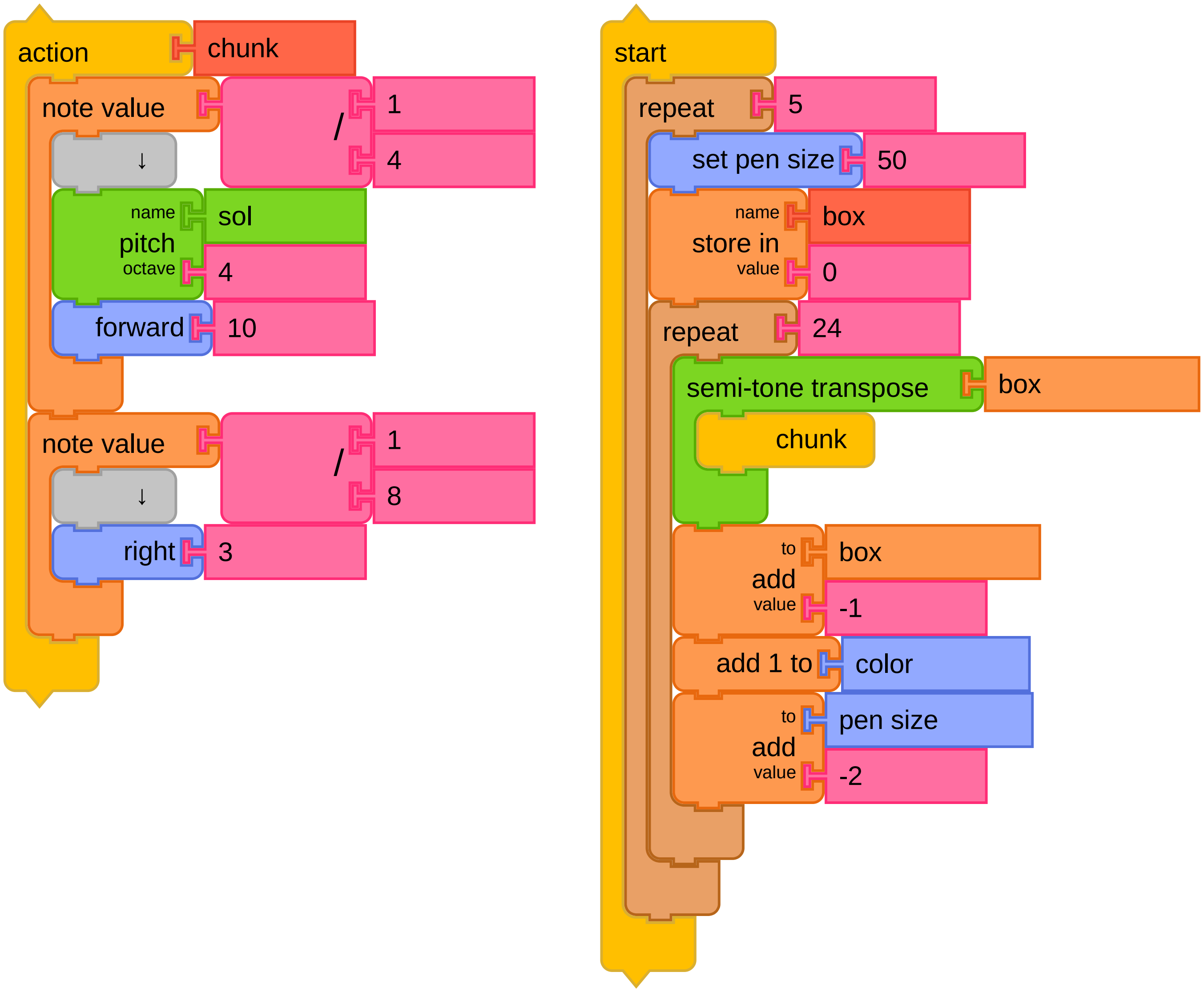
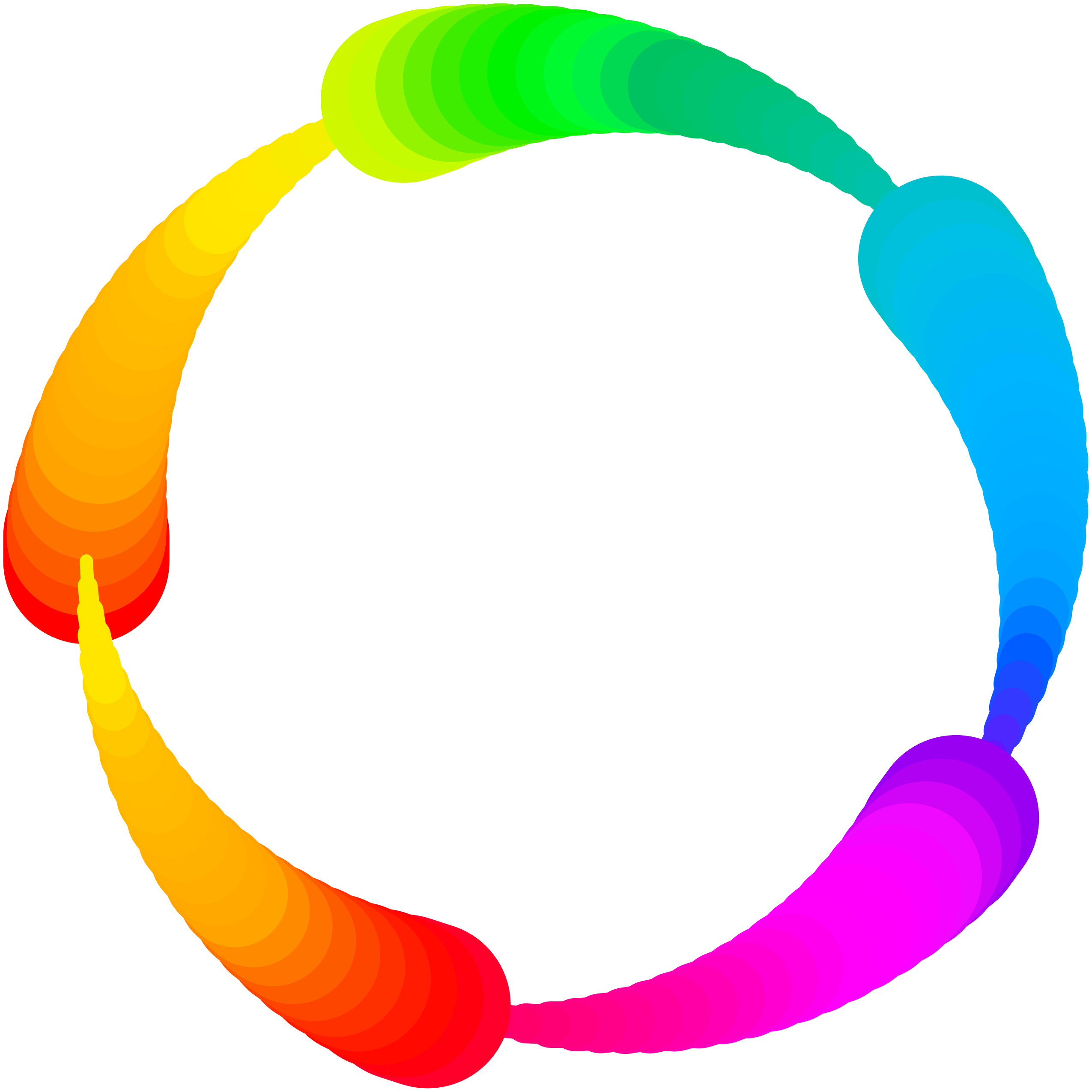
3.6 Adding graphics



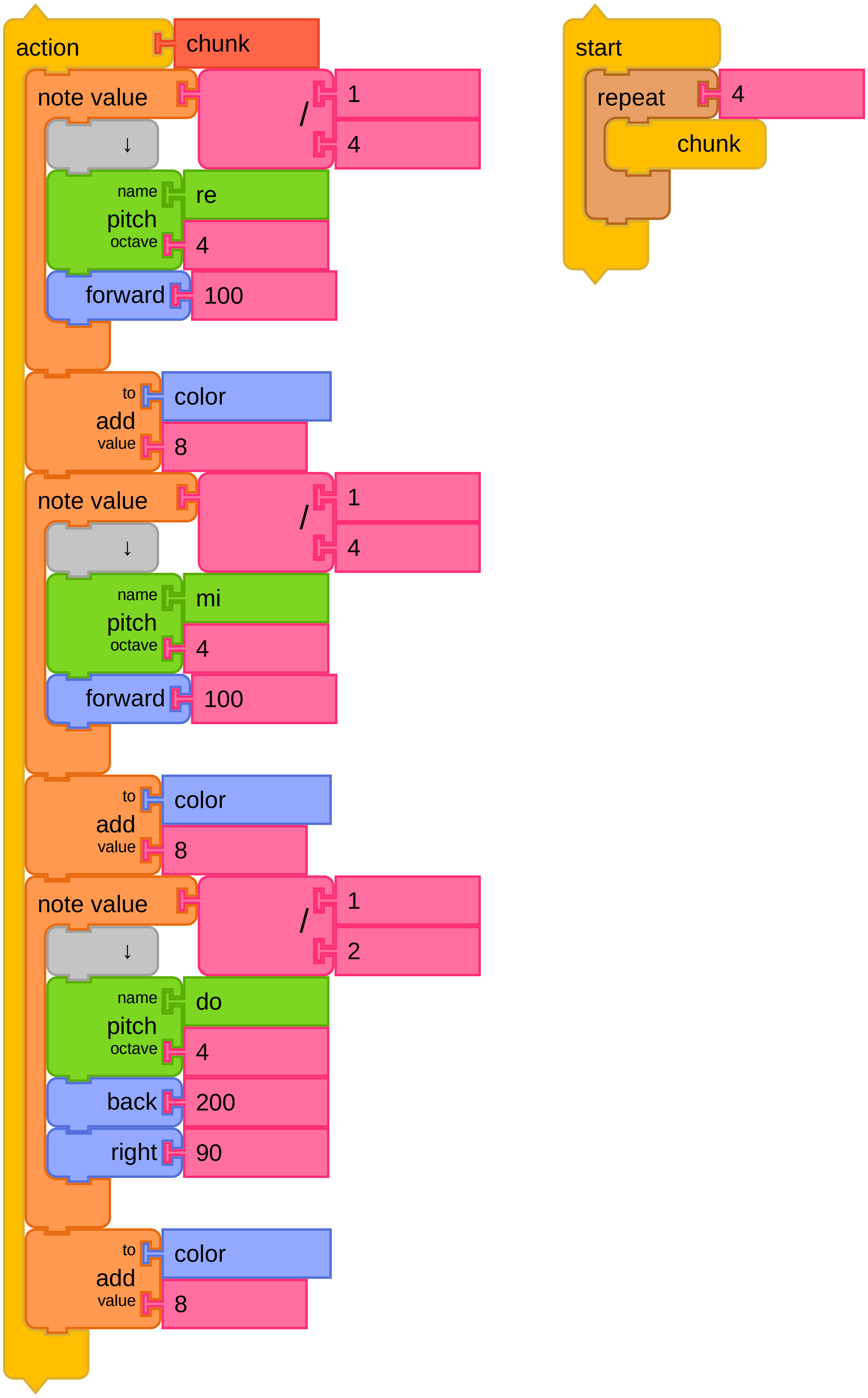
```
action chunk
    note value 1 / 4
        ↓
        pitch name sol octave 4
        forward 10
    note value 1 / 8
        ↓
        right 3
```

```
start
    repeat 5
        set pen size 50
        store in name box value 0
        repeat 24
            semi-tone transpose box
                chunk
            add to box value -1
            add 1 to color
            add to pen size value -2
```



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 decreased by one half step, the pen size decreases, and the pen color increases at each step in the inner repeat loop.

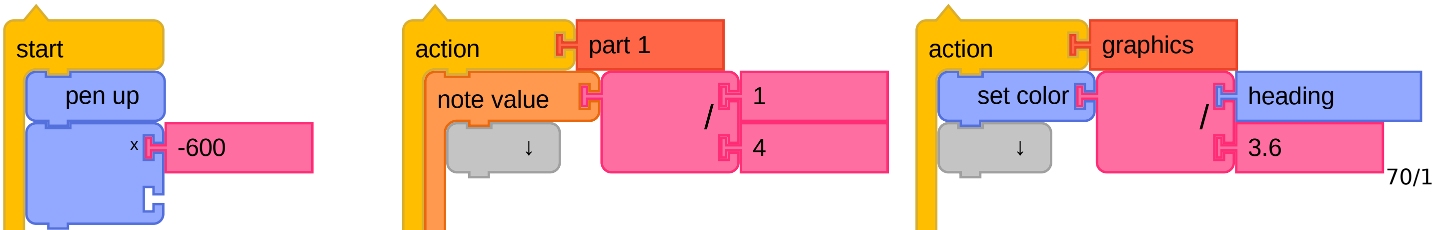
RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1523494709674021&run=True)



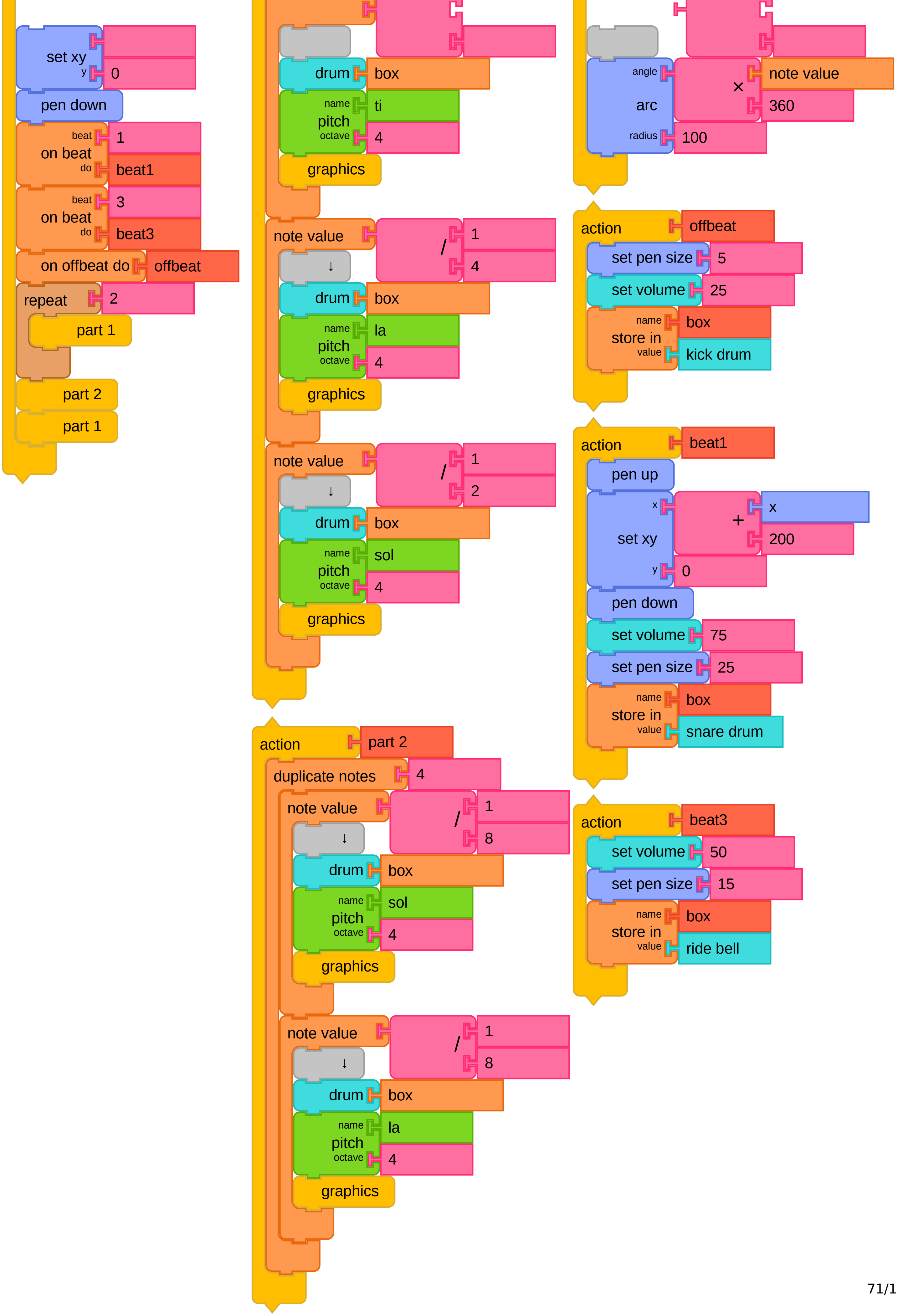
```
action chunk
  note value 1 / 4
    ↓
    pitch name re octave 4
    forward 100
  add to color value 8
  note value 1 / 4
    ↓
    pitch name mi octave 4
    forward 100
  add to color value 8
  note value 1 / 2
    ↓
    pitch name do octave 4
    back 200
    right 90
  add to color value 8

start
  repeat 4
    chunk
```

Another example of graphics synchronized to the music by placing the graphics commands inside of *Note value* blocks

RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1523106271018484&run=True)



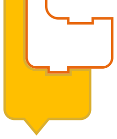
```
start
  pen up
  x -600

action part 1
  note value 1 / 4
    ↓

action graphics
  set color heading / 3.6
  ↓
```



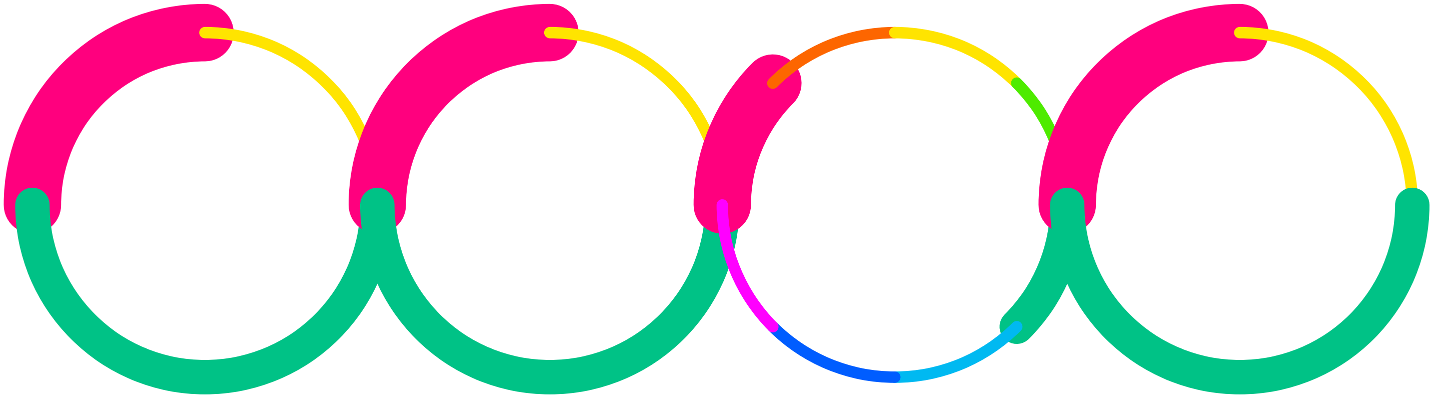
```
set xy
  y 0
pen down
on beat
  beat 1
  do beat1
on beat
  beat 3
  do beat3
on offbeat do offbeat
repeat 2
  part 1
part 2
part 1
```

```
  drum box
  pitch
    name ti
    octave 4
  graphics
note value 1 / 4
  ↓
  drum box
  pitch
    name la
    octave 4
  graphics
note value 1 / 2
  ↓
  drum box
  pitch
    name sol
    octave 4
  graphics

action part 2
  duplicate notes 4
    note value 1 / 8
      ↓
      drum box
      pitch
        name sol
        octave 4
      graphics
    note value 1 / 8
      ↓
      drum box
      pitch
        name la
        octave 4
      graphics
```

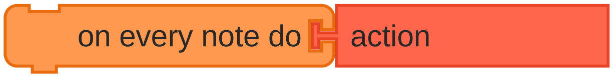
```
arc
  angle note value × 360
  radius 100

action offbeat
  set pen size 5
  set volume 25
  store in
    name box
    value kick drum

action beat1
  pen up
  set xy
    x x + 200
    y 0
  pen down
  set volume 75
  set pen size 25
  store in
    name box
    value snare drum

action beat3
  set volume 50
  set pen size 15
  store in
    name box
    value ride bell
```



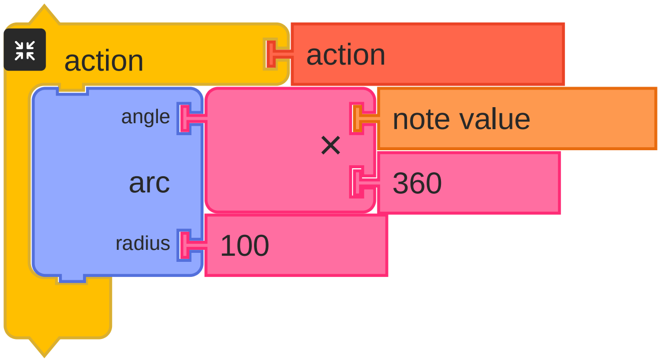
Another approach to graphics is to use modulate them based on the beat. In the example above, we call the same graphics action for each note, but the parameters associated with the action, such as pen width, are dependent upon which beat we are on. On Beat 1, the pen size is set to `50` and the volume to `75`. On Beat `3`, the pen size is set to `25` and the volume to `50`. On off beats, the pen size is set to `5` and the volumne to `5`. The resultant graphic is shown below.



The *On-Every-Note-Do* block lets you specify an action to take whenever a note is played. In the example above, the note value is used to determine the portion of an arc to draw, i.e., a 1/4 note draws a 1/4 circle, a 1/2 note draw 1/2 circle, and a whole note draws a full circle.

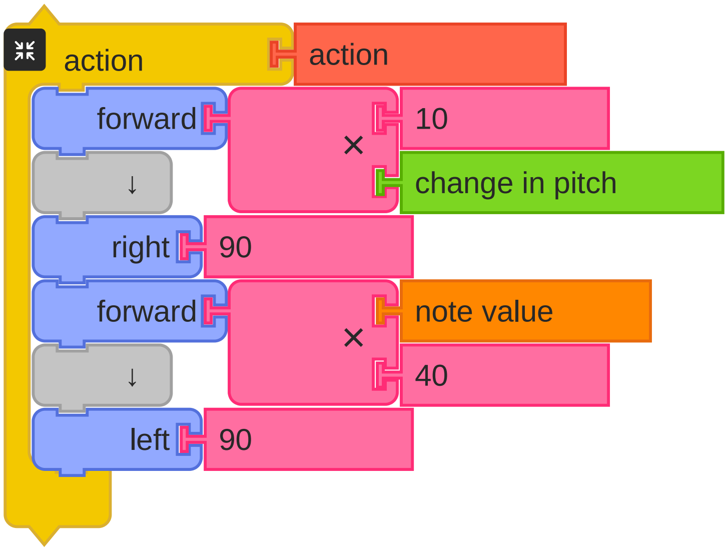


```
on every note do  action
```



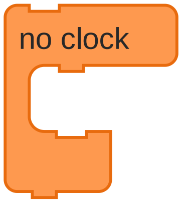
```
action  action
  arc  angle  ×  note value
                 360
       radius 100
```

The *On-Every-Note-Do* block is found in the Crab Canon project on the Planet to "plot the music". The mouse moves up and down based on the change in pich between notes and to the right in proportion to the note value.



RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1522885323588493)

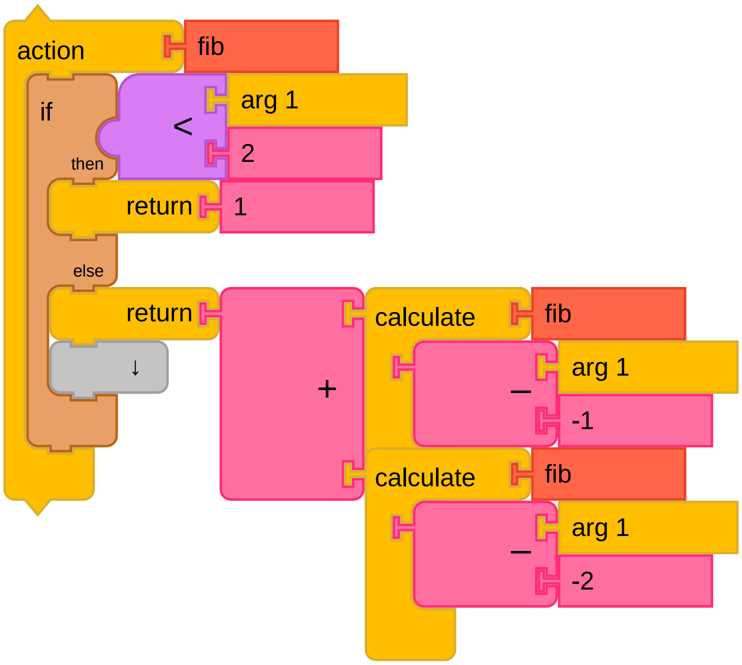
Music Blocks has an internal "conductor" maintaining the beat. When the Run button is clicked, the program begins and an internal master (or "conductor") clock starts up. All of the music tries to stay synced to that clock.



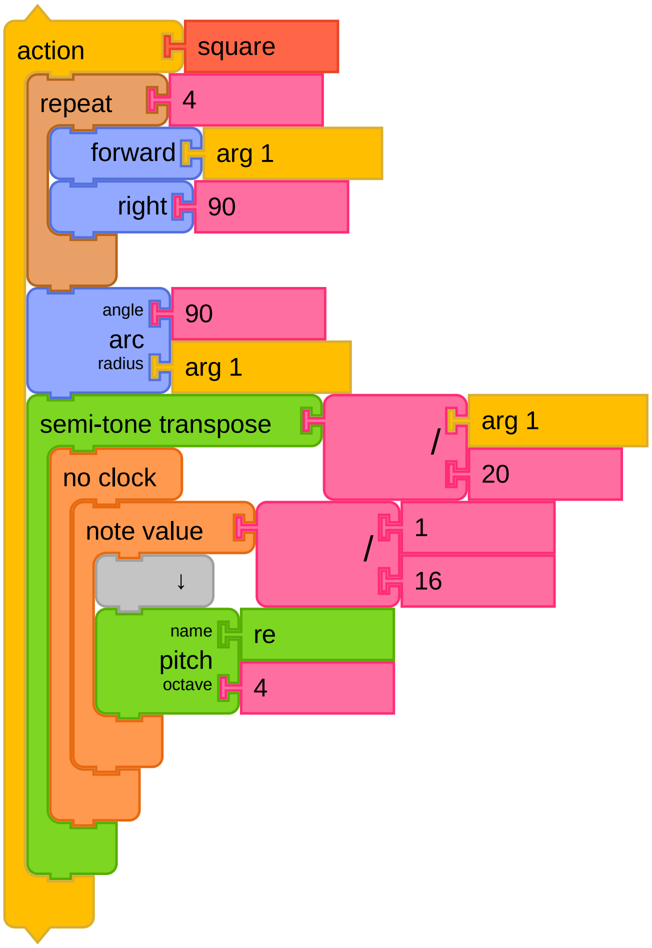
For example, if you have multiple voices (mice), they all share the same conductor in order to keep on the same beat. If a voice (mouse) is falling behind, Music Blocks tries to catch up on the next note by truncating it. If it is an 1/8 note behind and the next note is a 1/2 note, then only an 3/8 note would be played, so as to catch up. That is a somewhat extreme example—usually the timing errors are only very very small differences.

But in some situations, the timing errors can be very large. This is when the *No-clock* block is used.

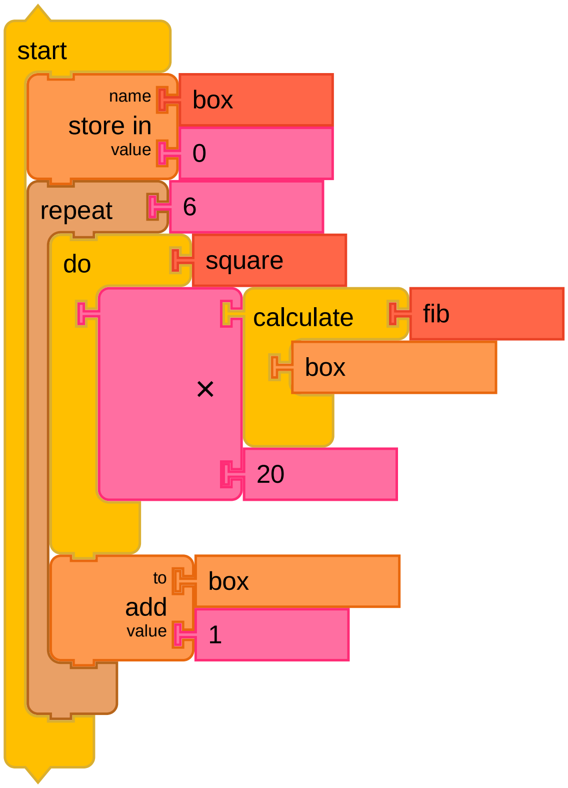
A typical problem is when the music is not played continuously. Imagine an interactive game where a hero is battling a monster. Our hero plays theme music whenever the monster is defeated. But that might occur at any time, hence it is not going to be in sync with the conductor. The offset could be tens of seconds. This would mean that all of the notes in the theme music might be consumed by trying to catch up with the conductor. The *No-clock* block essentially says, do your own thing and don't worry about the conductor.



```
action  fib
  if  <  arg 1
         2
  then
    return  1
  else
    return  +  calculate  fib
                  –  arg 1
                     -1
               calculate  fib
                  –  arg 1
                     -2
    ↓
```

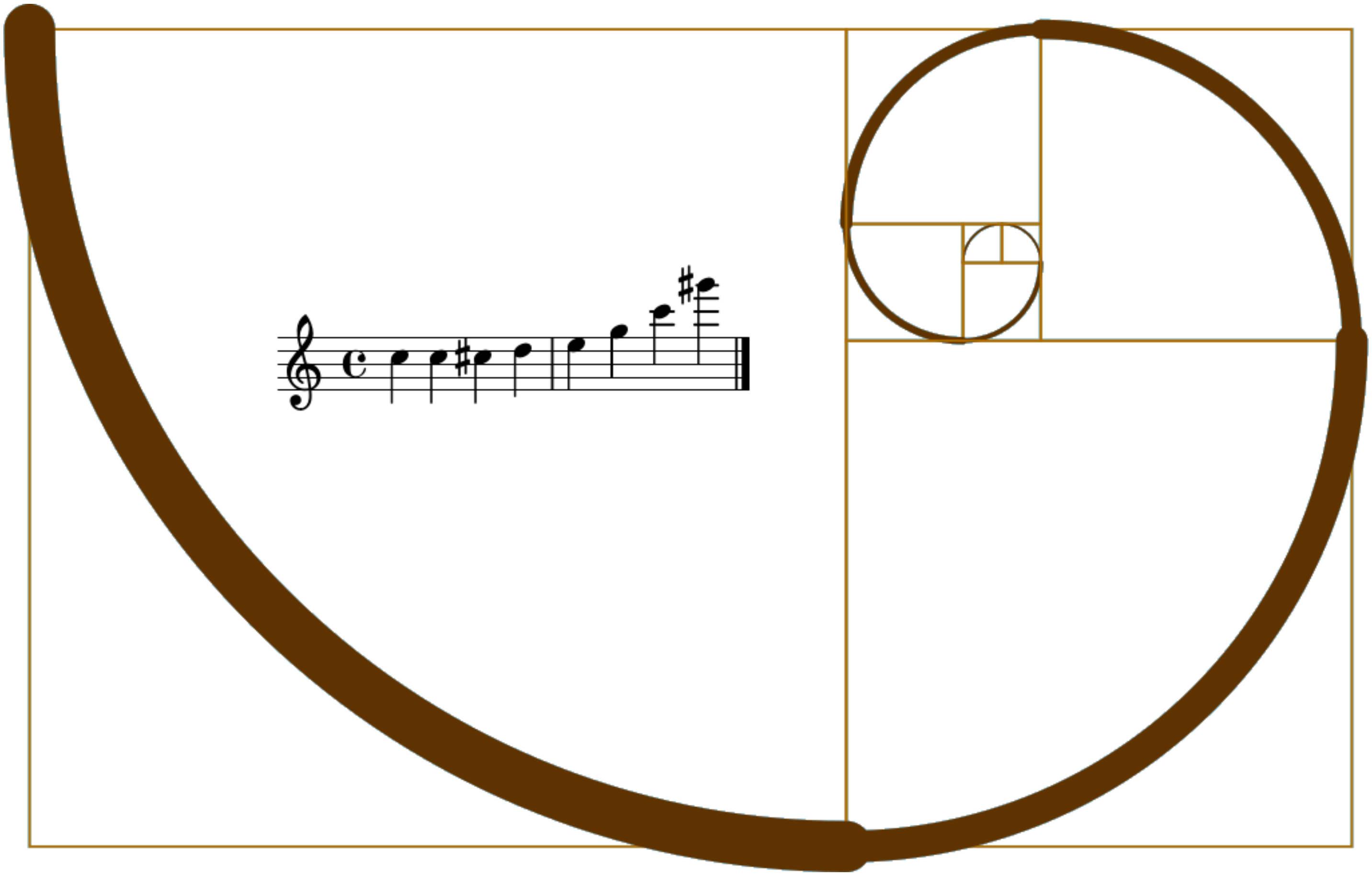


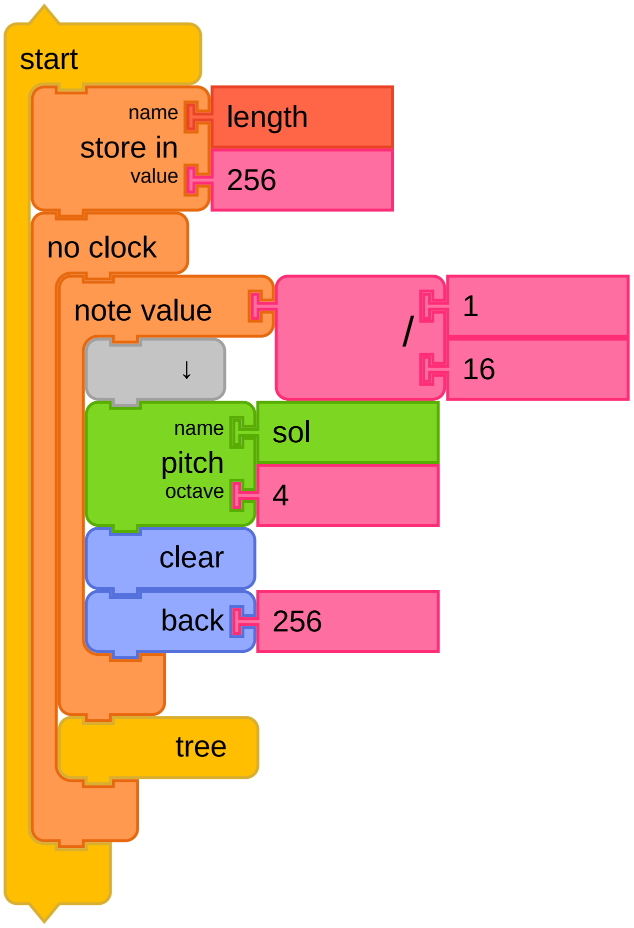
```
action  square
  repeat  4
    forward  arg 1
    right  90
  arc  angle   90
       radius  arg 1
  semi-tone transpose  /  arg 1
                          20
    no clock
      note value  /  1
                     16
        ↓
        pitch  name    re
               octave  4
```



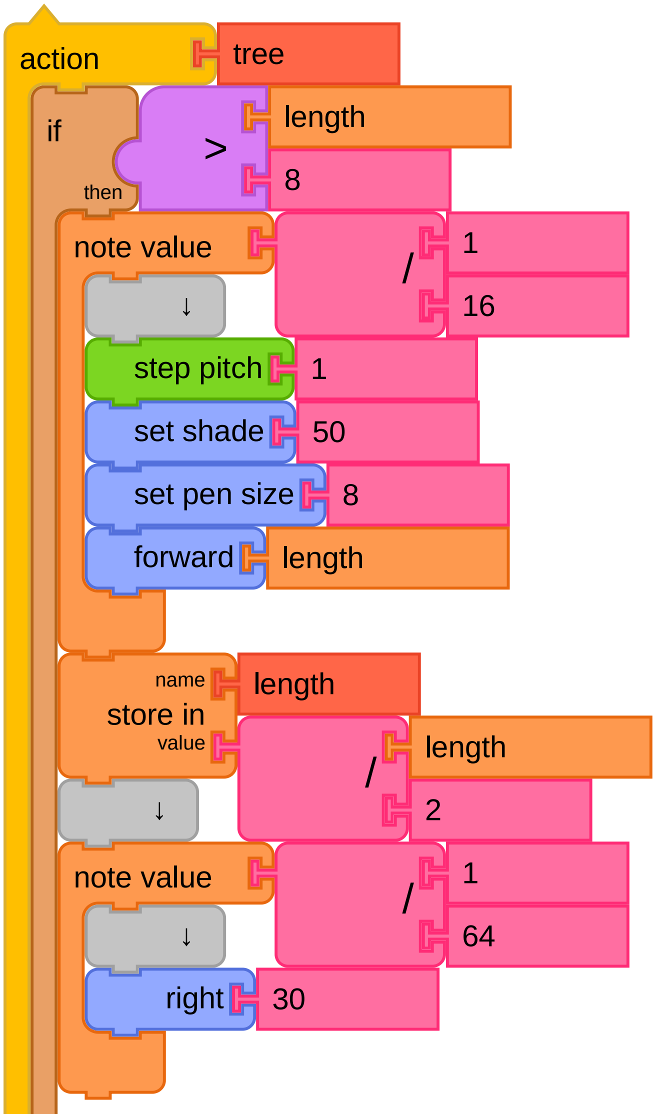
```
start
  store in  name   box
            value  0
  repeat  6
    do  square
        ×  calculate  fib
                      box
           20
    add  to     box
         value  1
```

In this example, because the computation and graphics are more complex, a *No-clock* block is used to decouple the graphics from the master clock. The "No-clock* block prioritizes the sequence of actions over the specified rhythm.

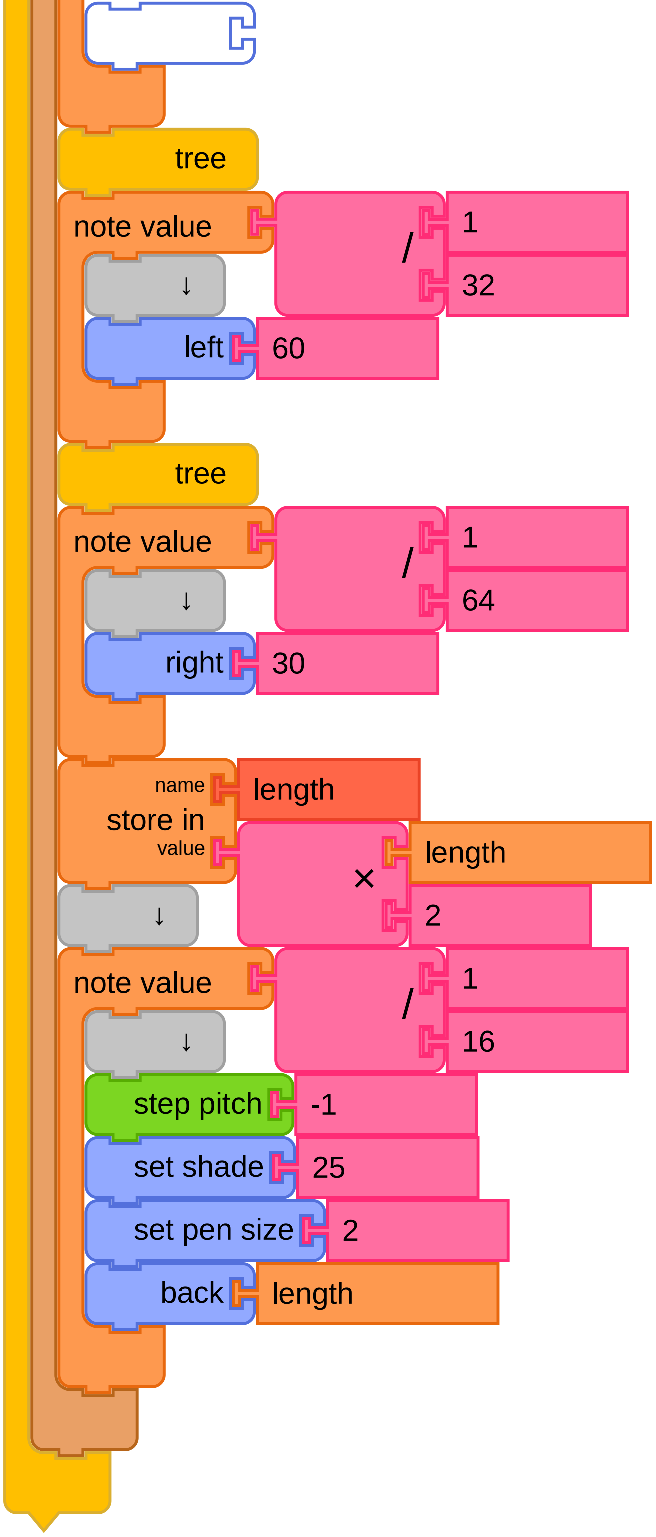




```
start
  store in  name: length  value: 256
  no clock
    note value  1 / 16
      pitch  name: sol  octave: 4
      clear
      back 256
    tree
```

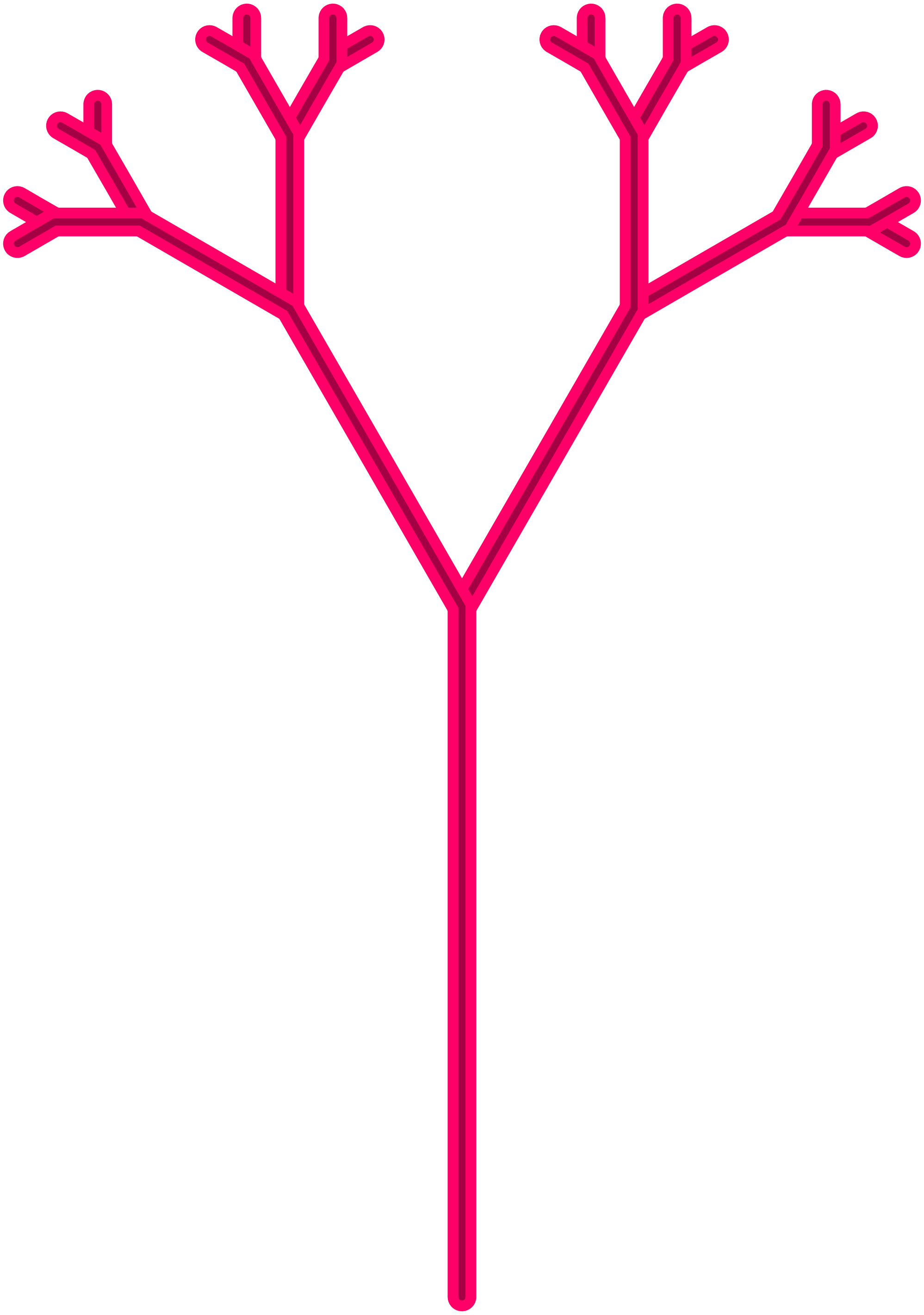


```
action  tree
  if  length > 8  then
    note value  1 / 16
      step pitch 1
      set shade 50
      set pen size 8
      forward length
    store in  name: length  value: length / 2
    note value  1 / 64
      right 30
```



```
tree
note value 1 / 32
    ↓
    left 60
tree
note value 1 / 64
    ↓
    right 30
store in name length value length × 2
↓
note value 1 / 16
    ↓
    step pitch -1
    set shade 25
    set pen size 2
    back length
```

Another example of embedding graphics into notes: in case, a recursive tree drawing, where the pitch goes up as the branches ascend.



RUN LIVE (https://musicblocks.sugarlabs.org/index.html?id=1523029986215035&run=True)