2.2.4 Picture Language

```
;; (define wave2 (beside wave (flip-vert wave)))
;; (define wave4 (below wave2 wave2))

(define (flipped-pairs painter)
  (let ((painter2 (beside painter (flip-vert painter))))
    (below painter2 painter2) ))

;; (define wave4 (flipped-pairs wave))

(define (right-split painter n)
  (if (= n 0)
      painter
      (let ((smaller (right-split painter (- n 1))))
        (beside painter (below smaller smaller)) )))

(define (corner-split painter n)
  (if (= n 0)
      painter
      (let ((up (up-split painter (- n 1)))
            (right (right-split painter (- n 1))) )
        (let ((top-left (beside up up))
              (bottom-right (below right right))
              (corner (corner-split painter (- n 1))) )
          (beside (below painter top-left)
                  (below bottom-right corner) )))))

(define (square-limit painter n)
  (let ((quarter (corner-split painter n)))
    (let ((half (beside (flip-horiz quarter) quarter)))
      (below (flip-vert half) half) )))
```

;; Higher-order operations

```
(define (square-of-four tl tr bl br)
  (lambda (painter)
    (let ((top (beside (tl painter) (tr painter)))
          (bottom (beside (bl painter) (br painter))) )
      (below bottom top) )))

(define (flipped-pairs painter)
  (let ((combine4 (square-of-four identity flip-vert
                                  identity flip-vert)))
    (combine4 painter) ))

(define flipped-pairs-new
   (square-of-four identity flip-vert identity flip-vert) )
```

/staff/umatani/local/bin/tustk,tustk2 TUS/TK **本体**
/staff/umatani/local/lib/tustk/demos **サンプル**
一部パスが違う可能性もあるので御自分で修正すること
(tk:loop) (tk:update) **画面のリフッシュ**
(load '**パス**/picl) **の後に** (load '**パス**/picl-test)
(make-canvas frm1)
((square-limit wave 2) frm1)

;; 傾いたフレーム picl.lsp

```
(define frm2
  (make-frame
    (make-vect 0 0)
    (make-vect 1.0 0)
    (make-vect 0.5 0.5) ))
```

;; wave 小宮先生が入手

```
(define wave
  (let ((p01 (make-vect 0.40 1.00))
        (p02 (make-vect 0.60 1.00))
        (p03 (make-vect 0.00 0.80))
        (p04 (make-vect 0.35 0.80))
        (p05 (make-vect 0.65 0.80))
        (p06 (make-vect 0.00 0.60))
        (p07 (make-vect 0.30 0.60))
        (p08 (make-vect 0.40 0.60))
        (p09 (make-vect 0.60 0.60))
        (p10 (make-vect 0.70 0.60))
        (p11 (make-vect 0.20 0.55))
        (p12 (make-vect 0.30 0.55))
        (p13 (make-vect 0.35 0.50))
        (p14 (make-vect 0.65 0.50))
        (p15 (make-vect 0.20 0.45))
        (p16 (make-vect 1.00 0.40))
        (p17 (make-vect 0.50 0.20))
        (p18 (make-vect 1.00 0.20))
        (p19 (make-vect 0.25 0.00))
        (p20 (make-vect 0.40 0.00))
        (p21 (make-vect 0.60 0.00))
        (p22 (make-vect 0.75 0.00)) )
    (segments->painter
      (list (make-segment p01 p04)
            (make-segment p04 p08)
            (make-segment p08 p07)
            (make-segment p07 p11)
            (make-segment p11 p03)
            (make-segment p06 p15)
            (make-segment p15 p12)
            (make-segment p12 p13)
            (make-segment p13 p19)
            (make-segment p20 p17)
            (make-segment p17 p21)
            (make-segment p22 p14)
            (make-segment p14 p18)
            (make-segment p16 p10)
            (make-segment p10 p09)
            (make-segment p09 p05)
            (make-segment p05 p02)
      ))))
```

```
;;  Frames
(define (frame-coord-map frame)
  (lambda (v)
    (add-vect
     (origin-frame frame)
     (add-vect (scale-vect (xcor-vect v) (edge1-frame frame))
               (scale-vect (ycor-vect v) (edge2-frame frame)) ))))

;: ((frame-coord-map a-frame) (make-vect 0 0))
;: (origin-frame a-frame)

(define (make-frame origin edge1 edge2) (list origin edge1 edge2))

;(define (make-frame origin edge1 edge2) (cons origin (cons edge1 edge2)))

;;  Painters

(define (segments->painter segment-list)
  (lambda (frame)
    (for-each
     (lambda (segment)
       (draw-line ((frame-coord-map frame) (start-segment segment))
                  ((frame-coord-map frame) (end-segment segment)) ))
     segment-list )))

(define (transform-painter painter origin corner1 corner2)
  (lambda (frame)
    (let ((m (frame-coord-map frame)))
      (let ((new-origin (m origin)))
        (painter
         (make-frame new-origin (sub-vect (m corner1) new-origin)
                                (sub-vect (m corner2) new-origin) ))))))

(define (flip-vert painter)
  (transform-painter painter (make-vect 0.0 1.0) (make-vect 1.0 1.0) (make-vect 0.0 0.0))
                             ;  new origin        ;  new end of edge1    ;  new end of edge2

(define (shrink-to-upper-right painter)
  (transform-painter painter
                     (make-vect 0.5 0.5) (make-vect 1.0 0.5) (make-vect 0.5 1.0) ))

(define (rotate90 painter)
  (transform-painter painter (make-vect 1.0 0.0) (make-vect 1.0 1.0) (make-vect 0.0 0.0)))

(define (squash-inwards painter)
  (transform-painter painter (make-vect 0.0 0.0)(make-vect 0.65 0.35)(make-vect 0.35 0.65)

(define (beside painter1 painter2)
  (let ((split-point (make-vect 0.5 0.0)))
    (let ((paint-left
           (transform-painter painter1
                              (make-vect 0.0 0.0)
                              split-point
                              (make-vect 0.0 1.0) ))
          (paint-right
           (transform-painter painter2
                              split-point
                              (make-vect 1.0 0.0)
                              (make-vect 0.5 1.0) )))
      (lambda (frame) (paint-left frame) (paint-right frame)) )))
```

3　宿題 – 〆切: 1月24日 (月) 午後5時 事務室レポート箱

- **Exercise 2.44 〜 2.52**
- **プログラムが動くことにより解の正しさを確認すること.**

(添削は試験後)