Instituto Tecnológico de Costa Rica
Ingeniería en Computadores

Lenguajes, Compiladores e intérpretes (CE3104)

Segundo Semestre 2019

Tarea# 1

Eduardo Solano Amador - 2016015318

Luis Daniel Prieto Sibaja - 2016072504

1.1. Descripción de las funciones implementadas.

Las funciones **(ventana-nombre-jugadores# cantjugadores)** enumeradas respectivamente como ...jugadores1 ...jugadores2 y ...jugadores3 se encargan de recibir la cantidad de jugadores que van a jugar y crea esa misma cantidad de textfield donde se escriben los nombres de los jugadores. Esta función crea una lista con los nombres y la envía a la función principal **bCEj** la cual llama a su función complementaria o auxiliar **(juego-blackjack listaNombres cantJugadores)** enviandole la lista y la cantidad de jugadores que entran al juego.

```
;*********************************************************************************************
(define (ventana-nombre-jugadores2 cantjugadores)
     ; Make a frame by instantiating the frame% class
     ; Frame para los text field de nombre de los jugadores
     (define frame3 (new frame% [label "Nombres de jugadores "]))

     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     ; Text field para el nombre del o los jugadores
     (define text-field-jug1
          (new text-field%
            (label "Nombre del jugador 1: ")
            (parent frame3)
            (init-value "")) )
     (define text-field-jug2
          (new text-field%
            (label "Nombre del jugador 2: ")
            (parent frame3)
            (init-value "")))

     ; Make a button in the frame
     (new button% [parent frame3]
             [label "Continuar"]
             ; Callback procedure for a button click:
             [callback (lambda (button event)
                         (send frame3 show #f)
                         (bCEj  (list (send text-field-jug1 get-value) (send text-field-jug2 get-value)))))])

     ; Show the frame by calling its show method
     (send frame3 show #t)
)
```

La función **(juego-blackjack listaNombres cantJugadores)** es donde se presenta la ventana del juego donde los jugadores interactúan, es en esta función donde se crean los botones para cada jugador, donde se presentan las cartas, donde se compite contra el crupier y contra los otros jugadores. Dentro de esta funcion tambien existen otras funciones como **(colocar-imagen imagen cordx cordy)** que se encarga de colocar una imagen o carta en la posición que se desee, también aquí dentro existe la función **(control click matriz)** que es la encargada de verificar el momento en que el mouse da click en algún botón y seguidamente hace verificaciones y sumando puntajes a los jugadores o ya sea el botón para que el jugador se plante.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; la funcion complementaria o auxiliar a la principal
; recibe una lista con los nombres de los jugadores
; y la cantidad de jugadores en esa lista
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(define (juego-blackjack listaNombres cantJugadores)


   (define ventana (open-viewport "ventana" 1040 720))
   ;::::::::::::::::::::::
   ((draw-viewport ventana) "black")
   ;::::::::::::::::::::::
   (define ventana2 (open-pixmap "eso no importa!" 1040 720 #|tamaño de la ventana|#))
   ;::::::::::::::::::::::
   ((draw-pixmap ventana2) "fondo.png" (make-posn 0.0 0.0) "black")
   ;::::::::::::::::::::::
   (copy-viewport ventana2 ventana)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Funcion que recibe en una lista con informacion necesaria de
; una imagen para cargarla y colocarla en cordenadas x,y especificadas
(define (colocar-imagen imagen cordx cordy)
  (cond
    ((equal? (car (cdr (cdr imagen))) "D") (aux-colocar-imagen (~r (- (car imagen) 1)) (~r 1) cordx cordy))
    ((equal? (car (cdr (cdr imagen))) "H") (aux-colocar-imagen (~r (- (car imagen) 1)) (~r 2) cordx cordy))
    ((equal? (car (cdr (cdr imagen))) "T") (aux-colocar-imagen (~r (- (car imagen) 1)) (~r 3) cordx cordy))
    ((equal? (car (cdr (cdr imagen))) "S") (aux-colocar-imagen (~r (- (car imagen) 1)) (~r 4) cordx cordy))))

(define (aux-colocar-imagen strvalor strsimbolo cordx cordy)
  ((draw-pixmap ventana2) (string-append "cartas/card-" strvalor "-" strsimbolo ".png")   (make-posn cordx cordy) "black"))
```

La función **(pantalla-final puntuacion listaNombres ganador cantJugadores)** es la encargada de crear una nueva ventana que muestra la pantalla final con las puntuaciones de cada jugador y de decir quien es el ganador de la partida.

```
;pantalla final donde se muestran todos la lista de puntajes y ganador final
(define (pantalla-final puntuacion listaNombres ganador cantJugadores)
  (define ventana (open-viewport "ventana Puntajes" 300 380))                    ;define y abre la ventana
   ;::::::::::::::::::::::
   ((draw-viewport ventana) "black")                                           ;color de "ventana"
   ;::::::::::::::::::::::
   (define ventana2 (open-pixmap "eso no importa!" 300 380 #|tamaño de la ventana|#))   ;define la ventana auxiliar (llevará el fondo)
   ;::::::::::::::::::::::
   ((draw-pixmap ventana2) "fondonegro.jpg" (make-posn 0.0 0.0) "black")         ;adicionar fondo a la ventana auxiliar
   ;::::::::::::::::::::::
   (copy-viewport ventana2 ventana)                                            ;copiamos el contenido de una ventana2 a ventana

   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Muestra los nombres de los jugadores;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;y crupier;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ((draw-string ventana2) (make-posn 70 50)"SE ACABÓ EL JUEGO!!!""red")
   ((draw-string ventana2) (make-posn 20 110) (string-append "EL GANADOR ES: " ganador) "red")


    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Muestra los nombres de los jugadores;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;y crupier;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   (cond
     ((equal? cantJugadores 1) ((draw-string ventana2) (make-posn 20 200)(string-append (car listaNombres) ": "  (~r(car puntuacion)))"red")            ;nombre y puntuacion jugac

                               ((draw-string ventana2) (make-posn 20 250)(string-append "CRUPIER: " (~r(car (cdr (cdr(cdr puntuacion)))))  "red")     ;puntuacion crupier
                               )

     ((equal? cantJugadores 2) ((draw-string ventana2) (make-posn 20 200)(string-append (car listaNombres) ": "  (~r(car puntuacion)))"red")            ;nombre y puntuacion jugac

                               ((draw-string ventana2) (make-posn 20 250)(string-append (car (cdr listaNombres)) ": " (~r(car (cdr puntuacion))))"red")  ;nombre y puntuacion jugac

                               ((draw-string ventana2) (make-posn 20 300)(string-append "CRUPIER: " (~r(car (cdr (cdr(cdr puntuacion)))))  "red")     ;puntuacion crupier
                               )

     ((equal? cantJugadores 3) ((draw-string ventana2) (make-posn 20 200)(string-append (car listaNombres) ": "  (~r(car puntuacion)))"red")            ;nombre y puntuacion jugac

                               ((draw-string ventana2) (make-posn 20 250)(string-append (car (cdr listaNombres)) ": " (~r(car (cdr puntuacion))))"red")  ;nombre y puntuacion jugac
```

1.2. Descripción de las estructuras de datos desarrolladas.

Se desarrolló una **matriz** por medio de una función llamada baraja, en la cual se guardan 52 **listas** representando las 52 cartas del mazo de cartas. Cada lista cuenta con 3 valores, uno es el valor de la carta, el otro es el nombre o número por el cual se identifica, por ejemplo Queen, Ace, 5, etc y el tercer valor es el símbolo de la carta siendo ¨S¨ el símbolo de espadas, "T" el de trébol "D" diamantes y "H" el de corazones. Con esta matriz se pueden buscar posiciones aleatorias para así poder barajar el mazo mediante otra función y con los valores de cada lista o carta se puede buscar en una carpeta la carta que corresponde y presentarla en la pantalla. También se usaron listas para guardar nombres de los jugadores, puntajes, cartas utilizadas por cada jugador.

```
(define(baraja id)
  (cond((zero? id)
        '())
       (else
        '((1 "Ace" "D")(1 "Ace" "S")(1 "Ace" "T")(1 "Ace" "H")
                        (2 "2" "D")(2 "2" "S")(2 "2" "T")(2 "2" "H")
                        (3 "3" "D")(3 "3" "S")(3 "3" "T")(3 "3" "H")
                        (4 "4" "D")(4 "4" "S")(4 "4" "T")(4 "4" "H")
                        (5 "5" "D")(5 "5" "S")(5 "5" "T")(5 "5" "H")
                        (6 "6" "D")(6 "6" "S")(6 "6" "T")(6 "6" "H")
                        (7 "7" "D")(7 "7" "S")(7 "7" "T")(7 "7" "H")
                        (8 "8" "D")(8 "8" "S")(8 "8" "T")(8 "8" "H")
                        (9 "9" "D")(9 "9" "S")(9 "9" "T")(9 "9" "H")
                        (10 "10" "D")(10 "10" "S")(10 "10" "T")(10 "10" "H")
                        (10 "Jack" "D")(10 "Jack" "S")(10 "Jack" "T")(10 "Jack" "H")
                        (10 "Queen" "D")(10 "Queen" "S")(10 "Queen" "T")(10 "Queen" "H")
                        (10 "King" "D")(10 "King" "S")(10 "King" "T")(10 "King" "H")))))
```

1.3. Descripción detallada de los algoritmos desarrollados.

-El algoritmo principal del juego se llama **(control click matriz puntuacion cantPlantes)** la cual funciona por recursividad, dentro de esa función se crearon eventos del mouse que funciona como botones, sea tomar carta o plantarse, cada vez que un botón sea presionado se hacen validaciones, en el caso del botón de pedir carta se baraja el mazo de cartas y se carga una imagen o carta aleatoriamente y se muestra en pantalla según el jugador del turno, además se llama recursivamente la función para guardar esas cartas elegidas en listas para cada jugador y así el programa vuelve a estar disponible para un nuevo evento con el click del mouse. Si se presiona el botón plantarse entonces se verifica si ya todos los jugadores se han plantado y continua el crupier y termina el juego, o se guarda ese plante y continua el juego con el siguiente jugador.

1.4. Problemas Conocidos.

No se logró hacer que se puedan jugar más de un jugador.

1.6. Problemas encontrados:

-Uno de los problemas fue con la GUI ya que primero se utilizó una librería **(require racket/gui)** para crear las ventanas, botones y textfields pero para cargar imágenes no logramos hacerlo con esa librería, entonces se procedió a utilizar otra librería **(require (lib "graphics.ss" "graphics"))** ligada a la anterior ya que había un gran avance utilizándola, y al empezar a usar esa nueva librería costó mucho la creación de botones, lo que llevó a muchas horas de trabajo solo entendiendo cómo hacerlo, hasta que al final se crearon cuadros en forma de botón y con una función para detectar los click del mouse se asemejaba al uso de botones.

-Otro de los problemas es el uso de variables, para poder hacer algo tipo variables se debieron crear funciones para utilizarlas como variables, o también se soluciona usando recursividad.

-El IDE de DrRacket cuando el .rkt contiene muchas líneas de código se pega, a veces simplemente es lo natural del IDE pero otras veces es porque después de correr el programa se cierra con la equis y no se preciona el boton de stop, es necesario siempre presionarlo para que no se pegue tanto.

1.7. Conclusiones y Recomendaciones del proyecto.

-La documentación oficial de DrRacket cuesta entenderla, se deben buscar otras fuentes.

-Para hacer programas más grandes se recomienda buscar otro lenguaje de programación el cual se pueda adaptar mejor a los requerimientos.

-Se logró completar la mayoría de lo pedido en los requerimientos del proyecto.

1.8. Bibliografía consultada en todo el proyecto

Guzman, J. E. (2006). Introducción a la programación con Scheme. Cartago: Editorial Tecnológica de Costa Rica.

Racket. (2017, 08 15). The Racket Graphical Interface Toolkit. Retrieved from Racket: http://download.racket-lang.org/releases/6.9/doc/gui/

https://docs.racket-lang.org/gui/windowing-overview.html#%28part._.Creating_.Windows%29

Ruiz, B. (2017). Compilación de Proyectos Finales de Programacion I en DrRacket. Adaptado de: https://github.com/h3ct0rjs/VideoJuego