

Guide to Programming with Music Blocks

Music Blocks is a programming environment for children interested in music and graphics. It expands upon Turtle Blocks in that it has a collection of features relating to pitch and rhythm.

The Turtle Blocks guide is a good place to start learning about the basics. In this guide, we illustrate the music features walking the reader through numerous examples.

Getting Started

Music Blocks is designed to run in a browser. Most of the development has been done in Chrome, but it should also work in Firefox (although you may need to disable hardware acceleration). You can run it from [github io](#) or by downloading a copy of the code and running directly from the file system of your computer.

For more details on how to use Music Blocks, see [Using Music Blocks](#) and for more details on how to use Turtle Blocks, see [Using Turtle Blocks JS](#).

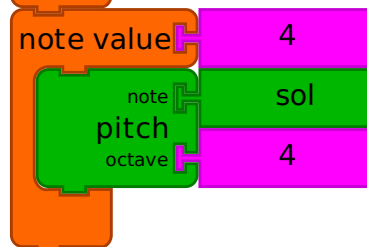
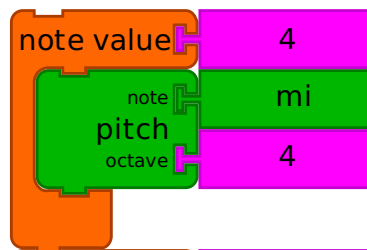
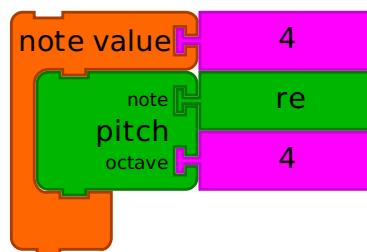
ABOUT THIS GUIDE

Many of the examples given in the guide have links to code you can run. Look for RUN LIVE links.

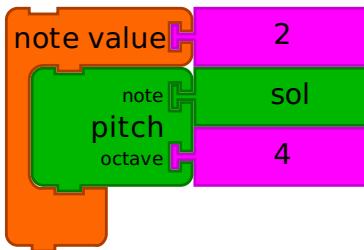
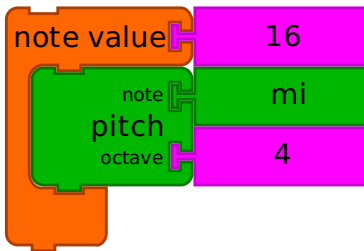
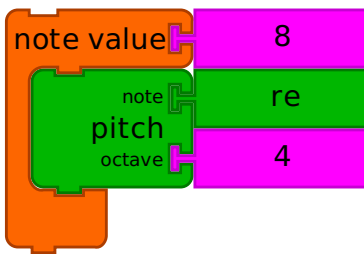
NOTES

Music Blocks exposed the common elements of music: pitch, rhythm, and sonic quality, e.g., loudness and softness, and to some degree, timbre and texture.

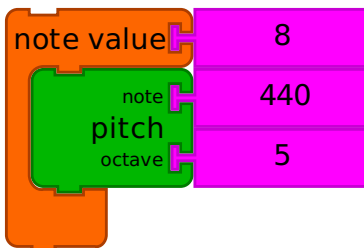
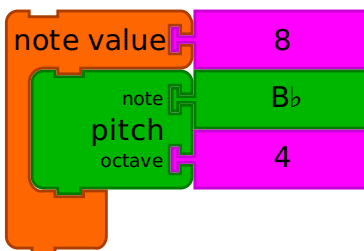
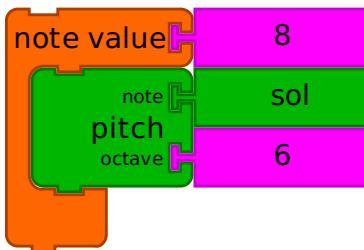
At the heart of Music Blocks is the *Note value* block. The *Note value* block is a container for a pitch that specifies the duration (note value) of the pitch. (The *Pitch* block is detailed below.)



At the top of the example above, a single *Note value* block is shown. The 4 is the note value, in this case, a quarter note. The pitch, specified by the *Pitch* block, contains a pitch, Re at Octave 4. At the bottom, two consecutive notes are shown.



In this example, different note values are shown. From top to bottom, 8 for an eighth note, 16 for a sixteenth note, and 2 for a half note.

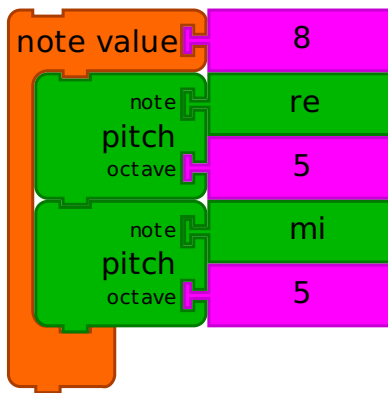


As we have seen, *Pitch* blocks are used inside the *Note value* blocks. The *Pitch* block specifies a pitch name and pitch octave that in combination determine the frequency at which a note is played.

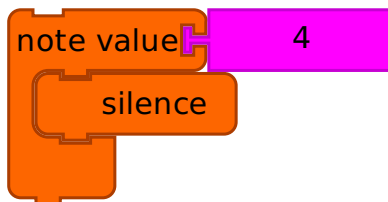
You can plug different values into the *Pitch* block name and octave slots. Some examples are shown above. Starting from the top, the pitch name block is specified using a *Solfège* block (*Sol* in *Octave 6*); the pitch name is specified using a *Ppppppitch-name* block (*B flat* in *Octave 4*); the pitch name is specified using a *Number* block (*440* Hertz).

The octave is specified using a number block and is restricted to whole numbers. In the case where the pitch name is specified by frequency, the octave is ignored.

Note that the pitch name can also be specified using a *Text* block.

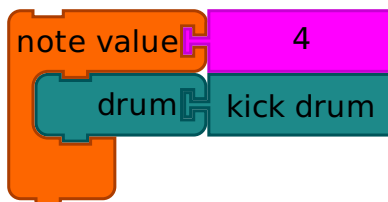


A chord (multiple, simultaneous pitches) can be specified by add multiple *Pitch* blocks to a *Note value* container.

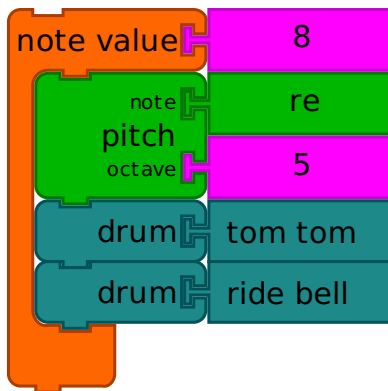


A rest of duration note value can be constructed using a *Silence* block.

Using drums

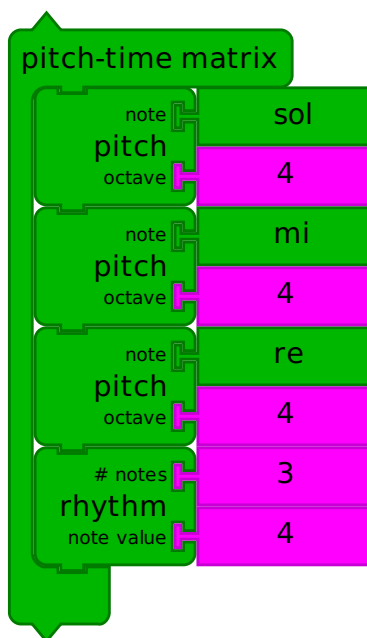


Anywhere you can use a *Pitch* block—e.g., inside of the matrix or a *Note value* block—you can also specify a drum sample. Currently there about two dozen different samples from which to choose. The default drum is a kick drum.



Just as in the chord example above, you can use multiple *Drum* blocks within a single *Note value* block and combine them with *Pitch* blocks.

THE PITCH-TIME MATRIX



Music Blocks provides a widget, the *Pitch-time Matrix*, as a scaffold for getting started.

Once you've launched Music Blocks in your browser, start by clicking on the *Pitch-time Matrix* stack that appears in the middle of the screen. (For the moment, ignore the *Start* block.) You'll see a grid organized vertically by pitch and horizontally by rhythm.

Solfa				
sol ₄				
mi ₄				
re ₄				
rhythmic note values	1/4	1/4	1/4	

The matrix in the figure above has three *Pitch* blocks and one *Rhythm* block, which is used to create a 3 x 3 grid of pitch and time.

Note that the default matrix has five *Pitch* blocks, hence, you will see five rows, one for each pitch. (A sixth row at the bottom is used for specifying the rhythms associated with each note.) Also by default, there are two *Rhythm* blocks, which specifies six quarter notes followed by one half note. Since the *Rhythm* blocks are inside of a *Repeat* block, there are fourteen (2 x 7) columns for selecting notes.

Solfa				
sol ₄				
mi ₄				
re ₄				
rhythmic note values	1/4	1/4	1/4	

By clicking on individual cells in the grid, you should hear individual notes (or chords if you click on more than one cell in a column). In the figure, three quarter notes are selected (black cells). First **Re 4**, followed by **Mi 4**, followed by **Sol 4**.



If you click on the *Play* button (found in the top row of the grid), you will hear a sequence of notes played (from left to right): **Re 4**, **Mi 4**, **Sol 4**.



Once you have a group of notes (a "chunk") that you like, click on the *Save* button (just to the right of the *Play* button). This will create a stack of blocks that can be used to play these same notes programmatically. (More on that below.)

You can rearrange the selected notes in the grid and save other chunks as well.



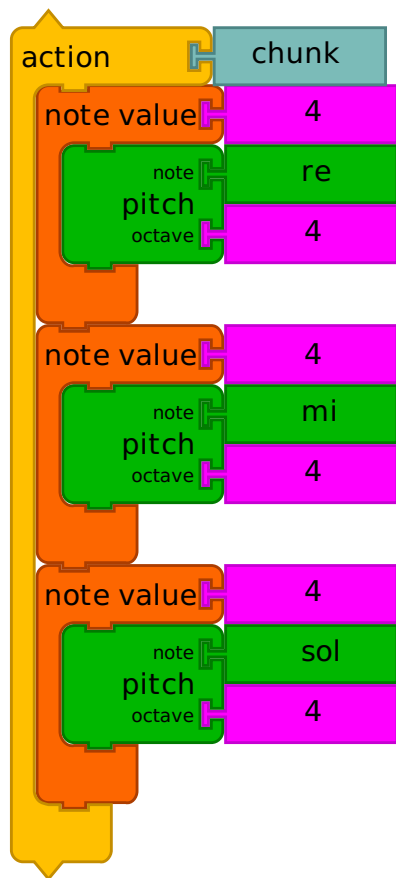
Or hide the matrix by clicking on the *Close* button (the right-most button in the top row of the grid.)



There is also an Erase button that will clear the grid.

Don't worry. You can reopen the matrix at anytime (it will remember its previous state) and since you can define as many chunks as you want, feel free to experiment.

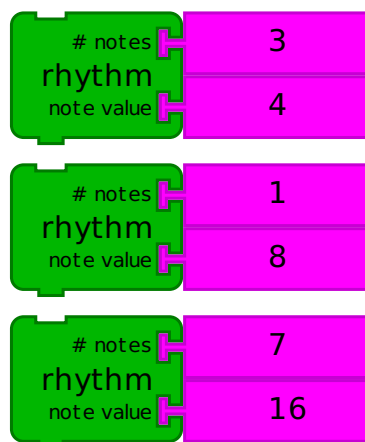
Tip: You can put a chunk inside a *Pitch-time Matrix* block to generate the matrix to corresponds to that chunk.



The chunk created when you click on the matrix is a stack of blocks. The blocks are nested: an *Action* block contains three *Note value* blocks, each of which contains a *Pitch* block. The *Action* block has a name automatically generated by the matrix, in this case, chunk. (You can rename the action by clicking on the name.). Each note has a duration (in this case 4, which represents a quarter note). Try putting different numbers in and see (hear) what happens. Each note block also has a pitch block (if it were a chord, there would be multiple *Pitch* blocks nested inside the Note block's clamp). Each pitch block has a pitch name (**Re** , **Mi** , and **Sol**), and a pitch octave; in this example, the octave is 4 for each pitch. (Try changing the pitch names and the pitch octaves.)

To play the chuck, simply click on the action block (on the word action). You should hear the notes play, ordered from top to bottom.

About the Rhythm Block



Rhythm blocks are used to generate rhythm patterns in the *Pitch-time Matrix* block. The top argument to the *Rhythm* block is the number of notes. The bottom argument is the duration of the note. In the top example above, three columns for quarter notes would be generated in the matrix. In the middle example, one column for an eighth note would be generated. In the bottom example, seven *columns for 16th notes would be generated.

pitch-time matrix

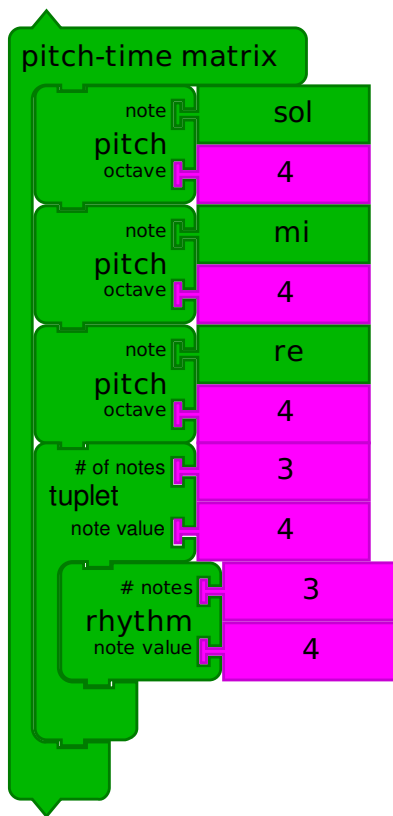
note	sol
pitch octave	4
note	mi
pitch octave	4
note	re
pitch octave	4
# notes	3
rhythm note value	4
# notes	6
rhythm note value	8

Solfa

sol ₄									
mi ₄									
re ₄									
rhythmic note values	1/4	1/4	1/4	1/8	1/8	1/8	1/8	1/8	1/8

You can use as many *Rhythm* blocks as you'd like inside the *Pitch-time Matrix* block. In the above example, two *Rhythm* blocks are used, resulting in three quarter notes and six eighth notes.

Creating Tuplets

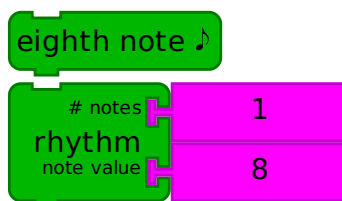
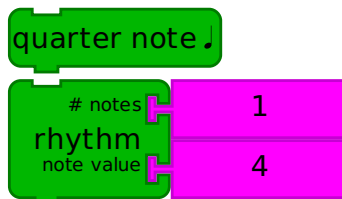
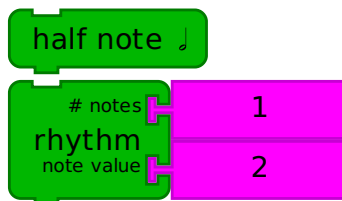


Solfa					
sol ₄					
mi ₄					
re ₄					
tuplet note values	1/12	1/12	1/12		
tuplet value	3				
rhythmic note values	1/4				

Tuplets are a collection of notes that get scaled to a specific duration. Using tuplets makes it easy to create groups of notes that are not based on a power of 2. In the example above, three quarter notes—defined in the *Rhythm* block—are played in the time of a single quarter note. The result is three twelfth notes.

You can mix and match *Rhythm* blocks and *Tuplet* blocks when defining your matrix.

Using individual notes in the matrix

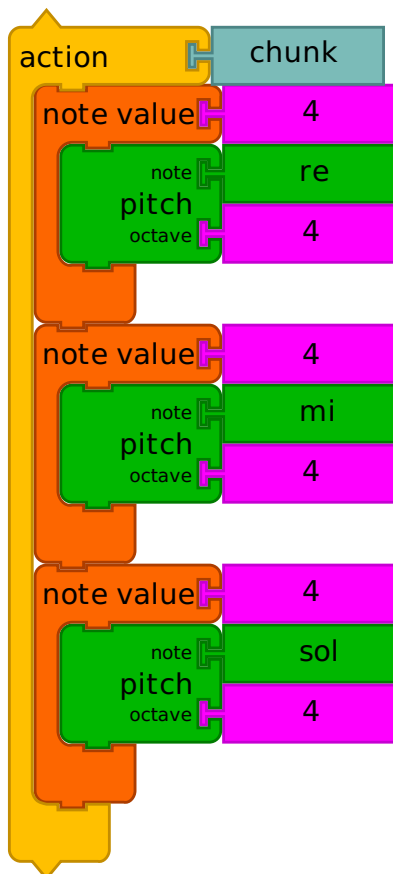


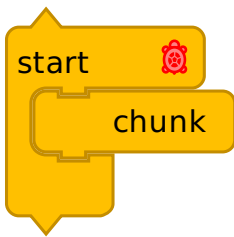
You can also use individual notes when defining the grid. These blocks will expand into *Rhythm* blocks with corresponding values.

PROGRAMMING WITH MUSIC

The remainder of this guide discusses how to use the chunks created by the *Pitch-time Matrix* when programming (You can also program with chunks you create and/or modify by hand).

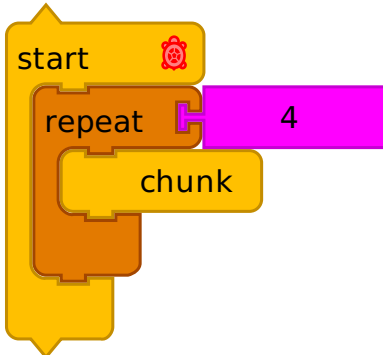
1. A chunk of notes



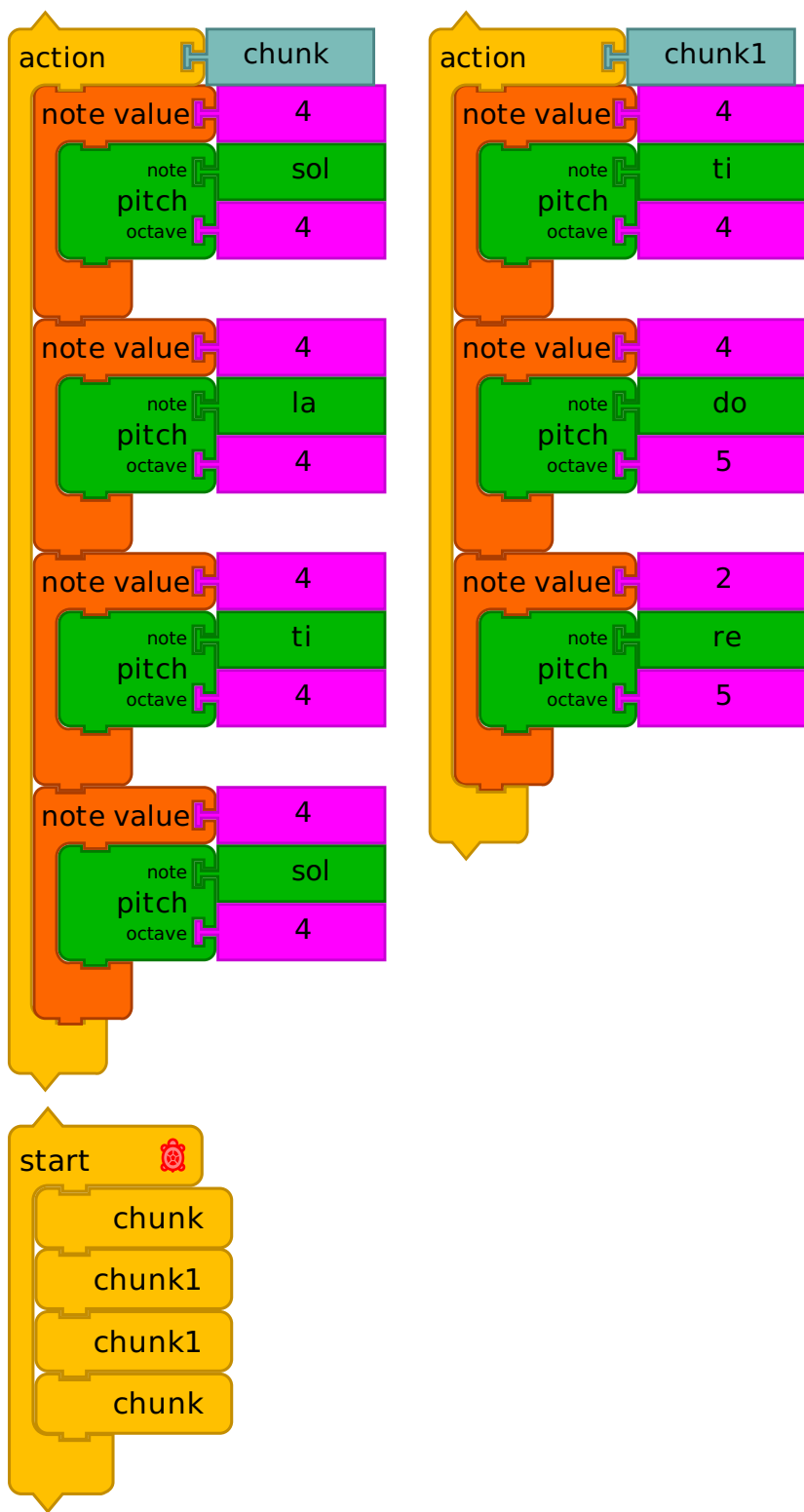


Every time you create a new *Action* stack, Music Blocks creates a new block specific to that stack. (The new block is found at the top of the *Block* palette, found on the left edge of the screen.) Clicking on this block is the same as clicking on your stack. By default, the new blocks are named `chunk`, `chunk1`, `chunk2` ... but you can rename them by editing the labels on the *Action* blocks.

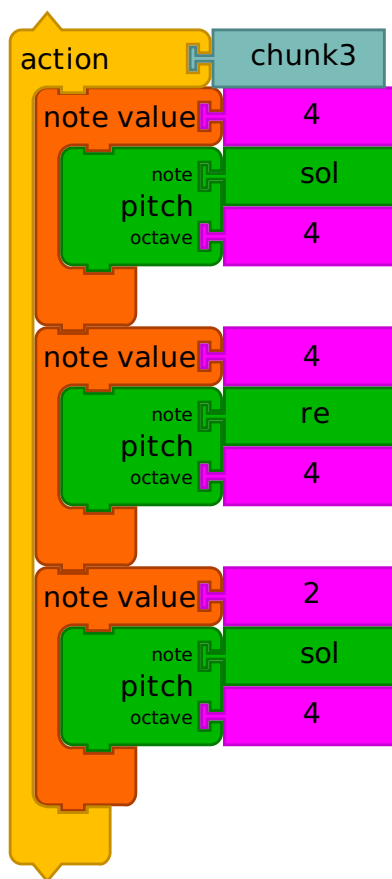
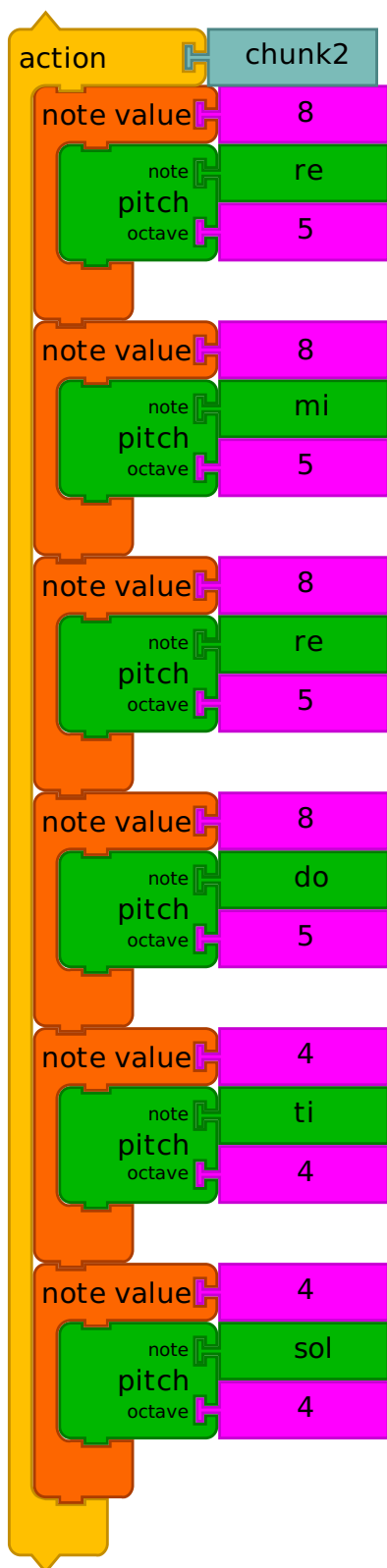
In the example above, the *Chunk* block is inside of a *Start* block, which ties it to the *Run* button in the upper-left corner of the screen (the “rabbit”). Try clicking on the *Run* button. Also try the *Run Slow* button (the “turtle”) and the *Step* button (the “snail”), which steps through the program one block per button press. There are also buttons for playing the music back slowly and for stepping one note per button press.

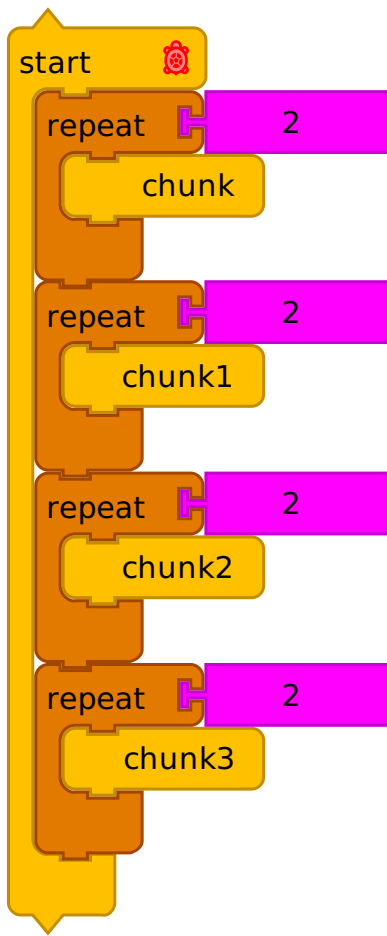


You can repeat chunks either by using multiple *Chunk* blocks or using a *Repeat* block.



You can also mix and match chunks. Here we play chunk, followed by chunk1 twice, and then chunk again.

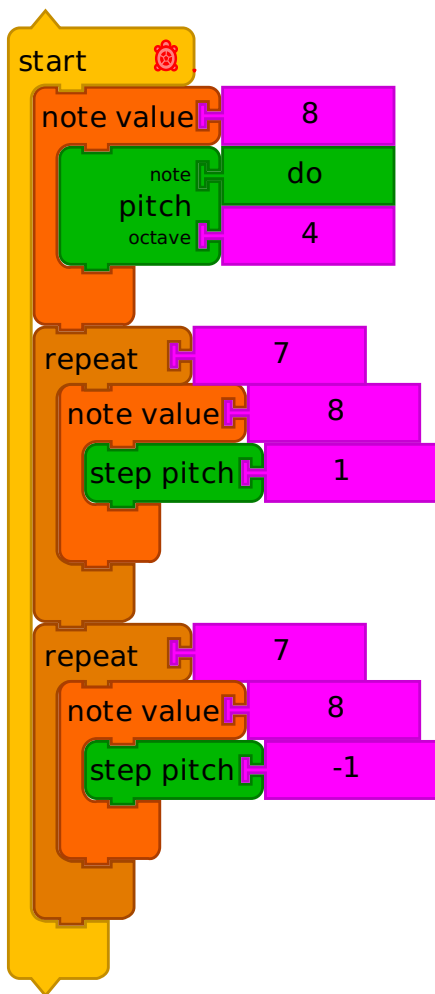




A few more chunks and we can make a song. (Can you read the block notation in order to guess what song we've programmed?)

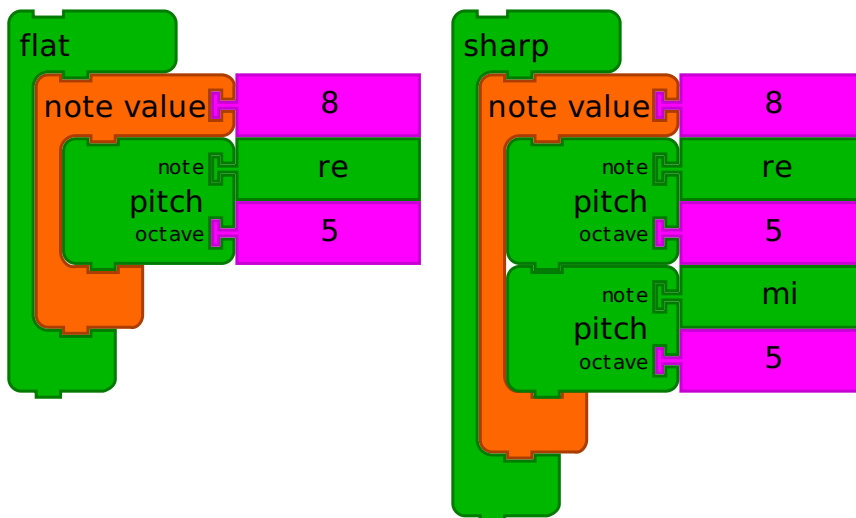
2. Transformations

There are many ways to transform pitch, rhythm, and other qualities of the sound.

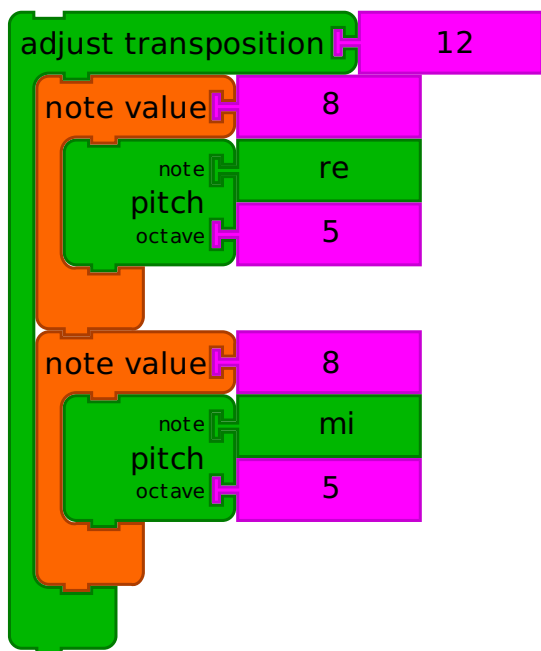


The *Step Pitch* block will move up or down notes in a scale from the current note. In the example above, *Step Pitch* blocks are used inside of *Repeat* blocks to play up and down a scale.

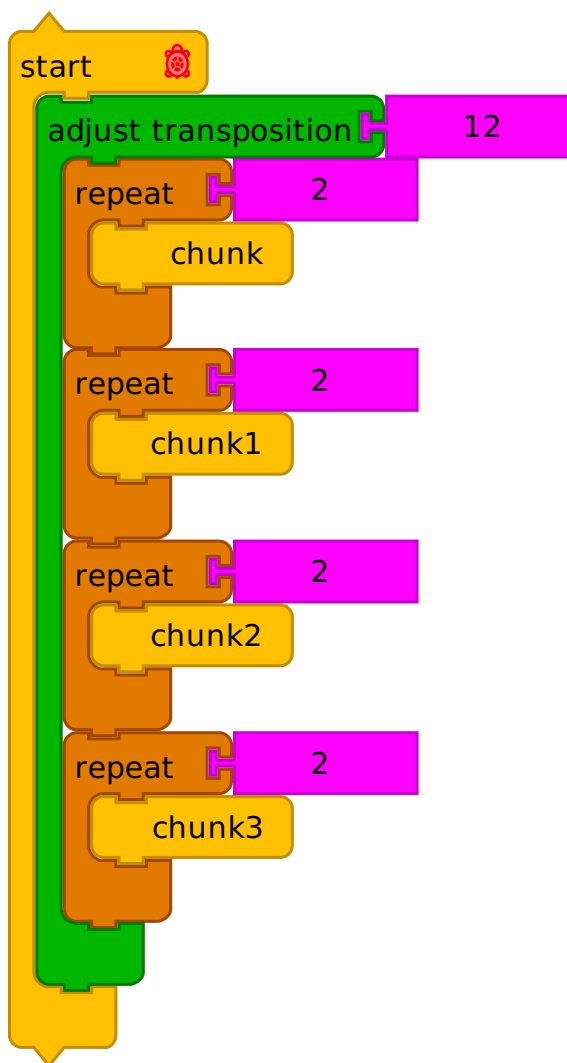
[RUN LIVE](#)



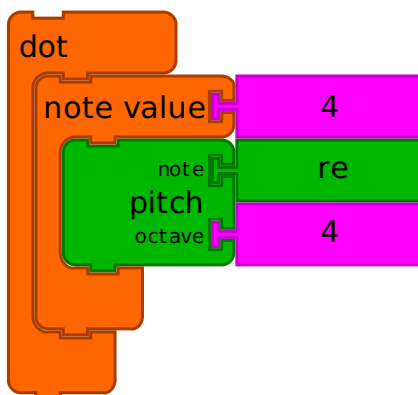
The *Sharp* and *Flat* blocks can be wrapped around *Pitch* blocks, *Note value* blocks, or chunks. A sharp will raise the pitch by one half step. A flat will lower by one half step. In the example, on the left, just the *Pitch* block **Mi** is lowered by one half step; on the right, both pitch blocks are raised by one half step.



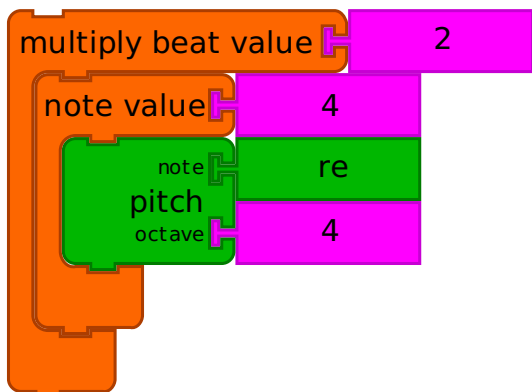
The *Adjust-transposition* block can be used to make larger shifts in pitch. To shift an entire octave, transpose by 12 half-steps up. -12 will shift an octave down.



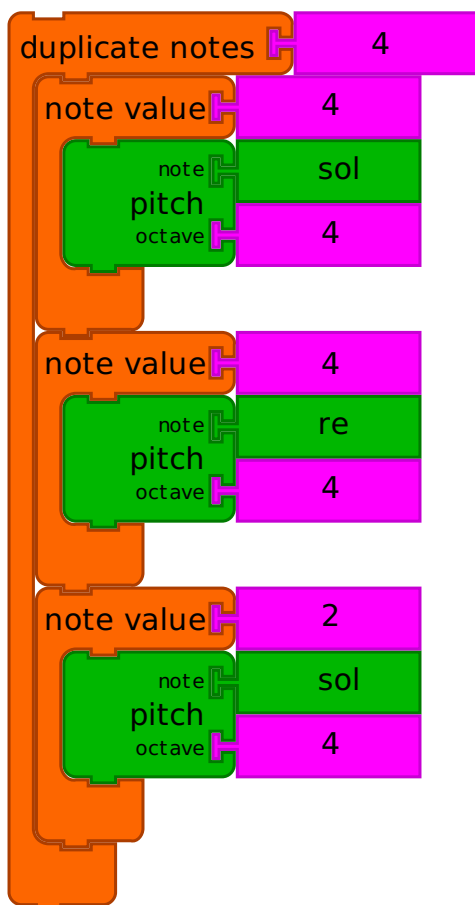
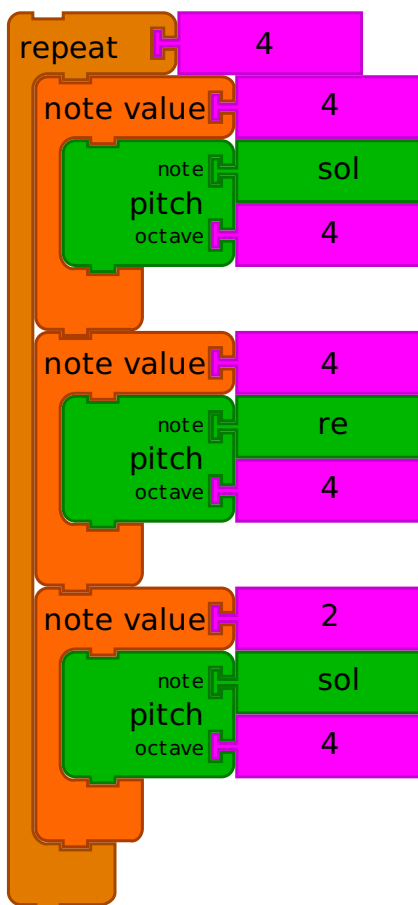
In the example above, we take the song we programmed previously and raise it by one octave.



You can “dot” notes using the *Dot* block. A dotted note extends by 50%. E.g., a dotted quarter note will play for 3/8 ($\frac{1}{4} + \frac{1}{8}$) of a beat. A dotted eighth note will play for 3/16 ($\frac{1}{8} + \frac{1}{16}$) of a beat.

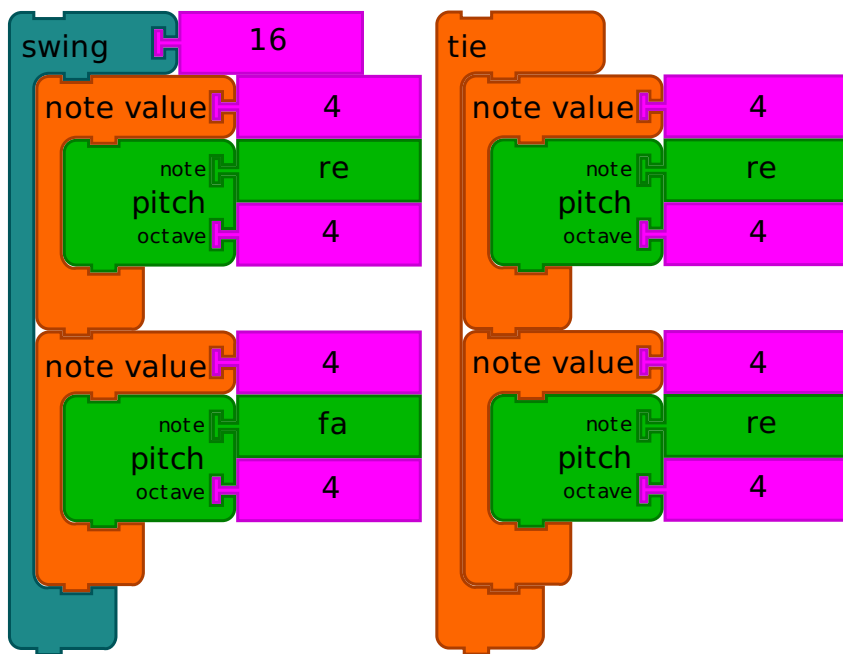


You can also multiply (or divide) the beat value, which will speed up or slowdown the notes.



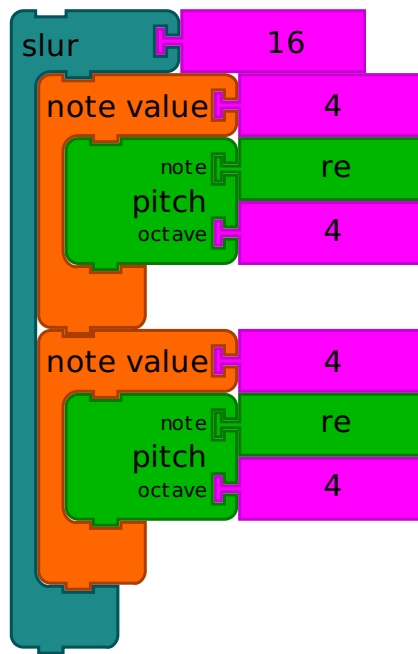
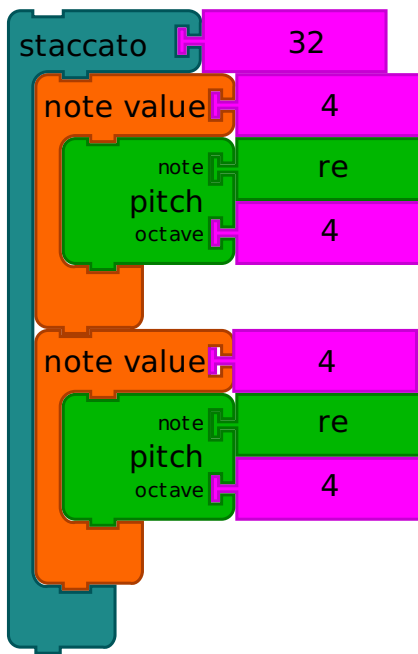
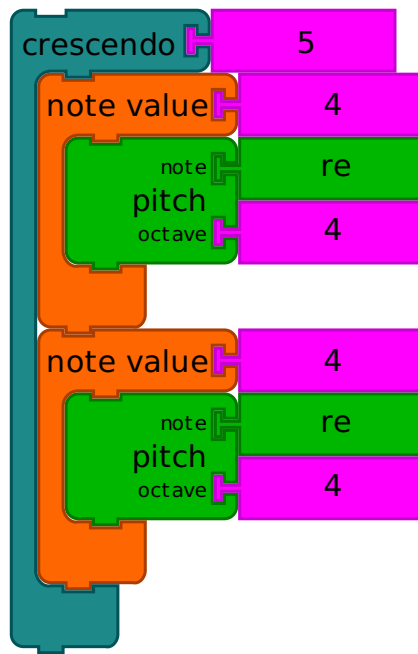
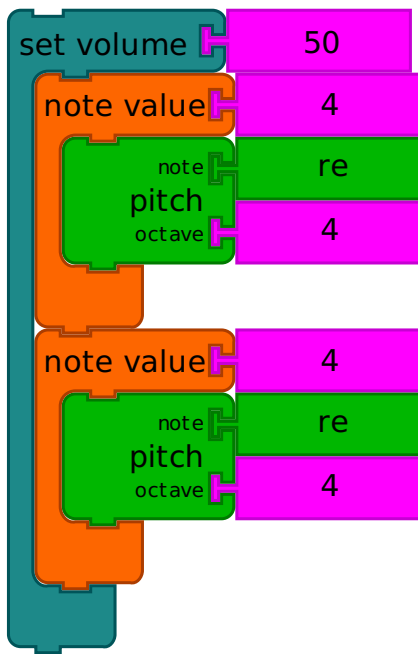
There are several ways to repeat notes. The *Repeat* block will play a sequence of notes multiple times; the *Duplicate* block will repeat each note in a sequence.

In the example, on the left, the result would be Sol, Re, Sol, Sol, Re, Sol, Sol, Re, Sol, Sol, Re, Sol ; on the right the result would be Sol, Sol, Sol, Sol, Re, Re, Re, Re, Sol, Sol, Sol, Sol .



The *Swing* block works on pairs of notes, adding some duration to the first note and taking the same amount from the second note.

Tie also works on pairs of notes, combining them into one note. (The notes must be identical in pitch, but can vary in rhythm.)

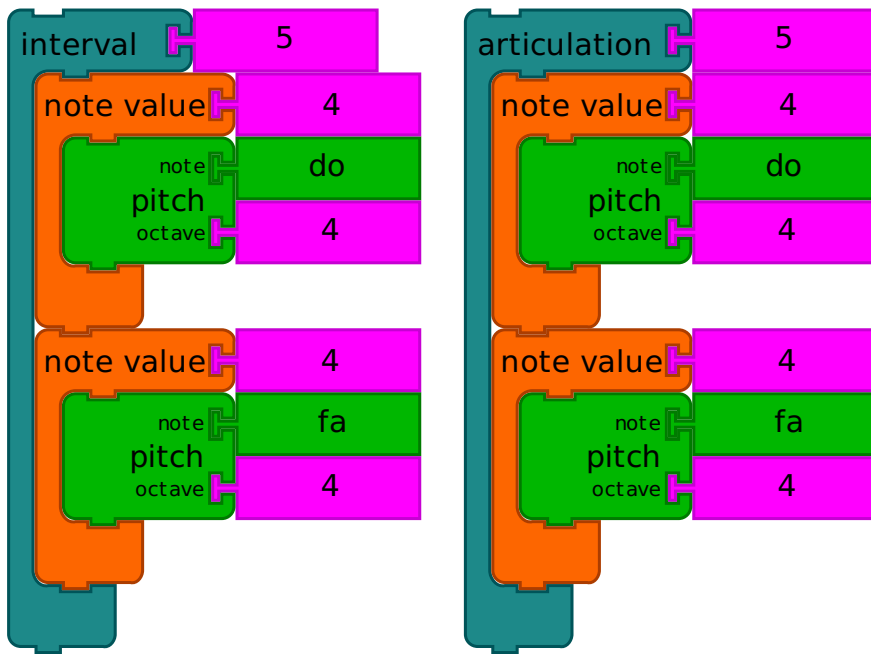


The *Set volume* block will change the volume of the notes. The default is 50; the range is 0 (silence) to 100 (full volume).

The *Crescendo* block will increase (or decrease) the volume of the contained notes by an amount specified.

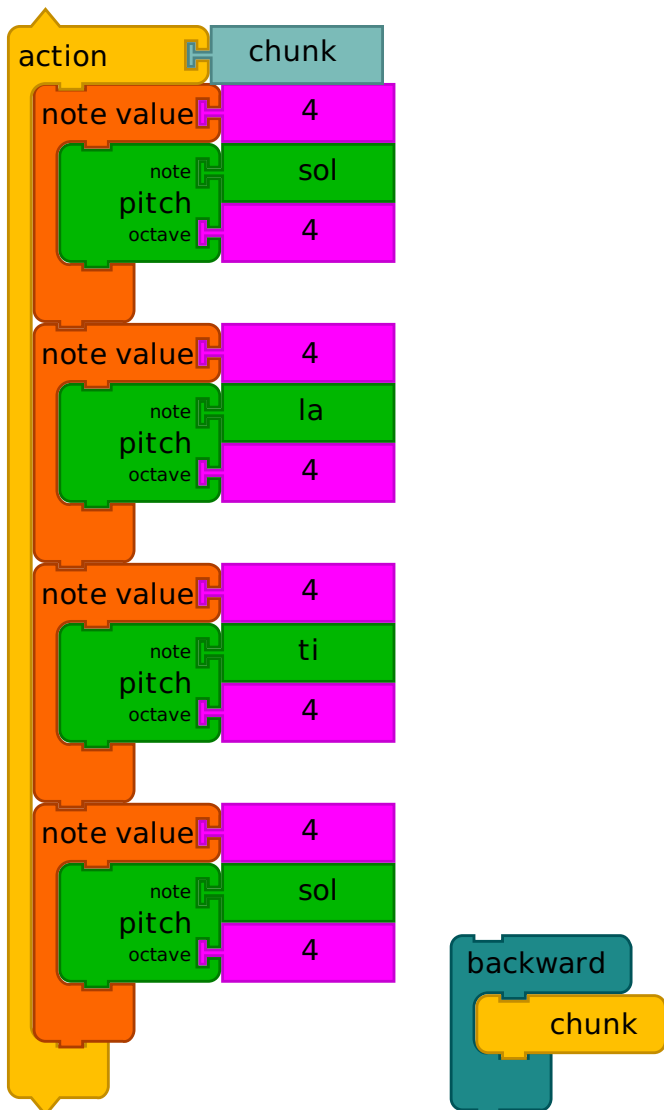
The *Staccato* block will play back notes in tight bursts while maintaining the specified rhythmic value of the notes.

The *Slur* block will run a note past its noted duration, blending it into the next note.



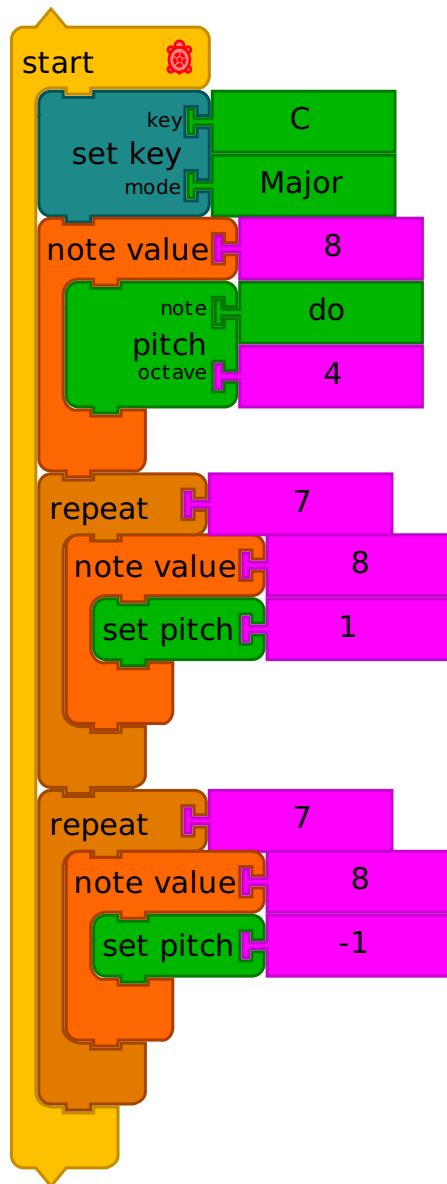
The *Interval* block calculates an interval, e.g., a fifth, and adds the additional pitches to a note. In the figure, we add **Sol** to **Do** and **Do** to **Fa**.

The *Articulation* block changes the volume of a group of notes.

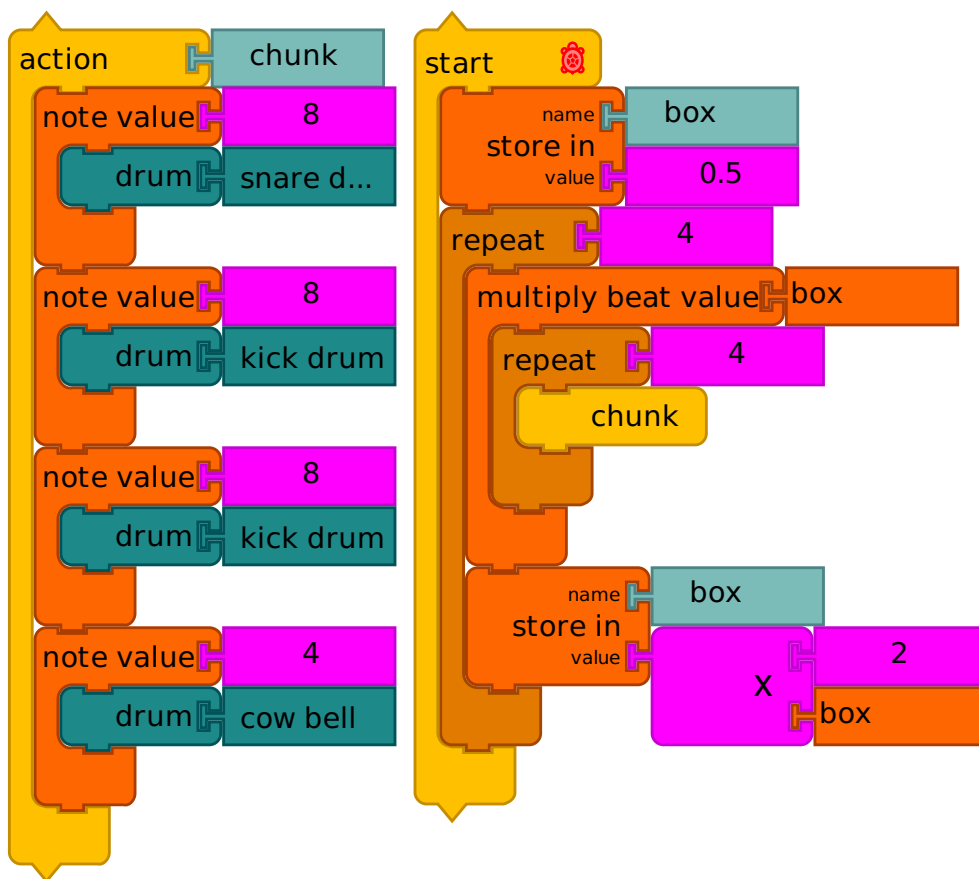


The *Backward* block will play the contained notes in reverse order. In the example above, the notes in *Chunk* are played as **Sol, Ti, La, Sol**, e.g., from the bottom to the top of the stack.

Note that *all* of the blocks inside a *Backward* block are reverse, so use this feature with caution if you include logic intermixed with notes.



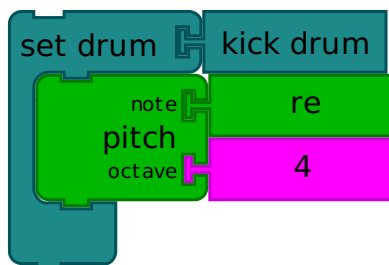
The *Set Key* block will change the key and mode of the mapping between solfege, e.g., **Do**, **Re**, **Mi**, to note names, e.g., **C**, **D**, **E**, when in C Major. Modes include Major and Minor, Chromatic, and a number of more exotic modes, such as Bebop, Geez, Maqam, et al.



In the above example, the sequence of drum beats is increased over time.

[RUN LIVE](#)






















The Pitch Drum Matrix

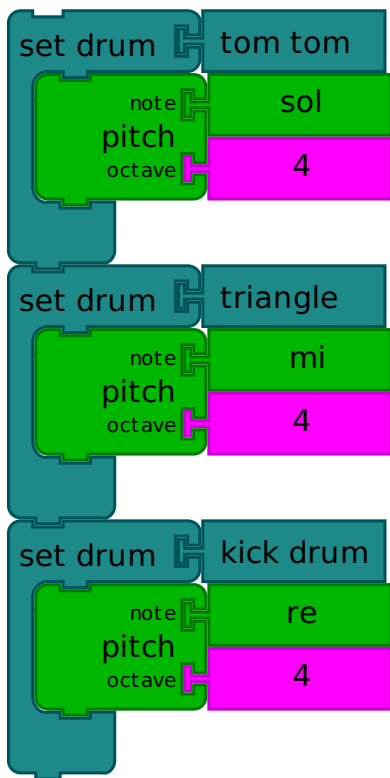


The *Set Drum* block is used to map the enclosed pitches into drum sounds. Drum sounds are played in a monopitch using the specified drum sample. In the example above, a **kick drum** will be substituted for each occurrence of a **Re 4**.

pitch-drum matrix

note	sol
pitch	
octave	4
note	mi
pitch	
octave	4
note	re
pitch	
octave	4
drum	kick drum
drum	tom tom
drum	ride bell
drum	triangle

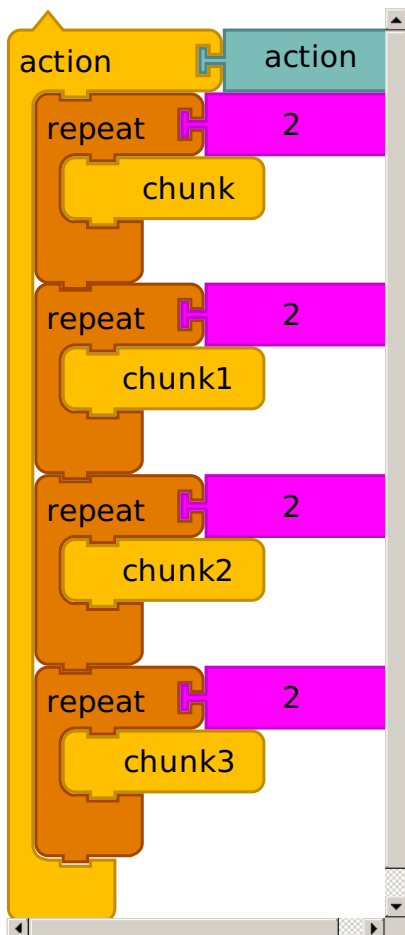
Solfa					
sol ₄					
mi ₄					
re ₄					
					
Solfa					
sol ₄					
mi ₄					
re ₄					
					



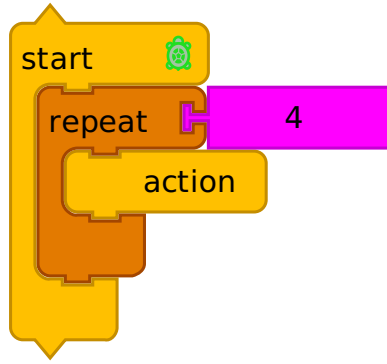
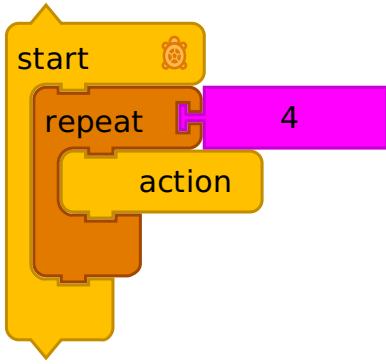
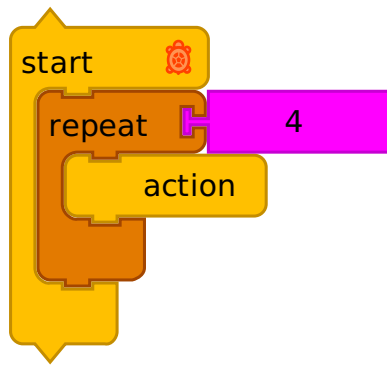
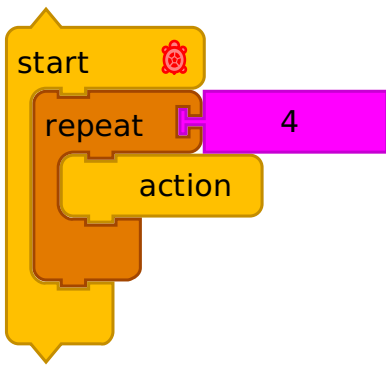
As an expedience for creating mapping with the *Set Drum* block, we provide the *Drum-Pitch* Matrix. You use it to map between pitches and drums. The output is a stack of *Set Dum* blocks.

3. Voices

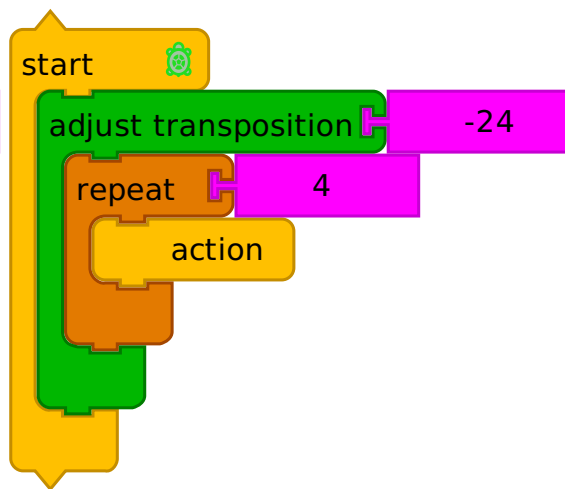
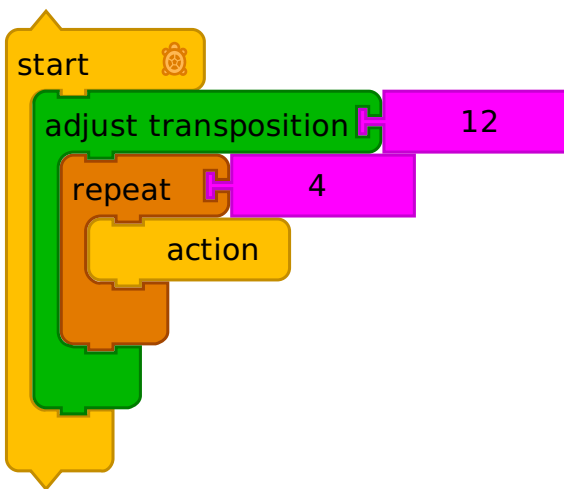
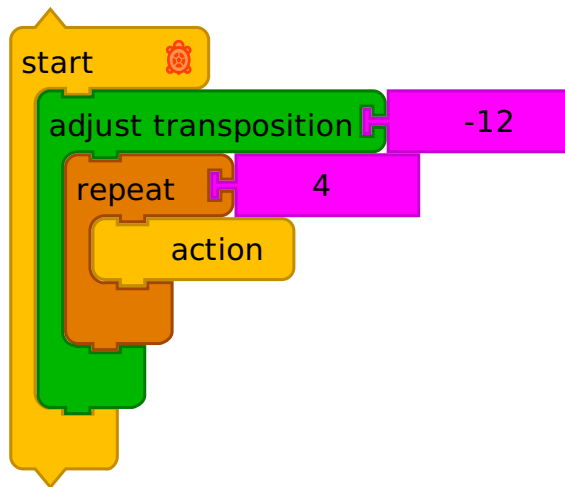
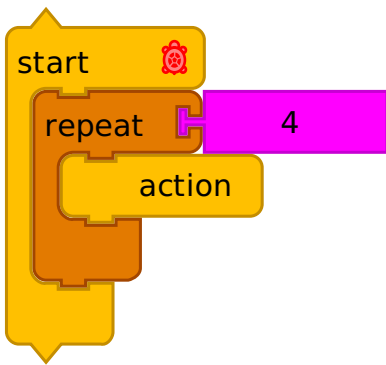
Each *Start* block runs as a separate voice in Music Blocks. (When you click on the Run button, all of the *Start* blocks are run concurrently.)



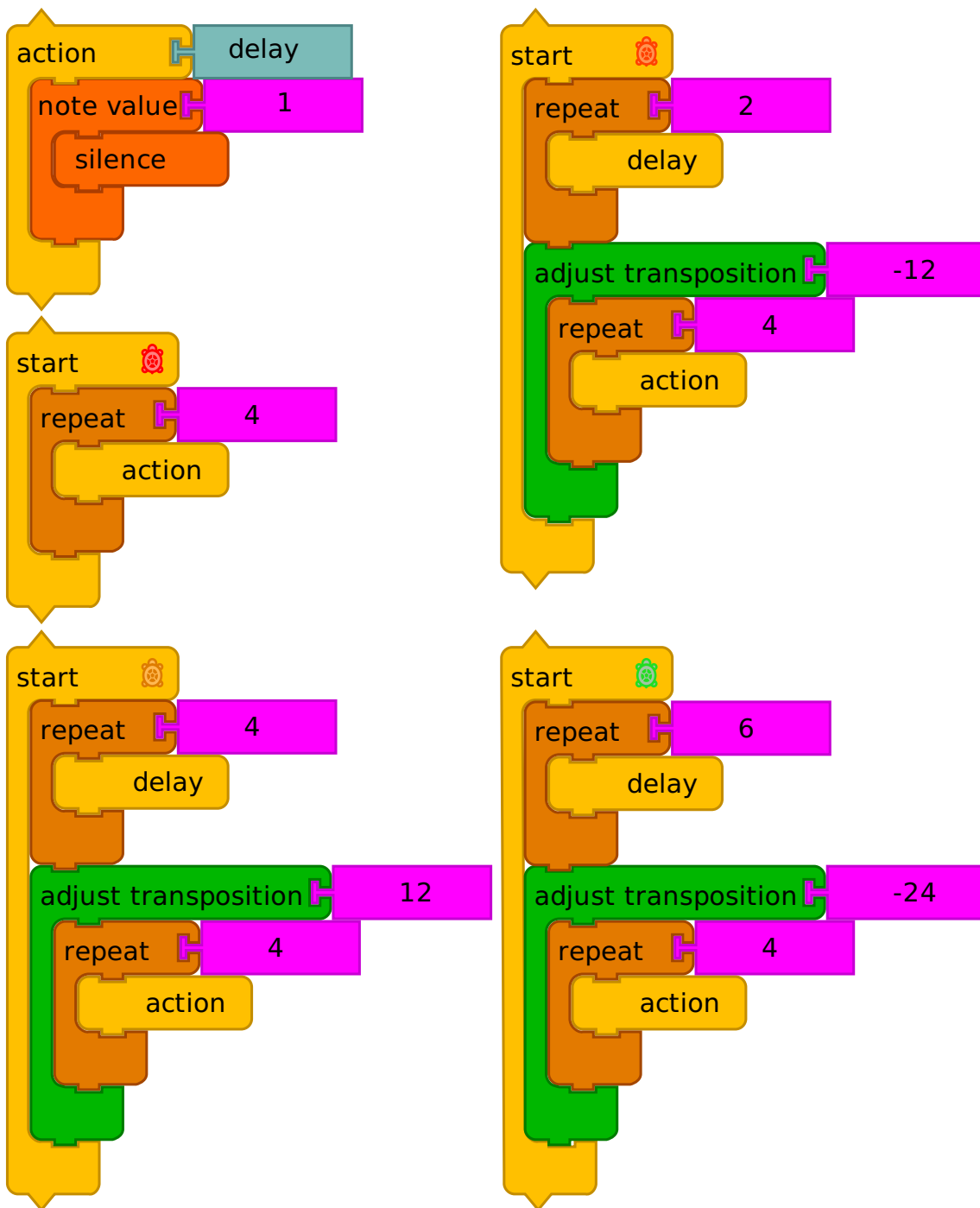
If we put our song into an action...



...we can run it from multiple *Start* blocks.

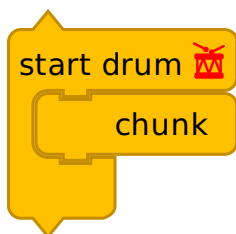


It gets more interesting if we shift up and down octaves.



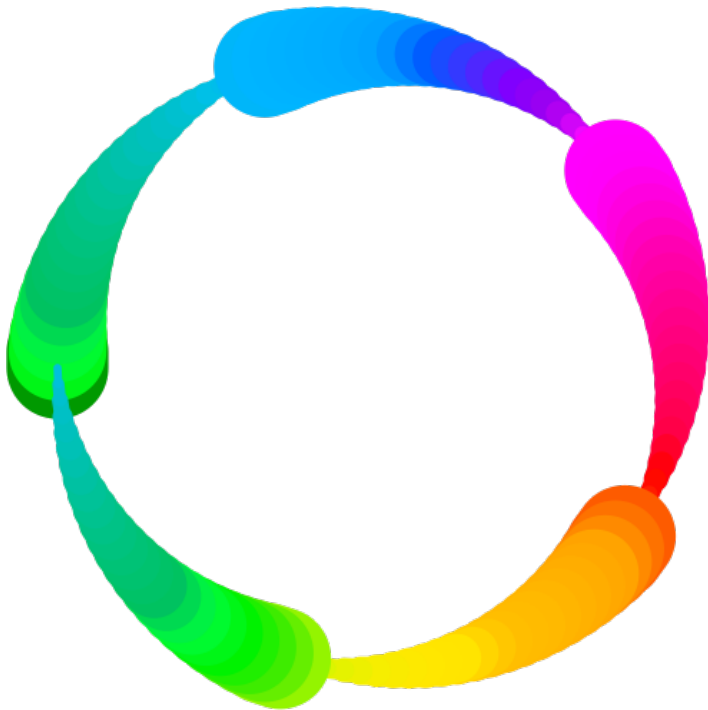
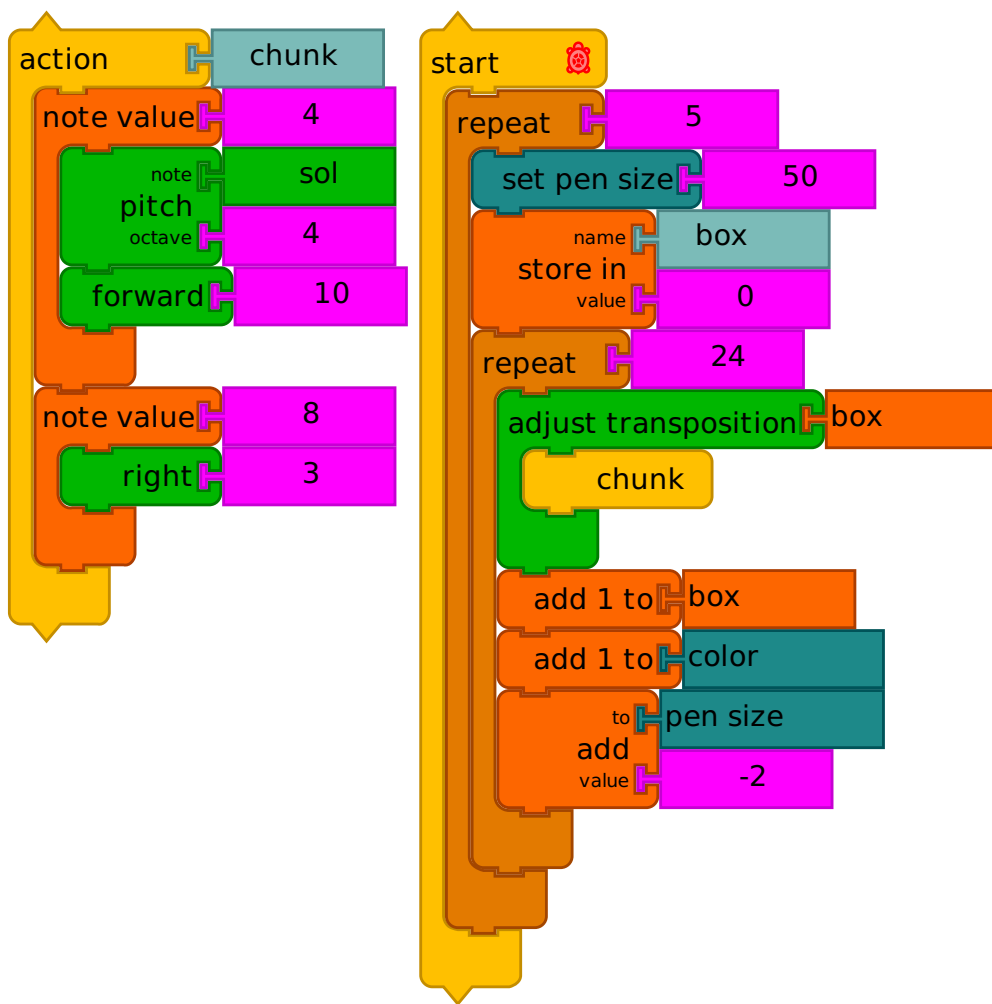
And even more interesting if we bring the various voices offset in time.

[RUN LIVE](#)

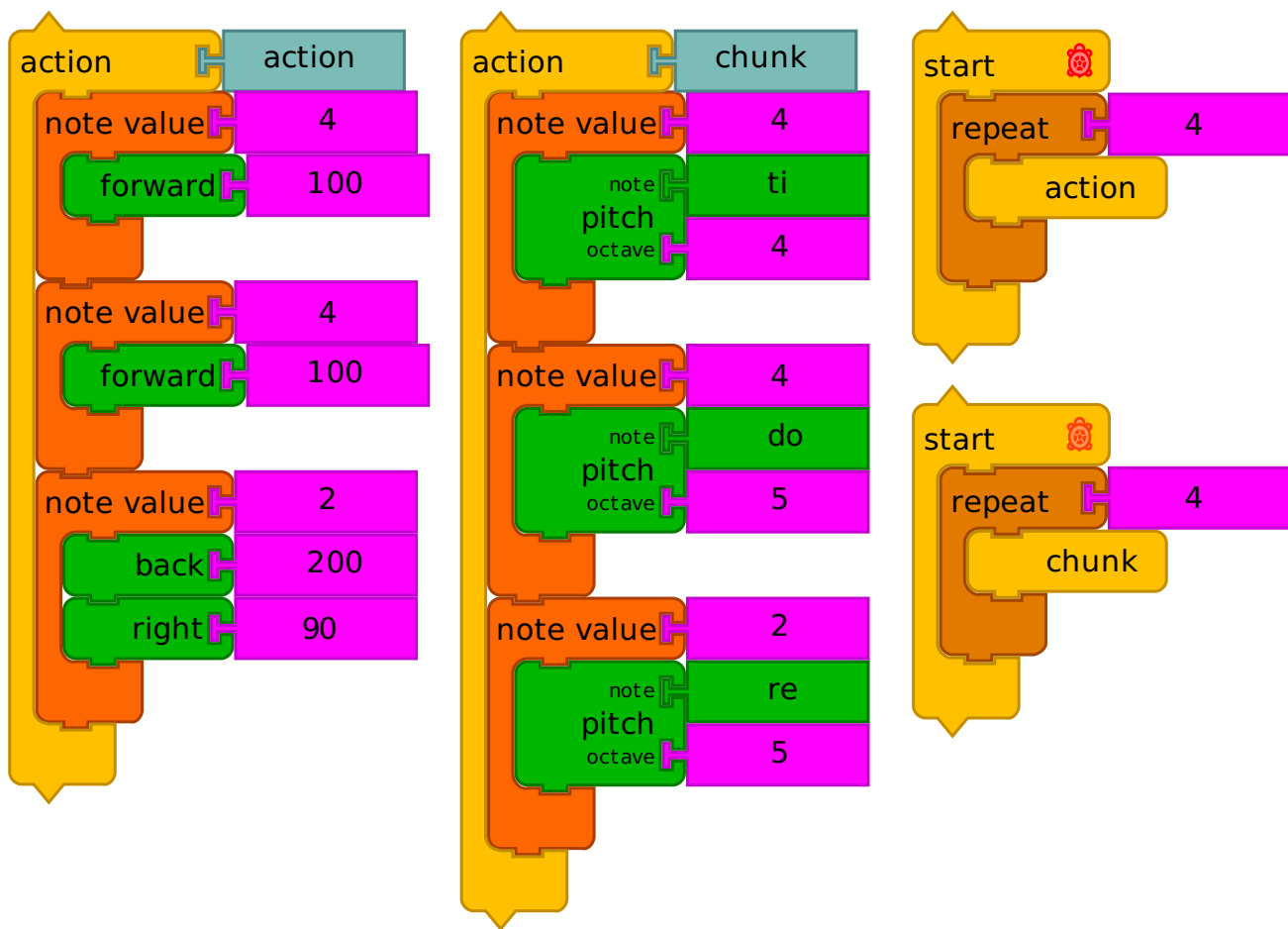


A special "drum" version of the *Start* block is available for laying down a drum track. Any pitch blocks encountered while starting from a drum will be played as **C2** with the default drum sample. In the example above, all of the notes in **chunk** will be played with a kick drum.

4. Adding graphics

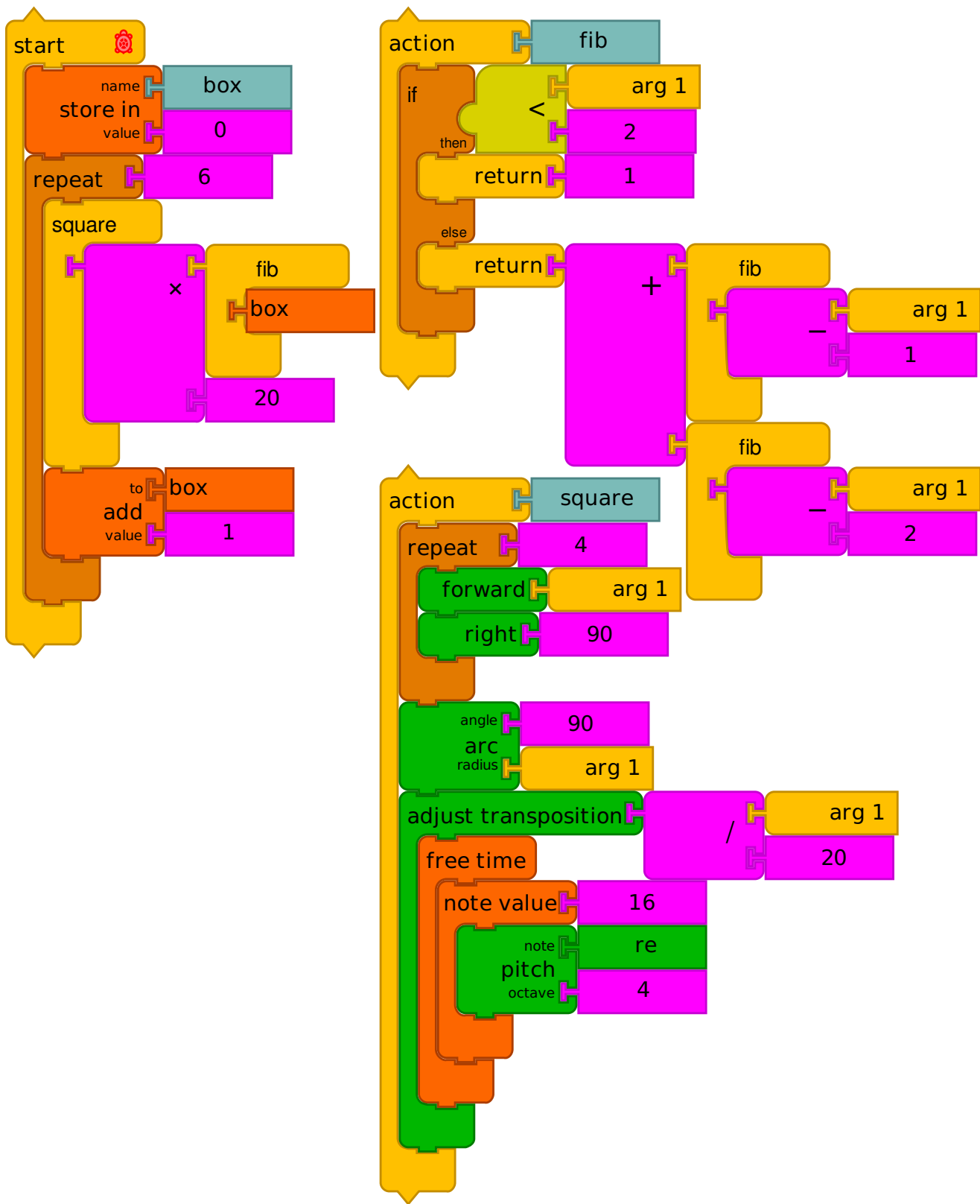


Turtle graphics can be combined with the music blocks. By placing graphics blocks, e.g., *Forward* and *Right*, inside of *Note value* blocks, the graphics stay in sync with the music. In this example, the turtle moves forward each time a quarter note is played. It turns right during the eighth note. The pitch is raised by one half step, the pen size decreases, and the pen color increases at each step in the inner repeat loop.

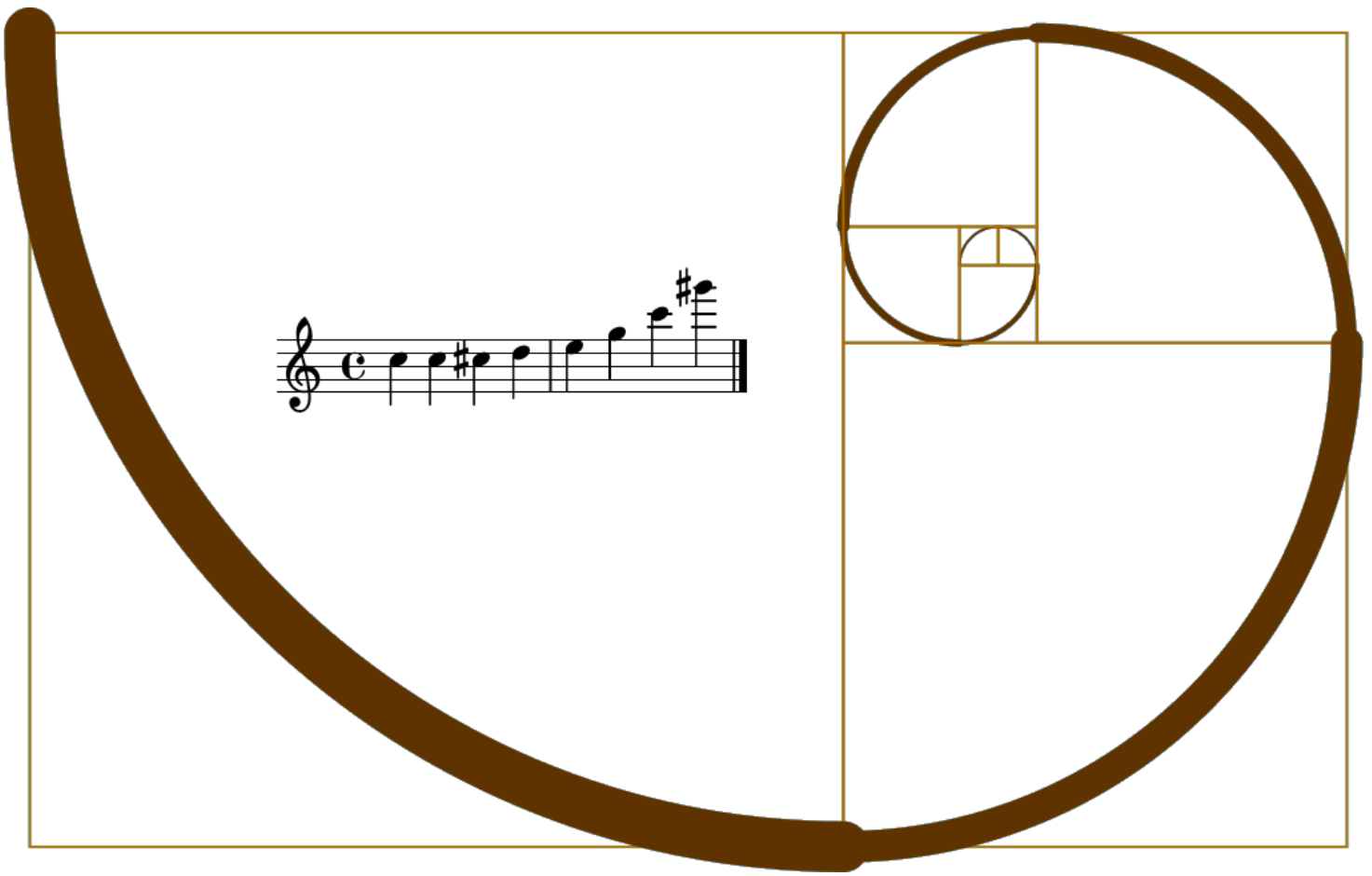


In this example, the graphics are synchronized to the music by placing the graphics commands inside of *Note value* blocks.

[RUN LIVE](#)

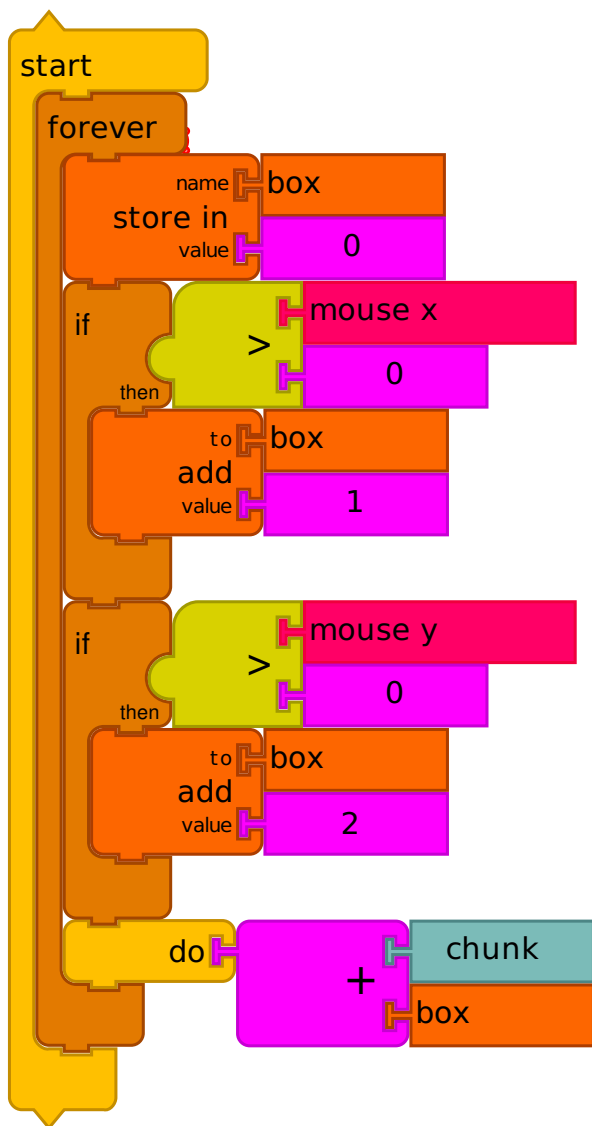


In this example, because the computation and graphics are more complex, a *Free-time* block is used to decouple the graphics from the master clock.



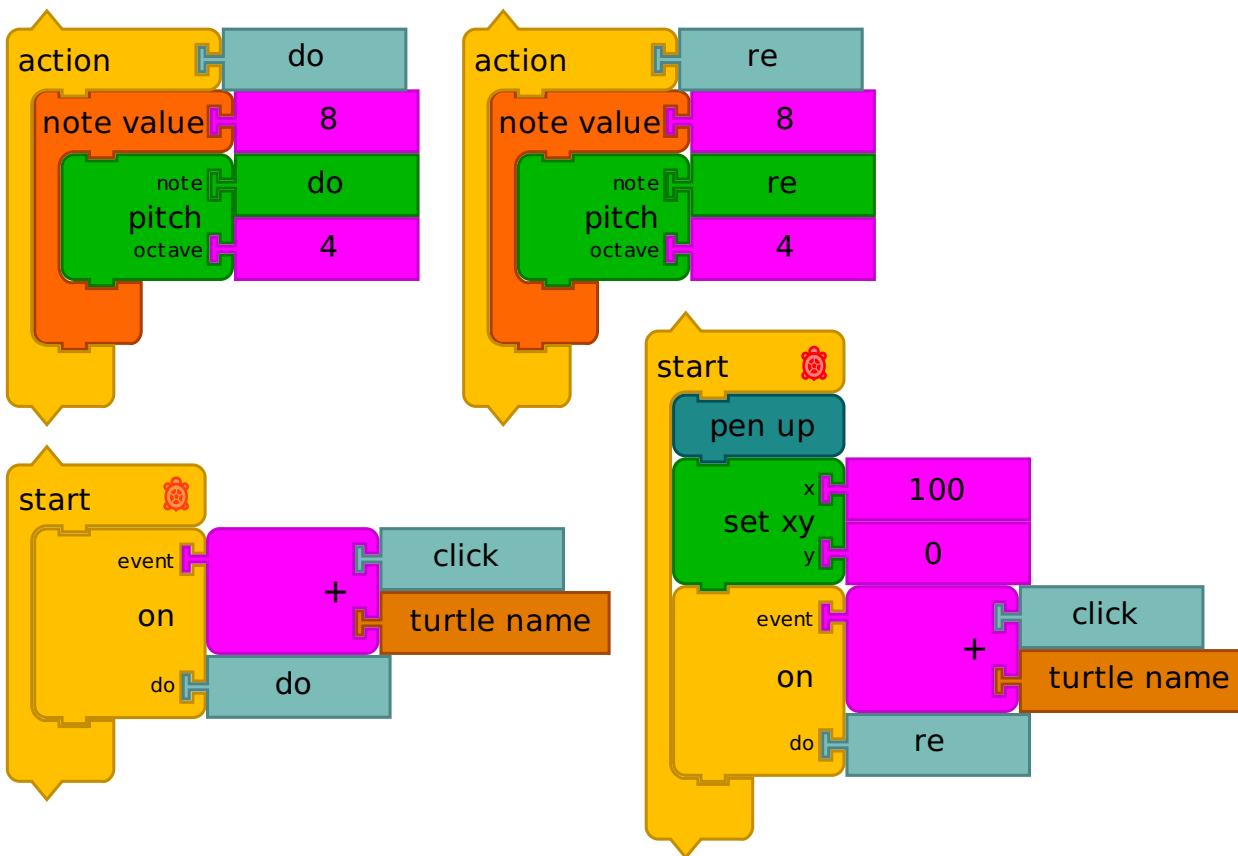
5. Interaction

There are many ways to interactive with Music Blocks, including tracking the mouse position to impact some aspect of the music.



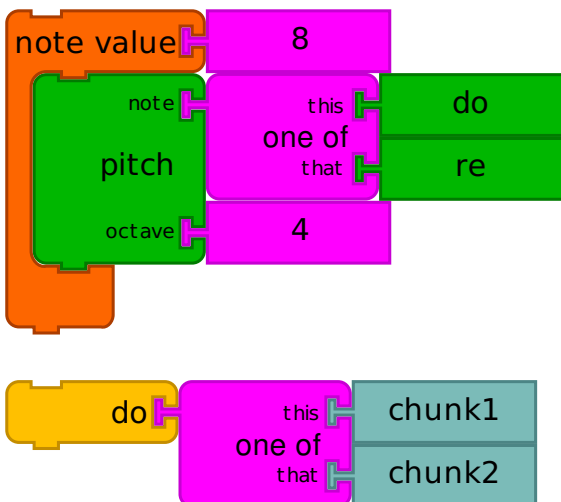
For example, we can launch the phrases (chunks) interactively. When the mouse is in the lower-left quadrant, `chunk` is played; lower-right quadrant, `chunk1`; upper-left quadrant, `chunk2`; and upper-right quadrant, `chunk3`.

[RUN LIVE](#)



In the example above, a simple two-key piano is created by associating *click* events on two different turtles with individual notes. Can you make an 8-key piano?

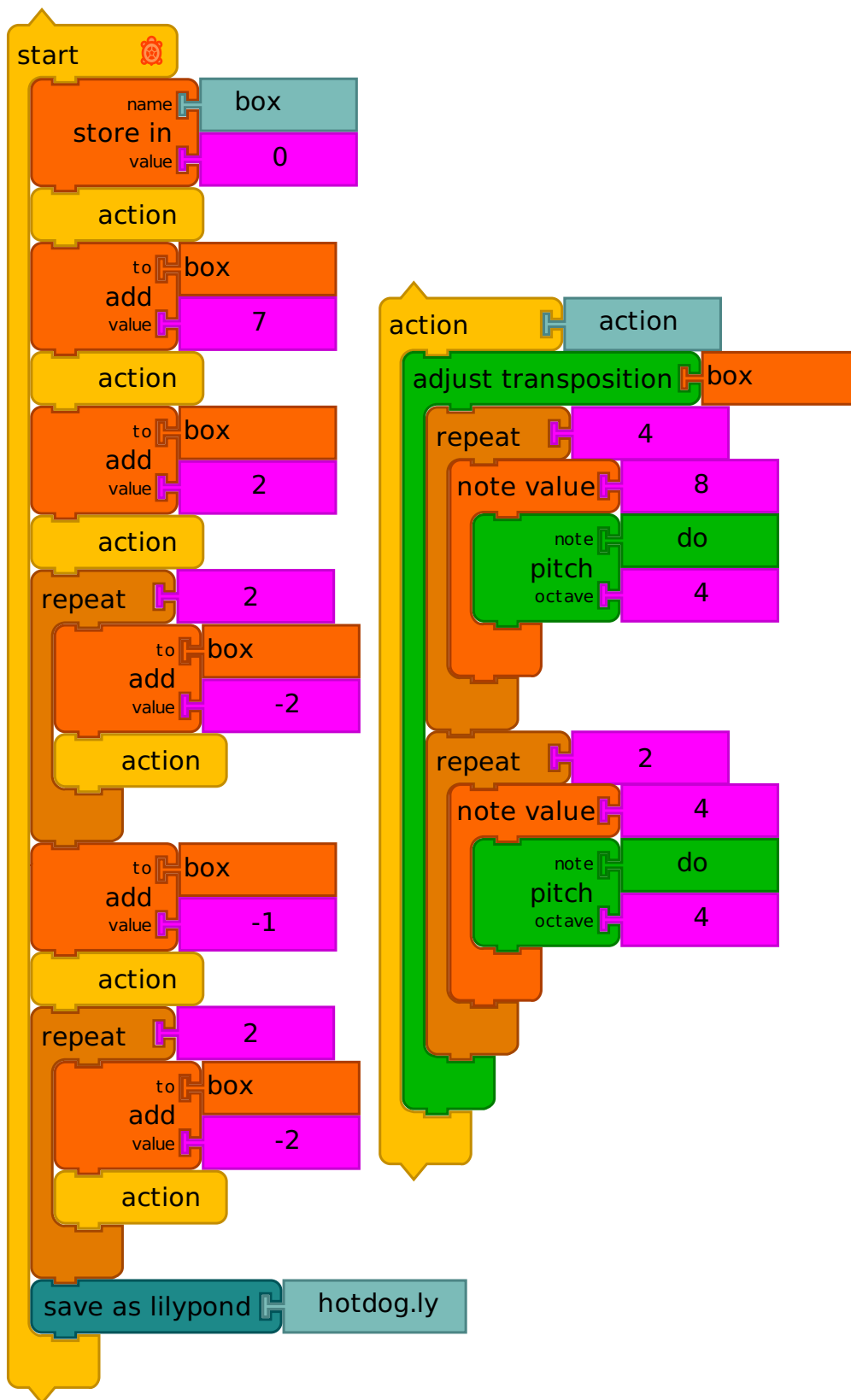
[RUN LIVE](#)



You can also add a bit of randomness to your music. In the top example above, the *One-of* block is used to randomly assign either *Do* or *Re* each time the *Note value* block is played. In the bottom example above, the *One-of* block is used to randomly select between *chunk1* and *chunk2*.

6. Beyond Music Blocks

Music Blocks is a waypoint, not a destination. One of the goals is to point the learner towards other powerful tools. One such tool is [Lilypond](#), a music engraving program.



The *Save as Lilypond* block will transcribe your composition. The output of the program above is saved to <Downloads/hotdog.ly> . There is also a *Save as Lilypond* button on the secondary toolbar.



\version "2.18.2"

```
mouse = {  
c'8 c'8 c'8 c'4 c'4 g'8 g'8 g'8 g'4 g'4 a'8 a'8 a'8 a'4  
a'4 g'8 g'8 g'8 g'4 g'4 f'8 f'8 f'8 f'4 e'8 e'8 e'8  
e'4 e'4 d'8 d'8 d'8 d'4 d'4 c'8 c'8 c'8 c'4 c'4  
}
```

```
\score {  
<<  
\new Staff = "treble" {  
\clef "treble"  
\set Staff.instrumentName = #"mouse" \mouse  
}  
>>  
\layout {}  
}
```



[RUN LIVE](#)