```
(define (next-eggs loe)
  (cond [(empty? loe) empty]
        [else
         ;temporarily ignore reference rule:
         ;(cons (make-egg (egg-x (first loe))
         ;                (+ (egg-y (first loe)) FALL-SPEED)
         ;                (+ (egg-r (first loe)) SPIN-SPEED))
         ;combination: cons is generic
         ;contribution: egg specific uses x, y and r
         ;---> combination in list fn, contribution in helper
         (cons (next-egg (first loe))
               (next-eggs (rest loe)))]))


(define (render-eggs loe)
  (cond [(empty? loe) MTS]
        [else
         ;temporarily ignore reference rule:
         ;(place-image (rotate (egg-r (first loe)) YOSHI-EGG)
         ;             (egg-x (first loe))
         ;             (egg-y (first loe))
         ;             (render-eggs (rest loe)))
         ;combination: place-image needs x, y
         ;contribution: uses r
         ;---> combination is in helper, also does contribution work
         (place-egg (first loe)
                    (render-eggs (rest loe)))]))
```