

Instituto Tecnológico de Costa Rica
Ingeniería en computadores
Lenguaje, compiladores e intérpretes
Grupo 1

Tic Tac Toe Documentación

Profesor:

Marco Rivera Meneses

Integrantes:

- Ashley Vásquez Concepción, 2018216773
- Eduardo Bolívar Minguet, 2020158103
- Marcelo Truque Montero, 2020426017

Semestre I, 2022

Manual de Usuario

Tic tac toe es un juego cuyo fin es marcar tres espacios seguidos en un tablero. Juega un turno el contrincante y otro turno el usuario. Gana quien logre conectar tres ya sea de forma horizontal vertical o diagonal, si ninguno lo logra y no hay más espacios en el tablero se considera un empate. Para jugar simplemente indique las dimensiones del tablero y haga click sobre el botón check. Acto seguido verá el tablero de juego, seleccionela posición que desea marcar e inmediatamente el contrincante eligirá el otro. Continúe así hasta que uno de los dos gane o hasta que se acabe el espacio en el tablero y exista un empate

Descripción detallada de los algoritmos/arquitectura:

1. Algoritmo de construcción de la matriz:

- El algoritmo de construcción de la matriz tiene dos finalidades, construir una matriz que contenga el tablero de juego, y construir una matriz que contenga los botones relacionados al tablero de juego. Ambas matrices están relacionadas, solamente que el algoritmo tratará con la matriz de datos y la interfaz con la matriz de botones. El algoritmo recibe como entradas la cantidad de filas y de columnas que debe de tener la matriz y comienza a operar recursivamente de modo que se construya en base a las dimensiones dadas un grid de botones y en el código una matriz de datos

2. Algoritmo de Administración del juego:

- El algoritmo que administra el juego tiene como fin detectar la finalización del juego y manejar el intercambio de información entre el algoritmo voraz y la interfaz que utiliza el usuario. El se divide principalmente en dos partes, una administra las decisiones del usuario y pide al algoritmo su movimiento. La otra administra las decisiones del algoritmo y notifica a la interfaz de los cambios que este hace sobre el tablero de juego. Cada vez que una de las dos partes (usuario- CPU) hace un cambio el administrador llama a una función que detecta la finalización del juego, y notifica a la interfaz del final y el resultado. Existen tres posibles resultados: gana el usuario, pierde el usuario, empate.

3. Algoritmo finalización:

- El algoritmo que detecta la finalización busca si en el tablero un jugador que marcó un espacio tenga tres marcas en fila. Para ello recibe como entradas la matriz sin su primera fila la primera fila de la matriz la columna desde la cual se evaluará (el algoritmo se asegura que sea 0 para que recorra toda la matriz) y el jugador que se evaluará si ganó. Comienza a recorrer a la matriz fila por fila hasta que se llega a una ficha colocada por el jugador que recibió de entrada (0 CPU | 1 usuario) una vez se llega a este se revisa si le siguen tres fichas iguales en fila, horizontal, diagonal derecha y diagonal izquierda. Si no encuentra la ficha del jugador o llega al límite de la matriz continúa hasta encontrar otra ficha, por otro lado si la encuentra se devuelve inmediatamente éxito lo cual indica que terminó. Si no ha terminado procede a evaluar la posibilidad de que haya empatado por lo que recorre la matriz en busca de espacios en blanco, si no encuentra continúa con el juego, si no encuentra notifica a la interfaz el empate. En caso que no empatara y no hubiera ganador el juego continúa.

4. Algoritmo voraz:

- El algoritmo voraz tiene tres etapas principales y su propósito es buscar las coordenadas de la matriz en la cual es mejor (según el criterio del algoritmo) colocar la ficha. Esta recibe como entrada la matriz que el usuario modificó

para evaluar la mejor posición de la ficha. Primeramente el algoritmo detecta si tiene una posición para ganar ya que sería en el momento la mejor decisión, si la tiene pone la ficha y gana. Si no busca si el contrincante tiene una ficha para ganar y lo bloquea en caso de que si sea. Si ninguna de las dos se cumple busca colocar una ficha cercana a una ficha enemiga para bloquear movimientos posibles dado que el usuario tiene ventaja por jugar primero.

Descripción de las funciones implementadas:

Función detección tres en fila: recorre una fila recursivamente, tiene entre sus entradas un conteo de cuántas fichas seguidas ha detectado en caso de que este número sea 3 devuelve True. Si recorrió la fila y no encontró tres seguidos devuelve falso.

Función detección tres en columna: recorre una matriz recursivamente, tiene entre sus entradas la columna que se está revisando y un conteo de cuántas fichas seguidas ha detectado en caso de que este número sea 3 devuelve True. Si todas las filas en la columna indicada y no hay tres fichas seguidas devuelve falso.

Función detección tres en diagonal derecha: recorre una matriz recursivamente, tiene entre sus entradas la columna y la fila que se está revisando y un conteo de cuántas fichas seguidas ha detectado en caso de que este número sea 3 devuelve True. La función revisa estrictamente el diagonal hacia abajo a la derecha, por ello en su llamada recursiva suma 1 a las filas y las columnas para continuar la revisión. Devuelve falso si no hay tres en fila o si se llegó al límite de la matriz

Función detección tres en diagonal izquierda: recorre una matriz recursivamente, tiene entre sus entradas la columna y la fila que se está revisando y un conteo de cuántas fichas seguidas ha detectado en caso de que este número sea 3 devuelve True. La función revisa estrictamente el diagonal hacia abajo a la izquierda, por ello en su llamada recursiva resta 1 a las filas y las columnas para continuar la revisión. Devuelve falso si no hay tres en fila o si se llegó al límite de la matriz

Función de empate: Recorre de forma recursiva la matriz buscando un espacio en blanco lo cual indicaría que aún hay posibilidades de continuar. Si existe la posibilidad, indica que no hay empate todavía. Si al recorrer la matriz no encontró ningún espacio jugable notifica a la interfaz el fin del juego y le indica que no hubo ganador.

Función administradora del user: La llaman los botones de la interfaz y le indican qué coordenadas eligió el usuario para colocar su ficha, este coloca en la matriz de datos la ficha, evalúa si se llegó a un estado de finalización. Si llega a este estado directamente llama a una función de finalización de la interfaz y le notifica quién fue el ganador. Si no llama a una función que se encarga de administrar el movimiento del CPU, el algoritmo voraz se utiliza en esta llamada pues el administrador recibe directamente las coordenadas de la ficha del CPU por lo que ese parámetro es el retorno del algoritmo voraz.

Función administración del CPU: Esta función se divide en dos, la primera parte notifica a la interfaz de la jugada del CPU y cambia los botones y la matriz que la interfaz contiene, la otra revisa directamente si esta matriz cambiada contiene estado de finalización en cuyo caso le indica a la interfaz nuevamente que el juego terminó y que el ganador es el CPU. Una vez se llega a esta última sea cual sea el estado, termina el ciclo de información de juego y queda a criterio del usuario si continuar o no o en qué momento eligirá una ficha para volver a empezar.

Función de selección del algoritmo voraz: La función tiene como objetivo retornar una de las posiciones adyacentes a la casilla marcada con X (para tapar) o con O (para armar fila). Esta depende de otras funciones:

Función de candidatos: Esta función se encarga de retornar una lista con todas las posiciones adyacentes posibles. Para esto, hace uso de dos funciones: una función greedy que localiza la fila en la que se encuentra el primer valor de X o de O encontrado; y una función greedy que busca la columna donde se encuentra la casilla. Con estas coordenadas, la función candidatos retorna la lista con las posiciones adyacentes, validando que estas efectivamente existan y que no contengan nada marcado.

Función de selección: esta función hace uso de los candidatos generados anteriormente y, mediante una función aleatoria que retorna valores del 0 al 7, escoge a algún candidato.

Descripción de la ejemplificación

Se usaron principalmente dos estructuras de datos, listas y matrices (que esencialmente son solo listas pero se diferenciarán para mayor sencillez a la hora de describir las estructuras). La interfaz utiliza dos matrices, una para tener registro de los movimientos que el CPU efectúa y para notificar a este de los movimientos que el usuario haga. La otra contiene los botones que se muestran en la interfaz de modo que se pueda acceder a estos y cambiar su estado si el CPU decidió colocar su ficha en este espacio. Las matrices son análogas ya que representan la misma información pero para diferentes finalidades. A continuación se presentan algunos ejemplos gráficos de esto:

Matriz Datos

```
((  ()  ())
((  ()  ())
((  ()  ())
```

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

Matriz Botones

```
-  -  -
-  -  -
-  -  -
```

```
X  -  O
X  -  -
-  O  -
```

En la interfaz se debía acceder y modificarlas como si fueran variables, sin embargo, en el resto de la estructura no se guardan, sino que directamente se efectúan tratamientos y análisis para determinar información pertinente al juego. Finalmente también se implementaron listas para la obtención de coordenadas de la matriz, la lista contiene un par que corresponde a (fila columna) y funciona como observa a continuación:

Coordenadas

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

Matriz

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

Problemas sin solución

Durante el desarrollo de la tarea no se dejaron problemas sin resolver.

Plan de actividades:

Estudiante a cargo:	Tarea asignada:	Descripción de la tarea:	Tiempo estimado de completitud:	Fecha de entrega:
Eduardo Bolívar Minguet	CPU	Lógica de las decisiones tomadas por la máquina a través del algoritmo voraz.	1 semana	6/5/2022
	Implementación de la matriz	Se construirá una matriz por medio de una lista de listas para guardar los valores de X u O de los jugadores.	6 días	5/5/2022
Ashley Vásquez	Construcción visual de la matriz	Mediante bibliotecas gráficas, realizar un tablero interactivo de TicTacToe.	1 semana	6/5/2022
	Interfaz	Interfaz con la que interactúa el usuario para jugar tic-tac-toe	5 días	11/5/2022
Marcelo Truque Montero	Administrador	Conexión entre la interfaz y las decisiones que toma el algoritmo heurístico de la Revisión de finalización del juego y administración del ganador o el caso de empate	1 semana	6/5/2022

		Manejo de la matriz de juego		
--	--	------------------------------	--	--

Problemas encontrados:

Paradigma Funcional: Adaptarse a no usar variables y tratar de forma recursiva prácticamente todos los datos implicó un mayor reto y un planteamiento más exhaustivo de los problemas y algoritmos desarrollados

Mouse Event: Se pretendía utilizar un evento que nos indicara donde hacía click el usuario para saber donde colocar la X en el tablero, sin embargo, la implementación nos generó muchos problemas ya que a pesar de que se realizaba de acuerdo a la interfaz no se logró obtener las coordenadas, por lo que decidimos hacerlo con botones.

Botones: El cómo acceder a botones creados de forma recursiva fue un reto al principio, tras enterarnos de que se podían usar variables únicamente para cuestiones de la interfaz, se generó una variable que contuviera a los botones en una matriz de forma que se pudiera acceder a ellos de la misma forma que se accede a la matriz de datos

Mejora de algoritmos para mayor eficiencia: Tras el planteamiento inicial consideré que se podía mejorar la eficiencia del programa pues estaba recorriendo la matriz hasta tres veces para la búsqueda de cosas que están estrechamente relacionadas. Tras un replanteamiento se unificó el recorrido a una única función y por ende un único recorrido que cuando encuentra un caso favorable busca si existen tres en línea y de lo contrario continúa con su recorrido. Cabe añadir que con el algoritmo anterior no hubo problemas, fue por cuestiones personales que se quiso mejorar la cantidad de recorridos sobre la matriz. Se consideró también unificar el recorrido por empate, sin embargo dado que este implica un llamado diferente a la interfaz se decidió dejar ambos recorridos por aparte

Selección de la posición para el algoritmo voraz: El algoritmo voraz presentó diversos problemas en la elección de la casilla en la cual marcar. Al principio escogía sólo la casilla a la izquierda de una X o de una O; para dar solución se realizó una función que recolecta todas las posibles casillas, y mediante una decisión aleatoria, se escogía una de ellas.

Conclusiones:

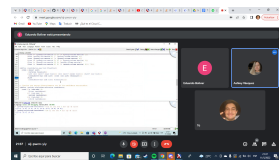
El paradigma funcional es capaz de reducir mucho la cantidad de líneas de un código, sin embargo al ser diferente de los paradigmas usuales conlleva un mayor tiempo de planteamiento de problemas y codificación y es un poco más difícil la detección inmediata de problemas. De igual forma esta manera de programar es sumamente eficiente en términos de memoria ya que erradica por completo el almacenamiento de variables, y deja exclusivamente almacenamiento para funciones y sus entradas. La recursividad compensa el hecho de no poder guardar variables más que las entrada ya que permite la operación y manipulación repetitiva de datos sin guardar en una variable los mismos, una vez recorrido el algoritmo se da la salida y se libera la memoria que ocupaban las variables de entrada de la función. En la interfaz es necesario el uso de variables puesto que de lo contrario se

pierden las referencias hacia estas y se mantienen prácticamente estáticas lo que le quita flexibilidad al código y complica modificaciones posteriores.

Recomendaciones:

- Definitivamente antes de adentrarse en el código es necesario entender realmente el paradigma funcional, sus ventajas y limitaciones, por ello se recomienda estudiar ampliamente el tema y las bibliotecas por utilizar
- Mantener al menos una variable que haga referencia a los widgets de la interfaz de forma que se pueda cambiar el mismo en cualquier momento del código y no solo dentro de la función que lo crea (ya que en el momento que sale de la función se pierde referencia del mismo)
- Hay mucha información en el manual de la GUI de racket acerca de la interfaz, sin embargo hay muy poca información de otros usuarios sobre la resolución de problemas comunes en esta, por lo que se recomienda leer ampliamente el manual y consultar experiencias de otros usuarios, pero se independiente de las mismas y hacer pruebas sobre las ventajas que ofrece la biblioteca.
- Hacer el código flexible de forma que reajustes necesarios por el avance del proyecto no conlleven una reestructuración compleja o llena de errores.

Minutas Reuniones

Fecha	Asistentes	Descripción Actividad	Evidencia
29/4/2022	Eduardo Bolívar Minguet Marcelo Truque Montero Ashley Vásquez	Reunión inicial y organización grupal División de tareas y Roles	
11/5/2022	Eduardo Bolívar Minguet Marcelo Truque Montero Ashley Vásquez	Reunión para unificación de las partes y solución de algunos problemas finales	

Bitácoras personales

Eduardo:

- **03/05/2022:**

Duración: De 6:00 p.m. a 9:30 p.m.

Se creó la función TTT (Tic Tac Toe) con los parámetros “filas” y “columnas”. La función verifica que el número de filas y de columnas sea mayor o igual que tres y menor o igual que diez, y prosigue a llamar a la función *build-matrix*, la cual construye la matriz (en la forma de una lista de listas) con base en los parámetros iniciales. También se definió una función que calcula el largo de una lista.

- **05/05/2022:**

Duración: De 1:00 p.m a 4:00 p.m.

Se trabajó en la función para insertar valores en la matriz dadas sus coordenadas, sin embargo, no se logró. También, se intentó hacer una función *value?* que verifica si un elemento es igual a otro dentro de la lista.

- **07/05/2022:**

Duración: De 4:00 p.m. a 10:00 p.m.

Se construyó una función de selección para el jugador que recibe como parámetros la fila y la columna donde insertar la X. Luego, se modificó para funcionar de forma general, para el jugador y para la máquina, recibiendo un parámetro más que corresponde a la ficha insertada (O o X).

- **10/05/2022:**

Duración: De 5:00 p.m. a 11:00 p.m.

Se empezó y se completó el algoritmo voraz para la elección de la casilla por parte de la máquina. Se subdividió en tres funciones principales:

1. *greedy-bot-selection*
2. *greedy-row*
3. *greedy-column*

También se construyeron funciones complementarias para facilitar el manejo. *Greedy-row* se modificó para tener dos posibles valores de retorno (la fila y el número de fila).

Se mejoró la función *value?* y se le dio el nombre de “*exist?*”, la cual verifica si un determinado valor pertenece a una lista.

- **11/05/2022:**

Duración: De 6:00 p.m. a 7:00 p.m.

Se realizó una función *random* que retorna valores de 0 a 7.

Duración: De 8:00 p.m. a 10:30 p.m.

Se realizó una mejor validación al algoritmo voraz para que pueda escoger entre más posibles alternativas alrededor de una determinada casilla.

Ashley:

- **02/05/2022:**

Duración: De 7:00 p.m. a 11:00 p.m.

Investigación inicial sobre la interfaz gráfica de racket por medio de video de ejemplos y de la documentación, se inicia a hacer pruebas para entender las funciones permitidas para la interfaz.

- **04/05/2022: De 2:00pm a 8:00pm.**

Creación de la ventana inicial donde se ingresa la cantidad de filas y columnas que se quieren para el juego y la validación de que estas sean máximo 10.

- **07/05/2022: De 3:00pm a 10:00pm.**

Creación de la ventana donde se generan las líneas de filas y columnas que dividen los espacios de juego, se intenta realizar el juego por medio de un *mouse event* que indique dónde hace click el usuario.

- **09/05/2022: De 4:00pm a 9:00pm.**

Se decide que es mejor realizar la interfaz por medio de botones ya que se tuvieron muchos problemas con la implementación del *mouse event*

- **11/05/2022: De 7:00pm a 9:00pm.**

Unir las partes de la interfaz gráfica, la ventana inicial donde se valida la cantidad de columnas y filas y la ventana donde se realiza el juego.

Marcelo

- **4/5/2022: De 9:00pm a 11:00pm**

Planteamiento inicial funciones recursivas para detección de finalización. Se hicieron funciones de prueba, se profundizó en conceptos del paradigma funcional, se reforzaron conceptos de recursividad de forma práctica. No se programó nada estrictamente de la tarea sino que se hicieron más que nada diferentes pruebas

- **7/5/2022: De 3:00pm a 7:00pm**

Se comenzó directamente la parte de administración de la tarea, se planteó una única función administrativa que se llama a si mismo y mantiene registro de jugadas de interfaz y CPU la función devuelve el identificador del jugador (que se definió en grupo fuera 1=usuario 0=CPU) una vez que finaliza. Se hizo una función que unifica la detección de finalización en una función booleana llamada *Fin?* la cual iba después a llamar a tres funciones por separado que recorren cada una la matriz buscando tres en fila correspondientes a: Filas Columnas Diagonal. De estas se implementó únicamente la que busca por filas Además se implementó una función

de empate que recorre la matriz por completo en búsqueda de espacios en blanco, si no encuentra devuelve 3 que sería el identificador de empate

- **8/5/2022:**

- De 5:00am a 7:00am**

- Se planteó como a manera mental el desarrollo de las funciones para detección de tres en filas y columnas, se consideró tras reflexionar que se estaba recorriendo la matriz múltiples veces para encontrar eventos sumamente similares por lo que se planteó el cambio de la lógica de la función Fin? para que esta recorriera la matriz y una vez encontrara la ficha de su interés llamara a las funciones Filas? Columnas? Diagonal? y estas revisaran si la ficha tenía tres seguidos en esta dirección. Se descarta la idea de unificar también el recorrido de empate ya que este debe dar una salida diferente a la función. Todo esto se planteó pero no se programó

- De 8:00am a 9:00am**

- Se cambió la lógica de Fin? tal y como se había especificado en el punto anterior, se redujo la cantidad de recorridos a solo dos uno para empate y otro para gane, se programaron las funciones que revisan tren en fila en columna o en diagonal, además se planteó una revisión más de diagonal hacia la izquierda ya que inicialmente esta solo se revisaba hacia la derecha. Se probaron todas las funciones y se terminó la parte administrativa

- **9/5/2022 De 10:00pm a 11:59pm:**

- Se replantea un cambio en la lógica de la interfaz, comienzo a investigar en la biblioteca para interfaz que tiene racket por defecto, se hacen pruebas menores para entender a mayor escala el funcionamiento de la interfaz gráfica

- **10/5/2022**

- De 12:00am a 3:00am:**

- Se genera una interfaz que crea recursivamente botones para generar un grid dada una cantidad de filas y columnas. Se hacen pruebas para comprobar el correcto funcionamiento de estos botones, se logra además cambiar configuraciones de estos para que se deshabiliten y cambie el texto cuando un usuario interactúa con ellos.

- De 10:00pm a 11:59pm:**

- Se intentó desarrollar una forma en que el CPU pudiera deshabilitar los botones y cambiar el texto considerando que no le puede hacer click automáticamente. Se resolvió generar una matriz que contenga los botones y una función que dadas unas coordenadas deshabilite el botón pertinente a las mismas y coloque una O en su texto.

- **11/5/2022**

- De 12:00am-2:00am**

- Se unificó la interfaz de botones con la administración del mismo, para ello se cambió levemente la lógica de la función administración de forma que ya no devolvía explícitamente un valor sino que llamaba a una función que administraba la condición de terminación en la interfaz. También se unificó la administración con el algoritmo voraz y se le dio un poco de estructura al proyecto. Se dividió la administración en dos, una para el usuario y otra para el CPU.

- De 5:00pm-6:00pm**

Se modificó ligeramente el algoritmo voraz, se añadieron a este dos funciones que se llaman primero, la primera evalúa si el CPU tiene posibilidad de ganar e inmediatamente elige estas coordenadas si hay posibilidad de gane. Si no busca si el contrincante tiene posibilidad de gane y lo bloquea inmediatamente. En caso contrario mantiene el criterio que tenía antes de la modificación.

Bibliografía:

<https://docs.racket-lang.org/gui/>

<https://stackoverflow.com/questions/40566422/random-function-in-racket>

