Music Blocks Lesson Plan
Mousetrap

Age:

7-12 years

Lesson duration:

90 minutes

- Introduction: The Mousetrap (15m)
- Part 1: On-event-do exercise (15m)
- Break (5m)
- Part 2: Building the Mousetrap (15m)
- Performance/Critique (10m)

Number of students:

Up to 10.

Rationale:

Students will build upon the concept of events, sensors, and broadcast. Then they'll use the on-event-do block to control a complex, multipart interactive game.

Objectives:

Students will understand what is meant by broadcast in computation and how it can be used in music. Students will be able to utilize broadcast and events in programming interactive games.

LESSON

Introduction:

Begin by asking students to sit in a circle and explain that in today's lesson they are going to build a better mousetrap.

Explain
The mousetrap game has three parts that have to work together: a mouse, a trap, and a start button. The different parts will communicate with each other by using a "broadcast". A broadcast is a message that everyone can hear. When you hear the broadcast you take some action. If

someone broadcasts “Time for dinner.”, the action we take is to come to the dinner table. We broadcast using the Broadcast Block. We listen with the On-Event-Do Block. (Both of these blocks are on the Action Palette.) We need to know what to message (or “event”) to listen for and what action to take when we hear the message.

The mouse button can also broadcast a special message called “click”. We use the Click block from the Sensor Palette to “listen” for mouse clicks.

Designing the mousetrap

In our game, (1) when the start button is clicked, the action it takes is to broadcast the message, “Drop the trap”; (2) when the mousetrap hears the message, “Drop the trap”, then the trap begins to fall; (3) the mouse escapes from the trap in response to a “click”. When you click on the mouse, it jumps to the side and shouts out “reset the trap”; and (4) when the mousetrap hears the message, “Reset”, it raises the mousetrap, so the mouse can escape.

Observe that the mousetrap has to listen for two different broadcasts and take two different actions.

Choose three students, one each to be the mouse, the mousetrap, and the start button.

The mousetrap student starts with her hands (the trap) raised above her head.

When you “click” on the start-button student, she shouts, “Drop the trap.”

When the mousetrap student hears “Drop the trap”, she slowly lowers her hands to her sides (to trap the mouse).

When you “click” on the mouse student, she steps aside and shouts “Reset.”

When the mousetrap student hears “Reset”, she raises her hands above her head again.

Have the students play their roles while you repeatedly click on either the start-button student or the mouse.

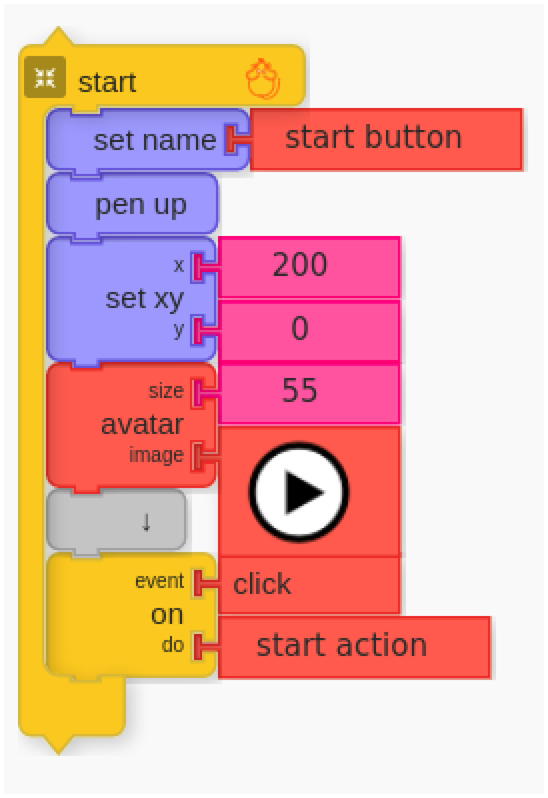
Time to build a better mousetrap.

Part 1:

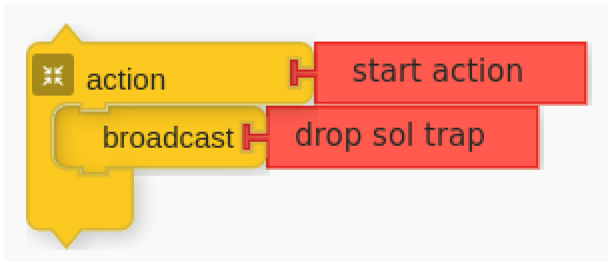
A. The Start-Button

We will start by building the Start Button. It will run in its own Start Block. It needs to listen for a mouse click and then take an action to send the broadcast.

1. Drag a Start Block from the Action Palette.
2. Use a Set Name Block from the Ensemble Palette to give your mouse a name.
3. Use a Pen Up Block and a Set XY Block to move the Start Button to one side.
4. Use an Avatar Block to make the button look like a start button.
5. Use an On-event-do Block to listen for a “click” (the Click Block is on the Sensor Palette) and do the “start action” in response to the event.

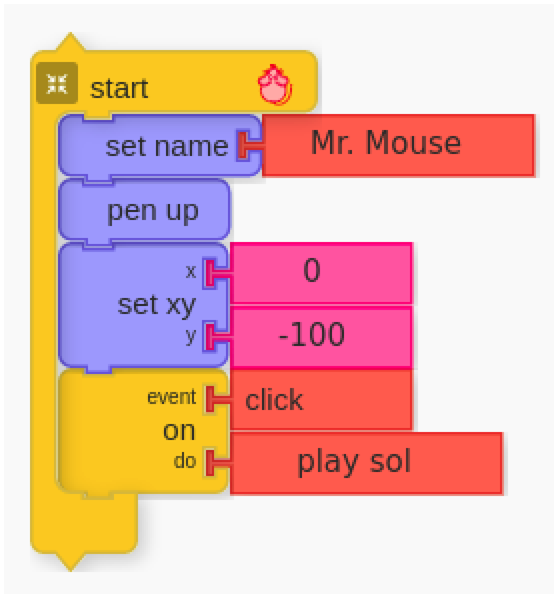


6. Make an Action Block (with the same name as the “do” action).
7. Put a Broadcast Block inside the action. We’ll broadcast the message: “drop sol trap”.



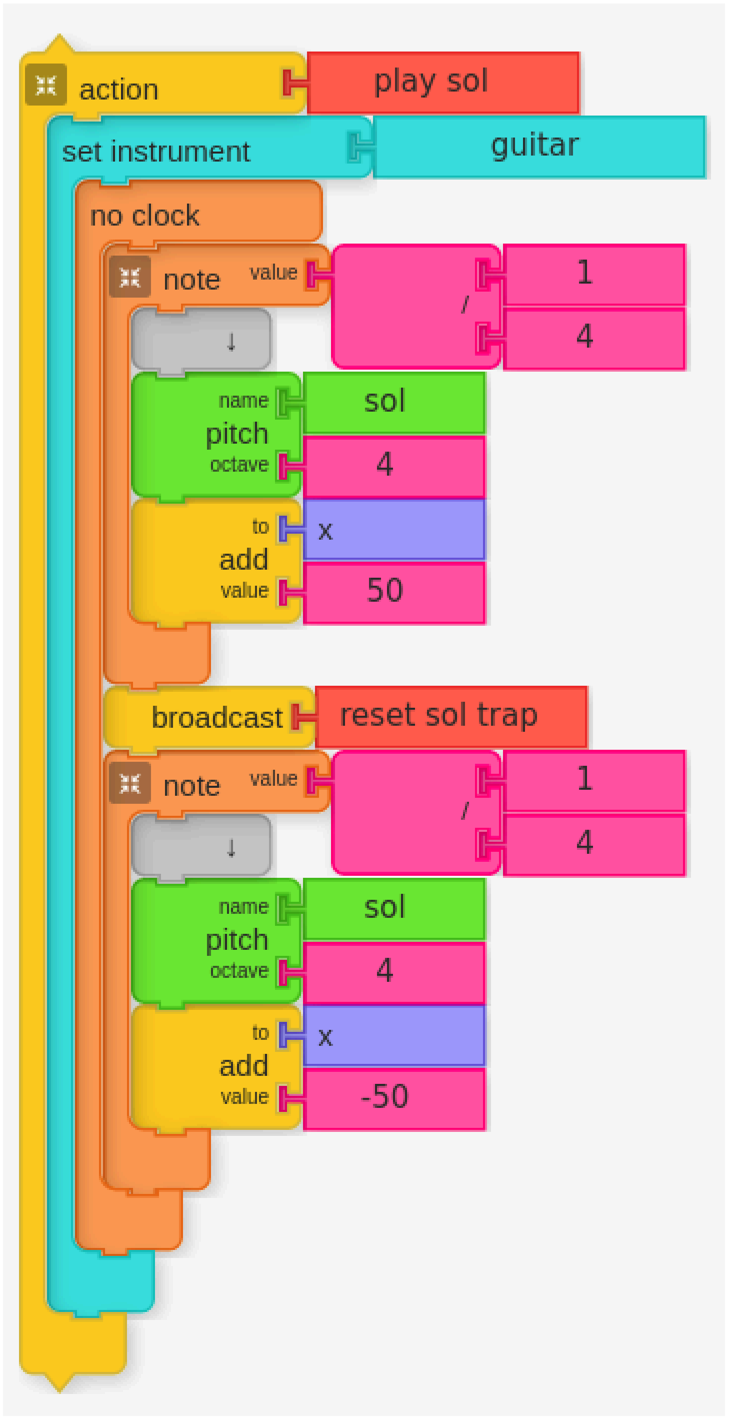
B. The Mouse

The Mouse is very similar to the Start Button. We don’t need to change its appearance, since it already looks like a mouse. It too listens for a “click” event and it does the “play sol” action when the event is received.



```
start
  set name  Mr. Mouse
  pen up
  set xy  x  0
          y  -100
  on  event  click
      do     play sol
```

The play sol action also needs to broadcast a message, “reset sol trap”. For fun, we also play a simple song (two sol notes) and do a dance by jumping to the right and then to the left (once the trap is raised.) Do you understand how the Add-to Block works?



```
action  play sol
  set instrument  guitar
    no clock
      note  value  1 / 4
        ↓
        pitch  name sol  octave 4
        add  to x  value 50
      broadcast  reset sol trap
      note  value  1 / 4
        ↓
        pitch  name sol  octave 4
        add  to x  value -50
```

,

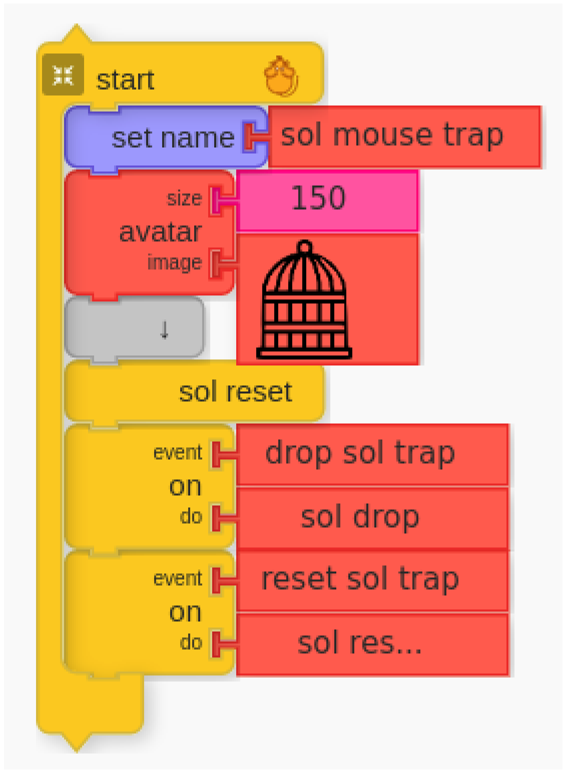
Break

Part 2

A. The Mousetrap

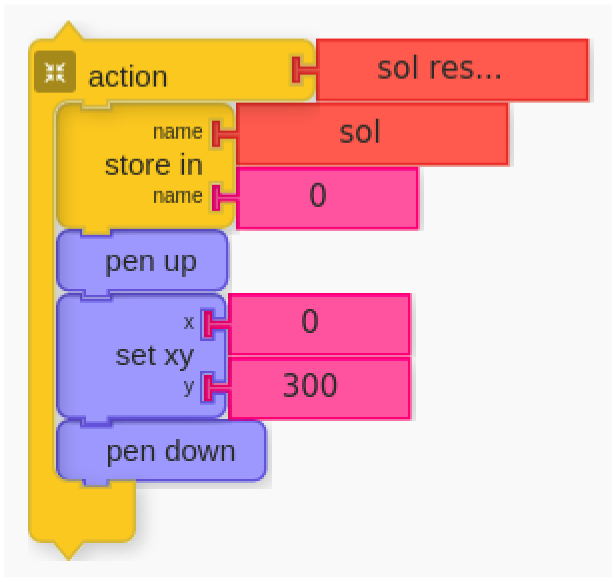
The mousetrap has to listen to two different events: a message to raise (or reset) the trap and a message to drop the trap.

1. Grab a Start Block and a Set Name block.
2. Use an Avatar Block to change the appearance to a cage to catch the mouse.
3. We need two On-event-do blocks, one for “drop sol trap” and one for “reset sol trap”.


```
start
  set name  sol mouse trap
  avatar  size 150
          image
  ↓
  sol reset
  on  event  drop sol trap
      do     sol drop
  on  event  reset sol trap
      do     sol res...
```

The “reset sol trap” event will result in the sol reset action.
The “drop sol trap” event will result in the sol drop trap action.

In the sol reset action, we reposition the trap. We also put the number 0 into a box. We can test to see if the trap is in the reset position by looking inside the box.


```
action  sol res...
  store in  name sol
            name 0
  pen up
  set xy  x 0
          y 300
  pen down
```

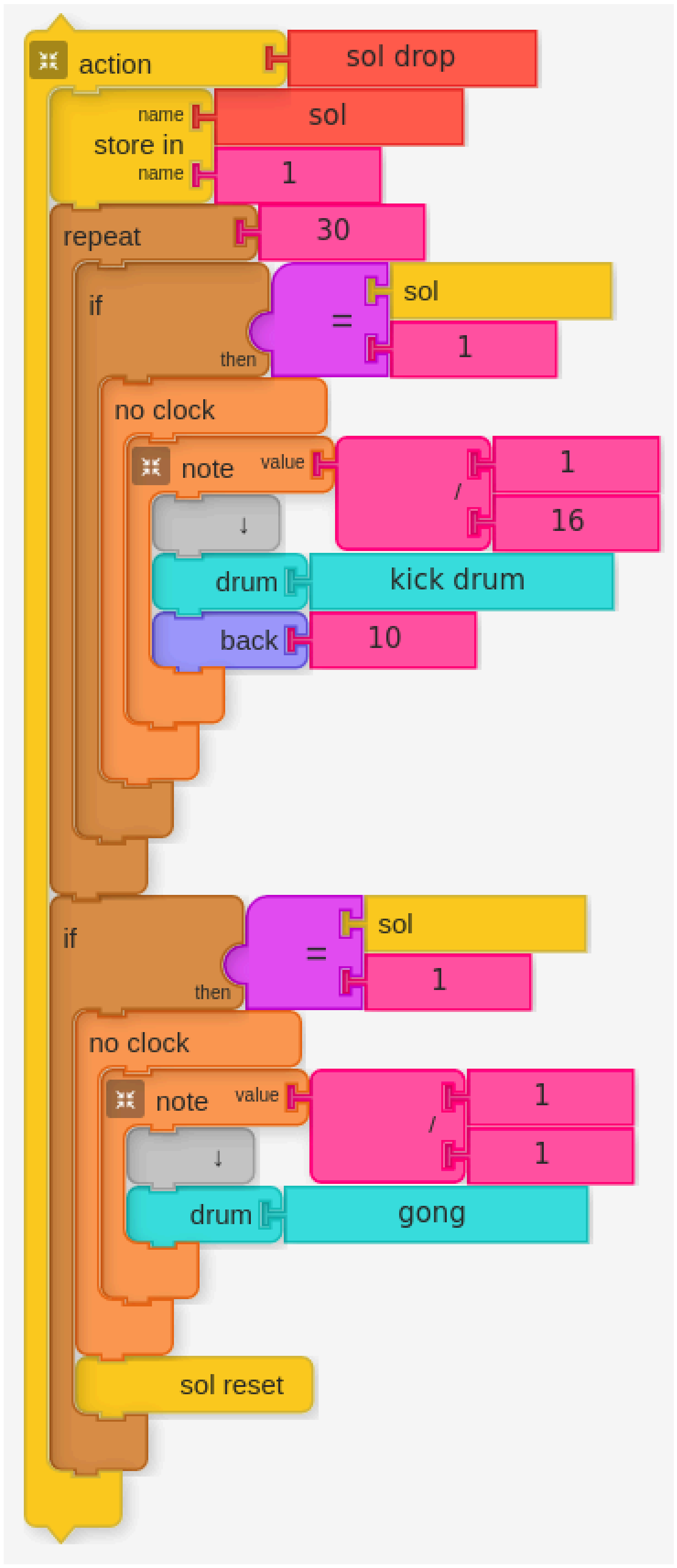
The sol drop trap action is more complex.

1. The first thing we do is to put a 1 in the box. This is so we know that the trap is falling.
2. Then we repeat an action 30 times: if the value inside the box is equal to 1, then play a drum sound and drop the cage by 10 pixels.

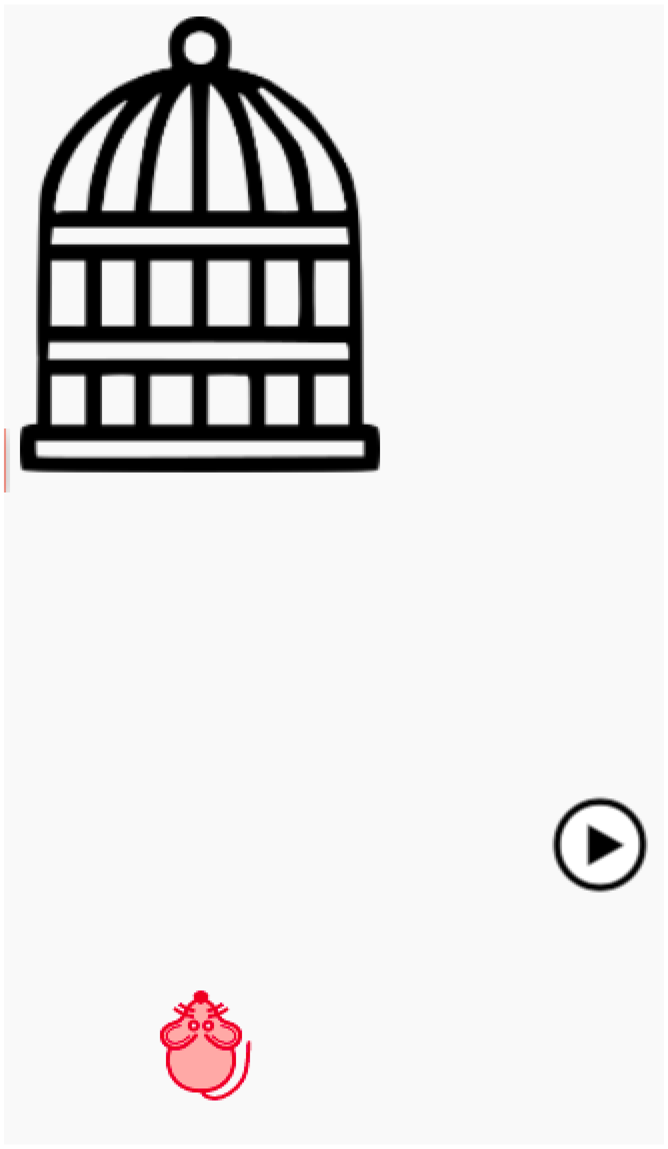
As long as the value in the box is 1, the cage will keep falling.

But if the sol reset action is taken, the value in the box will be set to 0 and the cage will stop dropping.

3. After the repeat loop completes, we check to value in the box once more. If it is still equal to 1, then we have caught the mouse and we play a gong sound.



```
action sol drop
  store in name sol
           name 1
  repeat 30
    if sol = 1 then
      no clock
        note value 1 / 16
          ↓
          drum kick drum
          back 10
  if sol = 1 then
    no clock
      note value 1 / 1
        ↓
        drum gong
    sol reset
```



B. Variations

Can you come up with dances for the mouse and the mouse trap? The start button could dance as well, each time it is clicked.

How about a victory song for the mouse if it avoids the trap?

And a victory dance for the cage, if it captures the mouse.

What could we do with some cheese?

Performance/Critique:

1. Have each student perform their variation of the mousetrap game.
2. Engage in a discussion about more things they could do with broadcast and events.

Key events:

- Introduction of key concept: broadcast
- The students create their own programs using events.

Materials:

- Music Blocks software

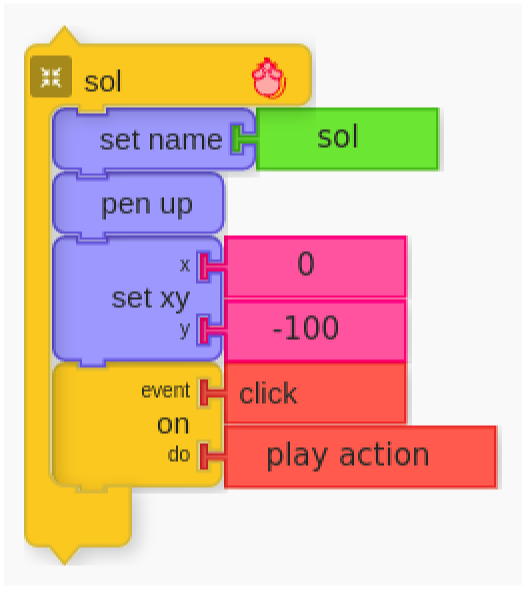
Assessment:

- Observe participation.
- Do the programs include creative use of events?

Extras: Generalizing the code.

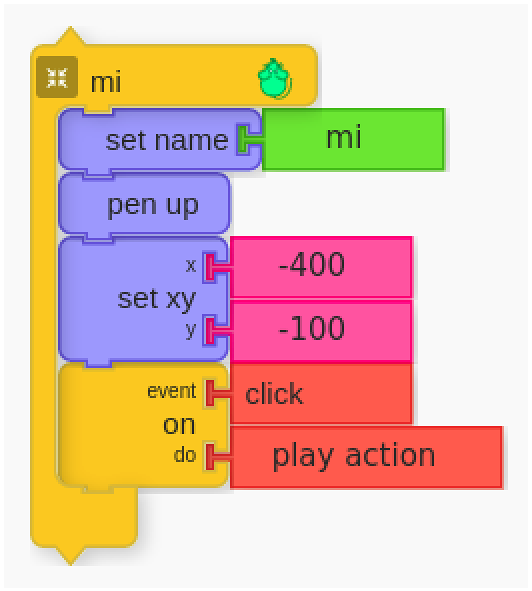
It is possible to construct the names of our actions using the + (plus) block. Doing this lets us expand to multiple mouse traps while reusing much of the code.

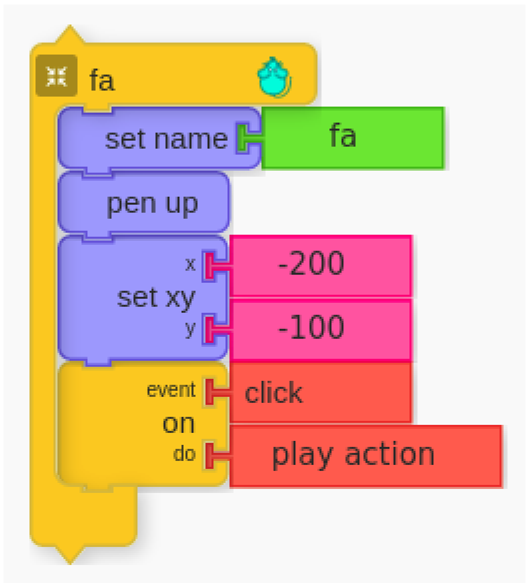
The code for the individual mice doesn’t change too much. But we construct the names of our actions and the names of the events to broadcast as per below.


```
sol
  set name  sol
  pen up
  set xy  x 0
          y -100
  on  event click
      do    play action
```

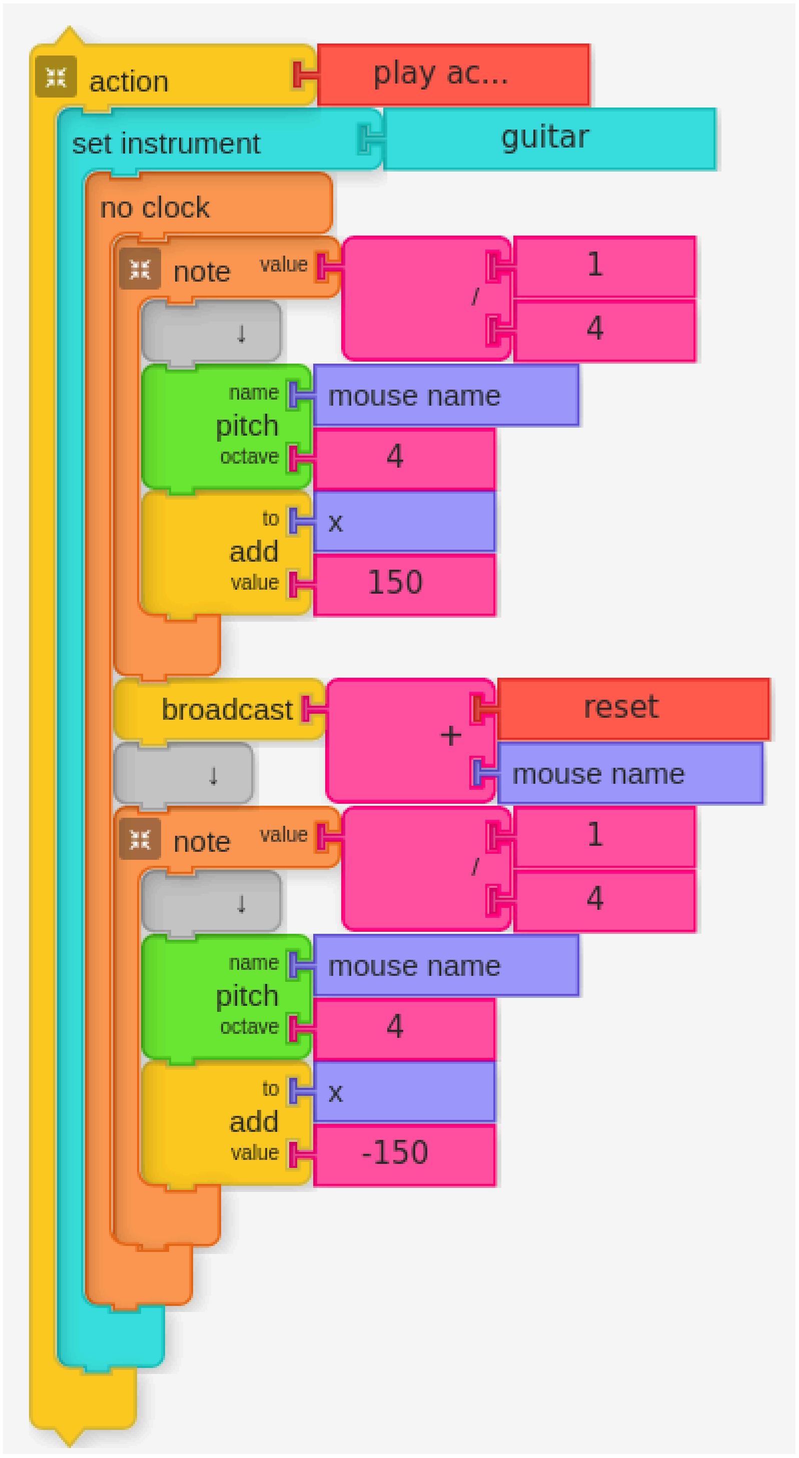
The name of the mouse is “mouse” + “sol”. The advantage of this approach is that when we want to add another mouse, all we need to do is copy this code (with the right-click menu) and then update “sol” to, say, “mi” or “fa” using the piemenu selector.

One additional change is to set the X position so that the mice aren’t on top of each other.


```
mi
  set name  mi
  pen up
  set xy  x -400
          y -100
  on  event click
      do    play action
```

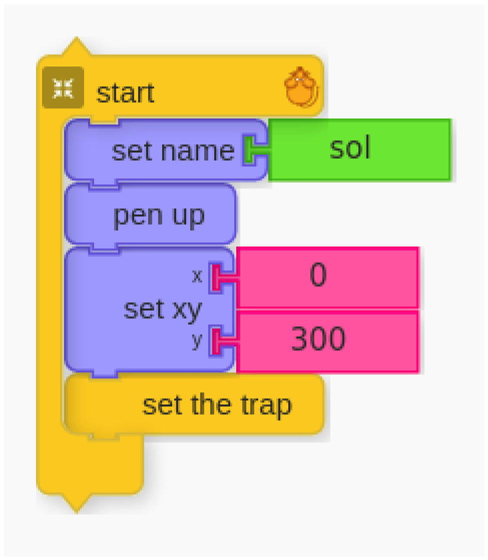

```
fa
  set name  fa
  pen up
  set xy  x -200
          y -100
  on  event click
      do    play action
```

The “play action” uses the mouse name to determine what note to play and also to construct the proper message to broadcast (so that the cage associated with the mouse that is playing will reset.)

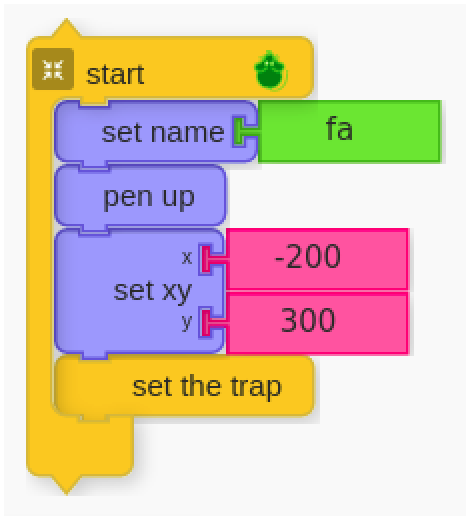


Note that the mouse name is constructed using + (plus) as is the message to broadcast.

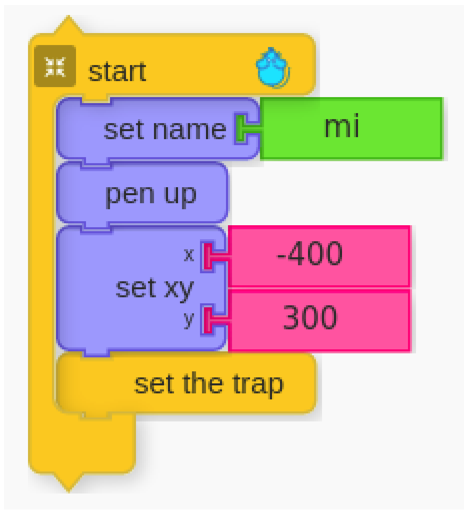
The mouse traps push this idea even further. We use the mouse name in order to share code among the individual mouse traps.



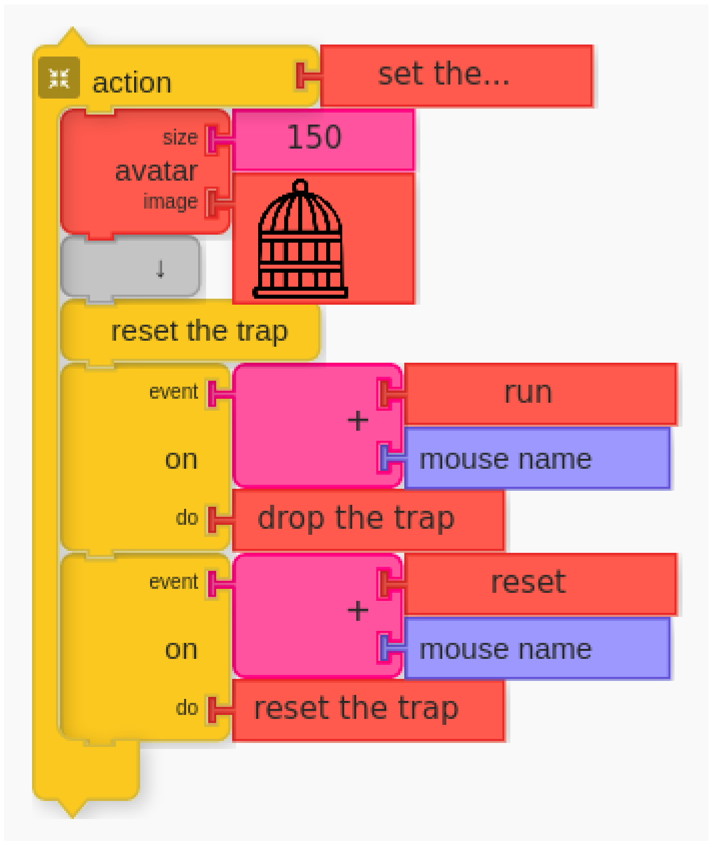
```
start
  set name sol
  pen up
  set xy x 0
         y 300
  set the trap
```



```
start
  set name fa
  pen up
  set xy x -200
         y 300
  set the trap
```

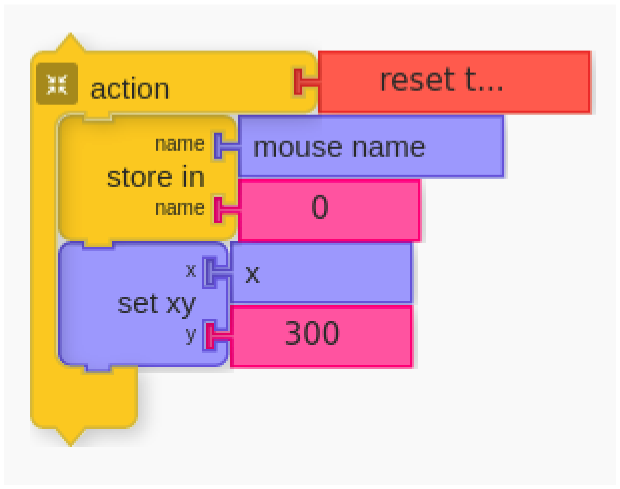


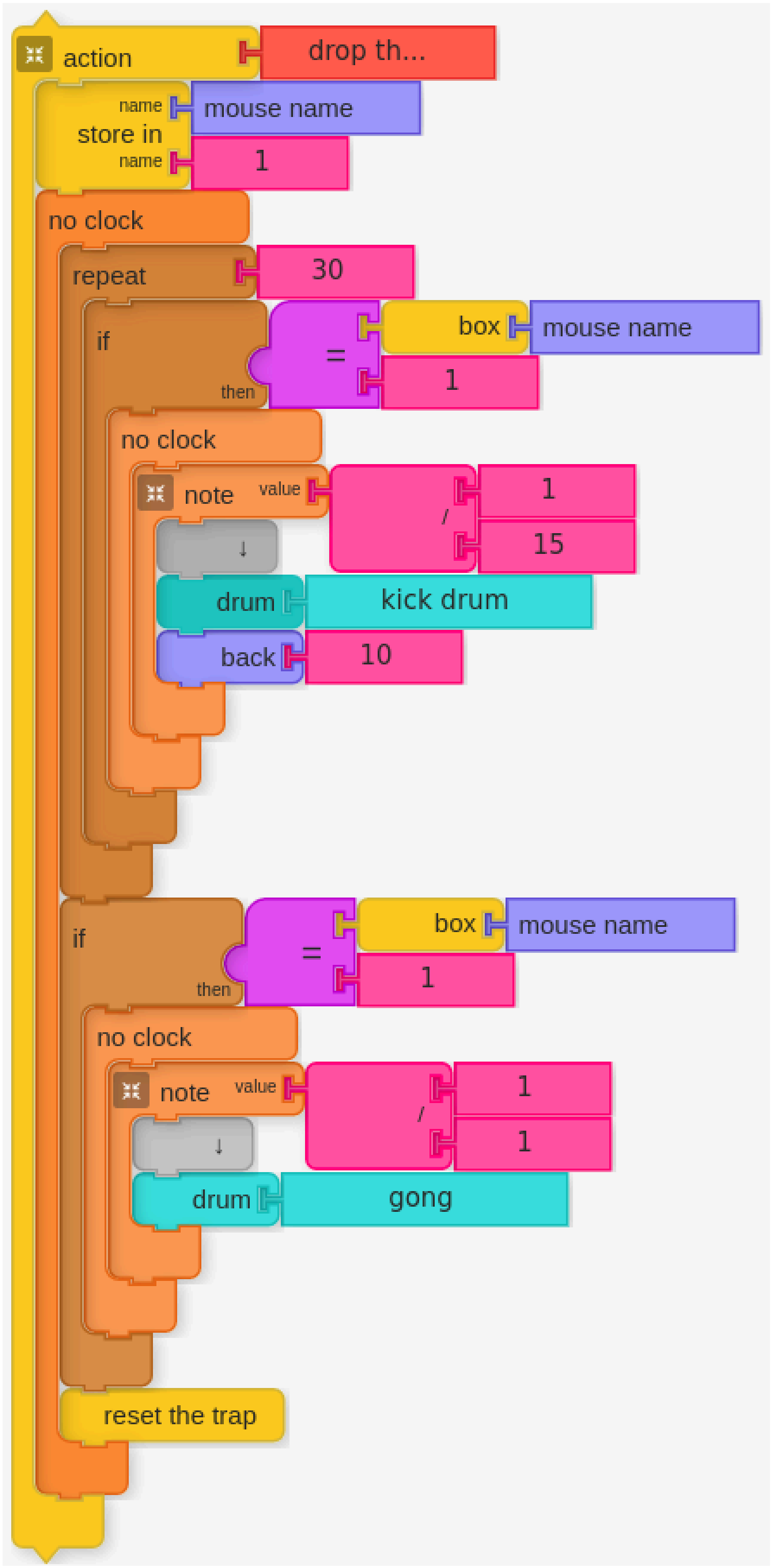
```
start
  set name mi
  pen up
  set xy x -400
         y 300
  set the trap
```


```
action  set the...
    avatar  size 150
            image
    ↓
    reset the trap
    on  event  run + mouse name
        do     drop the trap
    on  event  reset + mouse name
        do     reset the trap
```

The event names are constructed from the mouse name.

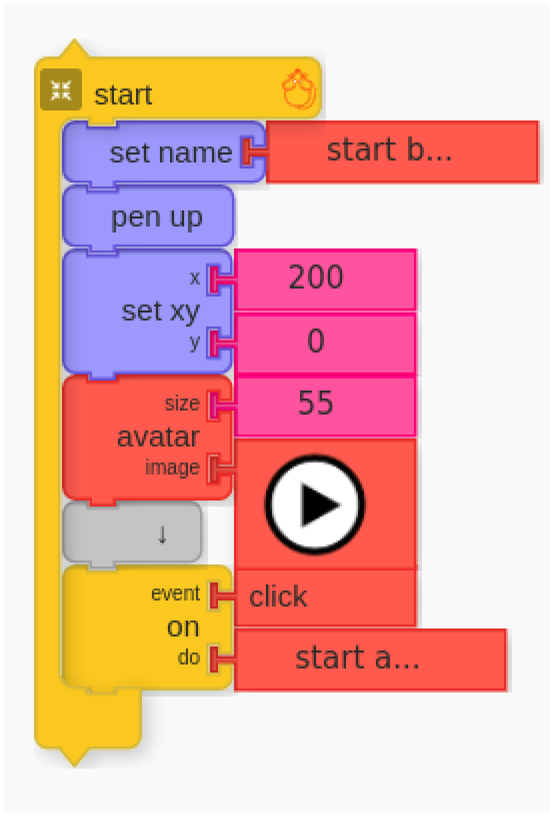
The “reset the trap” and “drop the trap” actions use the mouse name to ensure the proper cage is used.


```
action  reset t...
    store in  name mouse name
              name 0
    set xy    x x
              y 300
```

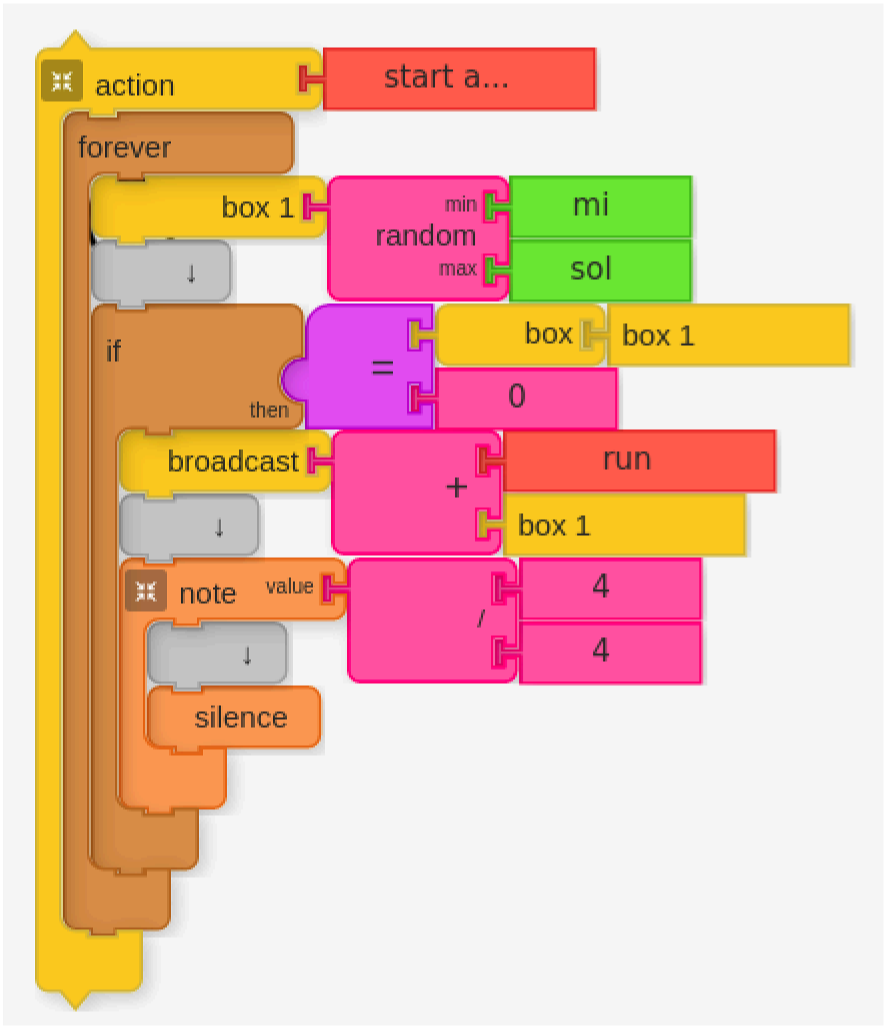


```
action  drop th...
  store in
    name  mouse name
    name  1
  no clock
    repeat  30
      if  box mouse name  =  1
      then
        no clock
          note  value  1 / 15
            ↓
            drum  kick drum
            back  10
    if  box mouse name  =  1
    then
      no clock
        note  value  1 / 1
          ↓
          drum  gong
    reset the trap
```

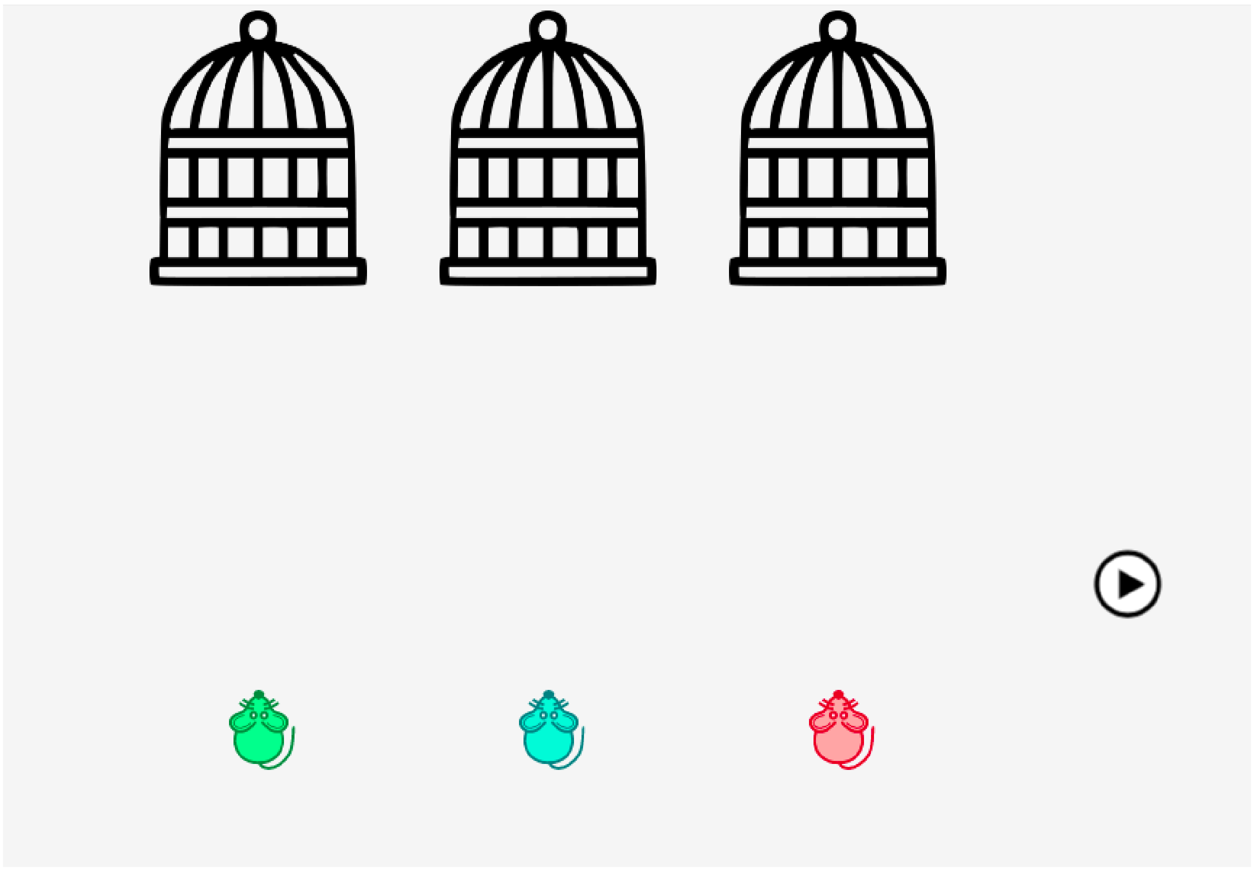
Finally we expand the range of the start button, using + (plus) to create the broadcast event. We also put in a forever loop so that the game would keep playing.



```
start
  set name  start b...
  pen up
  set xy  x 200
          y 0
  avatar  size 55
          image
  ↓
  on  event click
      do    start a...
```


```
action  start a...
  forever
    box 1  random  min mi
                   max sol
    ↓
    if  box  box 1  =  0
    then
      broadcast  run  +  box 1
      ↓
      note  value  4 / 4
        ↓
        silence
```

Note that we check to make sure that the cage we are dropping is not already falling. (If a cage is falling, its corresponding box has a 1 stored in it.)



See https://musicblocks.sugarlabs.org/index.html?id=1590616278785455