6.4

;Aufgabe 6-4

```
; a)

; A Shape is either:
; - a GCircle
; - a GSquare
; interp. a geometrical shape representing a circle or a rectangle


(define-struct gcircle (center radius reward))
; A GCircle is a structure (make-gcircle Posn Number Number)
; interp. the geometrical representation of a circle along with
;<reward>: the score, which the player gets when he collects a circle

(define-struct gsquare (corner-ul side reward))
; A GSquare is a structure (make-gsquare Posn Number Number)
; interp. the geometrical representation of a square along with
;<reward>: the score, which the player gets when he collects a square
; where corner-ul is the upper left corner
; and side is the length of the square's side

; b)

(define-struct level (stage goal))
; A Level is a structure (make-level String Number)
; interp. a level of the game
; where stage is the name of the current level
; and goal is the score required to pass the current level

(define-struct lst (first rest))
; A (List-of-Levels) is one of:
; - (cons Levels (List-of-Levels))
; - empty
;interp. the head and rest of a list of levels, or the empty list

; GAME


; A game is a structure: make-game( Posn Posn Number Number List-of-Levels)
; interp. the current state of the game, where:
; - loc is the current position of the rocket
; - vel is the moving-direction of the rocket
; - score is the player's current score
; - speed is the rocket's current speed
; - levels is the list of levels the player needs to play
(define-struct game (loc vel score speed levels))

;c
; Posn Posn -> Number
```

```
; Determines the distance betwwen two Points
(define (distance p1 p2)(sqrt (+
               (sqr (- (posn-x p1) (posn-x p2)))
               (sqr (- (posn-y p1) (posn-y p2))))))


; GCircle Posn Image-> Boolean
; Determines whether an image overlaps with a circle,
; where point is the center point of the image
(define (point-inside-circle circle point)
  (<= (distance point (gcircle-center circle)) (gcircle-radius circle)))

; GSquare Posn -> Boolean
; Determines whether a point is inside a square
(define (point-inside-square square point)
(and
(<= 0 (- (posn-x point) (posn-x (gsquare-corner-ul square))) (gsquare-side square))
(<= 0 (- (posn-y (gsquare-corner-ul square)) (posn-y point)) (gsquare-side square))))




; Shape WorldState -> Boolean
; check if the rocket hit an object of the currentlevel
(define (hit? shape ws)(cond [(gcircle? shape) (point-inside-circle shape (game-loc ws))]
                             [(gsquare? shape) (point-insied-square shape (game-loc ws))]))

(define WIDTH 400)
(define HEIGHT 400)
(define MTSCN (empty-scene WIDTH HEIGHT))
(define ROCKET rocket.png) ;rocket bild im racket

; Posn -> Posn
; Normalizes the given vector
(check-expect (vec-normalize (make-posn 1 0)) (make-posn 1 0))
(check-expect (vec-normalize (make-posn 2 0)) (make-posn 1 0))
(check-expect (vec-normalize (make-posn 0 1)) (make-posn 0 1))
(check-expect (vec-normalize (make-posn 0 2)) (make-posn 0 1))
(check-within (vec-normalize (make-posn 1 1)) (make-posn (/ 1 (sqrt 2)) (/ 1 (sqrt 2))) 0.00000001 )
(define (vec-normalize v)
  (make-posn
   (/ (posn-x v) (vec-len v) )
   (/ (posn-y v) (vec-len v) )))

; Posn -> Number
; Determines the length of the given vector
(check-expect (vec-len (make-posn 1 0)) 1)
(check-expect (vec-len (make-posn 2 0)) 2)
(check-expect (vec-len (make-posn 0 1)) 1)
(check-expect (vec-len (make-posn 0 2)) 2)
(check-within (vec-len (make-posn 1 1)) (sqrt 2) 0.00000001 )
(define (vec-len v)
        (sqrt (+
               (* (posn-x v) (posn-x v))
```

```
               (* (posn-y v) (posn-y v)))))

; Number Posn -> Posn
; Multiplies the given vector with the given scalar
(check-expect (vec-skal-mult 2 (make-posn 1 0)) (make-posn 2 0))
(check-expect (vec-skal-mult 0.5 (make-posn 2 0)) (make-posn 1 0))
(check-expect (vec-skal-mult 2 (make-posn 0 1)) (make-posn 0 2))
(check-expect (vec-skal-mult 0.5 (make-posn 0 2)) (make-posn 0 1))
(check-expect (vec-skal-mult 2 (make-posn 1 1)) (make-posn 2 2) )
(define (vec-skal-mult s v)
  (make-posn
   (* s (posn-x v))
   (* s (posn-y v))))

; Posn Posn -> Posn
; Computes v1 - v2
(check-expect (vec-sub (make-posn 1 0) (make-posn 1 0)) (make-posn 0 0))
(check-expect (vec-sub (make-posn 1 0) (make-posn 0 0)) (make-posn 1 0))
(check-expect (vec-sub (make-posn 0 0) (make-posn 0 1)) (make-posn 0 -1))
(define (vec-sub v1 v2)
  (make-posn
   (- (posn-x v1) (posn-x v2))
   (- (posn-y v1) (posn-y v2))))

; Posn Posn -> Posn
; Computes v1 + v2
(check-expect (vec-add (make-posn 1 0) (make-posn 1 0)) (make-posn 2 0))
(check-expect (vec-add (make-posn 1 0) (make-posn 0 0)) (make-posn 1 0))
(check-expect (vec-add (make-posn 0 0) (make-posn 0 1)) (make-posn 0 1))
(define (vec-add v1 v2)
  (make-posn
   (+ (posn-x v1) (posn-x v2))
   (+ (posn-y v1) (posn-y v2))))




; GAME


; A game is a structure: make-game( Posn Posn Number Number)
; interp. the current state of the game, where:
; - loc is the current position of the rocket
; - vel is the moving-direction of the rocket
; - score is the player's current score
; - speed is the rocket's current speed
(define-struct game (loc vel score speed))

; WorldState is game
; WorldState Posn -> WorldState
; Changes the velocity of the rocket towards the mouse-location
(check-expect (click/direction
  (make-game (make-posn 200 200) (make-posn 200 201) 30 31) (make-posn 300 200))
    (make-game (make-posn 200 200) (make-posn 1 0) 30 31))
```

```
(define (click/direction ws mouse)
  (make-game
       (game-loc ws)
       (vec-normalize (vec-sub mouse (game-loc ws)))
       (game-score ws)
       (game-speed ws)))

; WorldState -> WorldState
; Increases the speed of the rocket by 1
(check-expect (click/speed (make-game (make-posn 200 200) (make-posn 200 201) 30 31))
  (make-game
       (make-posn 200 200)
       (make-posn 200 201) 30 32))
(define (click/speed ws)
  (make-game
       (game-loc ws)
       (game-vel ws)
       (game-score ws)
       (+ 1 (game-speed ws))))

; WorldState Number Number MouseEvent -> WorldState
; Changes the rocket's direction --> Moves towards the position of the mouse-click
(check-expect (on-mouse-event (make-game (make-posn 200 200) (make-posn 200 201) 30 31)
  300 200 "button-up")(make-game (make-posn 200 200) (make-posn 200 201) 30 31))
(check-expect (on-mouse-event (make-game (make-posn 200 200) (make-posn 200 201) 30 31)
  300 200 "button-down")(make-game (make-posn 200 200)
                                   (make-posn 1 0) 30 32))
(define (on-mouse-event ws mouse-x mouse-y mouse-event)
  (if (string=? mouse-event "button-down")
      (click/speed (click/direction ws (make-posn mouse-x mouse-y)))
      ws))


; WorldState -> WorldState
; Moves the rocket according to its direction-vector and the current speed
(check-expect (tick/move (make-game (make-posn 200 200) (make-posn 200 201) 30 31))
  (make-game (make-posn 6400 6431) (make-posn 200 201) 30 31))
(define (tick/move ws)
  (make-game
   (vec-add (game-loc ws) (vec-skal-mult (game-speed ws) (game-vel ws) ))
   (game-vel ws)
   (game-score ws)
   (game-speed ws)))

; WorldState -> WorldState
; Increases the current score
(check-expect (tick/score  (make-game (make-posn 200 200) (make-posn 200 201) 30 31))
  (make-game (make-posn 200 200) (make-posn 200 201) 31 31))
(define (tick/score ws)
  (make-game
   (game-loc ws)
   (game-vel ws)
   (+ 1 (game-score ws))
   (game-speed ws)))
```

```
; WorldState -> WorldState
; Moves the rocket according to its direction-vector and the current speed
(check-expect (on-tick-event (make-game (make-posn 200 200) (make-posn 200 201) 30 31))
  (make-game (make-posn 6400 6431) (make-posn 200 201) 31 31))
(define (on-tick-event ws)
  (tick/score (tick/move ws)))

; WorldState -> Image
; Renders the game according to the current state
(check-expect (render (make-game (make-posn 200 200) (make-posn 200 201) 30 31))
  (place-image/align (text "Score: 30" 12 "blue")
    5 5 "left" "top"
    (place-image ROCKET 200 200 MTSCN)))
(check-expect (render (make-game (make-posn 500 500) (make-posn 200 201) 30 31))(place-image
  (above (text "GAME OVER" 30 "red")
  (text "Score: 30" 20 "blue"))
  200 200 MTSCN ))
(define (render ws)
  (if (end-of-world ws)
      (render/loose ws)
      (render/playing ws)))


; WorldState -> Image
; Renders the game while playing
(check-expect (render/playing (make-game (make-posn 200 200) (make-posn 200 201) 30 31))(place-image/ali
  5 5 "left" "top"
  (place-image ROCKET 200 200 MTSCN)))
(define (render/playing ws)
  (place-image/align (text (string-append "Score: " (number->string (game-score ws))) 12 "blue")
                     5 5 "left" "top"
                     (place-image ROCKET (posn-x (game-loc ws)) (posn-y (game-loc ws)) MTSCN)))

; WorldState -> Image
; Renders the screen, if the game is over
(check-expect (render/loose (make-game (make-posn 200 200) (make-posn 200 201) 30 31))(place-image
  (above (text "GAME OVER" 30 "red")
  (text "Score: 30" 20 "blue"))
  200 200 MTSCN ))
(define (render/loose ws)
  (place-image
   (above (text "GAME OVER" 30 "red")
          (text (string-append "Score: " (number->string (game-score ws))) 20 "blue"))
   (/ WIDTH 2) (/ HEIGHT 2) MTSCN ))

; WorldState -> Bool
; Checks whether the game is over (the rocket touches the scene-bounds)
(check-expect (end-of-world (make-game (make-posn 200 200) (make-posn 200 201) 30 31))false)
(check-expect (end-of-world (make-game (make-posn 500 500) (make-posn 200 201) 30 31))true)
(check-expect (end-of-world (make-game (make-posn 200 500) (make-posn 200 201) 30 31))true)
(check-expect (end-of-world (make-game (make-posn 500 200) (make-posn 200 201) 30 31))true)
(define (end-of-world ws)
  (not (and (<= 0 (posn-x (game-loc ws)) WIDTH)
```

```
            (<= 0 (posn-y (game-loc ws)) HEIGHT) ) ) )


; BIG BANG
(big-bang (make-game
           (make-posn (/ WIDTH 2) (/ HEIGHT 2))
           (make-posn 0 -1)
           0 1)
          (on-tick on-tick-event)
          (on-mouse on-mouse-event)
          (to-draw render)
          (stop-when end-of-world render/loose))
```