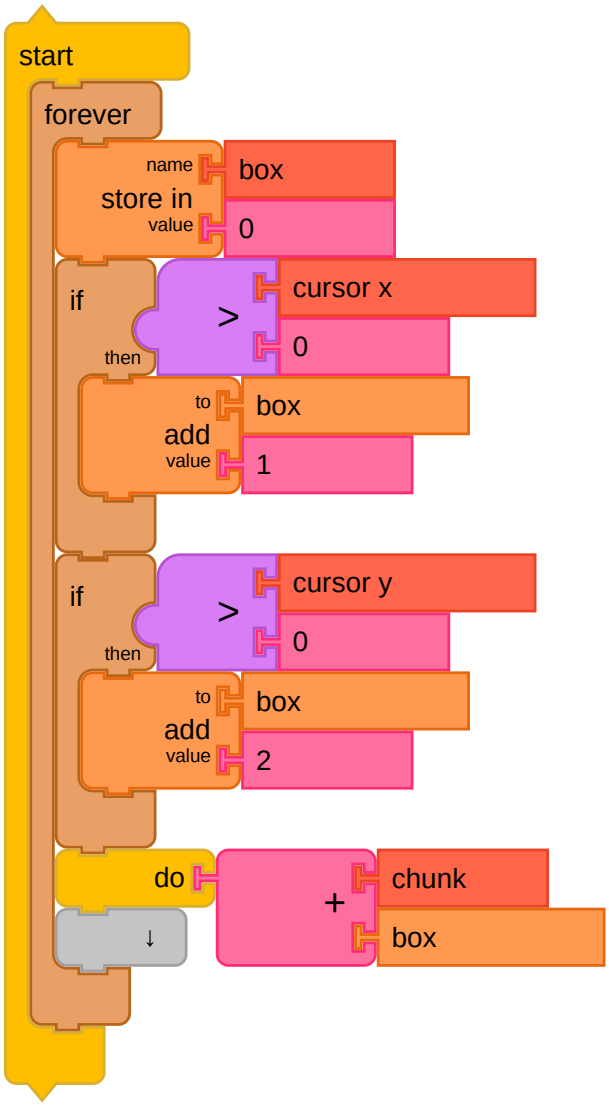
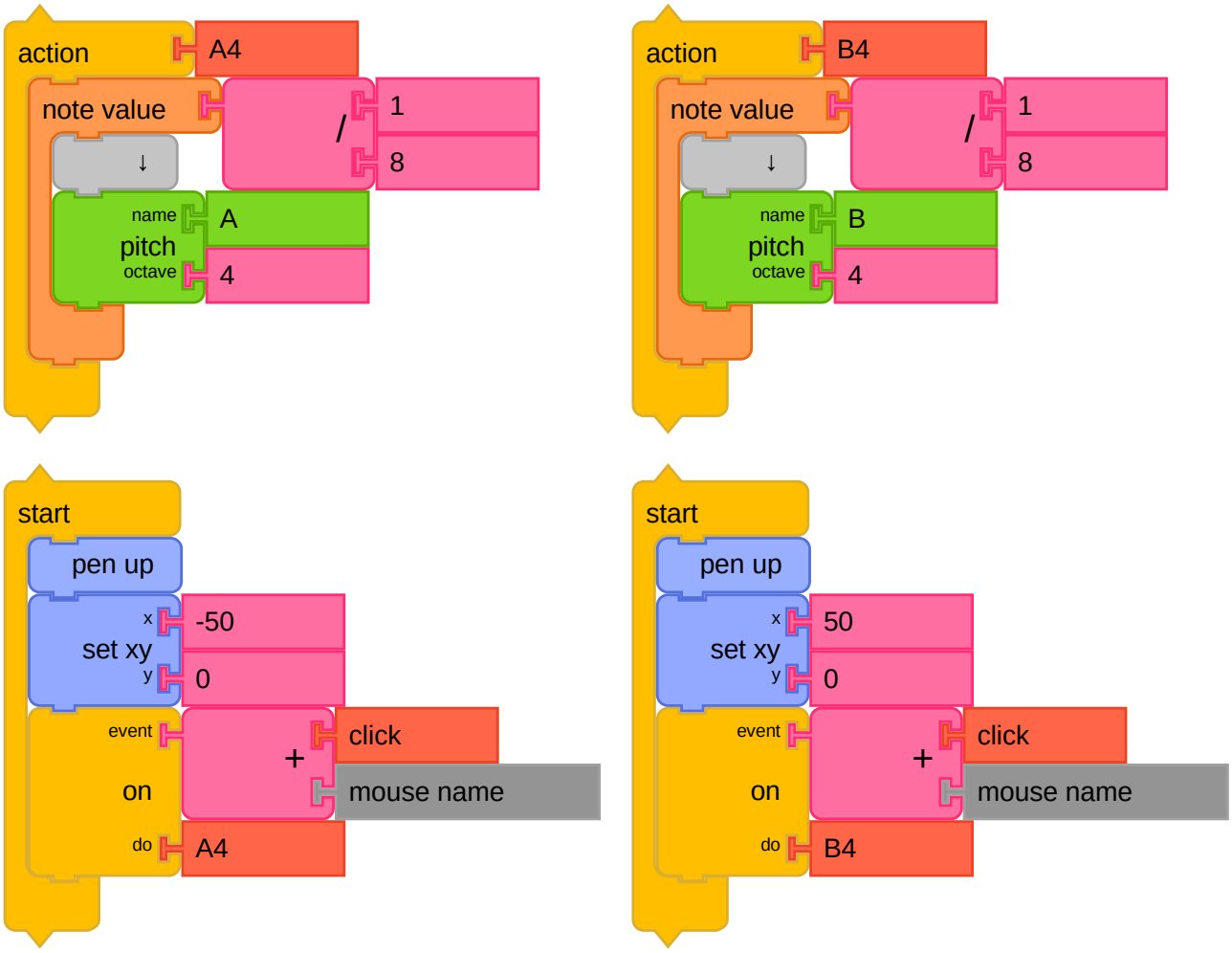
3.7 Interactions

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



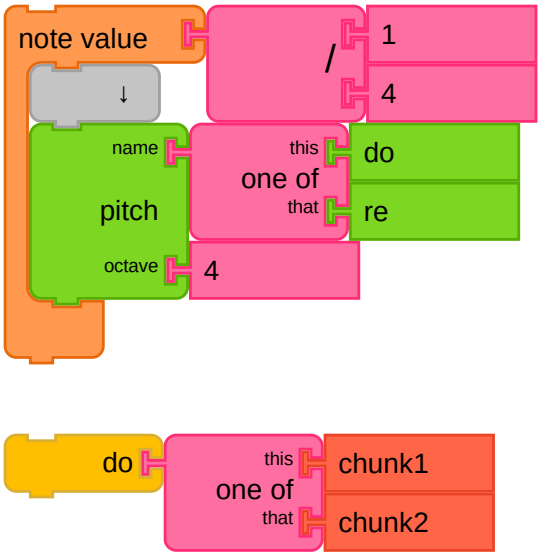
For example, we can launch the phrases (chunks) interactively. We use the mouse position to generate a suffix: `0`, `1`, `2`, or `3`, depending on the quadrant. When the mouse is in the lower-left quadrant, `chunk0` is played; lower-right quadrant, `chunk1`; upper-left quadrant, `chunk2`; and upper-right quadrant, `chunk3`.

RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1523028011868930&run=True)



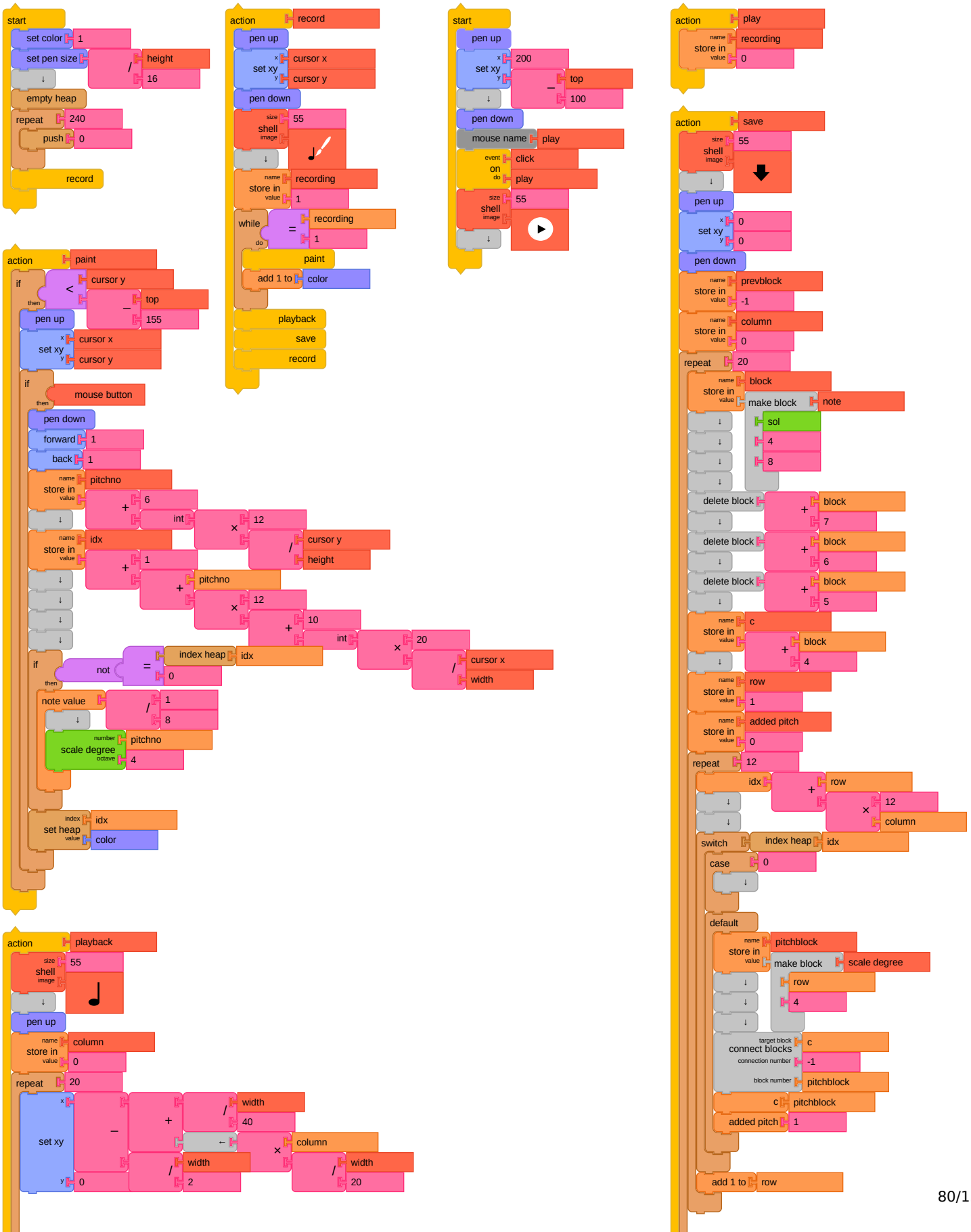
In the example above, a simple two-key piano is created by associating *click* events on two different turtles with individual notes. Can you make an 8-key piano?

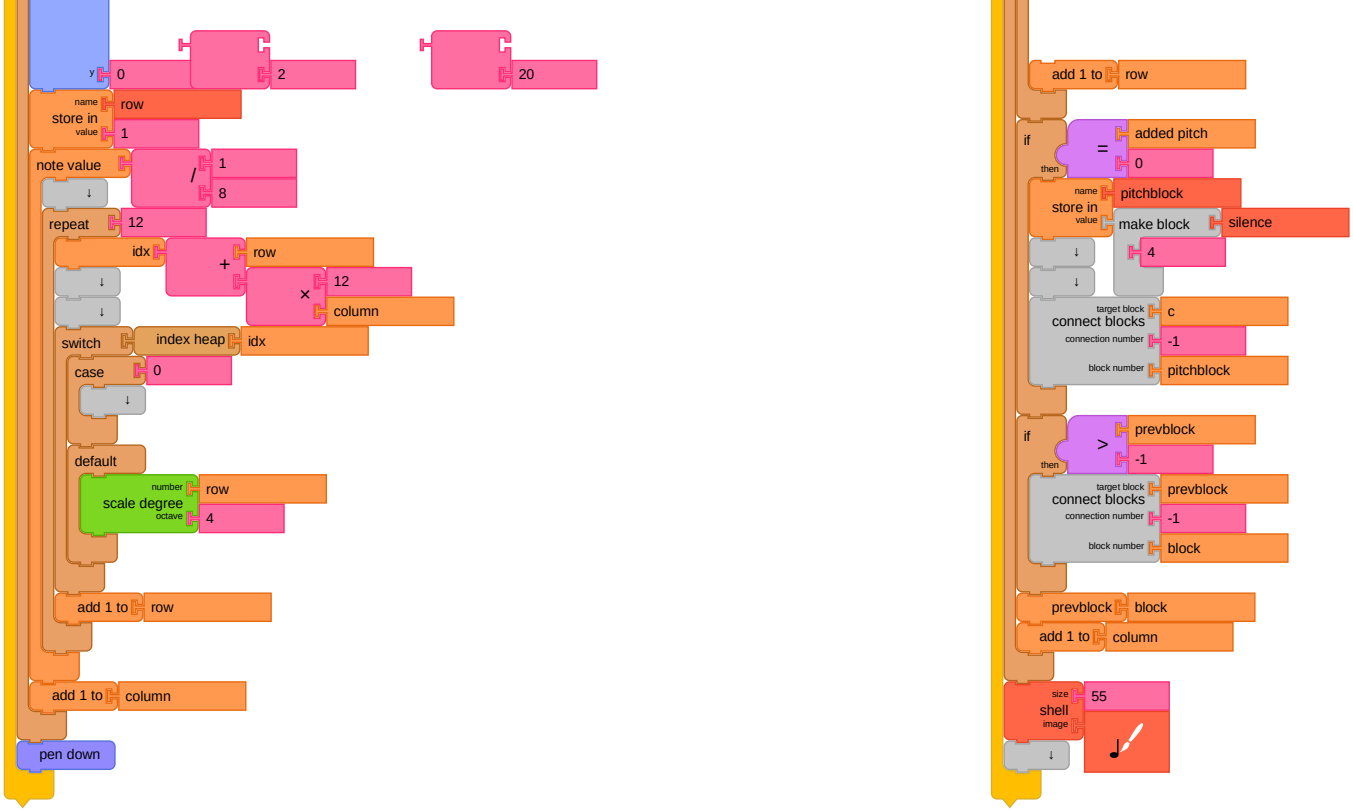
RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1523107390715125&run=True)



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`.

Musical Paint has been a popular activity dating back to programs such as Dan Franzblau's *Vidsizer* (1979) or Morwaread Farbood's *Hyperscore* (2002). Music Blocks can be used to create musical paint as well. In the somewhat ambitious example below, we go a step further than the typical paint program in that you can not only paint music (a la Vidsizer) and playback your painting as a composition (a la Hyperscore), but also generate *Note* blocks from your composition.





The program works by first creating an array from the heap that corresponds to a 20x12 grid of notes on the screen: 20 columns, representing time from left to right; and 12 rows, corresponding to scalar pitch values, which increase in value from the bottom to the top.

The *record* action repeatedly calls the *paint* action until the *playback* button is clicked.

The *paint* action tracks the mouse (*Set XY* to *cursor x* and *cursor y*) and, if the mouse button is pressed, marks an entry in the array corresponding to that note, plays the note, and leaves behind a "drop of paint".

The *playback* action is invoked by clicking on the *play* mouse, which sets *recording* to `0`, thus breaking out of the paint "while loop". Playback scans each column in the array from left to right for pitches to play and generates a chord of pitches for each column.

Once the *playback* action is complete, the *save* action is invoked. Again each column in the array is scanned, but this time, instead of playing notes, the *Make Block* block is called in order to generate a stack of notes that correspond to the composition. This stack can be copied and pasted into another composition.

While a bit fanciful, this example, which can be run by clicking on the link below, takes musical paint in a novel direction.

RUN LIVE (https://walterbender.github.io/musicblocks/index.html?id=1523896294964170&run=True&run=True)

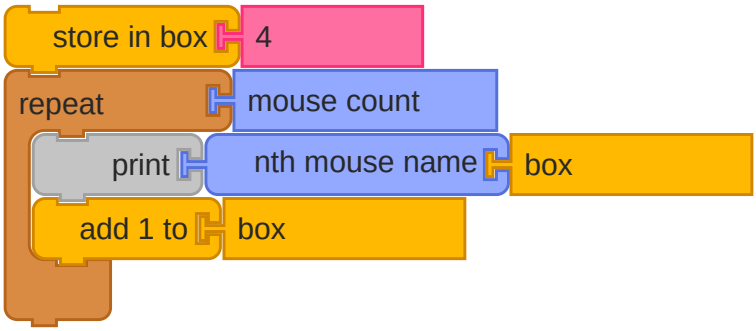
3.8 Ensemble

Much of music involves multiple instruments (voices or "mice" in Music Blocks) playing together. There are a number of special blocks that can be used to coordinate the actions of an ensemble of mice.

This section will guide about different ensemble blocks, which communicate the status of mice by name, including notes played, current pen color, pitch number, etc.

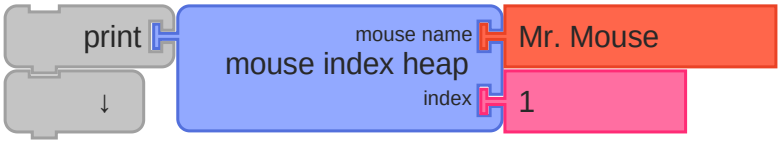
To use the ensemble blocks, you must assign a name to each mouse, as we will reference each mouse by its name.

Use the *Mouse count* block in combination with the *Nth mouse name* block to iterate through all of the mice.

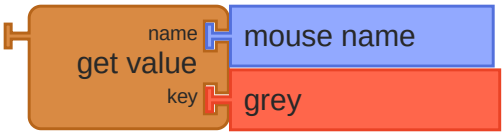


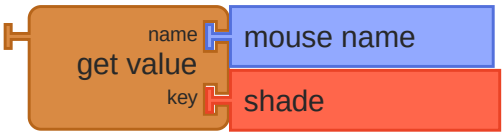
The *Mouse sync* block aligns the beat count between mice.

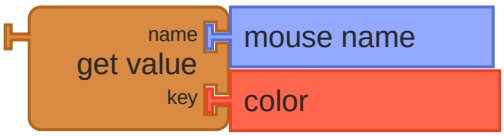
The *Mouse index heap* block returns a value in the heap at a specified location for a specified mouse.



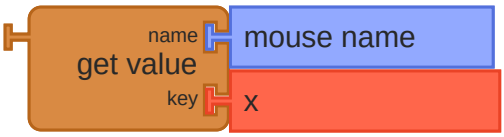
You can use the dictionary entries to data between mice. The *Get value* block lets you specify a mouse name and the value you want to access. For example, you can access a mouse's pen attributes, such as color, shade, and grey values.

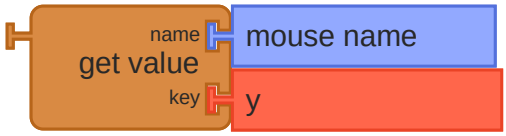


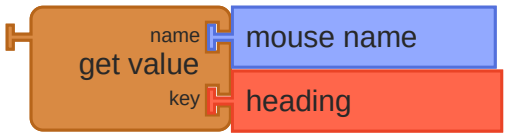


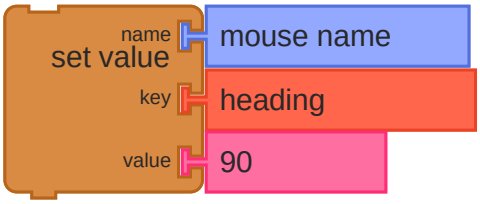


You can also access the mouse's graphics attributes, such as x, y, and heading. You can also set attributes of a mouse using the *Set value* block. In the example, a mouse's heading is set to 90.

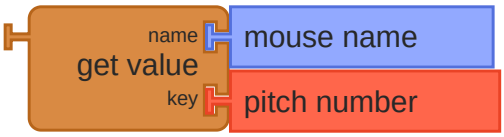


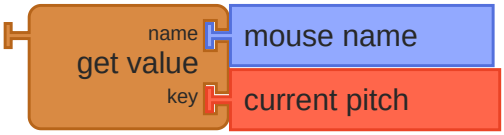


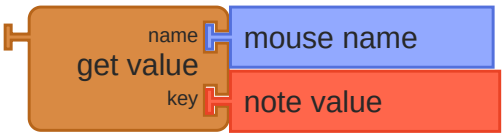


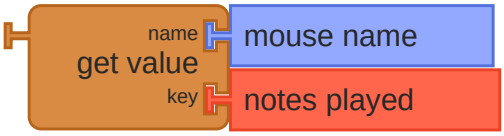


Some music status is also available through the dictionary. You can access a mouse's "current pitch", "pitch number", "note value", the number of "notes played".

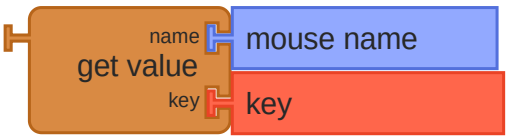








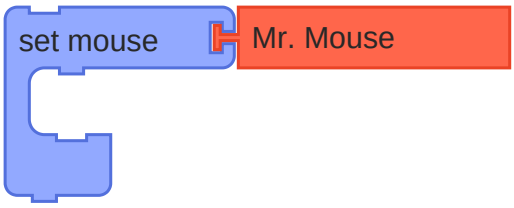
The dictionary can be used to share other things too. Just set a *key/value* pair with one mouse and access it from another.



Other Ensemble blocks include:

The *Found mouse* block will return true if the specified mouse can be found.

The *Set mouse* block sends a stack of blocks to be run by the specified mouse.


set mouse
Mr. Mouse