Introduction to big-bang Animations, Part III

We're still choosing (or making):

• an initial image
• a unary function with contract image -> image

(require picturing-programs)

Let's make a function that puts two copies of an image beside itself.

; double : image -> image
(define (double an-image)
  (beside an-image an-image))

(define a-bird)

; Predict and try the following:
; a-bird
; (double a-bird)
; (double (double a-bird))
; (double (double (double a-bird)))

That's getting bigger and bigger very fast (does it remind you of some
counting earlier in the course?).
Let's keep the size of the whole image the same.

; double-shrink : image -> image
(define (double-shrink an-image)
  (scale 1/2 (beside an-image an-image)))

; Predict and try the following:
; a-bird
; (double-shrink a-bird)
; (double-shrink (double-shrink a-bird))
; (double-shrink (double-shrink (double-shrink a-bird)))
; The doubling is only horizontal, so let's double it vertically as well.
; And for even more fun let's make it a rosette.

; The use of freeze in the function is for speed on these more complex images,
; which we'll discuss later when discussing image representation.

; rosette-shrink : image -> image
(define (rosette-shrink an-image)
  (freeze
   (scale 1/2
     (above
      (beside an-image (flip-horizontal an-image))
      (flip-vertical (beside an-image (flip-horizontal an-image)))))))

; EXERCISE: write out the expressions for the first few frames of the
; animation manually, and try to predict how they will look in advance.

; Start with a-bird, then once per second use rosette-shrink to produce a
; new image, five times:
(big-bang a-bird
  (on-tick rosette-shrink 1 5)
  (on-draw show-it))

; EXERCISE: Try that with double-shrink.