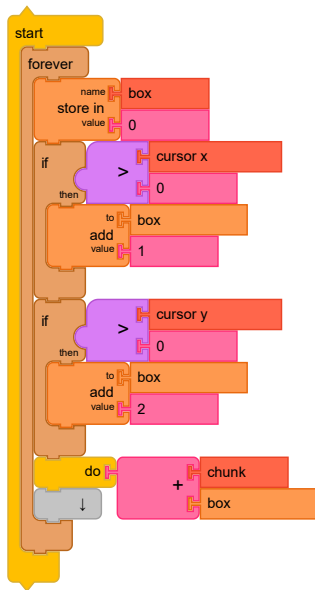## 3.7 Interactions

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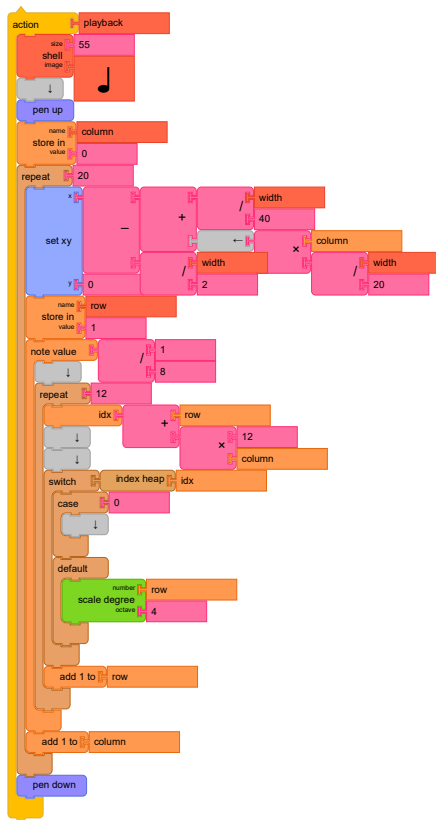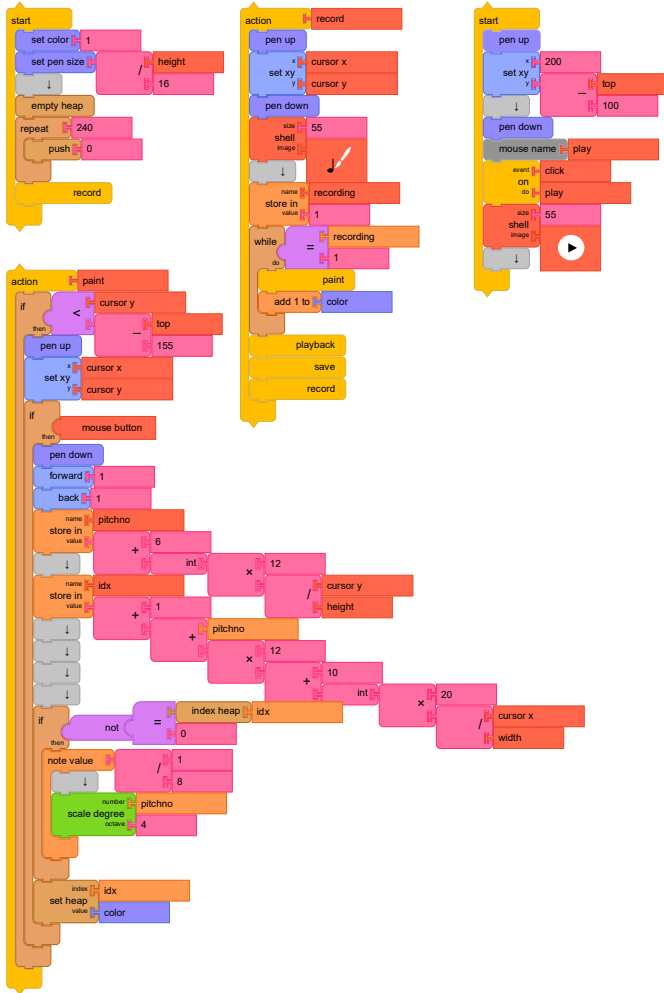
RUN LIVE
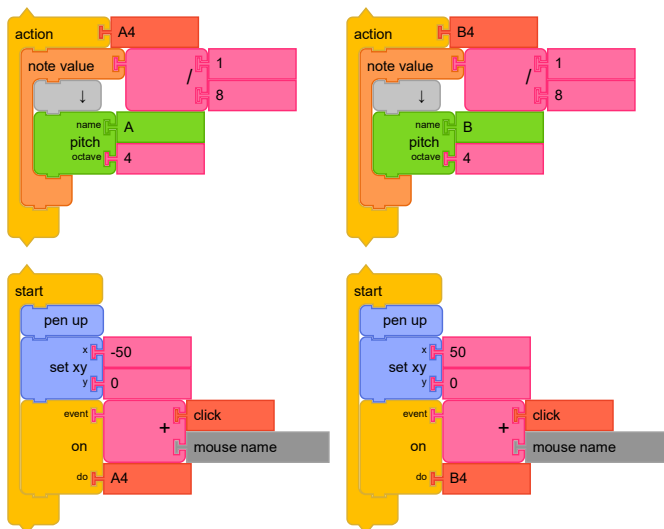
In the example above, a simple two-key piano is created by associating *click* events on two different turtles with individual notes. Can you make an 8-key piano?

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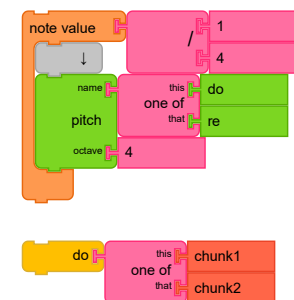
You can also add a bit of randomness to your music. In the top example above, the *One-of* block is used to randomly assign either `Do` or `Re` each time the *Note value* block is played. In the bottom example above, the *One-of* block is used to randomly select between `chunk1` and `chunk2`.

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.

start
- set color 1
- set pen size [height / 16]
- ↓
- empty heap
- repeat 240
  - push 0
- record

action record
- pen up
- set xy x: cursor x y: cursor y
- pen down
- shell size 55 image [brush]
- ↓
- store in name recording value 1
- while do [= recording 1]
  - paint
  - add 1 to color
- playback
- save
- record

start
- pen up
- set xy x: 200 y: [top − 100]
- pen down
- mouse name play
- on event click do play
- shell size 55 image [▶]
- ↓

action paint
- if then [cursor y < [top − 155]]
  - pen up
  - set xy x: cursor x y: cursor y
- if then [mouse button]
  - pen down
  - forward 1
  - back 1
  - store in name pitchno value [↓ [6 + [int [× 12 [cursor y / height]]]]]
  - store in name idx value [1 + [↓ [+ pitchno [× 12 [+ 10 [int [× 20 [cursor x / width]]]]]]]]
  - ↓
  - if then [not [= [index heap idx] 0]]
    - note value [1 / 8]
    - ↓
    - scale degree number pitchno octave 4
  - set heap index idx value color

action play
- store in name recording value 0

action playback
- shell size 55 image [♩]
- ↓
- pen up
- store in name column value 0
- repeat 20
  - set xy x: [− [+ [width / 40] [← / [× column [width / 20]]]] [width / 2]] y: 0
  - store in name row value 1
  - note value [1 / 8]
  - ↓
  - repeat 12
    - [idx + [row × 12 [column]]]
    - ↓
    - switch [index heap idx]
      - case 0
        - ↓
      - default
        - scale degree number row octave 4
    - add 1 to row
  - add 1 to column
- pen down

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