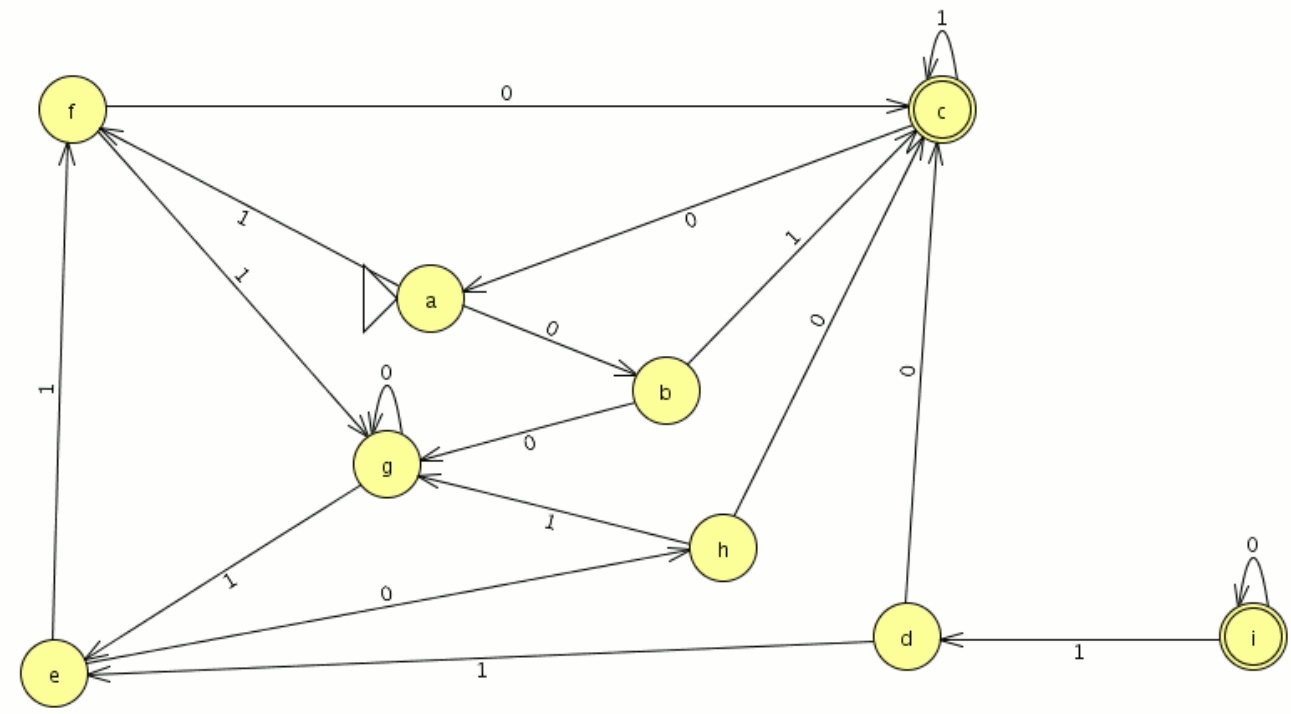
PROYECTO FINAL “Serpientes y escaleras”

TEORÍA DE LA COMPUTACIÓN



INTEGRANTES

- Madrigal Escoto Miguel Arturo **215767693** D17
- Flores Ontiveros Aide Sarahí **216457949** D17
- Fernández Román Kevin Arturo **216457949** D17

1. Planteamiento del problema

En la actualidad, estamos viviendo una etapa sumamente complicada, en donde debido a la contingencia sanitaria por el coronavirus SARS-COV2 nos vemos limitados al salir de nuestros hogares y a realizar ciertas actividades. Como resultado de esto, es riesgoso el reunirnos con la familia, conocidos, amigos, etc; lo cual puede llegar a provocar un alto nivel de estrés y ansiedad en la población, por lo tanto, la gran mayoría de las personas optan por buscar una distracción o entretenimiento, el cuál pueda ser disfrutado desde la comodidad de su hogar, entre ellos los juegos de mesa.

Los juegos de mesa son una forma de entretenimiento, los cuales son accesibles y mejoran tu habilidad mental; algunos otros son solo para pasar un buen rato con tus amigos. Sin embargo, hay ocasiones en las que obtener un juego de mesa puede ser complicado, sobre todo cuando no se tienen los recursos o las alternativas para adquirirlo. Por lo tanto, esta problemática puede ser solucionada por medio del uso de juegos de mesa a través de los dispositivos, como smartphones, tablets, computadoras, etc.

El juego de mesa que hemos decidido implementar es “serpientes y escaleras”, el cuál será representado a través del modelo de un autómata finito, debido a que estos pueden ser utilizados como reconocedores se secuencias de caracteres, lo cual nos permite reconocer la secuencia al avanzar a través del tablero del juego en las diferentes situaciones en las que un jugador se encuentre, es decir, cuando tire el dado y avance, cuando caiga en una serpiente o suba por una escalera.

2. Objetivo

Elaborar un modelo que represente el juego de “serpientes y escaleras” a través de un autómata finito.

Objetivos particulares

- Implementación del modelo del juego a través de un autómata finito en el software JFLAP.
- Implementación del funcionamiento del autómata finito en el lenguaje de programación Python.
- Creación de una interfaz gráfica por medio de la librería PySide2 para representar el tablero del juego y las transiciones que realiza el jugador al tirar un dado.
- Realización de una investigación sobre la teoría de autómatas finitos.

- Elaboración de un documento probatorio del proyecto final realizado.
- Comprobación del correcto funcionamiento del software tanto en la implementación con el lenguaje de programación Python como con casos de prueba adecuados en el software JFLAP.

3. Justificación

La realización de este proyecto pretende fungir como modelo ilustrativo de una aplicación de los autómatas finitos en un ejemplo de la vida real, con los conocimientos adquiridos en la asignatura de teoría de la computación y los saberes adquiridos en otras asignaturas como programación y métodos matemáticos, de tal forma que se pueda comprender el funcionamiento de un autómata finito y su aplicación en uno de los muchos ejemplos que se pueden representar por medio de la teoría de autómatas.

Además, la realización de este proyecto final será de mucha utilidad tanto para la carrera de computación como informática, sobre todo porque los miembros del equipo al pertenecer a la carrera de computación nos servirán como base para la resolución de problemas con mayor complejidad, sobre todo aquellos que se puedan presentar en la materia de Traductores de Lenguajes 2, donde aplicaremos los conocimientos adquiridos en Teoría de la Computación para diseñar nuestro propio compilador.

Cabe destacar que el poner en práctica los conocimientos adquiridos en el curso para resolver una problema de la vida real es de utilidad para no quedarnos solamente con el conocimiento teórico de cómo resolver un conjunto de ejercicios predefinidos, sino que mejora nuestras capacidades de análisis y abstracción para desenvolvernos de una manera adecuada en trabajos o proyectos de mayor magnitud, como es el caso de una tesis o un proyecto modular.

4. Marco teórico

Teoría de autómatas:

La teoría de los autómatas es una rama teórica de la informática y las matemáticas. Es el estudio de las máquinas abstractas y los problemas de cálculo que se pueden resolver utilizando estas máquinas. A una máquina abstracta se le llama autómata. La principal motivación detrás del desarrollo de la teoría de los autómatas fue crear métodos para describir y analizar el comportamiento dinámico de sistemas discretos.

Se entiende por autómata, por aquella máquina que toma alguna cadena como entrada y esta entrada pasa por un número finito de estados y puede entrar en el estado final. Este, consta de estados y transiciones; el estado está representado por círculos, y las transiciones por flechas.

En nuestro caso, se hará uso de un autómata finito, el cual está definido de la siguiente manera: modelo que realiza cómputos en forma automática sobre una entrada para producir una salida. Consta de una tupla de 5 elementos, la cual se detalla a continuación:

- Un conjunto finito de estados, a menudo designado como Q.
- Un conjunto finito de símbolos de entrada, a menudo designado como Σ.
- Una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado. La función de transición se designa habitualmente como δ.
- Un estado inicial, uno de los estados de Q.
- Un conjunto de estados finales o de aceptación F. El conjunto F es un subconjunto de Q.

La elección de este modelo matemático se debe en gran parte a que proporciona una solución simple y elegante a la problemática que se analiza en lo sucesivo. Aunado a lo anterior, dado que de una entrada(“tirar un dado”), se produce una salida(“nueva posición en la casilla”), el autómata finito se ajusta a las necesidades y condiciones del problema.

Problema a resolver: *juego “serpientes y escaleras”*

En el proceso de análisis y razonamiento para la representación del juego a través de un autómata finito, se identificó el siguiente conjunto de datos:

- *Estructura del tablero:* está compuesto por 25 casillas en una matriz de 5 x 5.
- *Estados:* se representan por medio de las casillas del tablero, es decir, cada casilla representa un estado q_n. El conjunto de estados estará formado por 25 elementos, donde q_0 es el estado inicial, hasta el estado de aceptación q_{24}.

  Además, existen tres subconjuntos de estados para su clasificación: *normal, serpiente y escalera*.

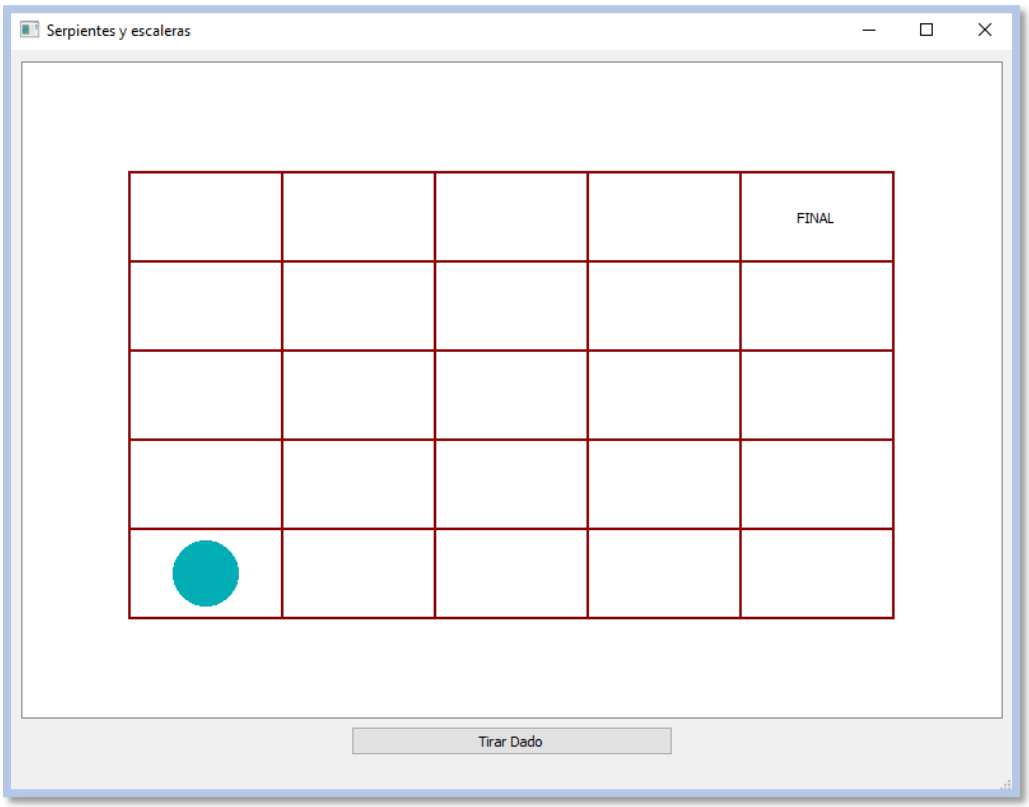
  Cada uno de los estados desde q_0 hasta q_{24} fue clasificado de manera arbitraria en uno de los subconjuntos; procurando que la mayoría fueran estados “normales” con el propósito de que la dificultad del juego no sea muy alta.

- *Transiciones:* el jugador se moverá a través de las casillas del tablero en función de un número obtenido al azar en un rango de [1, 6] (para simular el lanzamiento de un dado). La transiciones que se realizan en el juego dependen del estado actual del jugador:
    a) Si el jugador ha caído en una casilla (estado) categorizada como escalera, el símbolo de entrada es 0. Ejemplo: $q_4 \rightarrow q_{10}$.
    b) Si el jugador ha caído en una casilla (estado) categorizada como serpiente, el símbolo de entrada es 2. Ejemplo: $q_{16} \rightarrow q_{13}$.
    c) Si el jugador ha caído en una casilla (estado) que no está categorizada como serpiente o escalera, el símbolo de entrada es 1. Por lo tanto, en la transición se realizará el cambio de estado de q_n a q_{n+1}. Ejemplo: $q_7 \rightarrow q_8$.

5. Metodología

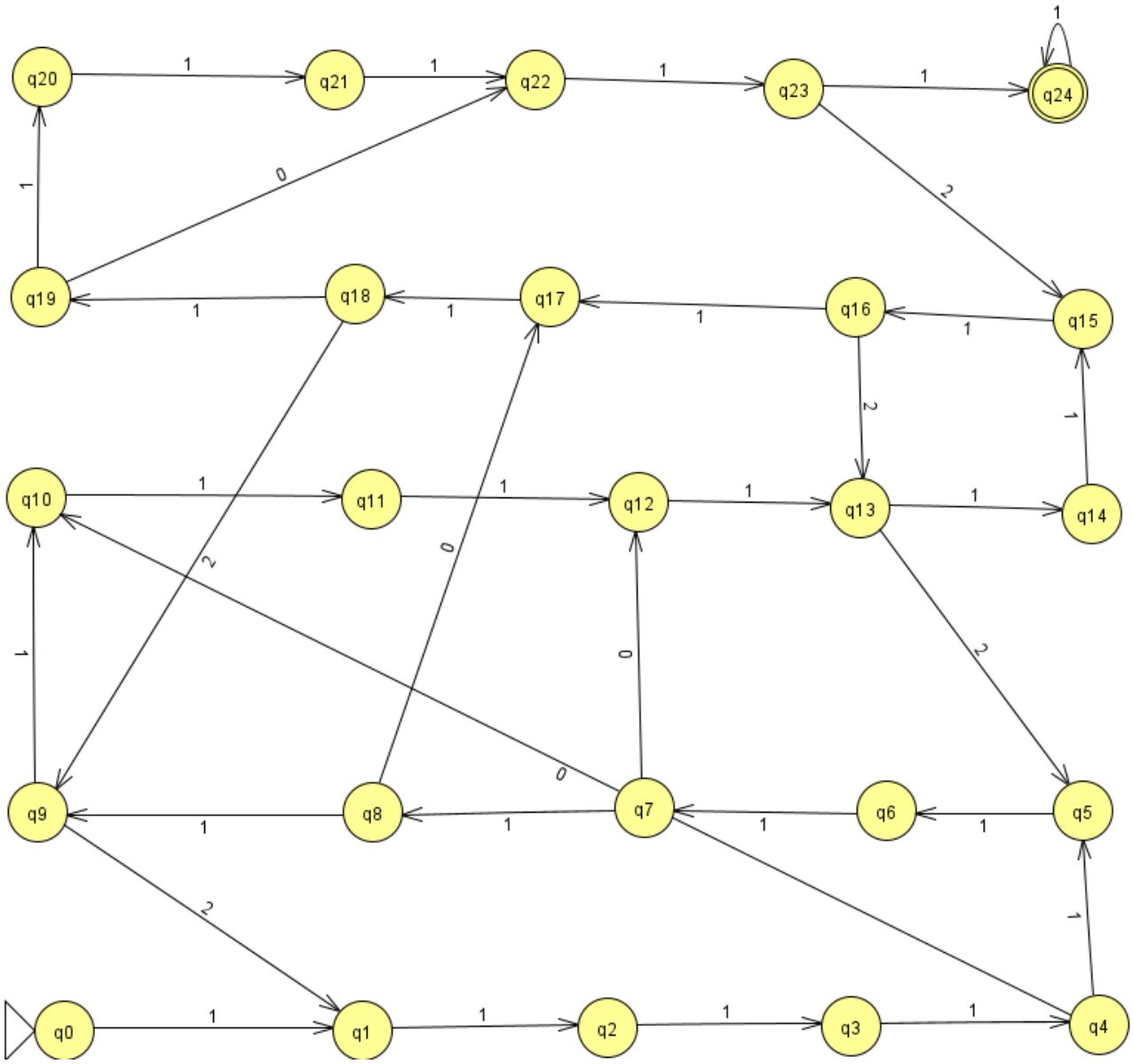
En primera instancia, se realizó el diseño del tablero de 5 x 5 por medio de la biblioteca PySide2, lo nos ayudó a tener una mejor perspectiva de la estructura del juego. Además, como equipo consideramos que el tablero no debe tener muchas casillas debido a que habría una saturación de estados y la tabla de transiciones quedaría muy extensa:

Diseño de la interfaz de usuario



En la elaboración del prototipo se anexó una circunferencia, la cual comienza en q_0 y esta se irá recorriendo conforme se vayan dando las transiciones. En la parte inferior se encuentra un botón para la obtención de un número al azar en el rango [1, 6], es decir, el lanzamiento de un dado. También en la casilla que representa el estado de aceptación $q_{24,}$ hay una etiqueta para denotar que cuando el jugador haya llegado a este punto habrá ganado la partida.

Diagrama de transición del proyecto



Posteriormente a la realización de la interfaz gráfica del proyecto, se realiza la implementación del autómata finito en el software JFLAP; del autómata debe ser determinista debido a que es requerido representar su funcionamiento a través de un lenguaje de programación.

Es importante recalcar que el modelo implementado en JFLAP debe coincidir con el número de estados y de transiciones del tablero, en este caso hay un total de 25 estados.

El AFD está formado por un conjunto de 25 estados, los cuales van desde q_0 como estado inicial hasta el estado final q_{24}. En el modelo se aprecia que sólo hay un estado de aceptación debido

a que este representa el estado, al cual el jugador debe llegar a través de una serie de transiciones para ganar el juego. Por lo tanto, los estados $q_{n,}$ donde $n < 24$ son considerados estados de no aceptación.

Por otro lado, los símbolos de entrada son el conjunto: { 1, 0, 2 }. En primer lugar, se tiene el símbolo de entrada 1, el cual se encuentra en las todas las transiciones del autómata debido a que representa un avance del jugador con respecto al valor obtenido del lanzamiento del dado. Por ejemplo, si un jugador se encuentra en el estado q_n saca un 5 haría una transición al estado q_{n+5}.

El siguiente símbolo de entrada es un 0, el cual se encuentra en las transiciones que representan una escalera en el juego, la cuales serán de ayuda para que el jugador se brinque una cierta cantidad de casillas. Por ejemplo, si el jugador cae en el estado q_4 haría una transición al estado q_{10} a través del símbolo de entrada 0.

Como tercer símbolo de entrada es un 2, el cual se encuentra en las transiciones que representan una serpiente en el juego, la cuales serán un obstáculo para hacer al jugador retroceder una cierta cantidad de casillas. Por ejemplo, si el jugador cae en el estado q_{18} haría una transición al estado q_9 a través del símbolo de entrada 2.

Por último, cabe recalcar que cualquier otro símbolo de entrada que no pertenezca al conjunto { 0, 1, 2 } será catalogado como inválido. Sin embargo, esta situación no suele suceder debido a que el usuario en ningún momento manipula los símbolos de entrada ya que el programa los agrega acorde con el estado actual, el número obtenido en el dado y la clasificación del estado (normal, escalera o serpiente).

Tabla de transición del proyecto

S / I	0	1	2
→ q_0	x	q_1	x
q_1	x	q_2	x
q_2	x	q_3	x
q_3	x	q_4	x
q_4	q_{10}	q_5	x
q_5	x	q_6	x
q_6	x	q_7	x
q_7	q_{12}	q_8	x
q_8	q_{17}	q_9	x
q_9	x	q_{10}	q_1
q_{10}	x	q_{11}	x
q_{11}	x	q_{12}	x
q_{12}	x	q_{13}	x
q_{13}	x	q_{14}	q_5
q_{14}	x	q_{15}	x
q_{15}	x	q_{16}	x
q_{16}	x	q_{17}	q_{13}
q_{17}	x	q_{18}	x
q_{18}	x	q_{19}	q_9
q_{19}	q_{22}	q_{20}	x
q_{20}	x	q_{21}	x
q_{21}	x	q_{22}	x
q_{22}	x	q_{23}	x
q_{23}	x	q_{24}	q_{15}
(q_{24})	x	q_{24}	x

Después de haber implementado el AFD en el software JFLAP se realiza la tabla de transición, la cuál es una manera diferente de visualizar el autómata, pero representan lo mismo.

El estado inicial se señala con una → en la primera columna, es decir, q_0. Además, el estado de aceptación se denota por una circunferencia a su alrededor (q_{24}).

En el encabezado de las columnas siguientes se encuentran los símbolos de entrada 0,1 y 2. Por lo tanto, la tabla de transición se puede interpretar como sigue:

- En la primera columna se encuentra el conjunto de todos los estados que forman parte del AFD.
- En la segunda columna se encuentra el conjunto de estados cuando el símbolo de entrada es 0 respecto a los estados de la primera columna. Si no existe una transición hacia el estado q_n, se denota con una 'x'.
- En la tercera columna se encuentra el conjunto de estados cuando el símbolo de entrada es 1 respecto a los estados de la primera columna. En este caso, se realizaría la transición de q_n a q_{n+1} si $n < 24$, debido a que si $n = 24$ nos quedaríamos en el mismo estado (transición a través del lazo).
- En la cuarta columna se encuentra el conjunto de estados cuando el símbolo de entrada es 2 respecto a los estados de la primera columna. Si no existe una transición hacia el estado q_n, se denota con una 'x'.

Estructura de datos bidimensional

0 (escalera)	1 (avanza)	2 (serpiente)	Aceptación
$f(q_0, 0) = \emptyset$	$f(q_0, 1) = q_1$	$f(q_0, 2) = \emptyset$	x
$f(q_1, 0) = \emptyset$	$f(q_1, 1) = q_2$	$f(q_1, 2) = \emptyset$	x
$f(q_2, 0) = \emptyset$	$f(q_2, 1) = q_3$	$f(q_2, 2) = \emptyset$	x
$f(q_3, 0) = \emptyset$	$f(q_3, 1) = q_4$	$f(q_3, 2) = \emptyset$	x
$f(q_4, 0) = q_{10}$	$f(q_4, 1) = q_5$	$f(q_4, 2) = \emptyset$	x
$f(q_5, 0) = \emptyset$	$f(q_5, 1) = q_6$	$f(q_5, 2) = \emptyset$	x
$f(q_6, 0) = \emptyset$	$f(q_6, 1) = q_7$	$f(q_6, 2) = \emptyset$	x
$f(q_7, 0) = q_{12}$	$f(q_7, 1) = q_8$	$f(q_7, 2) = \emptyset$	x
$f(q_8, 0) = q_{17}$	$f(q_8, 1) = q_9$	$f(q_8, 2) = \emptyset$	x
$f(q_9, 0) = \emptyset$	$f(q_9, 1) = q_{10}$	$f(q_9, 2) = q_1$	x
$f(q_{10}, 0) = \emptyset$	$f(q_{10}, 1) = q_{11}$	$f(q_{10}, 2) = \emptyset$	x
$f(q_{11}, 0) = \emptyset$	$f(q_{11}, 1) = q_{12}$	$f(q_{11}, 2) = \emptyset$	x
$f(q_{12}, 0) = \emptyset$	$f(q_{12}, 1) = q_{13}$	$f(q_{12}, 2) = \emptyset$	x
$f(q_{13}, 0) = \emptyset$	$f(q_{13}, 1) = q_{14}$	$f(q_{13}, 2) = q_5$	x
$f(q_{14}, 0) = \emptyset$	$f(q_{14}, 1) = q_{15}$	$f(q_{14}, 2) = \emptyset$	x
$f(q_{15}, 0) = \emptyset$	$f(q_{15}, 1) = q_{16}$	$f(q_{15}, 2) = \emptyset$	x
$f(q_{16}, 0) = \emptyset$	$f(q_{16}, 1) = q_{17}$	$f(q_{16}, 2) = q_{13}$	x
$f(q_{17}, 0) = \emptyset$	$f(q_{17}, 1) = q_{18}$	$f(q_{17}, 2) = \emptyset$	x
$f(q_{18}, 0) = \emptyset$	$f(q_{18}, 1) = q_{19}$	$f(q_{18}, 2) = q_9$	x
$f(q_{19}, 0) = q_{22}$	$f(q_{19}, 1) = q_{20}$	$f(q_{19}, 2) = \emptyset$	x
$f(q_{20}, 0) = \emptyset$	$f(q_{20}, 1) = q_{21}$	$f(q_{20}, 2) = \emptyset$	x
$f(q_{21}, 0) = \emptyset$	$f(q_{21}, 1) = q_{22}$	$f(q_{21}, 2) = \emptyset$	x
$f(q_{22}, 0) = \emptyset$	$f(q_{22}, 1) = q_{23}$	$f(q_{22}, 2) = \emptyset$	x
$f(q_{23}, 0) = \emptyset$	$f(q_{23}, 1) = q_{24}$	$f(q_{23}, 2) = q_{15}$	x
$f(q_{24}, 0) = \emptyset$	$f(q_{24}, 1) = q_{24}$	$f(q_{24}, 2) = \emptyset$	V

Posteriormente a la realización de la tabla de transición, se procede a hacer la representación del AFD por medio de una estructura de datos bidimensional.

Por lo tanto, la estructura de datos bidimensional se puede interpretar como sigue:

- En las primeras tres columnas se encuentran las funciones de estado siguiente. Por lo tanto, para cada función se recibe el estado actual, el símbolo de entrada y obtenemos un estado q_n. En caso de no haber un estado siguiente, obtenemos un ∅.
- La cuarta columna indica si se trata de un estado de aceptación (“V”) o un estado de no aceptación (“x”).

6. Pruebas y resultados

Primer caso de prueba

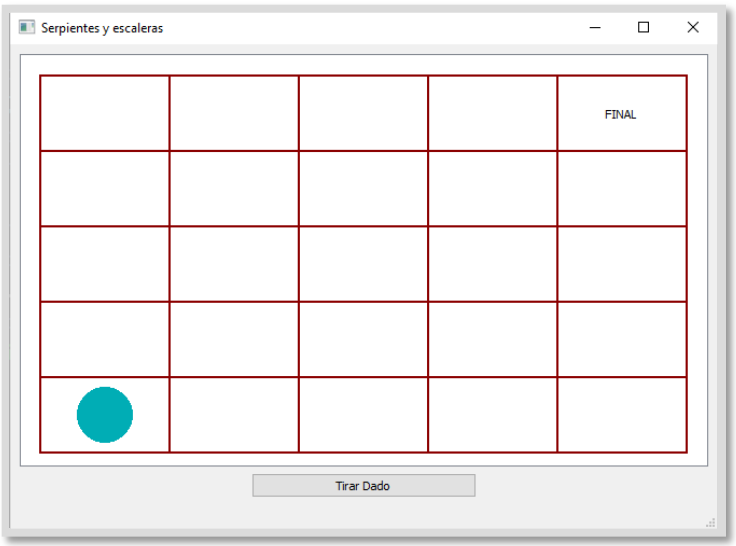


Fig 1.0 Se muestra la interfaz de usuario previo a tirar el dado, por lo que no hay símbolos de entrada; por lo tanto no hay transiciones.

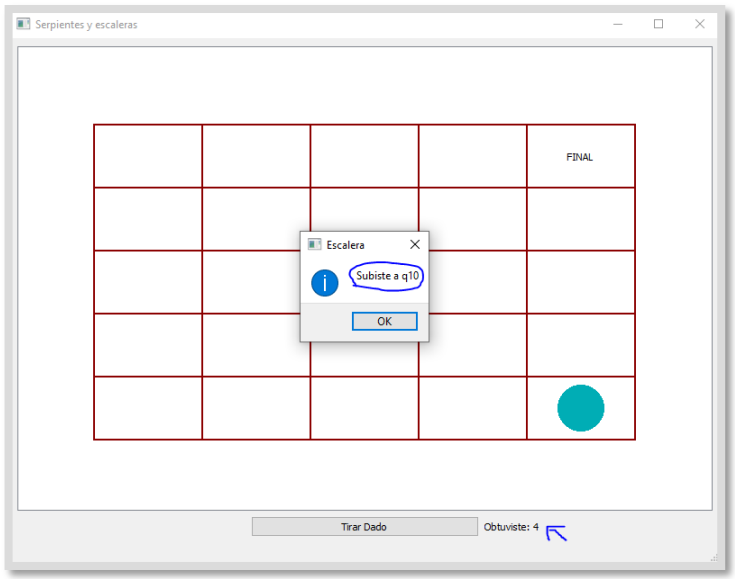


Fig 1.1 Posteriormente se realiza el primer lanzamiento del dado y se obtiene un 4. Por lo tanto los símbolos de entrada son: 1111 y se hace la transición de q_0 a q_4.

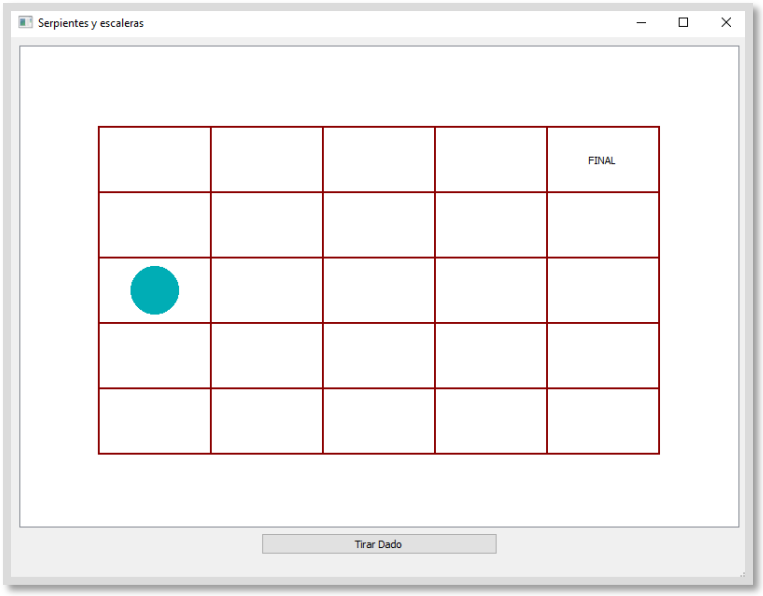


Fig 1.2 Al haber caído en un estado escalera (q_4), se agrega el símbolo de entrada 0 para realizar la transición de q_4 a q_{10}.

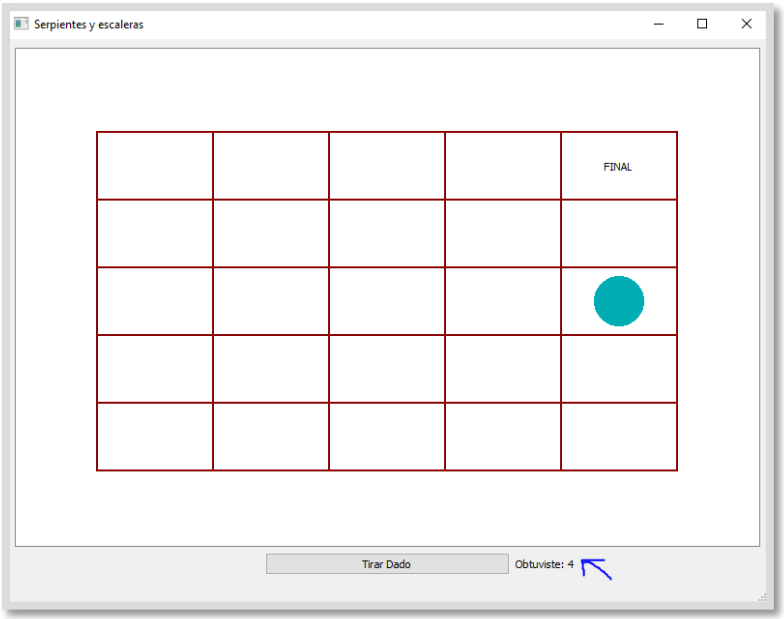


Fig 1.3 Después se vuelve a tirar el dado y sale un 4. Por lo tanto los símbolos de entrada que se agregan son: 1111 y se hace la transición de q_{10} a q_{14}.

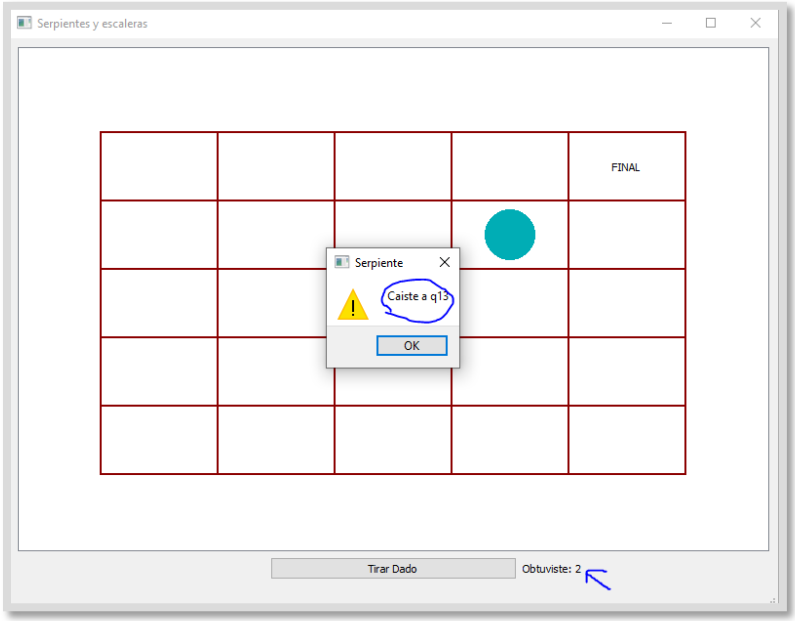


Fig 1.4 Después se vuelve a tirar el dado y sale un 2. Por lo tanto se agregan los símbolos de entrada: 11 y se hace la transición de q_{14} a q_{16}.

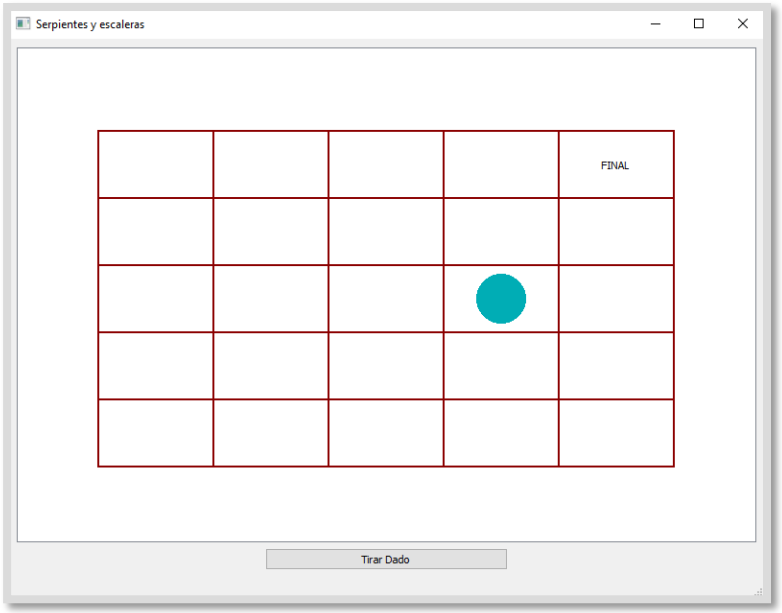


Fig 1.5 Al haber caído en un estado serpiente (q_{16}), se agrega el símbolo de entrada 2 para realizar la transición de q_{16} a q_{13}.

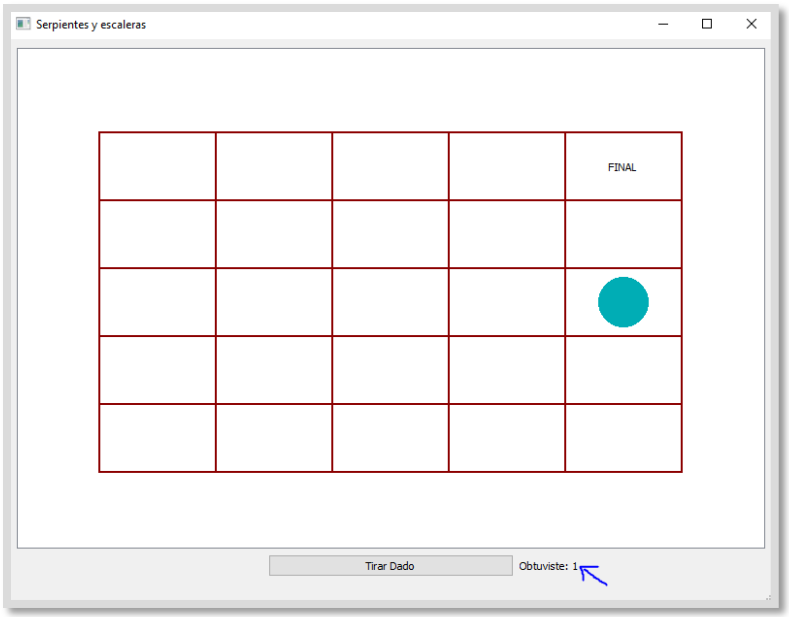


Fig 1.6 Después se vuelve a tirar el dado y sale un 1. Por lo que se agrega el símbolo de entrada: 1 y se hace la transición de q_{13} a q_{14}.

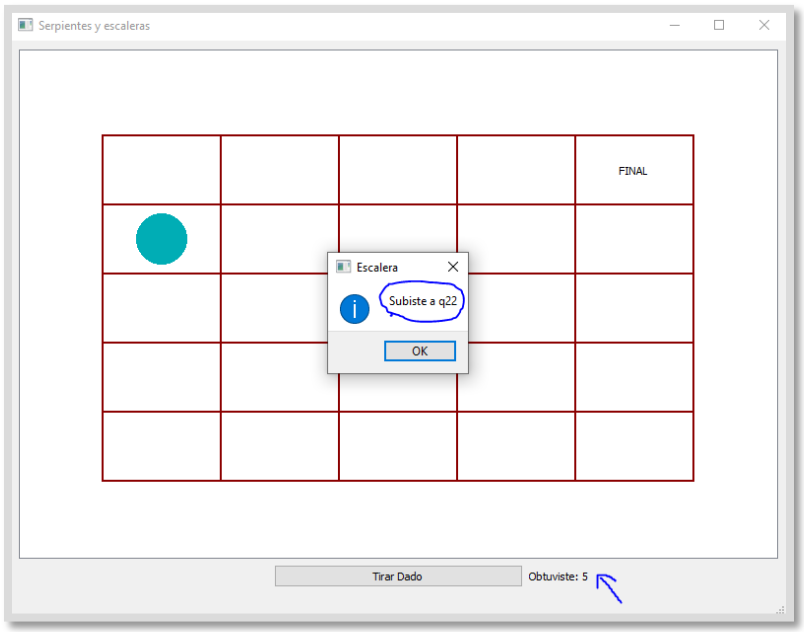


Fig 1.7 Después se vuelve a tirar el dado y sale un 5. Por lo tanto los símbolos de entrada que se agregan son: 11111 y se hace la transición de q_{14} a q_{19}.

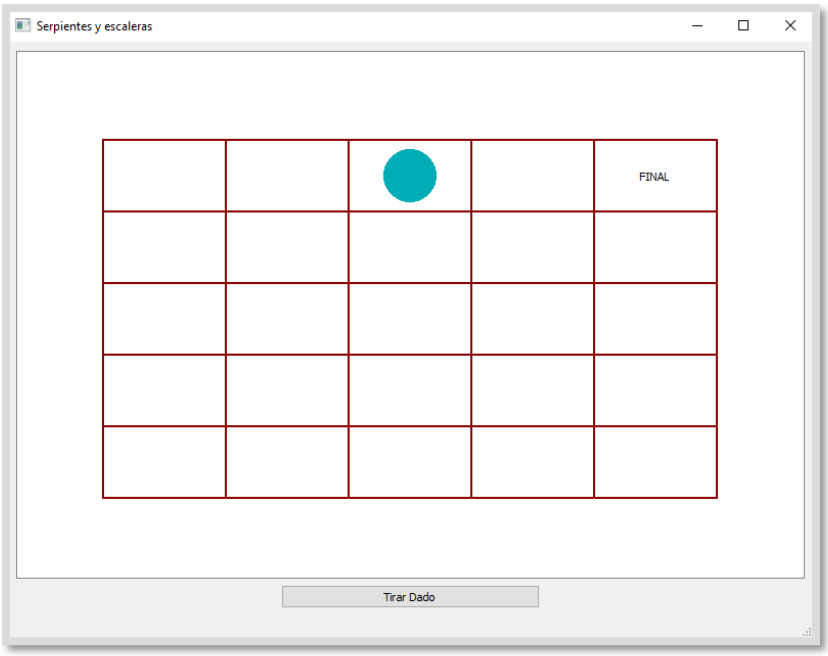


Fig 1.8 Al haber caído en un estado escalera (q_{19}), se agrega el símbolo de entrada 0 para realizar la transición de q_{19} a q_{22}.

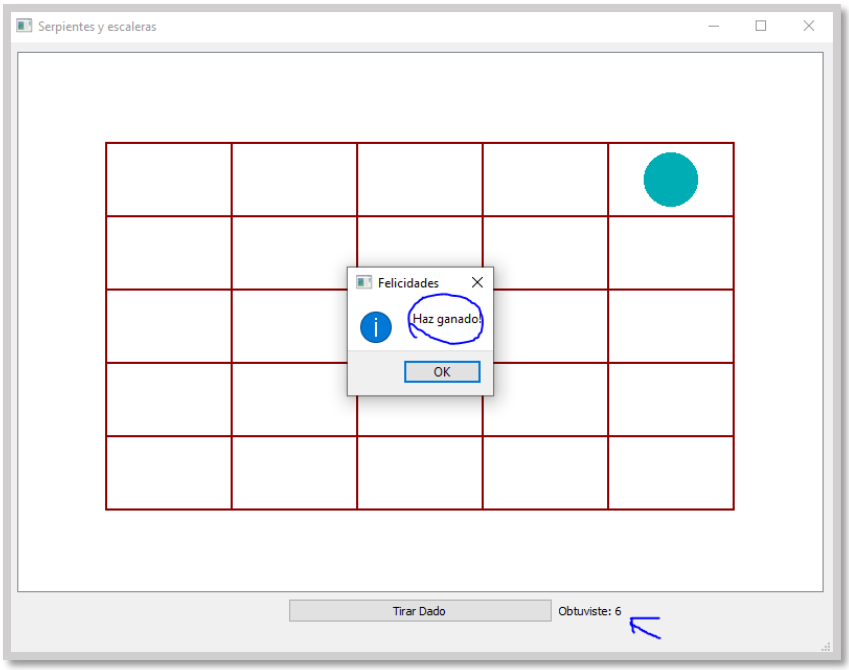
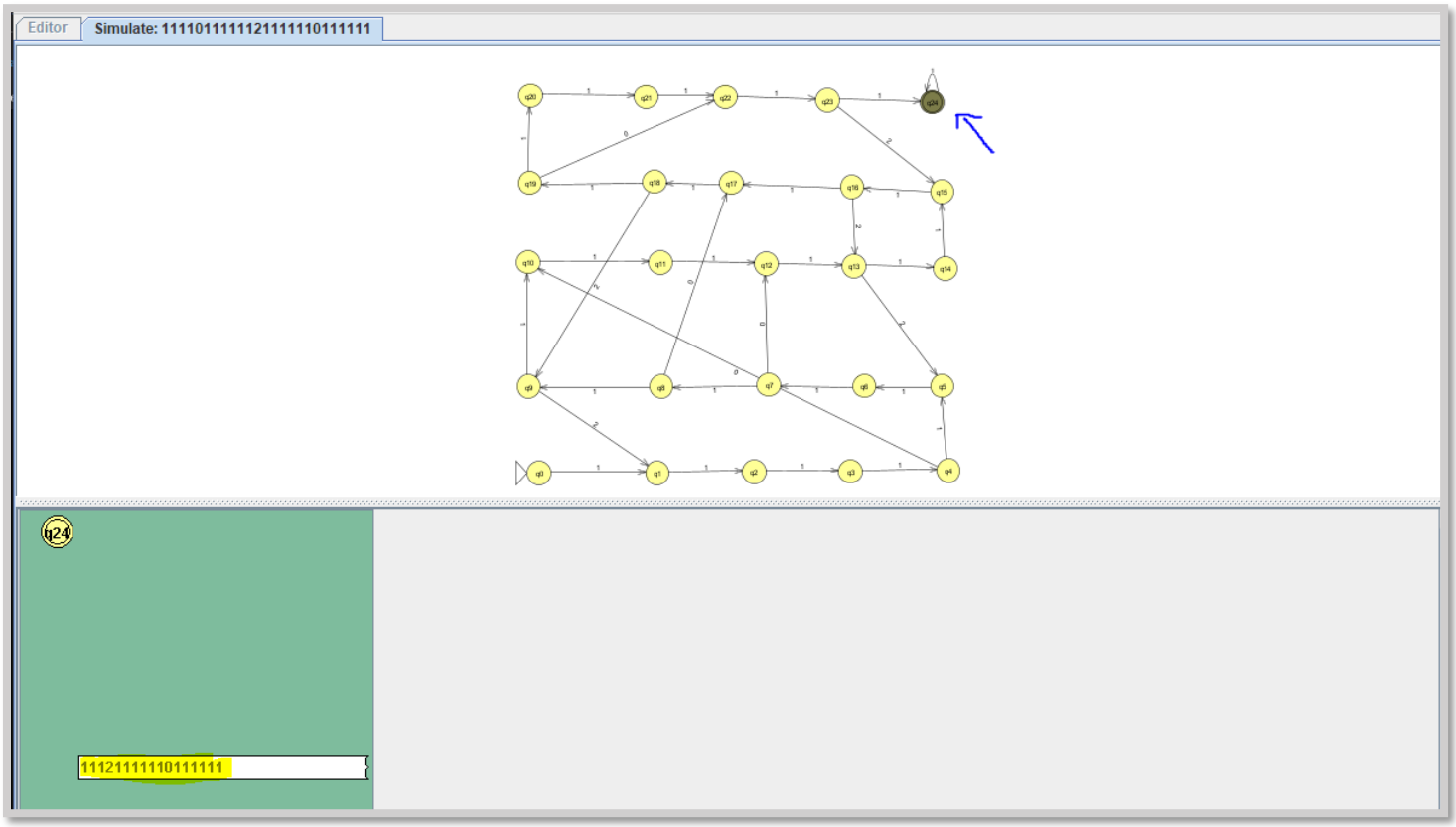


Fig 1.9 Después se vuelve a tirar el dado y sale un 6. Por lo tanto los símbolos de entrada que se agregan son: 111111 y se hace la transición de q_{22} a q_{24}. La transición previa se hace a través del lazo para los últimos 4 símbolos de entrada (1111), debido a que de q_{22} a q_{24} se hace la transición con los símbolos 11.

Por lo tanto, el jugador ha llegado al estado de aceptación q_{24} a través de la siguiente secuencia de símbolos de entrada en las transiciones:

```
[1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1]
```

Nota: representación de los símbolos de entrada que se utilizaron en la partida, los cuales se encuentran almacenados en una lista; cada elemento está separado por una coma.



Posteriormente se puede realizar la prueba en el software JFLAP para saber si el AFD acepta la secuencia de símbolos de entrada de las transiciones realizadas en el juego representado en la interfaz gráfica. Se puede comprobar que en ambos casos se llega al estado de aceptación q_{24}.

Segundo caso de prueba

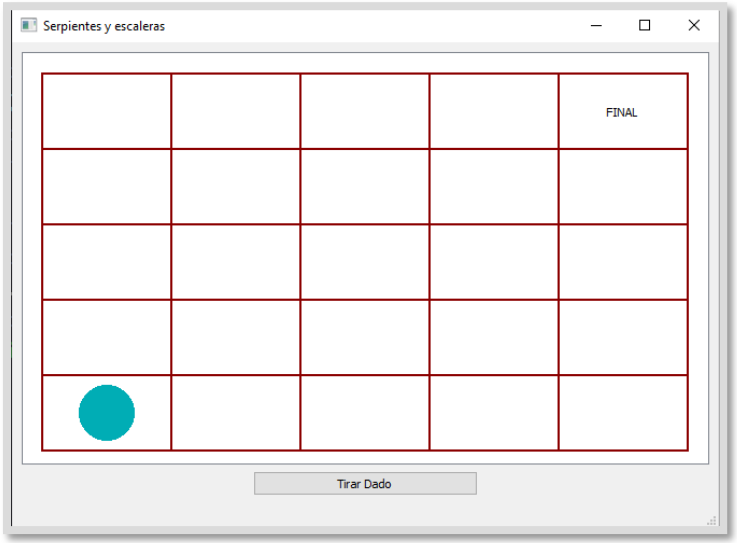


Fig 2.0 Se muestra la interfaz de usuario previo a tirar el dado, por lo que no hay símbolos de entrada; por lo tanto no hay transiciones.

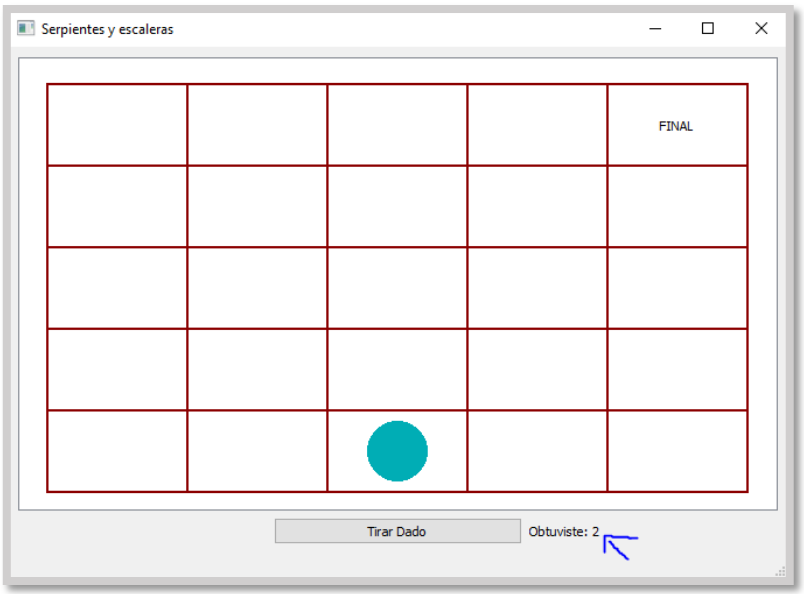


Fig 2.1 Posteriormente se realiza el primer lanzamiento del dado y se obtiene un 2. Por lo tanto los símbolos de entrada son: 11 y se hace la transición de q_0 a q_2.

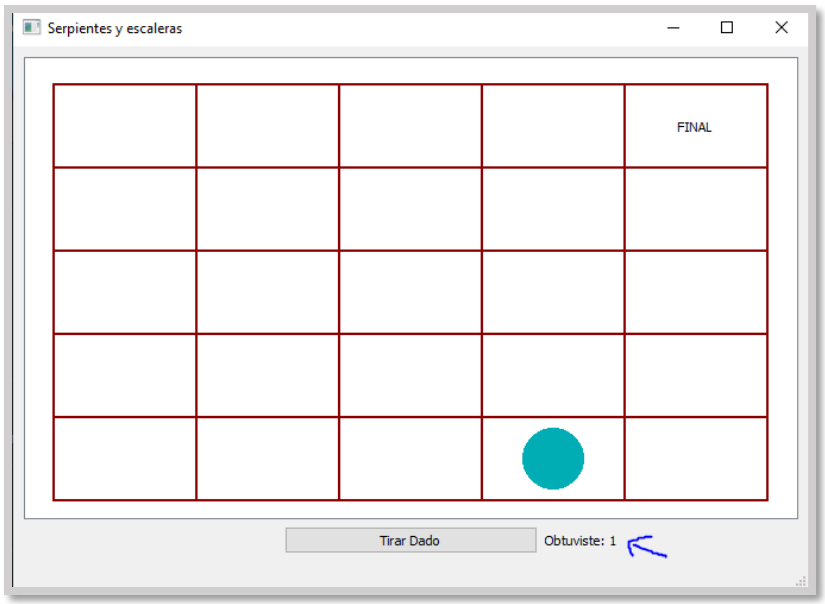


Fig 2.2 Después se vuelve a tirar el dado y sale un 1. Por lo que se agrega el símbolo de entrada: 1 y se hace la transición de q_2 a q_3.

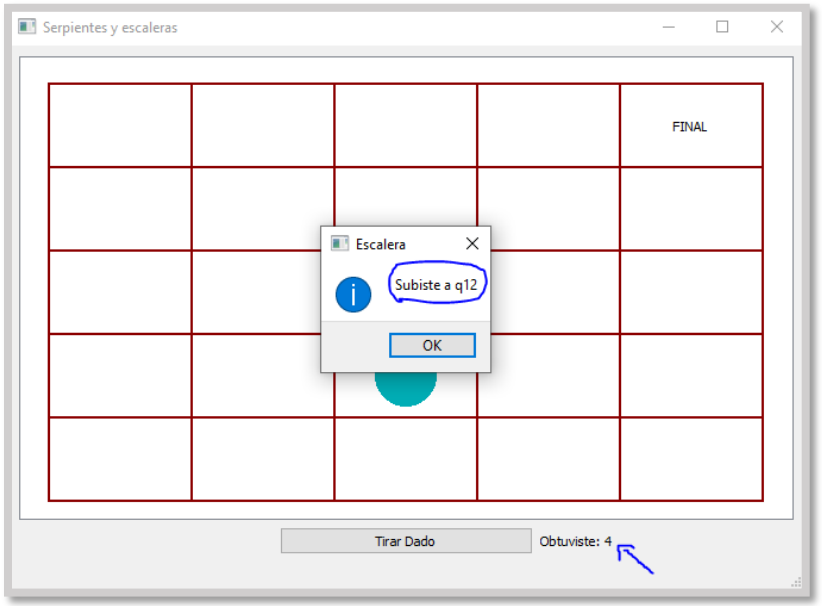


Fig 2.3 Después se vuelve a tirar el dado y sale un 4. Por lo tanto los símbolos de entrada que se agregan son: 1111 y se hace la transición de q_3 a q_7.

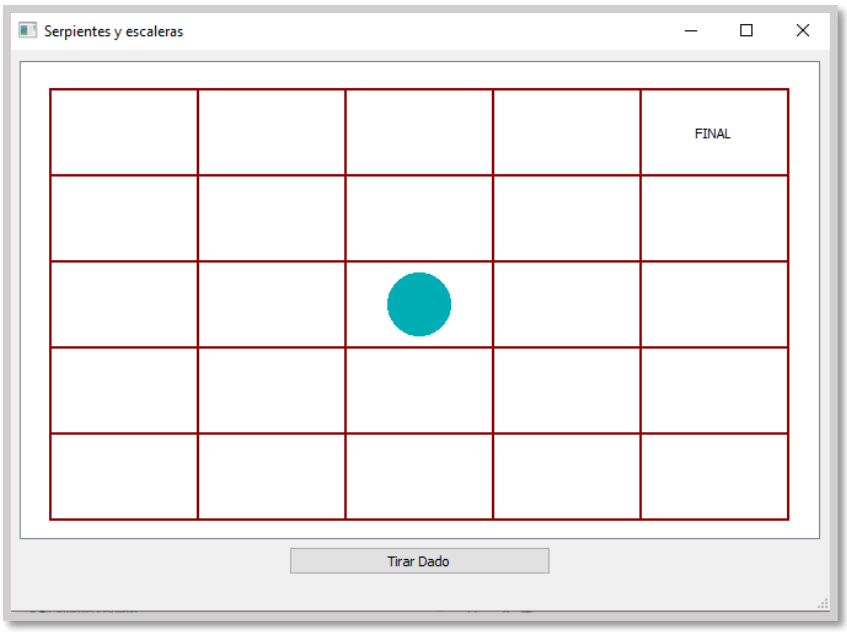


Fig 2.4 Al haber caído en un estado escalera (q_7), se agrega el símbolo de entrada 0 para realizar la transición de q_7 a q_{12}.

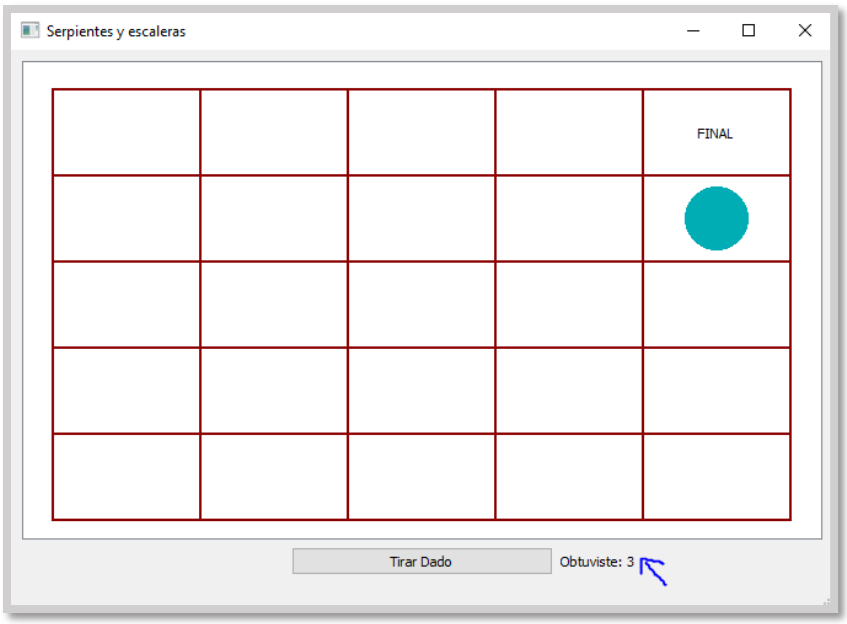


Fig 2.5 Después se vuelve a tirar el dado y sale un 3. Por lo tanto los símbolos de entrada que se agregan son: 111 y se hace la transición de q_{12} a q_{15}.

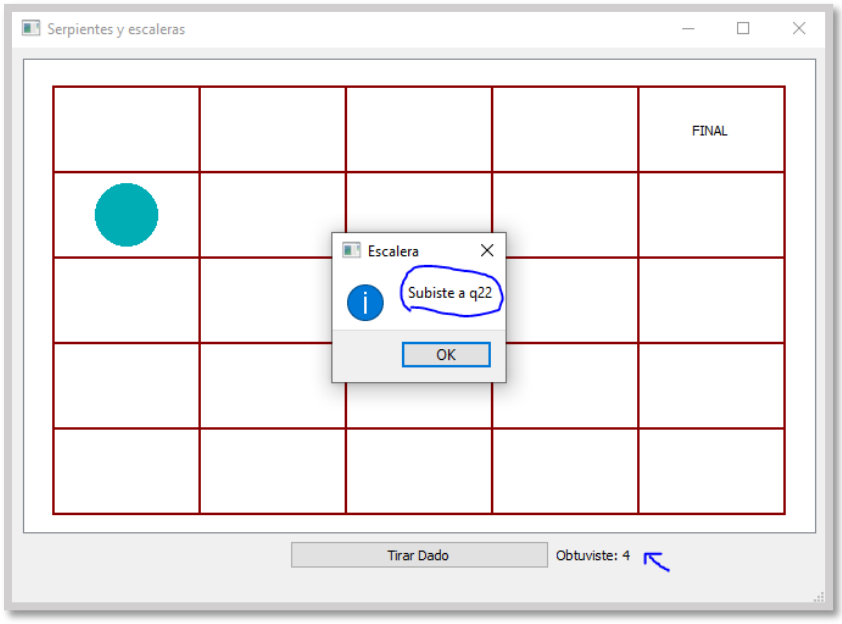


Fig 2.6 Después se vuelve a tirar el dado y sale un 4. Por lo tanto los símbolos de entrada que se agregan son: 1111 y se hace la transición de q_{15} a q_{19}.

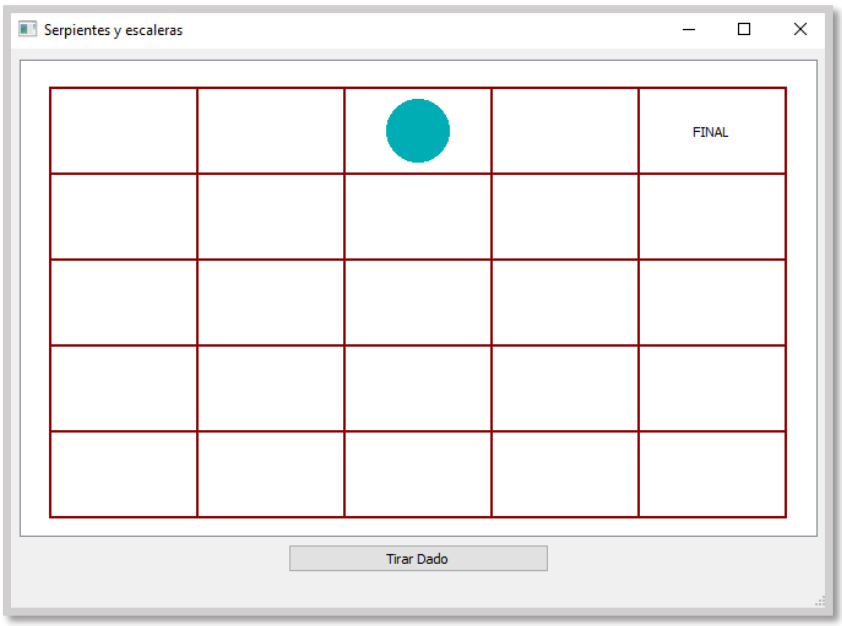


Fig 2.7 Al haber caído en un estado escalera (q_{19}), se agrega el símbolo de entrada 0 para realizar la transición de q_{19} a q_{22}.

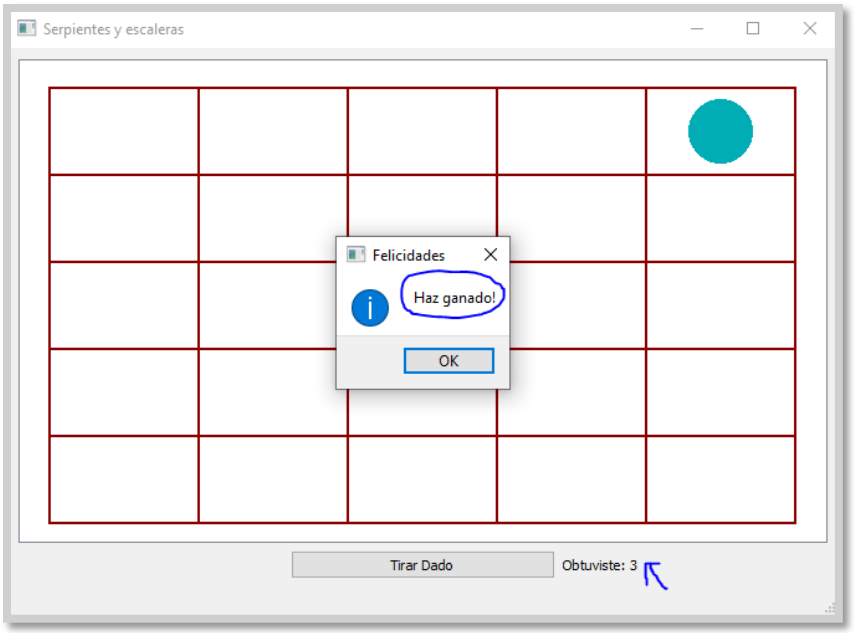
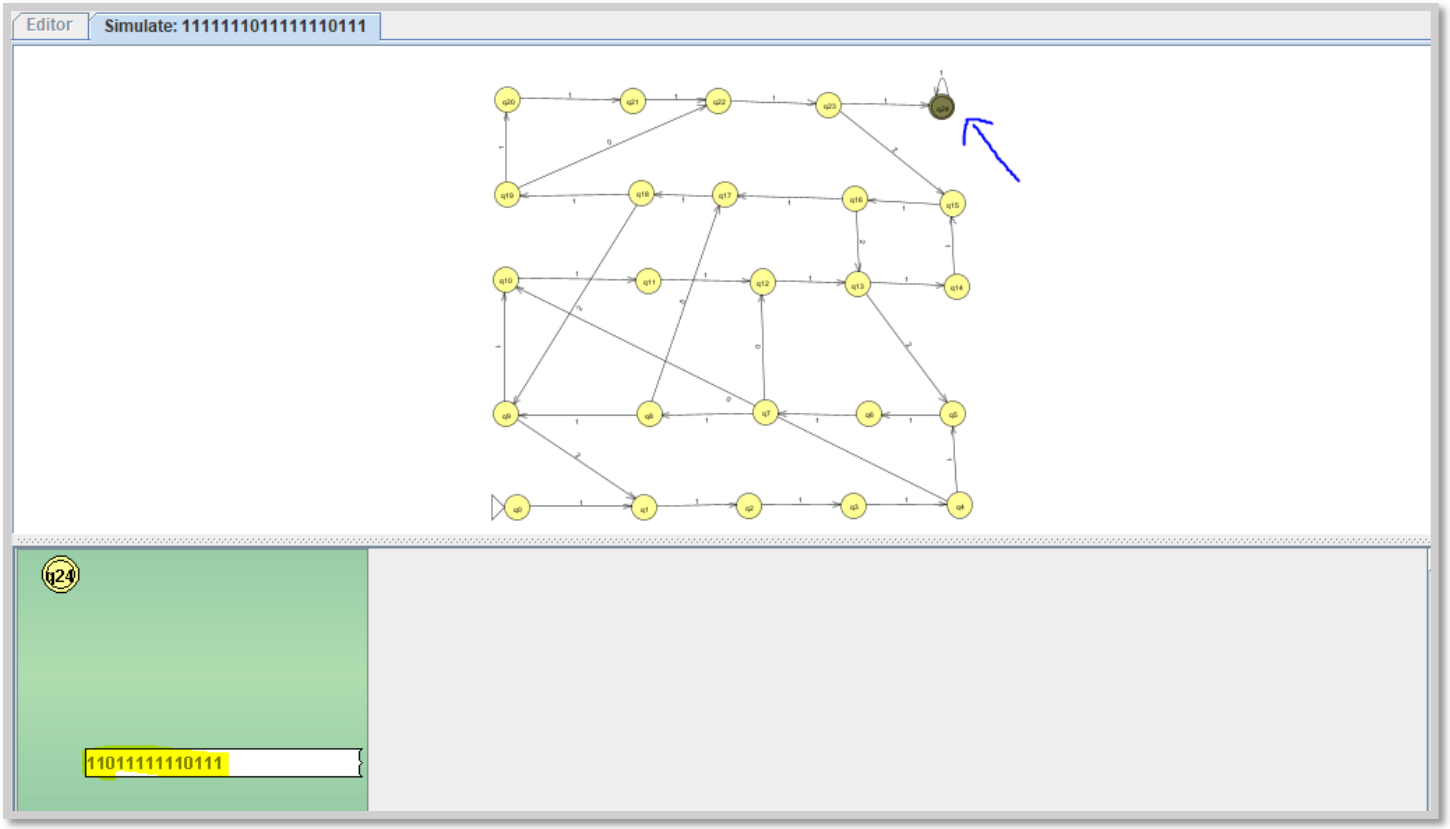


Fig 2.8 Después se vuelve a tirar el dado y sale un 3. Por lo tanto los símbolos de entrada que se agregan son: 111 y se hace la transición de q_{22} a q_{24}. La transición previa se hace a través del lazo para el último símbolo de entrada (1), debido a que de q_{22} a q_{24} se hace la transición con los símbolos 11.

Por lo tanto, el jugador ha llegado al estado de aceptación q_{24} a través de la siguiente secuencia de símbolos de entrada en las transiciones:

```
[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
```



Posteriormente se puede realizar la prueba en el software JFLAP para saber si el AFD acepta la secuencia de símbolos de entrada de las transiciones realizadas en el juego representado en la interfaz gráfica. Se puede comprobar que en ambos casos se llega al estado de aceptación q_{24}.

7. Conclusiones

A través de la realización de este proyecto se logró elaborar un modelo para representar el juego de “serpientes y escaleras” de forma digital a través de un autómata finito. Al ya tener los conocimientos base adquiridos acerca de los autómatas, nos fue sencillo implementar uno en el software de JFLAP, con el cual pudimos hacer pruebas de las cadenas aceptadas o rechazadas con el fin de averiguar si nuestro autómata finito funcionaba de forma correcta y eficiente para poder codificarlo en el lenguaje de programación Python.

La implementación del autómata finito en un lenguaje de programación resultó de forma óptima, pues la codificación de este se realizó de forma sencilla al hacer uso de una estructura de datos bidimensional (matriz). Para hacer uso de la matriz se necesitó de saber el estado actual, el

símbolo de entrada, que puede ser 1 para avanzar, 0 para serpiente y 2 para escalera, y con esto obtenemos el estado siguiente (si es que existe) de una forma práctica y eficiente.

El código implementado para el autómata es considerado el óptimo para este objetivo específico que era recrear el juego de mesa “Serpientes y Escaleras”. Sin embargo, debe quedar claro que se puede logar el mismo objetivo de distintas formas, como casi todos los programas que pueden ser codificados de diferente manera y se llega al mismo fin. En cuanto a nuestra experiencia obtenida en la realización de este proyecto, aprendimos que hacer uso de autómatas al codificar nos simplifica el trabajo, pues al ver el autómata representado en una estructura de datos bidimensional o una matriz, podemos manejar los estados del autómata fácilmente, tanto el estado actual como el estado siguiente, y ya no tenemos que implementar un mecanismo de control de selección como lo sería un switch o Ifs anidados para el caso específico del lenguaje Python.

Para finalizar, el proyecto cumplió con la expectativa planteada de un principio que era realizar un juego de mesa implementado un autómata. Además, se le agregó una interfaz gráfica para que sea más amigable con el usuario, de tal forma que el usuario presiona el botón de tirar dado, se le avisa el número que obtuvo en el dado que serían las casillas por avanzar y se le muestra la casilla donde quedó su ficha después del lanzamiento, por lo cual quedamos satisfechos con el trabajo realizado de principio a fin.

8. Referencias bibliográficas

- Create Executable from Python Script using Pyinstaller—Data to Fish. (s. f.). (1 diciembre 2021) Disponible en: https://datatofish.com/executable-pyinstaller/

- EcuRed (s. f.). (4 diciembre 2021) Autómata finito. Disponible en: https://www.ecured.cu/Aut%C3%B3mata_finito.

- Gomez, A. (2014). Introducción a la Teoría de Autómatas y Lenguajes Formales. Jalisco: TRAUCO.

- javatpoint (s. f.). (4 diciembre 2021) Theory of Automata. Disponible en: https://www.javatpoint.com/theory-of-automata.

- Team, Q. for P. (s. f.). (3 diciembre 2021). PySide2: Python bindings for the Qt cross-platform application and UI framework (5.15.2) [C++, Python; MacOS :: MacOS X, Microsoft, Microsoft :: Windows, POSIX, POSIX :: Linux]. Disponible en: https://www.pyside.org

Anexos

Pseudocódigo del algoritmo principal del programa

```
Algoritmo Tirar Dado //Algoritmo encargado de simular el lanzamiento del dado y la selección del estado actual en el autómata finito

    Definir random, estado, i Como Entero //random almacena un valor aleatorio entre 1 y 6, estado nos señala en qué estado actual estamos e i es una variable contador.

    //matriz de como estructura de datos bidimensional de la forma (x, y) = z, x = estado actual, y = símbolo de entrada, estado siguiente

    matriz = {

        (0, 1) : 1, (4, 0) : 10,  (9, 2)  : 1,

        (1, 1) : 2, (7, 0) : 12, (13, 2) : 5,

        (2, 1) : 3, (8, 0) : 17, (16, 2) : 13,

        (3, 1) : 4, (19, 0) : 22, (18, 2) : 9,

        (4, 1) : 5,                    (23, 2) : 15,

        (5, 1) : 6,

        (6, 1) : 7,

        (7, 1) : 8,

        (8, 1) : 9,

        (9, 1) : 10,

        (10, 1) : 11,

        (11, 1) : 12,

        (12, 1) : 13,

        (13, 1) : 14,

        (14, 1) : 15,

        (15, 1) : 16,

        (16, 1) : 17,
```

```
            (17, 1) : 18,
            (18, 1) : 19,
            (19, 1) : 20,
            (20, 1) : 21,
            (21, 1) : 22,
            (22, 1) : 23,
            (23, 1) : 24,
            (24, 1) : 24,
}
    random <- azar(1, 6) //Se genera un valor aleatorio comprendido entre el rango [1, 6], que simula el lanzamiento de un dado.
    Para j = 0 Hasta random Con Paso 1 Hacer //Iteración desde 0 hasta el valor de random
        autómata->agregarSimboloEntrada(1) //Por cada iteración, se agrega un símbolo de entrada al atributo autómata
    Imprimir “Obtuviste”, random //Se imprime el valor obtenido
    Mientras i < longitud(automata->getSimbolosEntrada) Hacer //MIentras el contador no supere el número de símbolos de entrada presentes en el autómata, se realiza la siguiente iteración
        estado <- matriz[(automata->getEstadoActual(), automata->getSimboloEntrada(i))] //Se obtiene el estado siguiente accediendo a la matriz bidimensional mediante los índices del estado actual presente en el autómata y el símbolo de entrada del mismo.
        dibujar_ficha() //Apartado gráfico: función para dibujar la ficha en el tablero.
        automata->setEstadoActual(estado) //Se coloca como estado actual en el autómata, aquel que fue obtenido de la matriz bidimensional.
        Imprimir “Estado actual”, automata->getEstadoActual() //Se imprime el estado actual en el que se encuentra el autómata finito.
        i = i + 1 //Se incrementa la variable i
    Si automata->getSimboloEntrada() == 4 Hacer
        automata->agregarSimboloEntrada(0)
    Si automata->getSimboloEntrada() == 7 Hacer
        automata->agregarSimboloEntrada(0)
    Si automata->getSimboloEntrada() == 8 Hacer
        automata->agregarSimboloEntrada(0)
```

```
Si automata->getSimboloEntrada() == 9 Hacer
        automata->agregarSimboloEntrada(2)
Si automata->getSimboloEntrada() == 13 Hacer
        automata->agregarSimboloEntrada(2)
Si automata->getSimboloEntrada() == 16 Hacer
        automata->agregarSimboloEntrada(2)
Si automata->getSimboloEntrada() == 18 Hacer
        automata->agregarSimboloEntrada(2)
Si automata->getSimboloEntrada() == 19 Hacer
        automata->agregarSimboloEntrada(0)
Si automata->getSimboloEntrada() == 23 Hacer
        automata->agregarSimboloEntrada(2)
mostrar_alertas() //Apartado gráfico: notificar al usuario si ha caído en una serpiente, escaleras; o indicar la victoria.
```