
- initialize note values ||
- instrument: clarinet, speed: twice normal (60 bpm) ||
- "Row,"  
"row,"  
"row"  
"your"  
"boat" ||



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

### Two Scripts

#### (3a) Main Script



The image shows a Scratch script for a main script. It starts with a 'when clicked' block, followed by a 'broadcast Initialize Note Values and wait' block. The script then contains a series of 'play note' blocks for notes G, A, B, C, and D, each with a specific duration in beats. There are three yellow callout boxes providing additional information:

- note the importance of using the "broadcast and wait" piece when performing the initialization
- instrument: clarinet, speed: twice normal (60 bpm)
- "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 two Scratch scripts side-by-side. The left script is titled 'when I receive Initialize Note Values' and contains a 'hide' block followed by 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 right script is titled 'when I receive Play G Major Scale' and 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 above the right script says 'Click the set of pieces below to test the variable values by hearing a G major scale'.

*end of Version 3*

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

### Three Scripts

#### (4a) Main Script



The script consists of the following blocks:

- 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

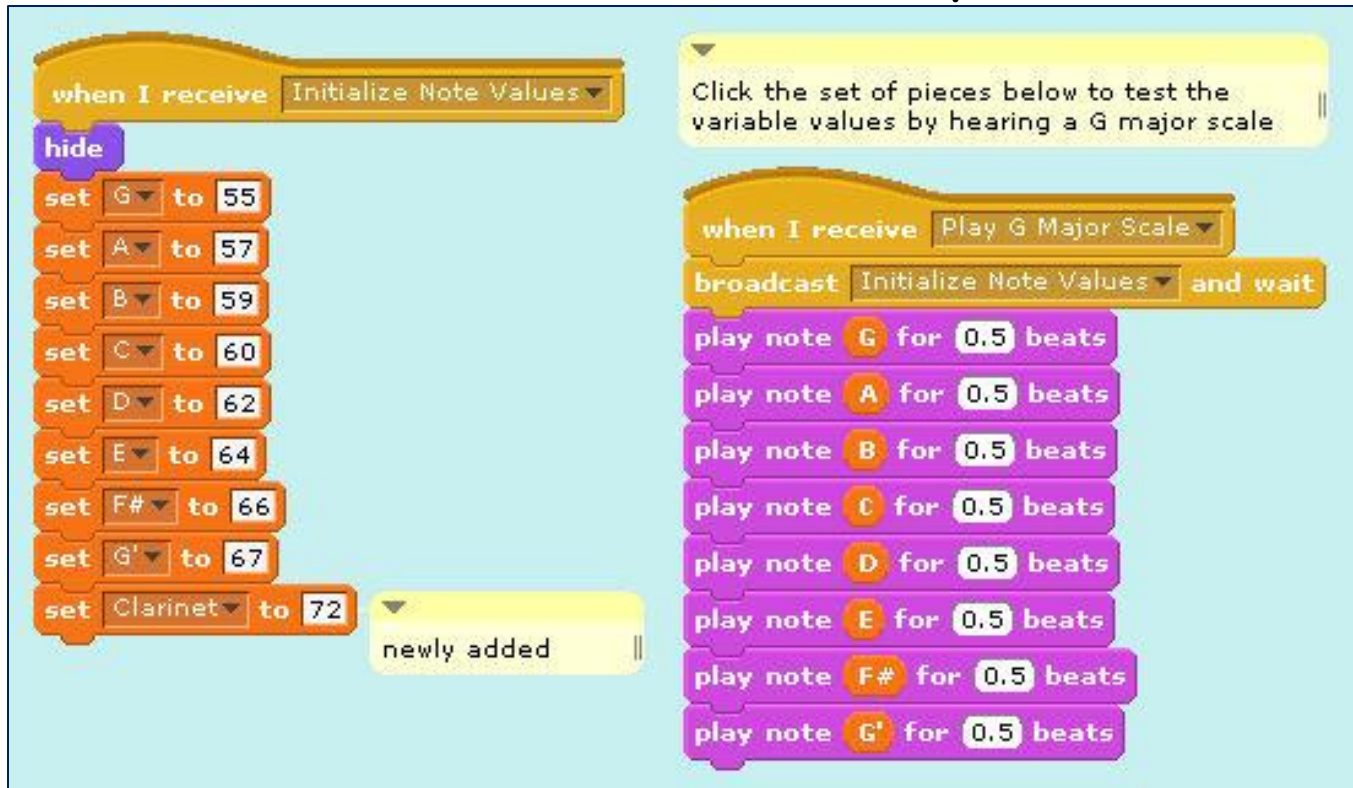
Annotations:

- 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 "newly added" tooltip is also visible.

```

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
  
```

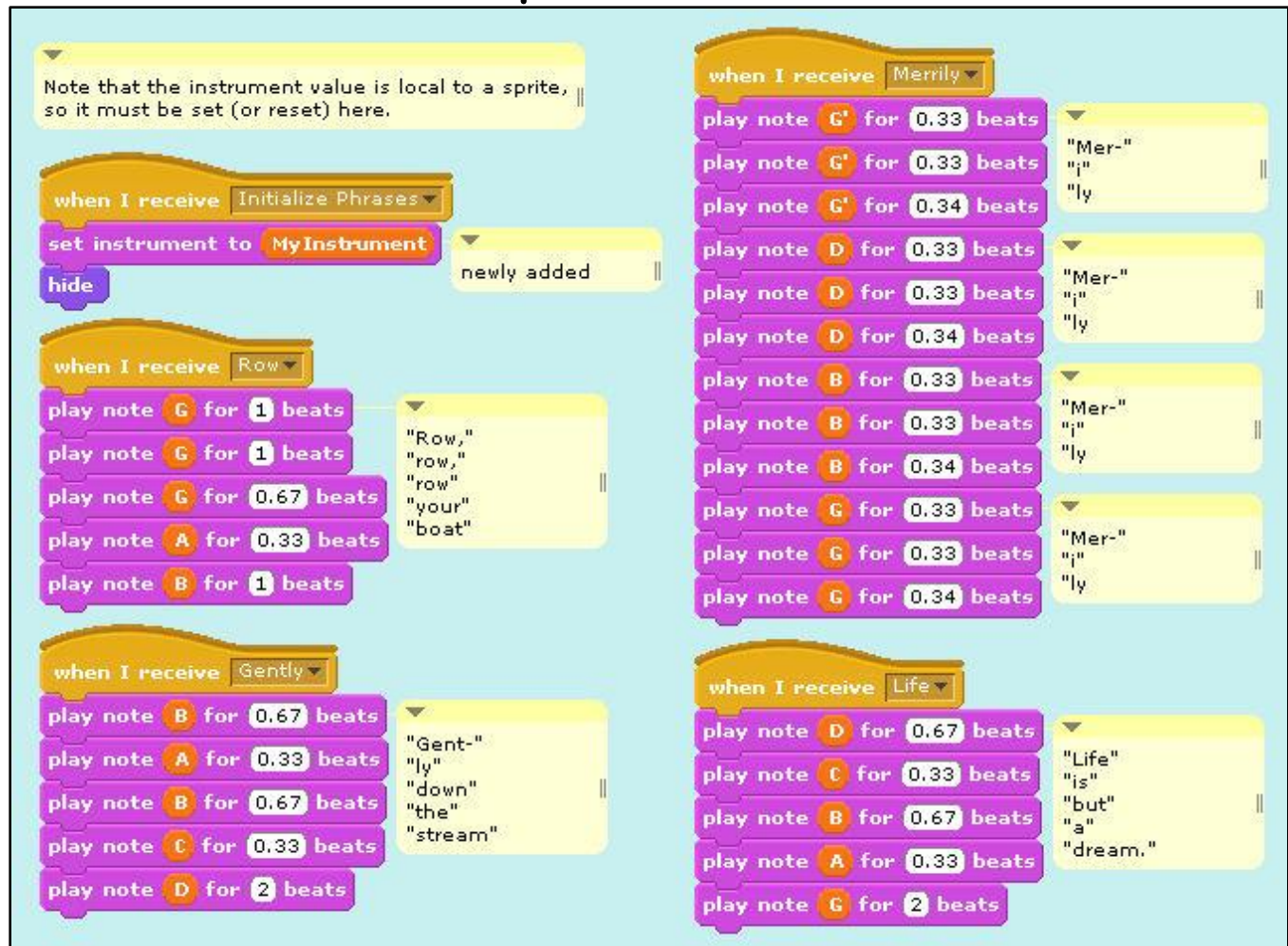
*continued on next page*

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

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

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
  "Mer-"
  "i"
  "ly"
  "Mer-"
  "i"
  "ly"
  "Mer-"
  "i"
  "ly"
  "Mer-"
  "i"
  "ly"

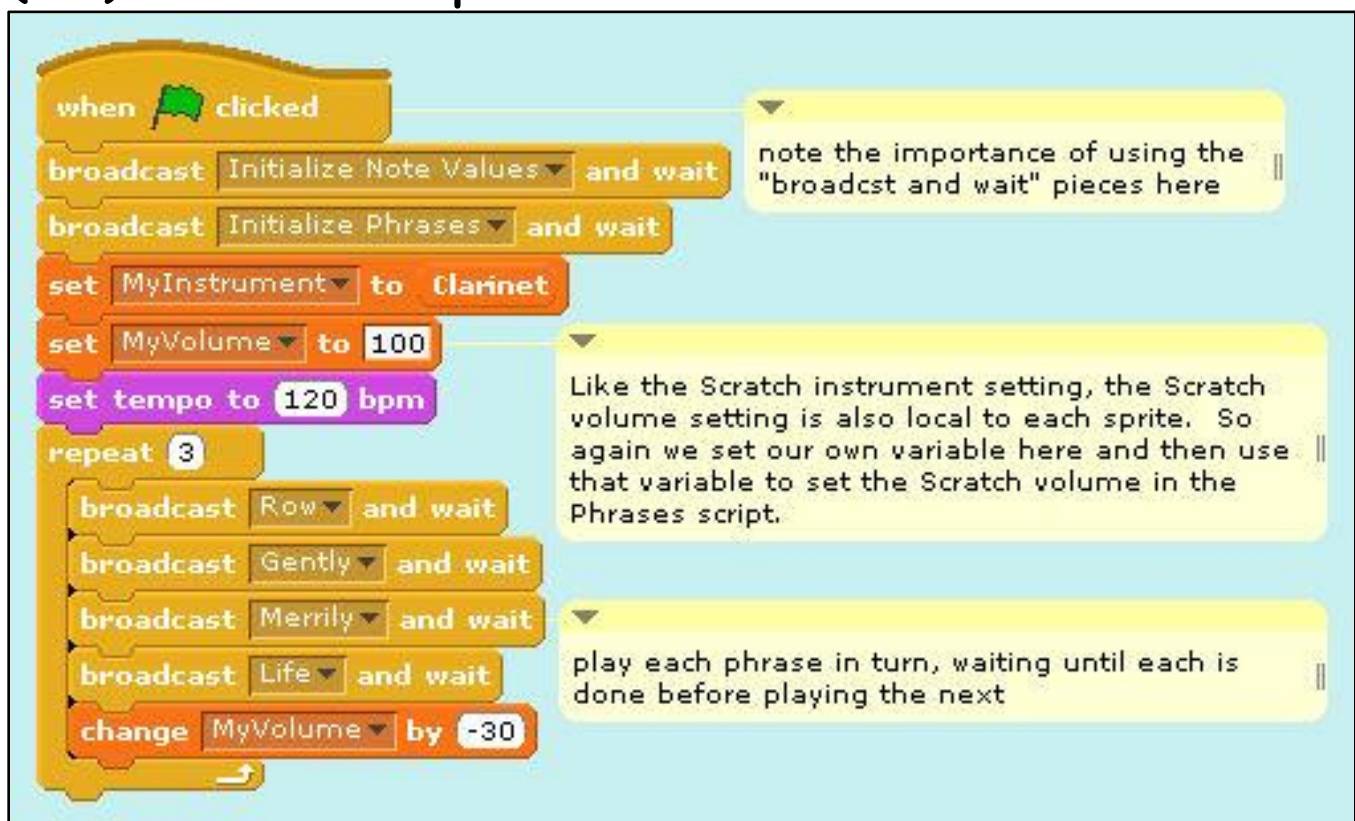
when I receive Life
  play note D for 0.67 beats
  play note C for 0.33 beats
  play note B for 0.67 beats
  play note A for 0.33 beats
  play note G for 2 beats
  "Life"
  "is"
  "but"
  "a"
  "dream."
  
```

*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. The script starts with a 'when green flag clicked' block, followed by two 'broadcast and wait' blocks for 'Initialize Note Values' and 'Initialize Phrases'. Then, it sets 'MyInstrument' to 'Clarinet' and 'MyVolume' to 100. The tempo is set to 120 bpm. A 'repeat 3' loop contains four 'broadcast and wait' blocks for 'Row', 'Gently', 'Merrily', and 'Life', followed by a 'change MyVolume by -30' block. Three yellow callout boxes provide annotations: the first notes the importance of 'broadcast and wait' pieces; the second explains that Scratch volume is local to each sprite and a custom variable is used to set it; the third states that phrases are played in turn, waiting for each to finish.

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

*(same as on page 28)*

*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' in a code editor. The first script is triggered by a 'when clicked' event and contains 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 next to the first two broadcast blocks says 'note the importance of using the "broadcast and wait" pieces here'. A second yellow callout box next to the 'set NoOfTimesToPlay' block says '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.' The second script is triggered by 'when I receive Part1' and contains: 'set Counter to 1', a 'repeat until Counter > NoOfTimesToPlay' loop containing 'broadcast Row and wait', an 'if Counter = 1' block containing 'broadcast Part2', 'broadcast Gently and wait', 'broadcast Merrily and wait', 'broadcast Life and wait', and 'change Counter by 1'. The third script is triggered by 'when I receive Part2' and contains a 'repeat NoOfTimesToPlay' loop with four 'broadcast' blocks: 'Row and wait', 'Gently and wait', 'Merrily and wait', and 'Life and wait'.



## 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 for initialization and playing a G major scale. The script is divided into two main sections:

- Initialization ("Init") Script:**
  - Starts with a "when I receive" trigger set to "Initialize Note Values".
  - Followed by a "hide" block.
  - A series of "set" blocks for notes: G (55), A (57), B (59), C (60), D (62), E (64), F# (66), G' (67), and Clarinet (72).
  - A "newly added" tooltip is visible next to the Clarinet block.
- Playing a G Major Scale Script:**
  - Starts with a "when I receive" trigger set to "Play G Major Scale".
  - Followed by a "broadcast" block: "broadcast Initialize Note Values and wait".
  - A series of "play note" blocks for notes G, A, B, C, D, E, F#, and G', each for 0.5 beats.

There are also two yellow callout boxes: one at the top right says "Click the set of pieces below to test the variable values by hearing a G major scale" and another at the bottom right says "newly added".

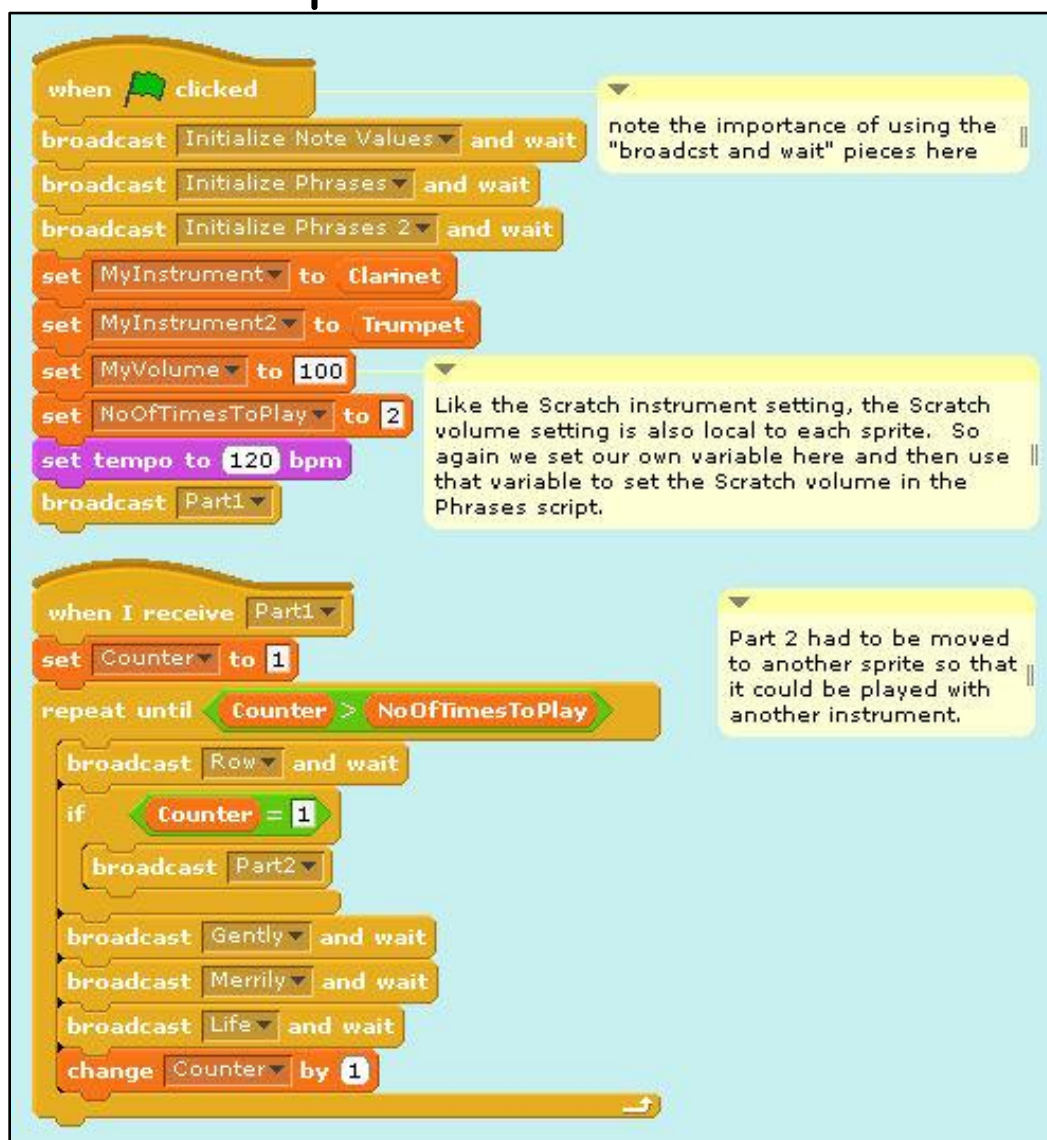
### (6c) Phrases Script (same as on page 31)

*end of Version 6*

## Row, Row, Row Your Boat Version 7: Playing a Round with Two Instruments

### Five Scripts

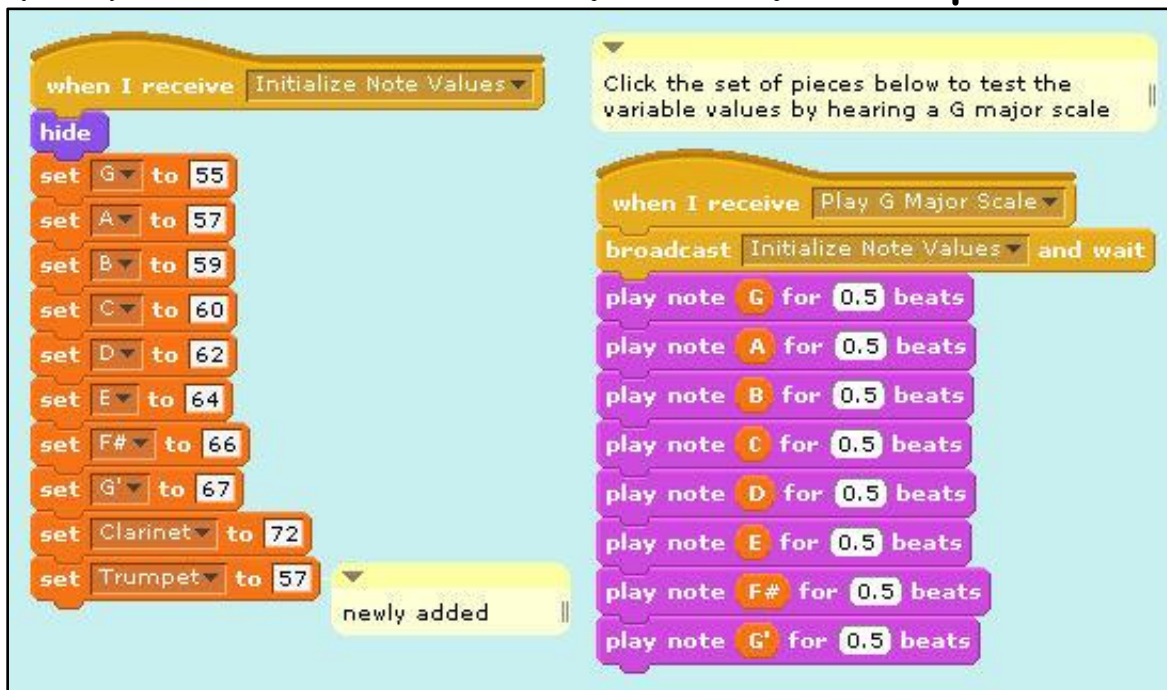
#### (7a) Main Script



The image shows a Scratch script for a 'Main Script' with two main sections. The first section is triggered by a 'when clicked' event and includes several initialization steps: broadcasting 'Initialize Note Values', 'Initialize Phrases', and 'Initialize Phrases 2' (all with 'and wait' blocks), setting 'MyInstrument' to 'Clarinet', 'MyInstrument2' to 'Trumpet', 'MyVolume' to 100, 'NoOfTimesToPlay' to 2, and 'tempo' to 120 bpm. It ends with a 'broadcast Part1' block. A callout box notes the importance of using 'broadcast and wait' pieces here. The second section is triggered by 'when I receive Part1' and starts with 'set Counter to 1'. It enters a 'repeat until' loop where 'Counter' is less than 'NoOfTimesToPlay'. Inside the loop, it broadcasts 'Row' (with 'and wait'), checks if 'Counter = 1', and if true, broadcasts 'Part2'. It then broadcasts 'Gently', 'Merrily', and 'Life' (all with 'and wait'), and finally changes 'Counter' by 1. A callout box explains that 'Part 2' had to be moved to another sprite so 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 notes G, A, B, C, D, E, F#, G', Clarinet, and Trumpet. The second part, triggered by 'when I receive Play G Major Scale', includes a 'broadcast Initialize Note Values and wait' block followed by ten 'play note' blocks for G, A, B, C, D, E, F#, and G'.

### (7c) Phrases Script (same as on page 31)

### (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' blocks: 'Row2 and wait', 'Gently2 and wait', 'Merrily2 and wait', and 'Life2 and wait'.

## 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 a background illustration of a boy rowing a boat. The boat has the text "Row, Row, Row Your Boat" written on its side. The artist's signature "Deborah Cavanaugh" is visible at the bottom of the illustration. In the top left corner, there is a "ListIndex" control with a value of 0. On the right side, there are two vertical lists:

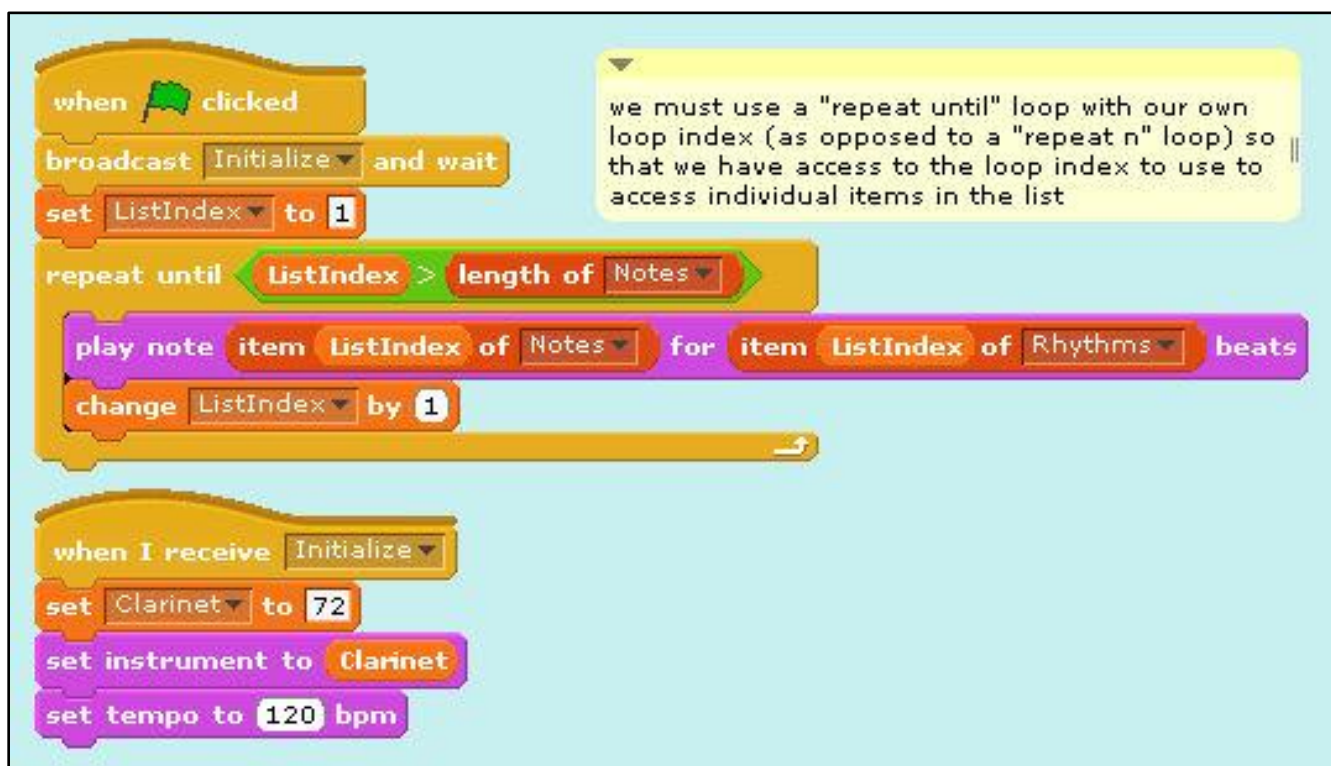
Notes		Rhythms	
1	55	1	1
2	55	2	1
3	55	3	0.67
4	57	4	0.33
5	59	5	1
6	59	6	0.67
7	57	7	0.33
8	59	8	0.67
9	60	9	0.33
10	62	10	2
11	67	11	0.33
12	67	12	0.33
13	67	13	0.34
14	62	14	0.33
15	62	15	0.33

At the bottom of each list, there is a scroll bar and a label: "+ length: 27".

*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

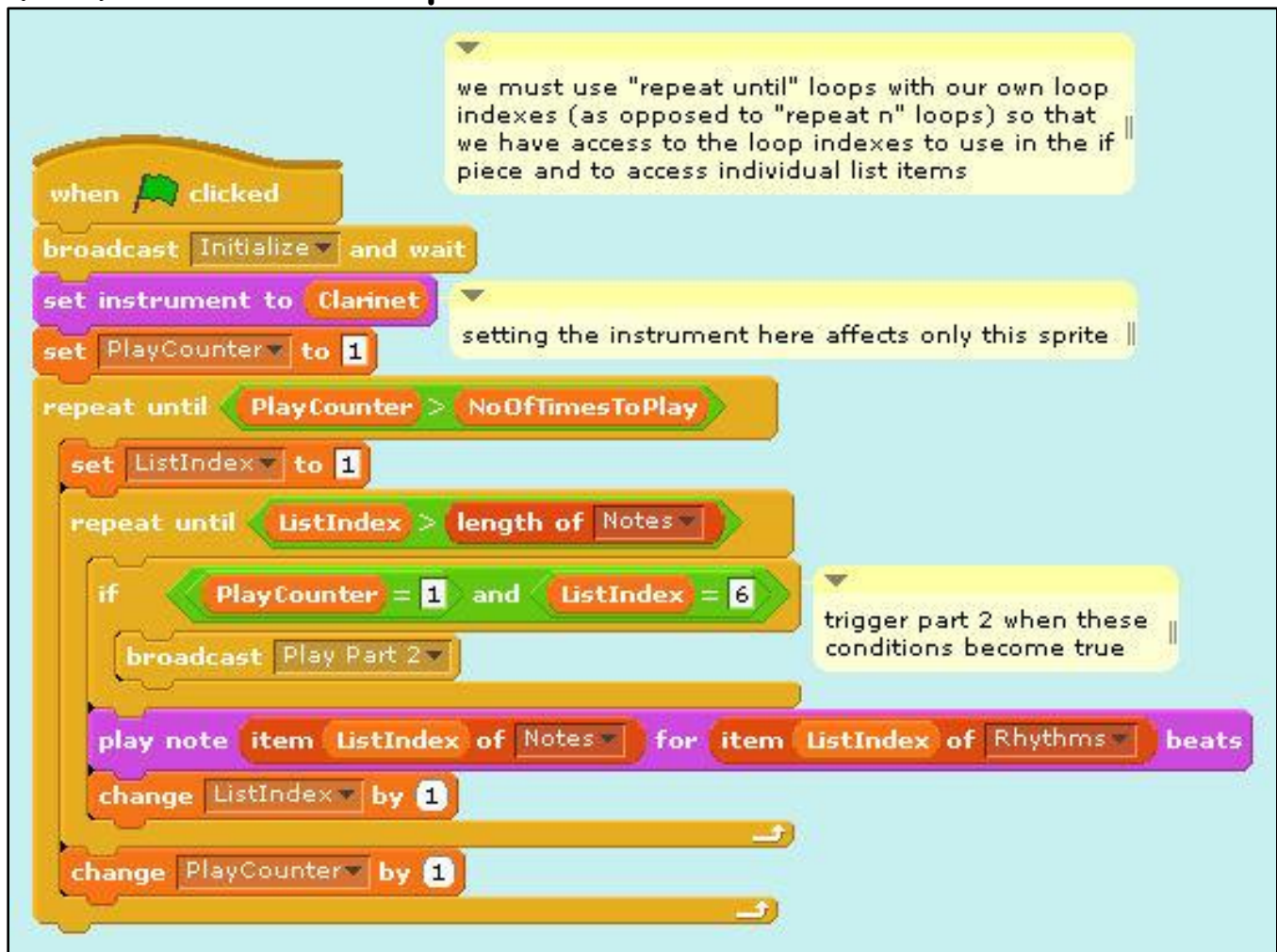
*end of Version 8*

## 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:

- 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
- setting the instrument here affects only this sprite
- 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)

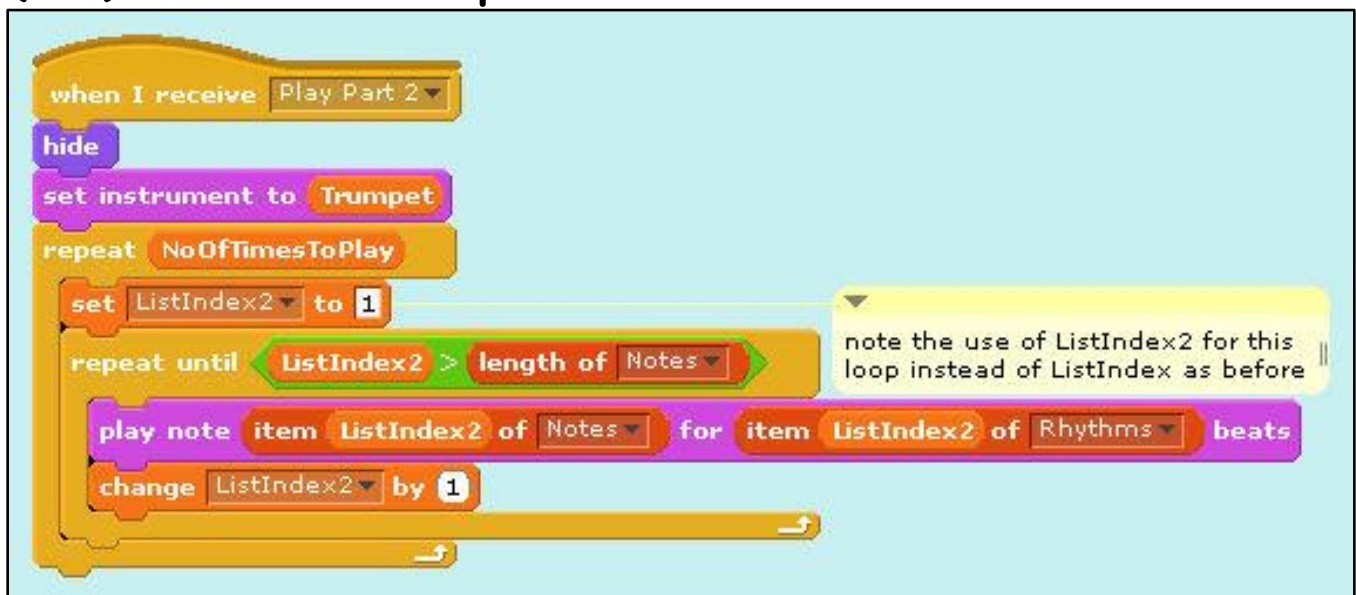
### (9b) Initialization ("Init") Script



```

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*

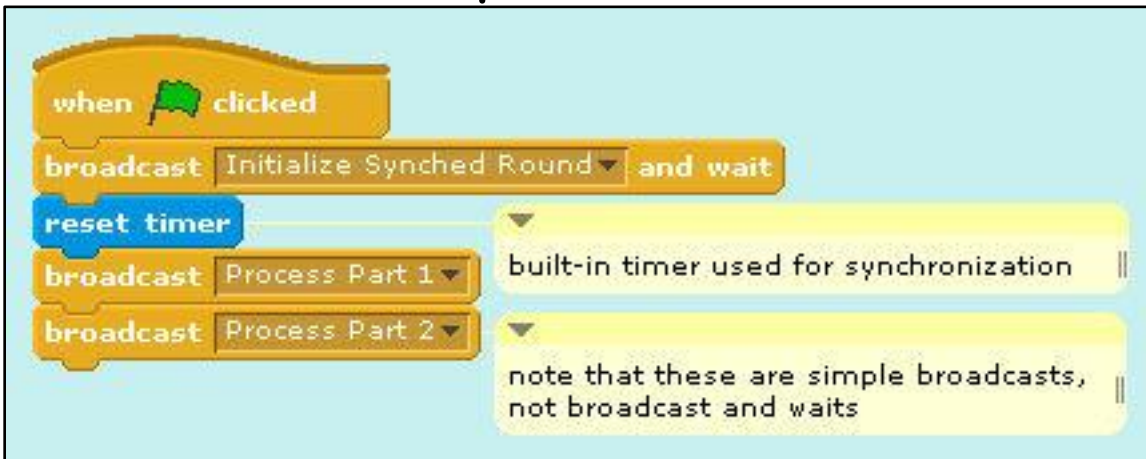


## Row, Row, Row Your Boat

# Version 10: Synchronizing Play from Lists

## Four Scripts

### (10a) Main Script



when clicked

broadcast Initialize Synched Round and wait

reset timer

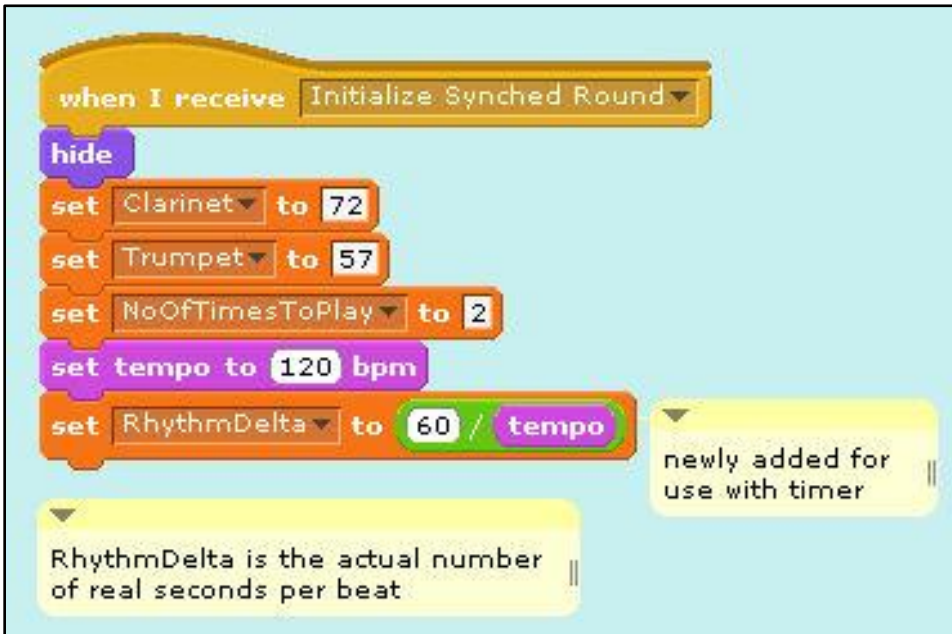
broadcast Process Part 1

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

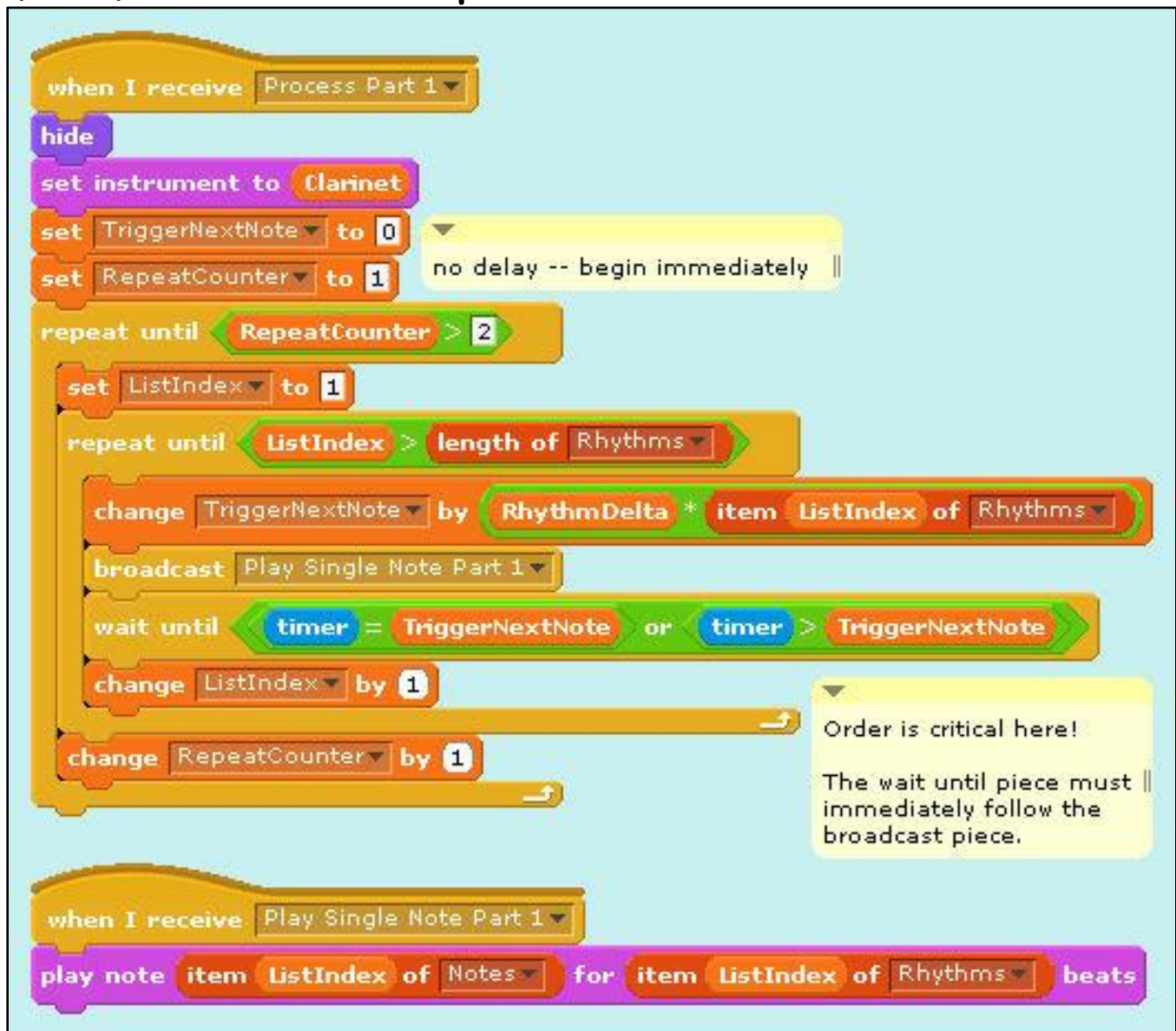
RhythmDelta is the actual number of real seconds per beat

*continued on  
next page*

## Row, Row, Row Your Boat

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

### (10c) Part 1 Script



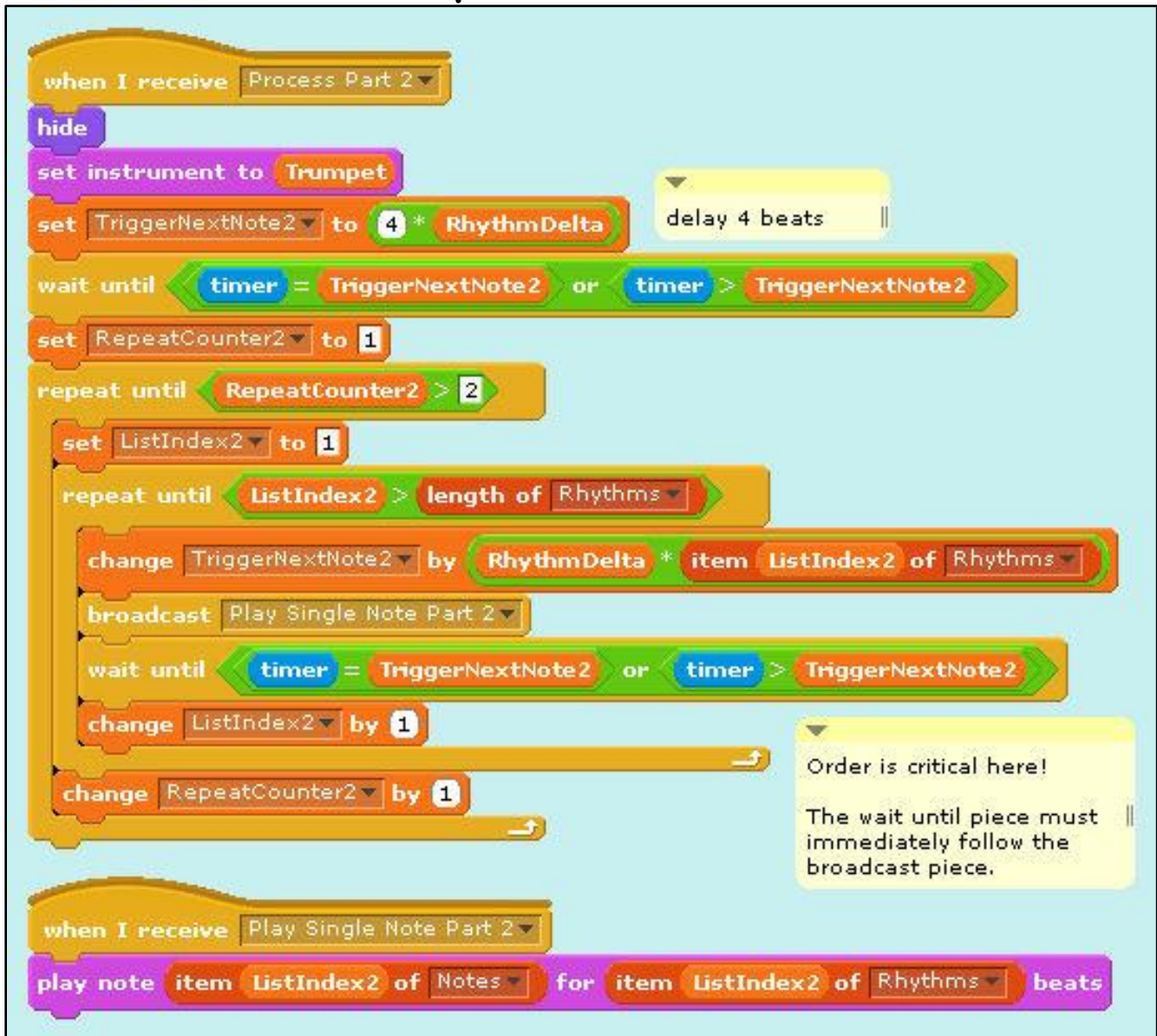
The script is divided into two main sections. The first section starts with a 'when I receive' block for 'Process Part 1'. It includes a 'hide' block, followed by 'set instrument to Clarinet'. Two 'set' blocks are used: 'set TriggerNextNote to 0' and 'set RepeatCounter to 1'. A yellow callout box next to the second 'set' block says 'no delay -- begin immediately ||'. This is followed by a 'repeat until' loop where 'RepeatCounter > 2'. Inside this loop, there is a 'set ListIndex to 1' block, another 'repeat until' loop where 'ListIndex > length of Rhythms'. Inside this inner loop, the following blocks are executed in order: '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', and 'change RepeatCounter by 1'. A yellow callout box next to the 'wait until' block says 'Order is critical here! The wait until piece must immediately follow the broadcast piece.' The second section starts with a 'when I receive' block for 'Play Single Note Part 1', followed by a 'play note' block: 'play note item ListIndex of Notes for item ListIndex of Rhythms beats'.

*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 frame. 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 yellow callout box next to the 'set' block contains the text 'delay 4 beats'. Below this is a 'wait until' block with the condition 'timer = TriggerNextNote2 or timer > TriggerNextNote2'. This is followed by a 'set RepeatCounter2 to 1' block and a 'repeat until RepeatCounter2 > 2' loop. Inside this loop, there is a 'set ListIndex2 to 1' block, a 'repeat until ListIndex2 > length of Rhythms' loop, and a 'change TriggerNextNote2 by RhythmDelta \* item ListIndex2 of Rhythms' block. This is followed by a 'broadcast Play Single Note Part 2' block, another 'wait until' block with the same condition as the first, a 'change ListIndex2 by 1' block, and a 'change RepeatCounter2 by 1' block. A yellow callout box next to the second 'wait until' block contains the text 'Order is critical here! The wait until piece must immediately follow the broadcast piece.' The script ends with a 'when I receive Play Single Note Part 2' block and a 'play note item ListIndex2 of Notes for item ListIndex2 of Rhythms beats' block.

*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 26 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:
 

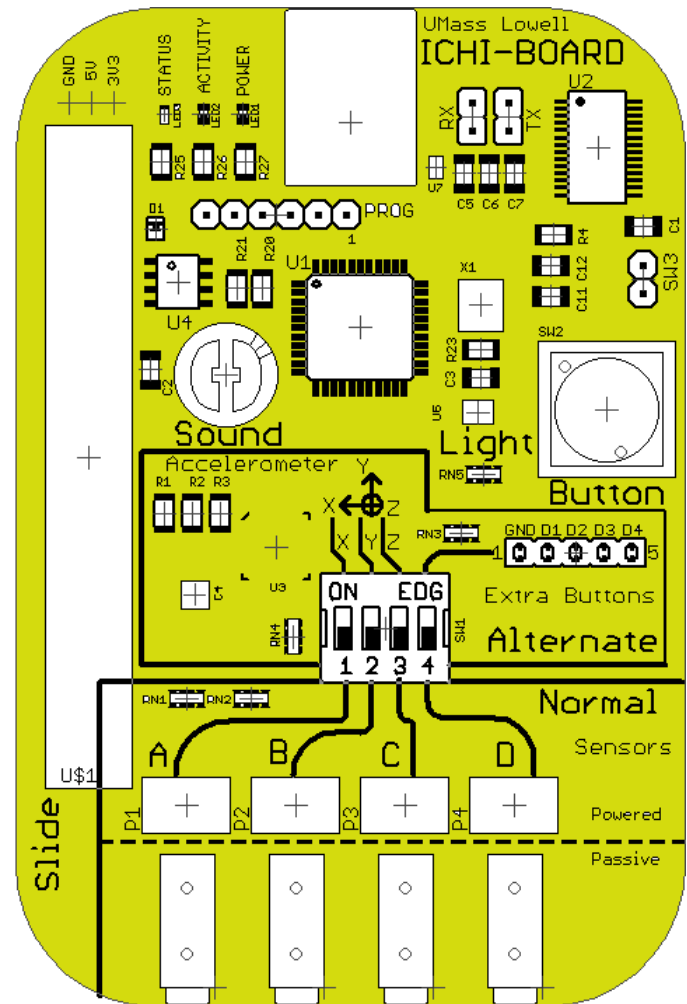
2, 1, 2, 2, 1, 3, 1
  - Create a new list containing these changes, but use the same loop that you created for the previous exercise to play this scale.
  
12. Create a program to play a major chord.
  - A major chord is the 1st, 3rd, and 5th notes of the scale, usually complemented by the octave above 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

```

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

# 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

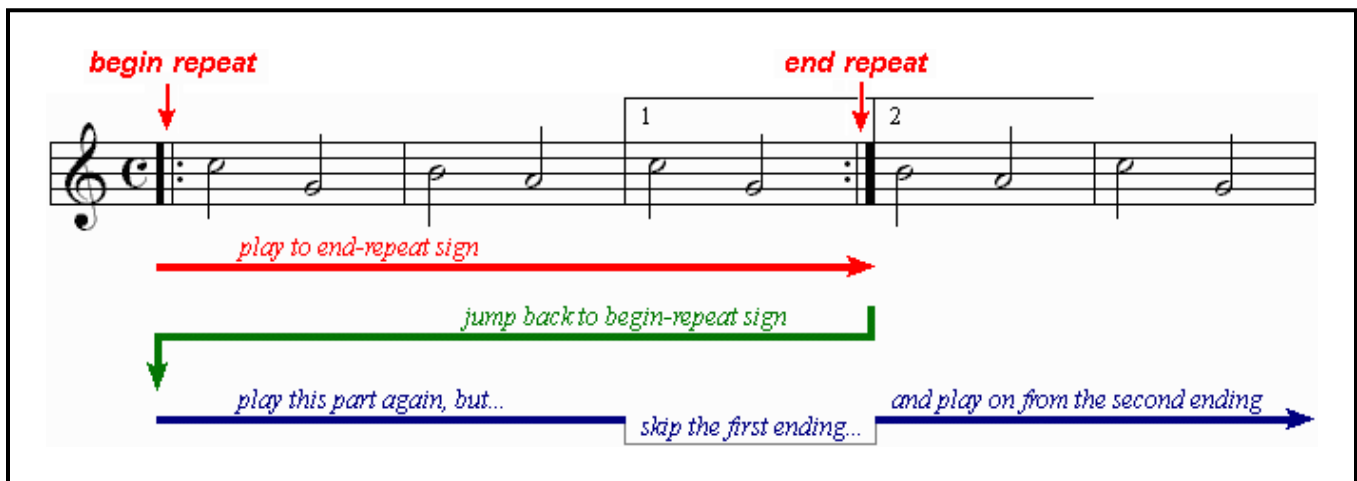
- positive and negative numbers
- real numbers
- decimal notation
- built-in functions with inputs
- angles
- Cartesian coordinates
- trigonometric operators
- random numbers

## 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

## 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  $\text{\%}$ . 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 and then to 3c, even though each of these programs plays exactly the same riff.

## Computing and Music (cont'd)

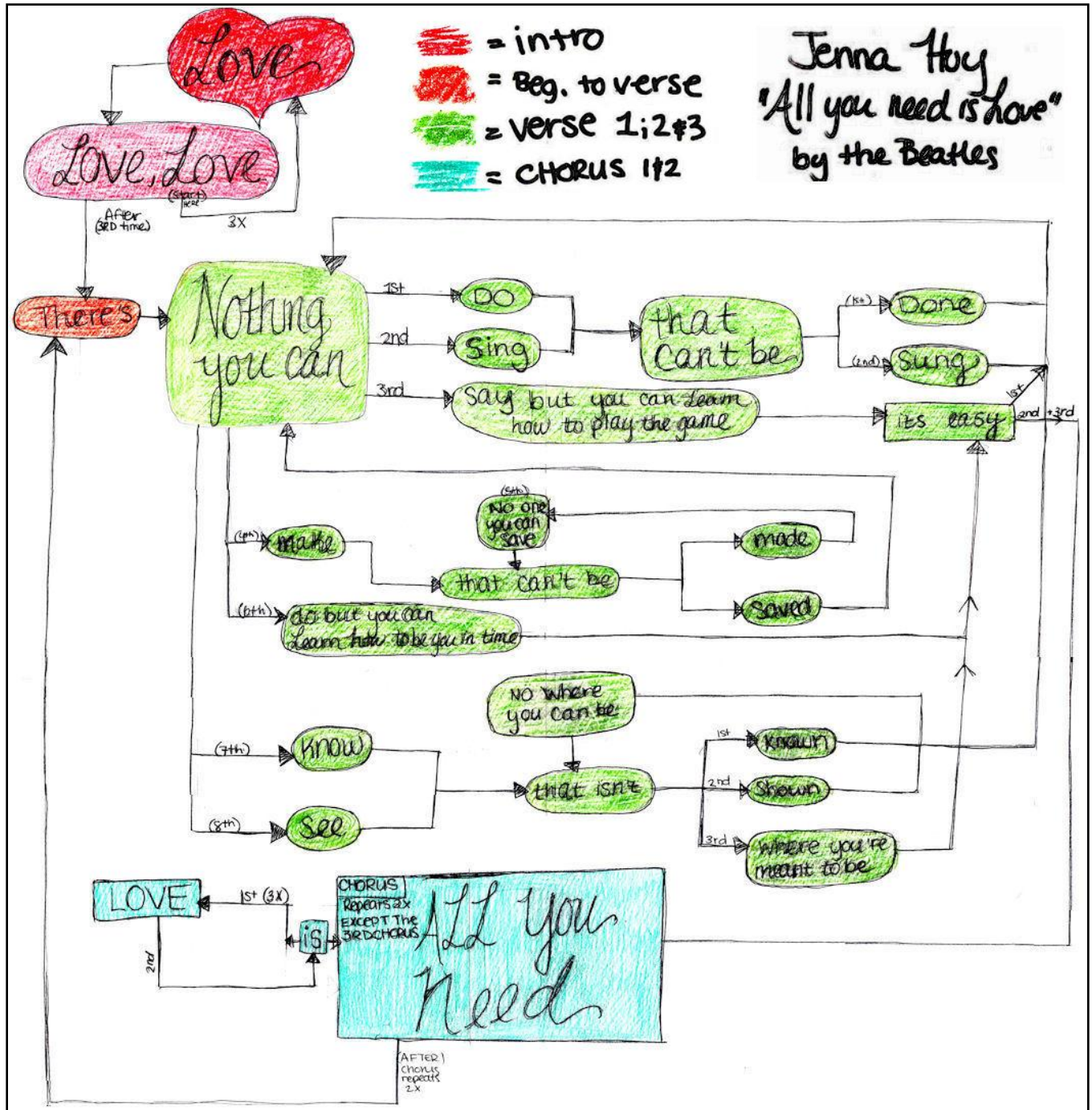
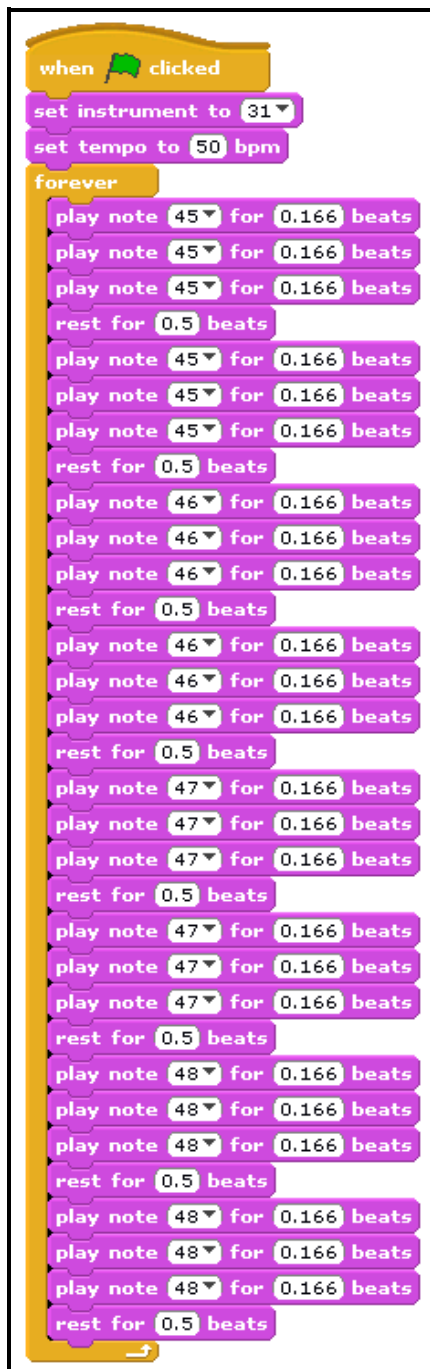


Figure 2. A song flowchart. [1]

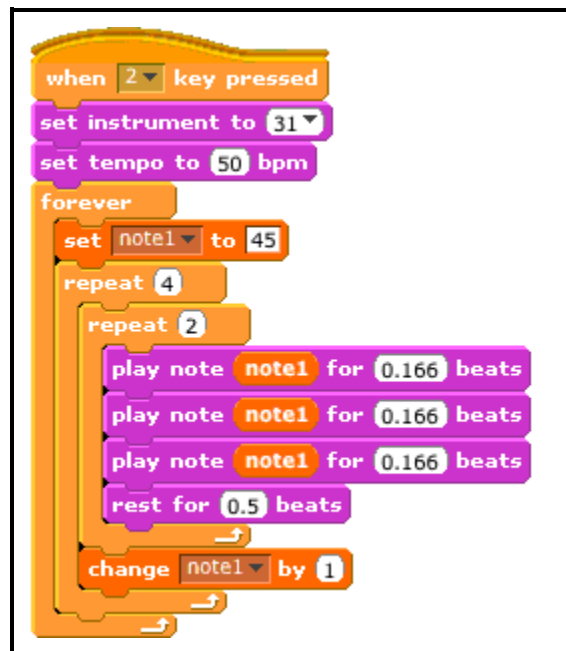
## Computing and Music (cont'd)



```

when clicked
  set instrument to 31
  set tempo to 50 bpm
  forever
    play note 45 for 0.166 beats
    play note 45 for 0.166 beats
    play note 45 for 0.166 beats
    rest for 0.5 beats
    play note 46 for 0.166 beats
    play note 46 for 0.166 beats
    play note 46 for 0.166 beats
    rest for 0.5 beats
    play note 47 for 0.166 beats
    play note 47 for 0.166 beats
    play note 47 for 0.166 beats
    rest for 0.5 beats
    play note 48 for 0.166 beats
    play note 48 for 0.166 beats
    play note 48 for 0.166 beats
    rest for 0.5 beats
  
```

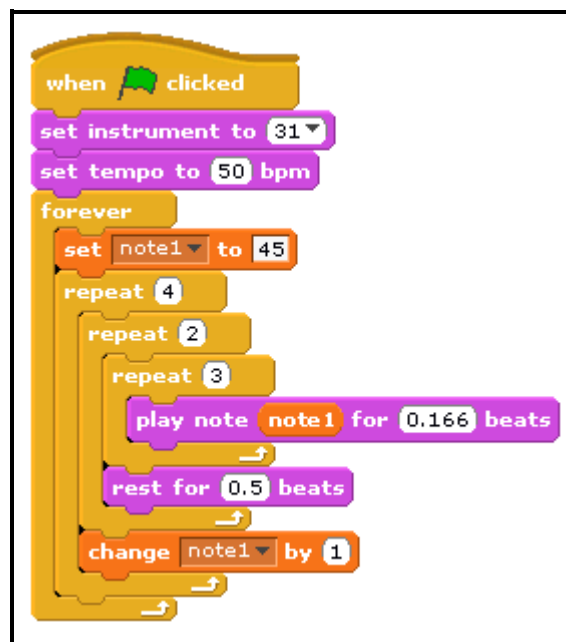
3a



```

when 2 key pressed
  set instrument to 31
  set tempo to 50 bpm
  forever
    set note1 to 45
    repeat 4
      repeat 2
        play note note1 for 0.166 beats
        play note note1 for 0.166 beats
        play note note1 for 0.166 beats
        rest for 0.5 beats
      change note1 by 1
    
```

3b



```

when clicked
  set instrument to 31
  set tempo to 50 bpm
  forever
    set note1 to 45
    repeat 4
      repeat 2
        repeat 3
          play note note1 for 0.166 beats
        rest for 0.5 beats
      change note1 by 1
    
```

3c

Figure 3. Three 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.

## Additional Readings

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.

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

## Additional Readings (cont'd)

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.

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



## Additional Readings (cont'd)

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.

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/Performamatics/>  
<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)