Othello

Javier García Jiménez, David Moreno López, Isabel Martínez Gómez
Laboratorio Inteligencia Artificial - Práctica 1
Universidad de Alcalá
Curso 2019-2020

1. Introducción

El trabajo a realizar consiste en la implementación de un código en Racket que resuelva el famoso juego Othello o también conocido como Reversi.
El Othello es un juego de dos personas, las cuales compartirán 64 fichas iguales y a cada uno de ellos se le asignará un color (o blanco o negro).

Este juego consiste por tanto en ir rellenando ficha a ficha (una por cada turno) un tablero de dimensión 8x8. Cuando se coloca una ficha en una posición válida, se voltean todas aquellas fichas que sean de distinto color a la colocada y que estén entre la ficha colocada y otra del mismo color que la misma.

El juego acaba cuando se rellena todo el tablero o cuando ambos jugadores se quedan sin movimientos posibles. El ganador del juego es el jugador con más fichas de su color al final de la partida.

2. Discusión y formulación del problema

Para la resolución del problema se han aportado dos soluciones:

- Para comenzar se ha desarrollado el modo de juego **1 contra 1,** a través del cual dos jugadores se enfrentan entre sí, sin intervención alguna de la máquina.
- El juego **jugador contra IA**: para la implementación de este otro modo de juego a través del cual un jugador se enfrenta a la máquina dotada con inteligencia artificial se ha optado por realizar la búsqueda minimax. Gracias a esta búsqueda, la máquina es capaz de realizar movimientos óptimos para buscar la victoria ante el jugador. Dependiendo de la dificultad que se le añada al juego, la máquina realizará mejores movimientos o movimientos no tan buenos.

Para que el usuario pueda disfrutar de una experiencia completa, se ha desarrollado también una interfaz gráfica que representa el transcurso del juego, y que permite al jugador (o a los jugadores en caso de que haya dos jugando entre sí), poder visualizar el tablero del juego con las correspondientes fichas y poder colocar las fichas haciendo "click" con el ratón en la posición deseada.

3. Método de resolución empleado

En este apartado explicaremos con detalle cada una de las funciones que han sido necesarias para la resolución del juego Othello.

- **Función busquedaMinimax**

```
(define (busquedaMinimax tablero turno profundidad)
  (foldl(lambda (valor maxValor)
     (if (> (car valor) (car maxValor)) valor maxValor)) '(-9999 -1 -1)
        (for/list ([i (crearListaPosiciones (movimientosPosibles
                                             (getCasillasColor 0 0 tablero turno) turno tablero))])
           (list (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
                          (cambiarTurno turno) profundidad) (car i) (cadr i)))))
```

Esta función se encarga de buscar una posición aplicando el método minimax. Para ello, hace uso de la función minimax. La base de esta función radica en el uso del foldl y for/list.
A través del for/list, se genera una lista de listas, cada una de las cuales con tres valores (valor, fila,columna). Estos tres valores se añaden a la lista cada vez que se realiza una iteración del for/list. Con esto, conseguimos conocer, cuales son los valores de realizar cada posible movimiento gracias a la función minimax. Tras esto, en la función foldl se realiza una comparación a través de una función lambda para recorrer toda la lista generada anteriormente y quedarse con el mejor valor de la misma, es decir. con los tres valores (valor, fila ,columna) en los que el valor sea el mayor de la misma.
Con esto, busquedaMinimax devuelve la fila y la columna donde la IA colocará su ficha.

- **Función minimax**

```
(define (minimax tablero turno profundidad)
  (if (or(finJuego tablero) (= profundidad 0)) (diferenciaPuntosMarcador(marcadorJuego tablero))
      (if (equal?(movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero)'())
          (minimax tablero (cambiarTurno turno) profundidad)
      (if (= turno 2)
          (foldl(lambda (valor maxValor)
                  (if (> valor maxValor) valor maxValor)) -9999
                  (for/list ([i (crearListaPosiciones (movimientosPosibles
                                                       (getCasillasColor 0 0 tablero turno) turno tablero))])
                     (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
                              (cambiarTurno turno) (- profundidad 1)) (car i) (cadr i)))
          (foldl(lambda (valor minValor)
                  (if (<  valor minValor) valor minValor)) 9999
                  (for/list ([i (crearListaPosiciones (movimientosPosibles
                                                       (getCasillasColor 0 0 tablero turno) turno tablero))])
                     (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
                              (cambiarTurno turno) (- profundidad 1)) (car i) (cadr i)))))))
```

Esta función emplea el algoritmo minimax para resolver el Othello. Para ello por cada movimiento calcula cual la opción en la que la diferencia del marcador es más favorable para él y realiza el primer movimiento de esa opción.

- **Función dibujarOthello**

```
(define (dibujarOthello n fila columna x y turno tablero)
  (if (= fila 8)
      (void)
      (if (even? n)
          (begin
            ((draw-solid-rectangle ventana) (make-posn x y) 70 70 "Dark green")
            ((draw-line ventana) (make-posn (+ x 70) y) (make-posn x y) "black")
            ((draw-line ventana) (make-posn x (+ y 70)) (make-posn x y) "black")
            (if (equal?(getValorCasilla fila columna tablero) 0) (void)
                (if (equal?(getValorCasilla fila columna tablero) 1)
                    ((draw-solid-ellipse ventana) (make-posn (+ x 10) (+ y 10)) 50 50 "white")
                    (if (equal?(getValorCasilla fila columna tablero) 2)((draw-solid-ellipse ventana)
                        (make-posn (+ x 10) (+ y 10)) 50 50 "black")
                        (if (equal? turno 1)
                            ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "white")
                            ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "black"))))))
          (begin
            ((draw-solid-rectangle ventana) (make-posn x y) 70 70 "Dark green")
            ((draw-line ventana) (make-posn (+ x 70) y) (make-posn x y) "black")
            ((draw-line ventana) (make-posn x (+ y 70)) (make-posn x y) "black")
            (if (equal? (getValorCasilla fila columna tablero) 0) (void)
                (if (equal?(getValorCasilla fila columna tablero) 1)
                    ((draw-solid-ellipse ventana) (make-posn (+ x 10) (+ y 10)) 50 50 "white")
                    (if (equal?(getValorCasilla fila columna tablero) 2)((draw-solid-ellipse ventana)
                        (make-posn (+ x 10) (+ y 10)) 50 50 "black")
                        (if (equal? turno 1)
                            ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "white")
                            ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "black")))))))))
  (if (= columna 7)
      (dibujarOthello n (+ fila 1) 0 0 (+ y 70) turno tablero)
      (if (= fila 8)
          (void)
          (dibujarOthello (+ n 1) fila (+ columna 1) (+ 70 x) y turno tablero))))
```

Esta función se encarga de dibujar el tablero del Othello mediante las funciones draw-solid-rectangle que dibuja los cuadrados del tablero y draw-line que se encarga de dibujar los bordes de cada cuadrado para delimitarlos.
Esta función recibe un estado del tablero y dibuja las fichas que se encuentren en el tablero en ese momento a través de la función draw-solid-ellipse y los posibles movimientos del jugador que se le indican con un círculo más pequeño.

- **Función getFila**

```
(define (getFila fila tablero)
  (list-ref tablero fila))
```

Esta función recibe un número de fila y el tablero devolviendo la fila correspondiente.

- **Función insertarFicha**

```
(define (insertarFicha fila columna valor tablero)
        (list-set tablero fila (list-set (getFila fila tablero) columna valor)))
```

Esta función recibe una posición en la que quieres insertar una ficha en el tablero mediante una fila y columna y un valor que insertar y mediante la función list-set la inserta en el tablero.

- **Función getCasillasColor**

```
(define (getCasillasColor fila columna tablero color)
  (if (> fila 7) '()
      (if ( > columna 7) (append (getCasillasColor (+ fila 1) 0 tablero color))
          (if (= color (getValorCasilla  fila columna tablero))
              (append (list(list fila columna)) (getCasillasColor fila (+ 1 columna) tablero color))
              (append (getCasillasColor fila (+ 1 columna) tablero color))))))
```

Esta función devuelve una lista con las posiciones de las fichas del color especificado en el parámetro color (1-para las blancas, 2-para las negras).

- **Función getPosiciones**

```
(define (getPosiciones fila columna color tablero)
  (append (comprobarEsquinaInferiorDerecha fila columna tablero 0 color)
          (comprobarEsquinaInferiorIzquierda fila columna tablero 0 color)
          (comprobarEsquinaSuperiorDerecha fila columna tablero 0 color)
          (comprobarEsquinaSuperiorIzquierda fila columna tablero 0 color)
          (comprobarFilaIzquierda fila columna tablero 0 color)
          (comprobarFilaDerecha fila columna tablero 0 color)
          (comprobarColumnaAbajo fila columna tablero 0 color )
          (comprobarColumnaArriba fila columna tablero 0 color)))
```

Esta función devuelve una lista con todos los posibles movimientos que puede hacer un jugador en el tablero con color indicado por parámetro en la fila y columna especificadas.

- **Función movimientosPosibles**

```
(define (movimientosPosibles listaFichasColor color tablero)
  (if (equal? listaFichasColor '() ) '()
      (append (getPosiciones (caar listaFichasColor) (cadr(car listaFichasColor)) color tablero)
              (movimientosPosibles (cdr listaFichasColor) color tablero))))
```

Esta función realiza una lista que contiene listas con cada una de las posiciones en las que un jugador puede insertar una ficha.

- **Función comprobarEsquinaInferiorDerecha**

```
(define (comprobarEsquinaInferiorDerecha fila columna tablero contador color)
  (if (or (= columna 7) (= fila 7)) '()
      (if (and (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) 0) (> contador 0))
          (append (list (+ fila 1)  (+ columna 1)))
           (if (and (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) 0) (= contador 0)) '()
               (if (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) color) '()
                   (append (comprobarEsquinaInferiorDerecha (+ fila 1)  (+ columna 1) tablero (+ 1 contador) color)))))))
```

Esta función comprueba si para la casilla especificada mediante fila y columna hay alguna casilla en su diagonal inferior derecha donde pueda meter una ficha el jugador especificado mediante color. Si la encuentra devuelve la posición de esa casilla mediante su fila y columna.

- **Funciones de comprobación**

```
(define (comprobarEsquinaInferiorIzquierda fila columna tablero contador color)
  (if (or (= columna 0) (= fila 7)) '()
        (if (and (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) 0) (> contador 0))
            (append (list (+ fila 1)  (- columna 1)))
             (if (and (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) 0) (= contador 0)) '()
                 (if (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) color) '()
                     (append (comprobarEsquinaInferiorIzquierda (+ fila 1)  (- columna 1) tablero (+ 1 contador) color)))))))

(define (comprobarEsquinaSuperiorDerecha fila columna tablero contador color)
  (if (or (= columna 7) (= fila 0)) '()
        (if (and (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) 0) (> contador 0))
            (append (list (- fila 1)  (+ columna 1)))
             (if (and (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) 0) (= contador 0)) '()
                 (if (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) color) '()
                     (append (comprobarEsquinaSuperiorDerecha (- fila 1)  (+ columna 1) tablero (+ 1 contador) color)))))))

(define (comprobarEsquinaSuperiorIzquierda fila columna tablero contador color)
  (if (or (= columna 0) (= fila 0)) '()
        (if (and (= (getValorCasilla (- fila 1)  (- columna 1) tablero) 0) (> contador 0))
            (append (list (- fila 1)  (- columna 1)))
             (if (and (= (getValorCasilla (- fila 1)  (- columna 1) tablero) 0) (= contador 0)) '()
                 (if (= (getValorCasilla (- fila 1)  (- columna 1) tablero) color) '()
                     (append (comprobarEsquinaSuperiorIzquierda (- fila 1)  (- columna 1) tablero (+ 1 contador) color)))))))

(define (comprobarFilaIzquierda fila columna tablero contador color)
  (if (= columna 0) '()
         (if (and (= (getValorCasilla fila  (- columna 1) tablero) 0) (> contador 0))
             (append (list fila  (- columna 1)))
              (if (and (= (getValorCasilla fila  (- columna 1) tablero) 0) (= contador 0)) '()
                  (if (= (getValorCasilla fila  (- columna 1) tablero) color) '()
                      (append (comprobarFilaIzquierda fila (- columna 1) tablero (+ 1 contador) color)))))))

(define (comprobarFilaDerecha fila columna tablero contador color)
  (if (= columna 7) '()
         (if (and (= (getValorCasilla fila  (+ columna 1) tablero) 0) (> contador 0))
             (append (list fila  (+ columna 1)))
              (if (and (= (getValorCasilla fila  (+ columna 1) tablero) 0) (= contador 0)) '()
                  (if (= (getValorCasilla fila  (+ columna 1) tablero) color) '()
                      (append (comprobarFilaDerecha fila (+ columna 1) tablero (+ 1 contador) color)))))))

(define (comprobarColumnaAbajo fila columna tablero contador color )
 (if (= fila 7) '()
         (if (and (= (getValorCasilla (+ fila 1) columna  tablero) 0) (> contador 0))
             (append (list (+ fila 1) columna))
              (if (and (= (getValorCasilla (+ fila 1) columna tablero) 0) (= contador 0)) '()
                  (if (= (getValorCasilla (+ fila 1) columna tablero) color) '()
                      (append (comprobarColumnaAbajo (+ fila 1) columna tablero (+ 1 contador) color)))))))

(define (comprobarColumnaArriba fila columna tablero contador color)
  (if (= fila 0) '()
         (if (and (= (getValorCasilla (- fila 1) columna  tablero) 0) (> contador 0))
             (append (list (- fila 1) columna))
              (if (and (= (getValorCasilla (- fila 1) columna tablero) 0) (= contador 0)) '()
                  (if (= (getValorCasilla (- fila 1) columna tablero) color) '()
                      (append (comprobarColumnaArriba (- fila 1) columna tablero (+ 1 contador) color)))))))
```

Estas funciones son igual que la anterior pero cada una de ellas con sus propias restricciones y dirección. Estas comprueban si hay una casilla en esa dirección donde se pueda insertar una ficha, siendo estas direcciones: columna arriba, columna abajo, fila izquierda, fila derecha, diagonal arriba izquierda, diagonal arriba derecha, diagonal abajo izquierda y diagonal abajo derecha.

- **Función actualizarTablero**

```
(define (actualizarTablero listaPosibles tablero)
  (if (equal? listaPosibles '()) tablero
      (actualizarTablero (cddr listaPosibles) (insertarFicha (car listaPosibles) (cadr listaPosibles) "X" tablero))))
```

Esta función inserta una "X" en los posibles movimientos que se pueden hacer con el tablero actual.

- **Función columnaArribaVolteable**

```
(define (columnaArribaVolteable fila columna tablero contador color)
  (if (<= fila 0) #f
      (if (= (getValorCasilla (- fila 1) columna tablero) 0) #f
          (if (and (= (getValorCasilla (- fila 1) columna tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (- fila 1) columna tablero) color) (> contador 0)) #t
                  (columnaArribaVolteable (- fila 1) columna tablero (+ 1 contador) color))))))
```

Esta función comprueba si en la columna de arriba de la casilla especificada en los parámetros fila y columna hay alguna ficha del contrincante que se pueda voltear.

- **Funciones que comprueban si pueden voltear fichas**

```
(define (columnaAbajoVolteable fila columna tablero contador color)
  (if (or (= fila 7) (< fila 0) (< columna 0)) #f
      (if (= (getValorCasilla (+ 1 fila) columna tablero) 0) #f
           (if (and (= (getValorCasilla (+ 1 fila) columna tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (+ 1 fila) columna tablero) color) (> contador 0)) #t
                   (columnaAbajoVolteable (+ 1 fila) columna tablero (+ 1 contador) color))))))

(define (filaDerechaVolteable fila columna tablero contador color)
  (if ( or (= columna 7)(< columna 0) (< fila 0)) #f
      (if (= (getValorCasilla fila (+ columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla fila (+ columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla fila (+ columna 1) tablero) color) (> contador 0)) #t
                  (filaDerechaVolteable fila (+ columna 1) tablero (+ 1 contador) color))))))

(define (filaIzquierdaVolteable fila columna tablero contador color)
  (if (= columna 0) #f
      (if (= (getValorCasilla fila (- columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla fila (- columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla fila (- columna 1) tablero) color) (> contador 0)) #t
                  (filaIzquierdaVolteable fila (- columna 1) tablero (+ 1 contador) color))))))

(define (diagonalArribaIzquierdaVolteable fila columna tablero contador color)
  (if ( or (= fila 0)(= columna 0)) #f
      (if (= (getValorCasilla (- fila 1) (- columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla (- fila 1) (- columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (- fila 1) (- columna 1) tablero) color) (> contador 0)) #t
                  (diagonalArribaIzquierdaVolteable (- fila 1) (- columna 1) tablero (+ 1 contador) color))))))

(define (diagonalArribaDerechaVolteable fila columna tablero contador color)
  (if ( or (= fila 0)(= columna 7)) #f
      (if (= (getValorCasilla (- fila 1) (+ columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color) (> contador 0)) #t
                  (diagonalArribaDerechaVolteable (- fila 1) (+ columna 1) tablero (+ 1 contador) color))))))

(define (diagonalAbajoDerechaVolteable fila columna tablero contador color)
  (if ( or (= fila 7)(= columna 7)) #f
      (if (= (getValorCasilla (+ fila 1) (+ columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color) (> contador 0)) #t
                  (diagonalAbajoDerechaVolteable (+ fila 1) (+ columna 1) tablero (+ 1 contador) color))))))

(define (diagonalAbajoIzquierdaVolteable fila columna tablero contador color)
  (if ( or (= fila 7)(= columna 0)) #f
      (if (= (getValorCasilla (+ fila 1) (- columna 1)  tablero) 0) #f
          (if (and (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color) (= contador 0)) #f
              (if (and (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color) (> contador 0)) #t
                  (diagonalAbajoIzquierdaVolteable (+ fila 1) (- columna 1) tablero (+ 1 contador) color))))))
```

Estas funciones son iguales que la anterior pero cada una comprueba en una de las 8 direcciones posibles que tiene una casilla.

- **Función voltearColumnaAbajo**

```
(define (voltearColumnaAbajo fila columna tablero color)
  (if (not (= (getValorCasilla (+ 1 fila) columna tablero) color))
      (voltearColumnaAbajo (+ 1 fila) columna (insertarFicha (+ 1 fila) columna color tablero) color)
      tablero))
```

Esta función voltea las fichas del color del contrario que se encuentren entre la ficha que acaba de poner y la ficha del jugador en la columna debajo de esta si es que la hay.

- **Funciones de voltear**

```
(define (voltearColumnaArriba fila columna tablero color)
  (if (not (= (getValorCasilla (- fila 1) columna tablero) color))
      (voltearColumnaArriba (- fila 1) columna (insertarFicha (- fila 1) columna color tablero) color)
      tablero))

(define (voltearFilaDerecha fila columna tablero color)
  (if (not (= (getValorCasilla fila ( + columna 1) tablero) color))
      (voltearFilaDerecha fila ( + columna 1) (insertarFicha fila ( + columna 1) color tablero) color)
      tablero))

(define (voltearFilaIzquierda fila columna tablero color)
  (if (not (= (getValorCasilla fila (- columna 1) tablero) color))
      (voltearFilaIzquierda fila (- columna 1) (insertarFicha fila (- columna 1) color tablero) color)
      tablero))

(define (voltearDiagonalArribaIzquierda fila columna tablero color)
  (if (not (= (getValorCasilla (- fila 1) (- columna 1) tablero) color))
      (voltearDiagonalArribaIzquierda  (- fila 1) (- columna 1) (insertarFicha (- fila 1) (- columna 1) color tablero) color)
      tablero))

(define (voltearDiagonalArribaDerecha fila columna tablero color)
  (if (not (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color))
      (voltearDiagonalArribaDerecha  (- fila 1) (+ columna 1) (insertarFicha (- fila 1) (+ columna 1) color tablero) color)
      tablero))

(define (voltearDiagonalAbajoDerecha fila columna tablero color)
  (if (not (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color))
      (voltearDiagonalAbajoDerecha  (+ fila 1) (+ columna 1) (insertarFicha (+ fila 1) (+ columna 1) color tablero) color)
      tablero))

(define (voltearDiagonalAbajoIzquierda fila columna tablero color)
  (if (not (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color))
      (voltearDiagonalAbajoIzquierda  (+ fila 1) (- columna 1) (insertarFicha (+ fila 1) (- columna 1) color tablero) color)
      tablero))
```

Estas funciones son iguales que la anterior pero cada una las voltea en una de las 8 direcciones posibles que tiene una casilla.

- **Función movimientoValido**

```
(define (movimientoValido fila columna listaPosibles)
  (if (equal? listaPosibles '()) #f
      (if (equal? (list fila columna) (list (car listaPosibles) (cadr listaPosibles))) #t
          (movimientoValido fila columna (cddr listaPosibles)))))
```

Esta función comprueba si el movimiento especificado se encuentra dentro de la lista de los movimientos posibles del jugador.

- **Función contarBlancas**

```
(define (contarBlancas tablero fila columna contador)
  (if (> fila 7) contador
      (if (> columna 7) (contarBlancas tablero (+ 1 fila) 0 contador)
          (if (= (getValorCasilla fila columna tablero) 1) (contarBlancas tablero fila (+ 1 columna) (+ 1 contador))
              (contarBlancas tablero fila (+ 1 columna) contador)))))
```

Función que cuenta el número de fichas blancas que hay en el tablero.

- **Función contarNegras**

```
(define (contarNegras tablero fila columna contador)
  (if (> fila 7) contador
      (if (> columna 7) (contarNegras tablero (+ 1 fila) 0 contador)
          (if (= (getValorCasilla fila columna tablero) 2) (contarNegras tablero fila (+ 1 columna) (+ 1 contador))
              (contarNegras tablero fila (+ 1 columna) contador)))))
```

Función que cuenta el número de fichas negras que hay en el tablero.

- **Función marcadorJuego**

```
(define (marcadorJuego tablero)
  (define blancas (contarBlancas tablero 0 0 0))
  (define negras (contarNegras tablero 0 0 0))
  (list blancas negras))
```

Esta función devuelve una lista con el marcador del juego.

- **Función imprimirMarcador**

```
(define (imprimirMarcador lista)
  (display "\nBlancas ") (display (car lista))
  (display " - ")
  (display "Negras ") (display (cadr lista)) (display "\n"))
```

Esta función imprime el marcador por pantalla.

- **Función diferenciaPuntosMarcador**

```
(define (diferenciaPuntosMarcador marcador)
  (- (cadr marcador) (car marcador)))
```

Esta función devuelve la diferencia de puntos entre los dos jugadores.

- **Función realizarMovimiento**

```
(define (realizarMovimiento fila columna tablero color)
  (when (equal? (columnaArribaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearColumnaArriba fila columna (insertarFicha fila columna color tablero) color)))
  (when (equal? (columnaAbajoVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearColumnaAbajo fila columna (insertarFicha fila columna color tablero) color)) )
  (when (equal? (filaDerechaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearFilaDerecha fila columna (insertarFicha fila columna color tablero) color)) )
  (when (equal? (filaIzquierdaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearFilaIzquierda fila columna (insertarFicha fila columna color tablero) color)) )
  (when (equal? (diagonalArribaIzquierdaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearDiagonalArribaIzquierda fila columna (insertarFicha fila columna color tablero) color)) )
  (when (equal? (diagonalArribaDerechaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearDiagonalArribaDerecha fila columna (insertarFicha fila columna color tablero) color)))
  (when (equal? (diagonalAbajoDerechaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearDiagonalAbajoDerecha fila columna (insertarFicha fila columna color tablero) color)) )
  (when (equal? (diagonalAbajoIzquierdaVolteable fila columna tablero 0 color) #t)
    (set! tablero (voltearDiagonalAbajoIzquierda fila columna (insertarFicha fila columna color tablero) color)) )
  tablero)
```

Función que voltea las fichas necesarias al hacer un movimiento llamando a todas las funciones explicadas anteriormente que volteaban las fichas siempre que pudiesen.

- **Función elegirMovimiento**

```
(define (elegirMovimiento tablero color)
  (define click (mouse-click-posn (get-mouse-click ventana)))
  (define fila (floor (/(posn-y click )70)))
  (define columna (floor (/ (posn-x  click) 70)))
  (if (movimientoValido fila columna (movimientosPosibles (getCasillasColor 0 0 tablero color) color tablero))
      (realizarMovimiento fila columna tablero color)
   (elegirMovimiento tablero color)))
```

Esta función recoge la posición de la casilla que el jugador ha pulsado, comprueba si es un movimiento válido mediante la función movimientoValido y si es válido realiza el movimiento llamando a la función realizarMovimiento.

- **Función cambiarTurno**

```
(define (cambiarTurno turno)
  (if (= turno 1)  2  1))
```

Esta función cambia el turno de un jugador al otro.

- **Función puedeJugar**

```
(define (puedeJugar tablero turno)
  (if (not (equal? (movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero) '())) #t #f))
```

Esta función comprueba si un jugador tiene movimientos posibles que realizar o no.

- **Función finJuego**

```
(define (finJuego tablero)
  (if (and (equal? (movimientosPosibles (getCasillasColor 0 0 tablero 1) 1 tablero) '())
           (equal? (movimientosPosibles (getCasillasColor 0 0 tablero 2) 2 tablero) '())) #t #f))
```

Esta función comprueba si hay movimientos posibles para alguno de los dos jugadores, si ninguno tiene movimientos posibles significa que se ha acabado el juego.

- **Función acabarJuego**

```
(define (acabarJuego tablero)
  (display "Se ha terminado el juego\n")
  (define marcador (marcadorJuego tablero))
  (imprimirMarcador marcador)
  marcador)
```

Esta función imprime el marcador al final del juego.

- **Función crearListaPosiciones**

```
(define (crearListaPosiciones listaPosibles)
  (if (equal? listaPosibles '()) '()
      (append (list(list (car listaPosibles) (cadr listaPosibles)))
              (crearListaPosiciones (cddr listaPosibles)))))
```

Esta función crea un lista en la que cada elemento de la lista es otra lista cuyos elementos son fila y columna de los posibles movimientos del jugador.

- **Función unoContraUno**

```
(define (unoContraUno tablero turno marcador)
  (dibujarOthello 0 0 0 0 0 turno
                  (actualizarTablero(movimientosPosibles(getCasillasColor 0 0 tablero turno) turno tablero) tablero))
  (display "Turno del jugador: ") (display turno)
  (display "\nB: ")(display (car marcador)) (display " - N:") (display ( cadr marcador)) (display "\n")
  (if (finJuego tablero) (acabarJuego tablero)
  (if (puedeJugar tablero turno)
       (unoContraUno  (elegirMovimiento tablero turno) (cambiarTurno turno) (marcadorJuego tablero))
      (unoContraUno tablero (cambiarTurno turno) marcador))))
```

Esta función se llama cuando juegan dos personas. Imprime cada movimiento del jugador si puede realizarlo, cambia el turno una vez un jugador haya realizado un movimiento e imprime el marcador. En cada paso comprueba si el juego se ha acabado para llamar a la función acabarJuego.

- **Función unoContraIAMinimax**

```
(define (unoContraIAMinimax tablero turno marcador profundidad)
  (dibujarOthello 0 0 0 0 0 turno
                  (actualizarTablero(movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero) tablero))
  (set! marcador (marcadorJuego tablero))
  (imprimirTablero 0 0 tablero)
  (imprimirMarcador marcador)
  (display "Turno del jugador ") (display turno) (display "\n")
  (if (equal? (finJuego tablero) #t) (acabarJuego tablero)
  (if(= turno 1)
        (if (puedeJugar tablero turno)
             (unoContraIAMinimax (elegirMovimiento tablero turno) (cambiarTurno turno) marcador profundidad)
             (unoContraIAMinimax tablero (cambiarTurno turno) marcador profundidad))
        (if (puedeJugar tablero turno)
    (let ([ia (busquedaMinimax tablero turno profundidad)])
    (unoContraIAMinimax (realizarMovimiento (cadr ia) (car (cddr ia)) tablero turno)
                        (cambiarTurno turno) marcador profundidad))
      (unoContraIAMinimax tablero (cambiarTurno turno) marcador profundidad)))))
```

Esta función se llama cuando juega una persona contra la IA. Va alternando entre el turno de la IA y el del jugador, imprime el tablero y actualiza e imprime el marcador en cada uno de ellos. En el turno del jugador si este puede realizar un movimiento espera hasta que lo realice y en el turno de la IA realiza el mejor movimiento para la situación del tablero actual calculado mediante el algoritmo Minimax.

- **Función dibujarMenu**

```
(define (dibujarMenu x1 y1 x2 y2 dificultad)
  (set! menu(open-viewport "menuPrincipal" 1002 666))
(((draw-pixmap-posn "fondo.jpg")menu) (make-posn 0 0))
  ((draw-ellipse menu) (make-posn x1 y1) 230 230 "Black")
  ((draw-ellipse menu) (make-posn x2 y2) 230 230 "Black")
  (define click (mouse-click-posn (get-mouse-click menu)))
  (when(and (> (floor (posn-x click) ) 24)
            (< (floor (posn-x click) ) 254)  (> (floor (posn-y click) ) 370 )
            (< (floor( posn-y click) ) 605) ) (juegoUnoVsIA dificultad) #f)
  (when(and (> (floor (posn-x click) ) 710)
            (< (floor (posn-x click) ) 964)  (> (floor (posn-y click) ) 370 )
            (< (floor( posn-y click) ) 605) ) (juegoUnoVsUno) #f))
```

Esta función muestra una pantalla que actúa de menú en la que hay dos botones con los que puedes elegir el modo de juego (uno vs uno o uno vs IA).

- **Función juegoUnoVsUno**

```
(define (juegoUnoVsUno)
  (set! ventana(open-viewport "Othello" 560 560))
  (close-viewport menu)
  (unoContraUno partida1 1 (list 0 0)))
```

Esta función es llamada después de pulsar el botón del menú uno vs uno. Cierra el menú principal y abre otra ventana con el modo de juego unos vs uno activado.

- **Función juegoUnoVsIA**

```
(define (juegoUnoVsIA dificultad)
  (close-viewport menu)
  (set! ventana(open-viewport "Othello" 560 560))
  (unoContraIAMinimax partida1 1 (list 0 0) dificultad))
```

Esta función es llamada después de pulsar el botón del menú uno vs IA. Cierra el menú principal y abre otra ventana con el modo de juego unos vs IA activado.

- **Función iniciarJuego**

```
(define (iniciarJuego)
  (display "Introduzca dificultad(1-5): ")
(define dificultad (read))
  (if (and (< dificultad 5) (> dificultad 0))(dibujarMenu 24 370 710 370 dificultad)
      (iniciarJuego)))
(iniciarJuego)
```

Esta función pide que introduzcas la dificultad por pantalla y a la función que abre el menú principal.

4. Código Racket

Archivo Othello.rkt

```
#lang racket
#|=========================================================================
 Práctica 2 - Othello
 Inteligencia Artificial
 Curso 2019/20
 Universidad de Alcalá
 Javier García Jiménez, David Moreno López, Isabel Martínez Gómez
 =========================================================================|#
(provide (all-defined-out))
(require (lib "graphics.ss" "graphics"))
(open-graphics)

(define ventana '())
(define partida1
 '((0 0 0 0 0 0 0 0)
   (0 0 0 0 0 0 0 0)
```

```
  (0 0 0 0 0 0 0 0)
  (0 0 0 1 2 0 0 0)
  (0 0 0 2 1 0 0 0)
  (0 0 0 0 0 0 0 0)
  (0 0 0 0 0 0 0 0)
  (0 0 0 0 0 0 0 0)))

(define (getValorCasilla fila columna tablero)
 (list-ref (list-ref tablero fila) columna))

;Dibuja el tablero con las fichas
(define (dibujarOthello n fila columna x y turno tablero)
 (if (= fila 8)
    (void)
    (if (even? n)
      (begin
       ((draw-solid-rectangle ventana) (make-posn x y) 70 70 "Dark green")
       ((draw-line ventana) (make-posn (+ x 70) y) (make-posn x y) "black")
       ((draw-line ventana) (make-posn x (+ y 70)) (make-posn x y) "black")
       (if (equal?(getValorCasilla fila columna tablero) 0) (void)
         (if (equal?(getValorCasilla fila columna tablero) 1)
           ((draw-solid-ellipse ventana) (make-posn (+ x 10) (+ y 10)) 50 50 "white")
           (if (equal?(getValorCasilla fila columna tablero) 2)((draw-solid-ellipse ventana)
             (make-posn (+ x 10) (+ y 10)) 50 50 "black")
             (if (equal? turno 1)
               ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "white")
               ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "black")))))))
      (begin
       ((draw-solid-rectangle ventana) (make-posn x y) 70 70 "Dark green")
       ((draw-line ventana) (make-posn (+ x 70) y) (make-posn x y) "black")
       ((draw-line ventana) (make-posn x (+ y 70)) (make-posn x y) "black")
       (if (equal? (getValorCasilla fila columna tablero) 0) (void)
         (if (equal?(getValorCasilla fila columna tablero) 1)
           ((draw-solid-ellipse ventana) (make-posn (+ x 10) (+ y 10)) 50 50 "white")
           (if (equal?(getValorCasilla fila columna tablero) 2)((draw-solid-ellipse ventana)
             (make-posn (+ x 10) (+ y 10)) 50 50 "black")
             (if (equal? turno 1)
               ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "white")
               ((draw-solid-ellipse ventana) (make-posn (+ x 33) (+ y 33)) 14 14 "black")))))))))
 (if (= columna 7)
    (dibujarOthello n (+ fila 1) 0 0 (+ y 70) turno tablero)
    (if (= fila 8)
      (void)
      (dibujarOthello (+ n 1) fila (+ columna 1) (+ 70 x) y turno tablero))))


(define (getFila fila tablero)
 (list-ref tablero fila))

(define (insertarFicha fila columna valor tablero)
     (list-set tablero fila (list-set (getFila fila tablero) columna valor)))
```

```
(define (getCasillasColor fila columna tablero color)
 (if (> fila 7) '()
   (if ( > columna 7) (append (getCasillasColor (+ fila 1) 0 tablero color))
     (if (= color (getValorCasilla  fila columna tablero))
       (append (list(list fila columna)) (getCasillasColor fila (+ 1 columna) tablero color))
       (append (getCasillasColor fila (+ 1 columna) tablero color))))))

(define (getPosiciones fila columna color tablero)
 (append (comprobarEsquinaInferiorDerecha fila columna tablero 0 color)
      (comprobarEsquinaInferiorIzquierda fila columna tablero 0 color)
      (comprobarEsquinaSuperiorDerecha fila columna tablero 0 color)
      (comprobarEsquinaSuperiorIzquierda fila columna tablero 0 color)
      (comprobarFilaIzquierda fila columna tablero 0 color)
      (comprobarFilaDerecha fila columna tablero 0 color)
      (comprobarColumnaAbajo fila columna tablero 0 color )
      (comprobarColumnaArriba fila columna tablero 0 color)))

;Imprime cuales son los movimientos posibles para un jugador
(define (movimientosPosibles listaFichasColor color tablero)
 (if (equal? listaFichasColor '() ) '()
   (append (getPosiciones (caar listaFichasColor) (cadr(car listaFichasColor)) color tablero)
       (movimientosPosibles (cdr listaFichasColor) color tablero))))

;Imprime el formato del tablero
(define (imprimirTablero fila columna tablero)
 (when (> columna 7) (printf "\n"))
 (when (and (= columna 0)(= fila 0)) (display "\n  0 1 2 3 4 5 6 7\n"))
 (when (and (= columna 0) (<= fila 7)) (display fila)(display " "))
 (when (and (<= fila 7) (<= columna 7))
 (display (getValorCasilla fila columna tablero)) (display " "))
 (if (> fila 7) (void)
   (if ( > columna  7)(imprimirTablero (+ 1 fila) 0 tablero)
     (imprimirTablero fila (+ 1 columna) tablero))) )


;#-------------------------------------------------------------------------------#
(define (comprobarEsquinaInferiorDerecha fila columna tablero contador color)
 (if (or (= columna 7) (= fila 7)) '()
     (if (and (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) 0) (> contador 0))
       (append (list (+ fila 1)  (+ columna 1)))
        (if (and (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) 0) (= contador 0)) '()
          (if (= (getValorCasilla (+ fila 1)  (+ columna 1) tablero) color) '()
            (append (comprobarEsquinaInferiorDerecha (+ fila 1)  (+ columna 1) tablero (+ 1 contador)
color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarEsquinaInferiorIzquierda fila columna tablero contador color)
 (if (or (= columna 0) (= fila 7)) '()
     (if (and (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) 0) (> contador 0))
       (append (list (+ fila 1)  (- columna 1)))
        (if (and (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) 0) (= contador 0)) '()
```

```
            (if (= (getValorCasilla (+ fila 1)  (- columna 1) tablero) color) '()
               (append (comprobarEsquinaInferiorIzquierda (+ fila 1)  (- columna 1) tablero (+ 1 contador)
color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarEsquinaSuperiorDerecha fila columna tablero contador color)
  (if (or (= columna 7) (= fila 0)) '()
      (if (and (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) 0) (> contador 0))
         (append (list (- fila 1)  (+ columna 1)))
          (if (and (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) 0) (= contador 0)) '()
            (if (= (getValorCasilla (- fila 1)  (+ columna 1) tablero) color) '()
               (append (comprobarEsquinaSuperiorDerecha (- fila 1)  (+ columna 1) tablero (+ 1 contador)
color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarEsquinaSuperiorIzquierda fila columna tablero contador color)
  (if (or (= columna 0) (= fila 0)) '()
      (if (and (= (getValorCasilla (- fila 1)  (- columna 1) tablero) 0) (> contador 0))
         (append (list (- fila 1)  (- columna 1)))
          (if (and (= (getValorCasilla (- fila 1)  (- columna 1) tablero) 0) (= contador 0)) '()
            (if (= (getValorCasilla (- fila 1)  (- columna 1) tablero) color) '()
               (append (comprobarEsquinaSuperiorIzquierda (- fila 1)  (- columna 1) tablero (+ 1 contador)
color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarFilaIzquierda fila columna tablero contador color)
  (if (= columna 0) '()
      (if (and (= (getValorCasilla fila  (- columna 1) tablero) 0) (> contador 0))
         (append (list fila  (- columna 1)))
          (if (and (= (getValorCasilla fila  (- columna 1) tablero) 0) (= contador 0)) '()
            (if (= (getValorCasilla fila  (- columna 1) tablero) color) '()
               (append (comprobarFilaIzquierda fila (- columna 1) tablero (+ 1 contador) color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarFilaDerecha fila columna tablero contador color)
  (if (= columna 7) '()
      (if (and (= (getValorCasilla fila  (+ columna 1) tablero) 0) (> contador 0))
         (append (list fila  (+ columna 1)))
          (if (and (= (getValorCasilla fila  (+ columna 1) tablero) 0) (= contador 0)) '()
            (if (= (getValorCasilla fila  (+ columna 1) tablero) color) '()
               (append (comprobarFilaDerecha fila (+ columna 1) tablero (+ 1 contador) color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarColumnaAbajo fila columna tablero contador color )
 (if (= fila 7) '()
      (if (and (= (getValorCasilla (+ fila 1) columna  tablero) 0) (> contador 0))
         (append (list (+ fila 1) columna))
          (if (and (= (getValorCasilla (+ fila 1) columna tablero) 0) (= contador 0)) '()
            (if (= (getValorCasilla (+ fila 1) columna tablero) color) '()
               (append (comprobarColumnaAbajo (+ fila 1) columna tablero (+ 1 contador) color)))))))
;#-------------------------------------------------------------------------------#
(define (comprobarColumnaArriba fila columna tablero contador color)
  (if (= fila 0) '()
      (if (and (= (getValorCasilla (- fila 1) columna  tablero) 0) (> contador 0))
         (append (list (- fila 1) columna))
          (if (and (= (getValorCasilla (- fila 1) columna tablero) 0) (= contador 0)) '()
```

```
          (if (= (getValorCasilla (- fila 1) columna tablero) color) '()
            (append (comprobarColumnaArriba (- fila 1) columna tablero (+ 1 contador) color)))))))

;Actualiza el tablero mostrando las posiciones con una X donde un jugador puede insertar una ficha
(define (actualizarTablero listaPosibles tablero)
 (if (equal? listaPosibles '()) tablero
   (actualizarTablero (cddr listaPosibles) (insertarFicha (car listaPosibles) (cadr listaPosibles) "X"
tablero))))

;Imprime el tablero con los posibles movimientos para el jugador
(define (imprimirPosiblesMovimientos tablero turno)
 (imprimirTablero 0 0 (actualizarTablero(movimientosPosibles
                        (getCasillasColor 0 0 tablero turno) turno tablero) tablero)))


(define (columnaArribaVolteable fila columna tablero contador color)
 (if (<= fila 0) #f
   (if (= (getValorCasilla (- fila 1) columna tablero) 0) #f
     (if (and (= (getValorCasilla (- fila 1) columna tablero) color) (= contador 0)) #f
       (if (and (= (getValorCasilla (- fila 1) columna tablero) color) (> contador 0)) #t
         (columnaArribaVolteable (- fila 1) columna tablero (+ 1 contador) color))))))

(define (columnaAbajoVolteable fila columna tablero contador color)
 (if (or (= fila 7) (< fila 0) (< columna 0)) #f
   (if (= (getValorCasilla (+ 1 fila) columna tablero) 0) #f
     (if (and (= (getValorCasilla (+ 1 fila) columna tablero) color) (= contador 0)) #f
       (if (and (= (getValorCasilla (+ 1 fila) columna tablero) color) (> contador 0)) #t
         (columnaAbajoVolteable (+ 1 fila) columna tablero (+ 1 contador) color))))))

(define (filaDerechaVolteable fila columna tablero contador color)
 (if ( or (= columna 7)(< columna 0) (< fila 0)) #f
   (if (= (getValorCasilla fila (+ columna 1)  tablero) 0) #f
     (if (and (= (getValorCasilla fila (+ columna 1) tablero) color) (= contador 0)) #f
       (if (and (= (getValorCasilla fila (+ columna 1) tablero) color) (> contador 0)) #t
         (filaDerechaVolteable fila (+ columna 1) tablero (+ 1 contador) color))))))

(define (filaIzquierdaVolteable fila columna tablero contador color)
 (if (= columna 0) #f
   (if (= (getValorCasilla fila (- columna 1)  tablero) 0) #f
     (if (and (= (getValorCasilla fila (- columna 1) tablero) color) (= contador 0)) #f
       (if (and (= (getValorCasilla fila (- columna 1) tablero) color) (> contador 0)) #t
         (filaIzquierdaVolteable fila (- columna 1) tablero (+ 1 contador) color))))))

(define (diagonalArribaIzquierdaVolteable fila columna tablero contador color)
 (if ( or (= fila 0)(= columna 0)) #f
   (if (= (getValorCasilla (- fila 1) (- columna 1)  tablero) 0) #f
     (if (and (= (getValorCasilla (- fila 1) (- columna 1) tablero) color) (= contador 0)) #f
       (if (and (= (getValorCasilla (- fila 1) (- columna 1) tablero) color) (> contador 0)) #t
         (diagonalArribaIzquierdaVolteable (- fila 1) (- columna 1) tablero (+ 1 contador) color))))))

(define (diagonalArribaDerechaVolteable fila columna tablero contador color)
```

```
 (if ( or (= fila 0)(= columna 7)) #f
    (if (= (getValorCasilla (- fila 1) (+ columna 1)  tablero) 0) #f
       (if (and (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color) (= contador 0)) #f
          (if (and (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color) (> contador 0)) #t
             (diagonalArribaDerechaVolteable (- fila 1) (+ columna 1) tablero (+ 1 contador) color))))))

(define (diagonalAbajoDerechaVolteable fila columna tablero contador color)
 (if ( or (= fila 7)(= columna 7)) #f
    (if (= (getValorCasilla (+ fila 1) (+ columna 1)  tablero) 0) #f
       (if (and (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color) (= contador 0)) #f
          (if (and (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color) (> contador 0)) #t
             (diagonalAbajoDerechaVolteable (+ fila 1) (+ columna 1) tablero (+ 1 contador) color))))))

(define (diagonalAbajoIzquierdaVolteable fila columna tablero contador color)
 (if ( or (= fila 7)(= columna 0)) #f
    (if (= (getValorCasilla (+ fila 1) (- columna 1)  tablero) 0) #f
       (if (and (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color) (= contador 0)) #f
          (if (and (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color) (> contador 0)) #t
             (diagonalAbajoIzquierdaVolteable (+ fila 1) (- columna 1) tablero (+ 1 contador) color))))))

(define (voltearColumnaAbajo fila columna tablero color)
 (if (not (= (getValorCasilla (+ 1 fila) columna tablero) color))
    (voltearColumnaAbajo (+ 1 fila) columna (insertarFicha (+ 1 fila) columna color tablero) color)
    tablero))

(define (voltearColumnaArriba fila columna tablero color)
 (if (not (= (getValorCasilla (- fila 1) columna tablero) color))
    (voltearColumnaArriba (- fila 1) columna (insertarFicha (- fila 1) columna color tablero) color)
    tablero))

(define (voltearFilaDerecha fila columna tablero color)
 (if (not (= (getValorCasilla fila ( + columna 1) tablero) color))
    (voltearFilaDerecha fila ( + columna 1) (insertarFicha fila ( + columna 1) color tablero) color)
    tablero))

(define (voltearFilaIzquierda fila columna tablero color)
 (if (not (= (getValorCasilla fila (- columna 1) tablero) color))
    (voltearFilaIzquierda fila (- columna 1) (insertarFicha fila (- columna 1) color tablero) color)
    tablero))

(define (voltearDiagonalArribaIzquierda fila columna tablero color)
 (if (not (= (getValorCasilla (- fila 1) (- columna 1) tablero) color))
    (voltearDiagonalArribaIzquierda  (- fila 1) (- columna 1) (insertarFicha (- fila 1) (- columna 1) color
tablero) color)
    tablero))

(define (voltearDiagonalArribaDerecha fila columna tablero color)
 (if (not (= (getValorCasilla (- fila 1) (+ columna 1) tablero) color))
    (voltearDiagonalArribaDerecha  (- fila 1) (+ columna 1) (insertarFicha (- fila 1) (+ columna 1) color
tablero) color)
    tablero))
```

```
(define (voltearDiagonalAbajoDerecha fila columna tablero color)
  (if (not (= (getValorCasilla (+ fila 1) (+ columna 1) tablero) color))
    (voltearDiagonalAbajoDerecha  (+ fila 1) (+ columna 1) (insertarFicha (+ fila 1) (+ columna 1) color
tablero) color)
    tablero))

(define (voltearDiagonalAbajoIzquierda fila columna tablero color)
  (if (not (= (getValorCasilla (+ fila 1) (- columna 1) tablero) color))
    (voltearDiagonalAbajoIzquierda  (+ fila 1) (- columna 1) (insertarFicha (+ fila 1) (- columna 1) color
tablero) color)
    tablero))

(define (movimientoValido fila columna listaPosibles)
  (if (equal? listaPosibles '()) #f
    (if (equal? (list fila columna) (list (car listaPosibles) (cadr listaPosibles))) #t
      (movimientoValido fila columna (cddr listaPosibles)))))

;Cuenta el numero de fichas blancas que hay en el tablero
(define (contarBlancas tablero fila columna contador)
  (if (> fila 7) contador
    (if (> columna 7) (contarBlancas tablero (+ 1 fila) 0 contador)
      (if (= (getValorCasilla fila columna tablero) 1) (contarBlancas tablero fila (+ 1 columna) (+ 1
contador))
        (contarBlancas tablero fila (+ 1 columna) contador)))))

;Cuenta el numero de fichas negras que hay en el tablero
(define (contarNegras tablero fila columna contador)
  (if (> fila 7) contador
    (if (> columna 7) (contarNegras tablero (+ 1 fila) 0 contador)
      (if (= (getValorCasilla fila columna tablero) 2) (contarNegras tablero fila (+ 1 columna) (+ 1
contador))
        (contarNegras tablero fila (+ 1 columna) contador)))))

;Crea una lista con el marcador
(define (marcadorJuego tablero)
  (define blancas (contarBlancas tablero 0 0 0))
  (define negras (contarNegras tablero 0 0 0))
  (list blancas negras))

;Imprime el marcador
(define (imprimirMarcador lista)
  (display "\nBlancas ") (display (car lista)) (display " - ") (display "Negras ") (display (cadr lista))
(display "\n"))

;Diferencia de puntos en el marcador
(define (diferenciaPuntosMarcador marcador)
  (- (cadr marcador) (car marcador)))


(define (realizarMovimiento fila columna tablero color)
```

```
 (when (equal? (columnaArribaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearColumnaArriba fila columna (insertarFicha fila columna color tablero)
color)))
 (when (equal? (columnaAbajoVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearColumnaAbajo fila columna (insertarFicha fila columna color tablero) color)) )
 (when (equal? (filaDerechaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearFilaDerecha fila columna (insertarFicha fila columna color tablero) color)) )
 (when (equal? (filaIzquierdaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearFilaIzquierda fila columna (insertarFicha fila columna color tablero) color)) )
 (when (equal? (diagonalArribaIzquierdaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearDiagonalArribaIzquierda fila columna (insertarFicha fila columna color
tablero) color)) )
 (when (equal? (diagonalArribaDerechaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearDiagonalArribaDerecha fila columna (insertarFicha fila columna color
tablero) color)))
 (when (equal? (diagonalAbajoDerechaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearDiagonalAbajoDerecha fila columna (insertarFicha fila columna color tablero)
color)) )
 (when (equal? (diagonalAbajoIzquierdaVolteable fila columna tablero 0 color) #t)
  (set! tablero (voltearDiagonalAbajoIzquierda fila columna (insertarFicha fila columna color
tablero) color)) )
 tablero)


(define (elegirMovimiento tablero color)
 (define click (mouse-click-posn (get-mouse-click ventana)))
 (define fila (floor (/(posn-y click )70)))
 (define columna (floor (/ (posn-x  click) 70)))
 (display "fila click") (display fila) (display "\n")
 (display "columna click") (display columna) (display "\n")
 (display  "\nJugador con fichas ") (display color) (display "\n Introduzca fila: ")
 (if (movimientoValido fila columna (movimientosPosibles (getCasillasColor 0 0 tablero color) color
tablero))
    (realizarMovimiento fila columna tablero color)
  (elegirMovimiento tablero color)))

(define (cambiarTurno turno)
 (if (= turno 1)  2  1))

(define (puedeJugar tablero turno)
 (if (not (equal? (movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero) '())) #t #f))

(define (finJuego tablero)
 (if (and (equal? (movimientosPosibles (getCasillasColor 0 0 tablero 1) 1 tablero) '())
       (equal? (movimientosPosibles (getCasillasColor 0 0 tablero 2) 2 tablero) '())) #t #f))

(define (acabarJuego tablero)
 (display "Se ha terminado el juego\n")
 (define marcador (marcadorJuego tablero))
 (imprimirMarcador marcador)
 marcador)
```

```
(define moveMinimax '())

(define (crearListaPosiciones listaPosibles)
  (if (equal? listaPosibles '()) '()
    (append (list(list (car listaPosibles) (cadr listaPosibles)))
        (crearListaPosiciones (cddr listaPosibles)))))

(define (busquedaMinimax tablero turno profundidad)
  (foldl(lambda (valor maxValor)
    (if (> (car valor) (car maxValor)) valor maxValor)) '(-9999 -1 -1)
      (for/list ([i (crearListaPosiciones (movimientosPosibles
                          (getCasillasColor 0 0 tablero turno) turno tablero))])
        (list (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
              (cambiarTurno turno) profundidad) (car i) (cadr i)))))

(define (minimax tablero turno profundidad)
  (if (or(finJuego tablero) (= profundidad 0)) (diferenciaPuntosMarcador(marcadorJuego tablero))
    (if (equal?(movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero)'())
      (minimax tablero (cambiarTurno turno) profundidad)
    (if (= turno 2)
      (foldl(lambda (valor maxValor)
          (if (> valor maxValor) valor maxValor)) -9999
          (for/list ([i (crearListaPosiciones (movimientosPosibles
                          (getCasillasColor 0 0 tablero turno) turno tablero))])
            (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
                (cambiarTurno turno) (- profundidad 1)) (car i) (cadr i)))
      (foldl(lambda (valor minValor)
          (if (<  valor minValor) valor minValor)) 9999
          (for/list ([i (crearListaPosiciones (movimientosPosibles
                          (getCasillasColor 0 0 tablero turno) turno tablero))])
            (minimax (realizarMovimiento (car i) (cadr i) tablero turno)
                (cambiarTurno turno) (- profundidad 1)) (car i) (cadr i)))))))

;Modo de juego: jugador contra jugador
(define (unoContraUno tablero turno marcador)
  (imprimirTablero 0 0 tablero)
  (if (finJuego tablero) (acabarJuego tablero)
  ((display "Turno del jugador: ") (display turno)
  (display "\nB: ")(display (car marcador)) (display " - N:") (display ( cadr marcador)) (display "\n")
  (if (puedeJugar tablero turno)
    ((dibujarOthello 0 0 0 0 0 turno
            (actualizarTablero(movimientosPosibles(getCasillasColor 0 0 tablero turno) turno
tablero) tablero))
     (imprimirPosiblesMovimientos tablero turno)
     (unoContraUno  (elegirMovimiento tablero turno) (cambiarTurno turno) (marcadorJuego
tablero)))
    (unoContraUno tablero (cambiarTurno turno) marcador)))))

(define movimiento '())
(define ia 0)
```

```
;Modo de juego: jugador contra maquina
(define (unoContraIAMinimax tablero turno marcador profundidad)
  (dibujarOthello 0 0 0 0 0 turno
            (actualizarTablero(movimientosPosibles (getCasillasColor 0 0 tablero turno) turno tablero)
tablero))
  (set! marcador (marcadorJuego tablero))
  (imprimirTablero 0 0 tablero)
  (imprimirMarcador marcador)
  (display "Turno del jugador ") (display turno) (display "\n")
  (if (equal? (finJuego tablero) #t) (acabarJuego tablero)
  (if(= turno 1)
      (if (puedeJugar tablero turno)
          (unoContraIAMinimax (elegirMovimiento tablero turno) (cambiarTurno turno) marcador
profundidad)
          (unoContraIAMinimax tablero (cambiarTurno turno) marcador profundidad))
      (if (puedeJugar tablero turno)
    (let ([ia (busquedaMinimax tablero turno profundidad)])
    (unoContraIAMinimax (realizarMovimiento (cadr ia) (car (cddr ia)) tablero turno)
                (cambiarTurno turno) marcador profundidad))
     (unoContraIAMinimax tablero (cambiarTurno turno) marcador profundidad)))))

(define menu '())

(define (dibujarMenu x1 y1 x2 y2 dificultad)
  (set! menu(open-viewport "menuPrincipal" 1002 666))
(((draw-pixmap-posn "fondo.jpg")menu) (make-posn 0 0))
  ((draw-ellipse menu) (make-posn x1 y1) 230 230 "Black")
  ((draw-ellipse menu) (make-posn x2 y2) 230 230 "Black")
  (define click (mouse-click-posn (get-mouse-click menu)))
  (when(and (> (floor (posn-x click) ) 24)
        (< (floor (posn-x click) ) 254)  (> (floor (posn-y click) ) 370 )
        (< (floor( posn-y click) ) 605) ) (juegoUnoVsIA dificultad) #f)
  (when(and (> (floor (posn-x click) ) 710)
        (< (floor (posn-x click) ) 964)  (> (floor (posn-y click) ) 370 )
        (< (floor( posn-y click) ) 605) ) (juegoUnoVsUno) #f))

(define (juegoUnoVsUno)
  (set! ventana(open-viewport "Othello" 560 560))
  (close-viewport menu)
  (unoContraUno partida1 1 (list 0 0)))

(define (juegoUnoVsIA dificultad)
  (close-viewport menu)
  (set! ventana(open-viewport "Othello" 560 560))
  (unoContraIAMinimax partida1 1 (list 0 0) dificultad))

(define (iniciarJuego)
  (display "Introduzca dificultad(1-5): ")
(define dificultad (read))
```

```
 (if (and (< dificultad 5) (> dificultad 0))(dibujarMenu 24 370 710 370 dificultad)
   (iniciarJuego)))
(iniciarJuego)
```

Archivo pruebas.rkt

```
#lang racket/base
(require rackunit)
(require (file "Othello.rkt"))
;Para poder ver únicamente el resultado de las pruebas debe comentar en el archivo Othello.rkt las 4 últimas funciones (menu,juegoUnoVsIA,juegoUnoVsUno y dibujarMenu)
(check-equal? (getValorCasilla 3 3 partida1)1)
;Esta prueba va a dar error:(check-equal? (getValorCasilla 3 3 partida1)2)
(check-equal? (getFila 3 partida1)'(0 0 0 1 2 0 0 0))

(check-equal? (insertarFicha 3 2 2 partida1)'((0 0 0 0 0 0 0 0)
                                              (0 0 0 0 0 0 0 0)
                                              (0 0 0 0 0 0 0 0)
                                              (0 0 2 1 2 0 0 0)
                                              (0 0 0 2 1 0 0 0)
                                              (0 0 0 0 0 0 0 0)
                                              (0 0 0 0 0 0 0 0)
                                              (0 0 0 0 0 0 0 0)))
(check-equal? (getCasillasColor 3 3 partida1 1)'((3 3) (4 4)))
(check-equal? (getPosiciones 3 3 1 partida1)'(3 5 5 3))
(check-equal? (movimientosPosibles (getCasillasColor 3 3 partida1 1) 1 partida1)'(3 5 5 3 4 2 2 4))
(check-equal? (comprobarEsquinaInferiorDerecha 3 2 partida1 0 1)'(5 4))
(check-equal? (comprobarEsquinaInferiorIzquierda 3 2 partida1 0 1)'())
(check-equal? (comprobarEsquinaSuperiorDerecha 3 2 partida1 0 1)'())
(check-equal? (comprobarEsquinaSuperiorIzquierda 3 2 partida1 0 1)'())
(check-equal? (comprobarFilaIzquierda 3 2 partida1 0 1)'())
(check-equal? (comprobarFilaDerecha 3 2 partida1 0 1)'())
(check-equal? (comprobarColumnaAbajo 3 2 partida1 0 1)'())
(check-equal? (comprobarColumnaArriba 3 2 partida1 0 1)'())
(check-equal? (actualizarTablero (movimientosPosibles(getCasillasColor 0 0 partida1 1) 1 partida1) partida1)'((0 0 0 0 0 0 0 0)
                                                                                                                (0 0 0 0 0 0 0 0)
                                                                                                                (0 0 0 0 "X" 0 0 0)
                                                                                                                (0 0 0 1 2 "X" 0 0)
                                                                                                                (0 0 "X" 2 1 0 0 0)
                                                                                                                (0 0 0 "X" 0 0 0 0)
                                                                                                                (0 0 0 0 0 0 0 0)
                                                                                                                (0 0 0 0 0 0 0 0)))
(check-equal? (columnaArribaVolteable 3 3 partida1 3 1)#f)
(check-equal? (columnaAbajoVolteable 3 3 partida1 3 1)#f)
(check-equal? (filaDerechaVolteable 3 3 partida1 3 1)#f)
(check-equal? (filaIzquierdaVolteable 3 3 partida1 3 1)#f)
(check-equal? (diagonalArribaIzquierdaVolteable 3 3 partida1 3 1)#f)
(check-equal? (diagonalArribaDerechaVolteable 3 3 partida1 3 1)#f)
(check-equal? (diagonalAbajoDerechaVolteable 3 3 partida1 3 1)#t)
```

```
(check-equal? (diagonalAbajoIzquierdaVolteable 3 3 partida1 3 1)#f)
(check-equal? (movimientoValido 3 3 (movimientosPosibles(getCasillasColor 0 0 partida1 1) 1
partida1))#f)
(check-equal? (contarBlancas partida1 0 0 0)2)
(check-equal? (contarNegras partida1 0 0 0)2)
(check-equal? (marcadorJuego partida1)'(2 2))
(check-equal? (diferenciaPuntosMarcador '(10 7))-3)
(check-equal? (realizarMovimiento 3 5 partida1 1)'((0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 1 1 1 0 0)
                                                   (0 0 0 2 1 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)))
(check-equal? (cambiarTurno 1)2)
(check-equal? (puedeJugar partida1 1)#t)
(check-equal? (finJuego partida1)#f)
(check-equal? (acabarJuego partida1)'(2 2))
(check-equal? (crearListaPosiciones (movimientosPosibles(getCasillasColor 0 0 partida1 1) 1
partida1))'((3 5) (5 3) (4 2) (2 4)))
(check-equal? (busquedaMinimax partida1 1 3)'(5 3 5))
(check-equal? (minimax partida1 1 3)2)
(check-equal? (voltearFilaIzquierda 3 5 partida1 1)'((0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 1 1 0 0 0)
                                                     (0 0 0 2 1 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)))

(check-equal? (voltearColumnaAbajo 1 3 partida1 1)'((0 0 0 0 0 0 0 0)
                                                    (0 0 0 0 0 0 0 0)
                                                    (0 0 0 1 0 0 0 0)
                                                    (0 0 0 1 2 0 0 0)
                                                    (0 0 0 2 1 0 0 0)
                                                    (0 0 0 0 0 0 0 0)
                                                    (0 0 0 0 0 0 0 0)
                                                    (0 0 0 0 0 0 0 0)))

(check-equal? (voltearColumnaArriba 6 3 partida1 1)'((0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 1 2 0 0 0)
                                                     (0 0 0 1 1 0 0 0)
                                                     (0 0 0 1 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)
                                                     (0 0 0 0 0 0 0 0)))
```

```
(check-equal? (voltearFilaDerecha 4 2 partida1 1)'((0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 1 2 0 0 0)
                                                   (0 0 0 1 1 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)
                                                   (0 0 0 0 0 0 0 0)))

(check-equal? (voltearDiagonalArribaIzquierda 5 5 partida1 1)'((0 0 0 0 0 0 0 0)
                                                               (0 0 0 0 0 0 0 0)
                                                               (0 0 0 0 0 0 0 0)
                                                               (0 0 0 1 2 0 0 0)
                                                               (0 0 0 2 1 0 0 0)
                                                               (0 0 0 0 0 0 0 0)
                                                               (0 0 0 0 0 0 0 0)
                                                               (0 0 0 0 0 0 0 0)))

(check-equal? (voltearDiagonalArribaDerecha 4 3 partida1 2)'((0 0 0 0 0 0 0 0)
                                                             (0 0 0 0 0 0 0 0)
                                                             (0 0 0 0 0 0 0 0)
                                                             (0 0 0 1 2 0 0 0)
                                                             (0 0 0 2 1 0 0 0)
                                                             (0 0 0 0 0 0 0 0)
                                                             (0 0 0 0 0 0 0 0)
                                                             (0 0 0 0 0 0 0 0)))

(check-equal? (voltearDiagonalAbajoDerecha 2 2 partida1 1)'((0 0 0 0 0 0 0 0)
                                                            (0 0 0 0 0 0 0 0)
                                                            (0 0 0 0 0 0 0 0)
                                                            (0 0 0 1 2 0 0 0)
                                                            (0 0 0 2 1 0 0 0)
                                                            (0 0 0 0 0 0 0 0)
                                                            (0 0 0 0 0 0 0 0)
                                                            (0 0 0 0 0 0 0 0)))

(check-equal? (voltearDiagonalAbajoIzquierda 2 5 partida1 2)'((0 0 0 0 0 0 0 0)
                                                              (0 0 0 0 0 0 0 0)
                                                              (0 0 0 0 0 0 0 0)
                                                              (0 0 0 1 2 0 0 0)
                                                              (0 0 0 2 1 0 0 0)
                                                              (0 0 0 0 0 0 0 0)
                                                              (0 0 0 0 0 0 0 0)
                                                              (0 0 0 0 0 0 0 0)))
```