Planteamiento del problema.

El problema a resolver para este proyecto será mostrar cómo se encuentran presentes los autómatas finitos deterministas en el mundo de los videojuegos, usando como ejemplo un videojuego clásico llamado “Pacman”.

Objetivo.

El propósito central del proyecto será recrear el videojuego “Pacman” con la ayuda del software de desarrollo de videojuegos llamado “Game Maker”.
Esto con la finalidad de ayudarnos a poder observar con algo gráfico que los autómatas finitos deterministas se encuentran presentes en el campo del gaming desde hace décadas.
A su vez, la recreación de este famoso videojuego nos ayudará a poder comprender y plasmar cómo sería el autómata finito que se encontraría en este, siendo capaces de comprender a fondo este mecanismo de juego tan particular que, a pesar de ser algo relativamente sencillo, esconde un gran esfuerzo detrás de sí y se torna más complejo mientras más se profundiza en él.

Justificación.

Esta idea puede ayudar a los alumnos y personas en general a comprender un poco mejor el mundo implícito de la programación y la teoría de la computación presente no solo en “Pacman” sino en todo el campo de los videojuegos, ayudando también a tener ideas cercanas acerca del funcionamiento de otros campos como podrían ser el desarrollo web y aplicaciones para teléfonos celulares y computadores.
Esto sería posible gracias al comportamiento primitivo que podía tener el jugador en la época original de “Pacman”, siendo limitado por los recursos de aquellos años, el cual ahora nos permite presenciar de forma más sencilla el funcionamiento de los autómatas finitos en estos, a diferencia de los videojuegos actuales que gracias a los avances tecnológicos se han vuelto cada vez más complejos pero que, sin embargo, aún hacen uso de estas herramientas en lo más profundo de su codificación.

Marco teórico:

Autómatas finitos no deterministas:
La definición formal de AFND se basa en la consideración de que a menudo según los algoritmos de transformación de expresiones y gramáticas regulares a AF terminan obteniéndose autómatas con transiciones múltiples para un mismo símbolo o transiciones vacías. Independientemente que sean indeseables, sobre todo para la implementación material, fundamentalmente mecánica, de los autómatas finitos, son imprescindibles durante la modelación

de analizadores lexicográficos de los elementos gramaticales de los lenguajes de programación, llamados tókens, como literales numéricos, identificadores, cadenas de texto, operadores, etc.
Los AFND son definiciones no tan deseables dentro de los lenguajes regulares porque dificultan su implementación tanto mecánica como informática; aunque en la mayoría de las transformaciones a lo interno de los LR (expresiones regulares a AF, gramáticas regulares a AF) conducen a AFND. Los AFND, por tanto, son imprescindibles en el análisis lexicográfico y el diseño de los lenguajes de programación.
Haciendo la analogía con los AFDs, en un AFND puede darse cualquiera de estos dos casos:

- Que existan transiciones del tipo $\delta(q,a)=q1$ y $\delta(q,a)=q2$ siendo $q1 \neq q2$;
- Que existan transiciones del tipo, $\delta(q,\varepsilon)$ siendo un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Cuando se cumple el segundo caso, se dice que el autómata es un autómata finito no determinista con transiciones vacías o transiciones ε (abreviado AFNDε). Estas transiciones permiten al autómata cambiar de estado sin procesar ningún símbolo de entrada.
Sea un autómata finito definido por la quíntupla $A=(Q,\Sigma,q0,\delta,F)$ donde:

Q: es el conjunto finito de estados

Σ: es un alfabeto finito

$q0 \in Q$: es el estado inicial

$F \subseteq Q$: es un conjunto de estados finales o de aceptación

$\delta: Qx(\sum \cup \{\varepsilon\}) \rightarrow P(Q)$: cuando contiene transiciones vacías

$\delta: Qx\sum \rightarrow P(Q)$ ó $\delta: \{qi,x,qj \mid qi \ 〚,q〛 j \in Q; x \in \sum\}$ - (del estado qi mediante el terminal x se va a qj), es la función de transición.

Esto significa que los autómatas finitos deterministas son un caso particular de los no deterministas, puesto que Q pertenece al conjunto P (Q).
Se dice que es un autómata finito no determinista (AFND) si y sólo si existen en δ al menos una de las siguientes transiciones:

AFND`s pueden presentarse mediante tablas de transiciones.

En las filas los estados.

Si el estado es final lo antecederá un *.

Si el estado es inicial lo marcaremos con una →

En las columnas se pondrán los símbolos del alfabeto de entrada y se añadirá una columna adicional para ε .

En la intersección (celdas de la matriz) se indicarán las transiciones.

AFND en Pacman:
Este videojuego es bien conocido tanto por aquellos que se encuentran inmersos en el mundo de los videojuegos como por aquellos que no se encuentran demasiado relacionados con este.
Un videojuego aparentemente sencillo pero que resultó ser demasiado innovador en la época en la cual fue creado.
Lo que buscamos aquí es recrear lo más posible y con los conocimientos que tenemos ese videojuego para ayudarnos a demostrar la forma en que trabajan los autómatas finitos en conjunto con el jugador en esta obra.
Realmente no es nada complicado comprender el funcionamiento de estos AFND en este videojuego en particular pues los estados en los que se puede encontrar el jugador, quien será el foco de estudio en esta ocasión, son realmente limitados y sencillos de entender.
Para esto solamente se necesitará entender lo básico acerca de la teoría de autómatas finitos no determinas.
De esta forma esperamos que se pueda comprender fácilmente y por cualquier persona el funcionamiento de los AFND no solo en Pacman sino en diversos videojuegos y aplicaciones electrónicas en general, ayudando a otras personas a poder realizar tanto actividades teóricas como prácticas acerca de este tema.

Metodología:

Bien, pues una vez explicado lo anterior, procederemos a explicar de la manera más breve y sencilla de entender el funcionamiento de este proyecto.
Para comprender el comportamiento de este autómata le daremos los siguientes nombres a los seis estados presentes en este:

- Estado q0: Inicio del juego.
- Estado q1: Animación de Pacman moviéndose hacia la derecha.
- Estado q2: Animación de Pacman moviéndose hacia la izquierda.
- Estado q3: Animación de Pacman moviéndose hacia arriba.
- Estado q4: Animación de Pacman moviéndose hacia abajo.
- Estado q5: Animación de Pacman muriendo.
- Estado q6: Ganar.

Como es bien sabido, Pacman inicia cada nivel en un punto específico del mapa, siendo totalmente inerte hasta que el jugador decida hacia dónde desea llevar a la pequeña bola amarilla.
Este vendría siendo el estado q0, donde Pacman no ha recibido ninguna

entrada aún.
Después del estado q0 vendrían seis estados más, los cuales serían: Pacman yendo hacia arriba, Pacman yendo hacia abajo, Pacman yendo hacia la izquierda, Pacman yendo hacia la derecha derecha, Pacman muriendo y el fin del juego cuando el jugador gana el nivel.
El jugador podría mover a Pacman en cuatro direcciones distintas, dependiendo de la entrada que el autómata reciba.
Para esto también debemos aclarar cuáles son las entradas que este puede recibir:

- R: Flecha derecha.
- L: Flecha izquierda.
- U: Flecha arriba.
- D: Flecha abajo.
- P: Fin de puntos.
- C: Colisión con fantasmas (Cuando estos pueden dañar al jugador).
- A: Reaparición del jugador.

De esta forma, una vez que el juego inicia con el estado q0, el autómata puede recibir cualquier entrada de las ya propuestas, con excepción de la entrada “P”, la cual solo será posible una vez que el jugador logre obtener todos los puntos. Para poder hacer esto más claro, podemos imaginar que Pacman tiene la posibilidad de moverse en cualquiera de las cuatro direcciones posibles, aunque eso signifique colisionar incluso con alguna pared del nivel. Además de eso, Pacman puede colisionar con un fantasma que logre llegar hacia él, lo que permitiría dar paso a la entrada “A”, que se encargaría de hacer que el nivel se repita, dando inicio una vez más al estado q0.
Esto es solo imaginando una de las tantas combinaciones que el autómata puede recibir, siendo capaz de intercalar entre distintos estados una y otra vez hasta que logre llegar al estado q5 o q6, a los cuales puede acceder solamente si obtiene todos los puntos distribuidos por el mapa o si por el contrario, el jugador es alcanzado por uno de los fantasmas y muere.
Para poder comprender mejor las posibilidades del autómata se diseñó una tabla de transición la cual se podrá ver a continuación, donde se pueden ver las entradas que puede recibir cada estado propuesto y cómo eso lo lleva a un estado distinto, donde podrá recibir otra entrada una y otra vez.

Código para los movimientos de Pacman:

```
//Crear

// Iniciar sprite

sprite_index = spr_pacman_right;

image_speed = 0;

image_index = 0;
```

```
// Variable para la velocidad
v = 4;
/// Movimientos

if(global.Golpe) or audio_is_playing(snd_intro) exit;
// Movimientos del personaje

if keyboard_check(vk_right) and place_snapped(32,32)
     {
     direction = 0;
     speed = v;//cambiar velocidad
     }

if keyboard_check(vk_left) and place_snapped(32,32)
     {
     direction = 180;
     speed = v;
     }

if keyboard_check(vk_up) and place_snapped(32,32)
     {
     direction = 90;
     speed = v;
     }

if keyboard_check(vk_down) and place_snapped(32,32)
     {
     direction = 270;
     speed = v;
     }
```

```
//verificar direccion y velocidad para cambiar el sprite
if speed > 0
     {
          image_speed = 1;
     }
else
     {

          image_speed = 1;


     }

switch(direction)
     {
          case 0:
          sprite_index = spr_pacman_right;
          break;

          case 90:
          sprite_index = spr_pacman_up;
          break;

          case 180:
          sprite_index = spr_pacman_left;
          break;

          case 270:
          sprite_index = spr_pacman_down;
          break;
     }
```

Pruebas y resultados:

δ	R	L	U	D	P	A	C
q_0	q_1	q_2	q_3	q_4	q_6	-	q_5
q_1	q_1	q_2	q_3	q_4	q_6	-	q_5
q_2	q_1	q_2	q_3	q_4	q_6	-	q_5
q_3	q_1	q_2	q_3	q_4	q_6	-	q_5
q_4	q_1	q_2	q_3	q_4	q_6	-	q_5
q_5	-	-	-	-	-	q_0	-
q_6	-	-	-	-	-	q_0	-

$q_0 = Inicio\ del\ juego$

$q_1 = Animación\ de\ PacMan\ hacia\ la\ derecha$

$q_2 = Animación\ de\ PacMan\ hacia\ la\ izquierda$

$q_3 = Animación\ de\ PacMan\ hacia\ arriba$

$q_4 = Animación\ de\ PacMan\ hacia\ abajo$

$q_5 = Animación\ de\ PacMan\ muriendo$

$q_6 = Ganar$

$R = Tecla\ de\ flecha\ derecha$

$I = Tecla\ de\ flecha\ izquierda$

$U = Tecla\ de\ flecha\ arriba$

$D = Tecla\ de\ flecha\ abajo$

$P = Fin\ de\ puntos/\ comida$

$C = Colisión\ con\ fantasma\ (fantasma\ inmune)$

$A = Reaparición$

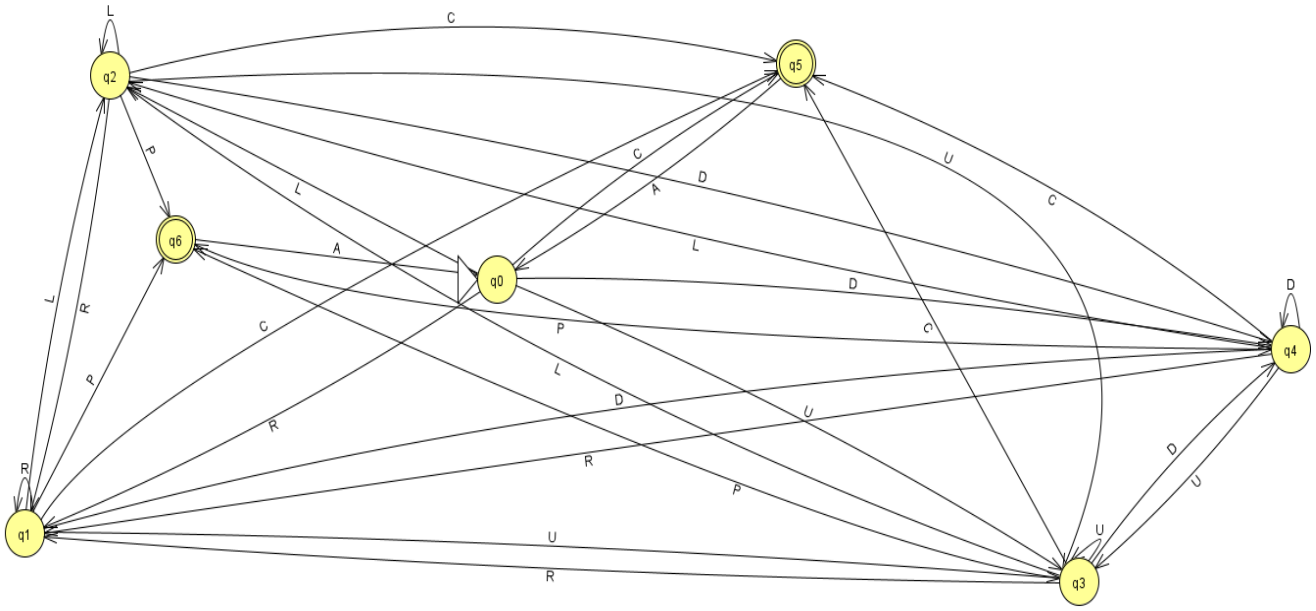
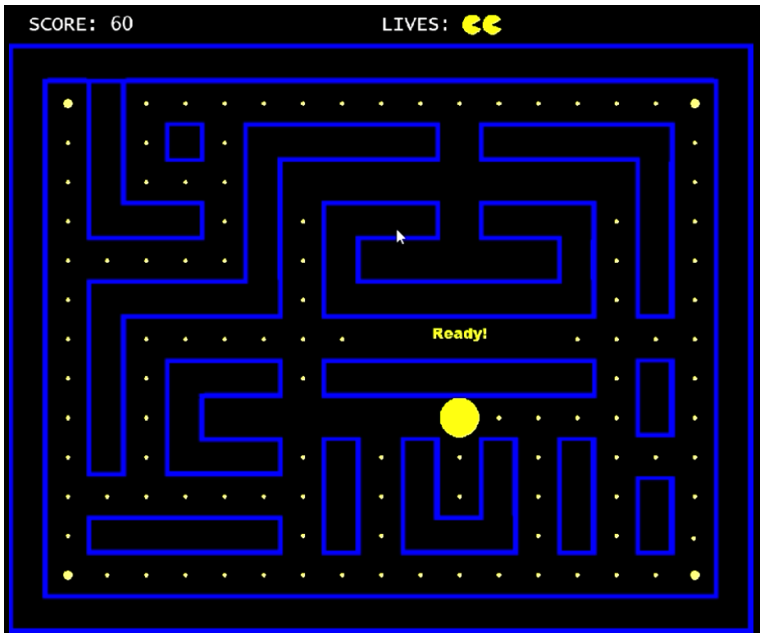
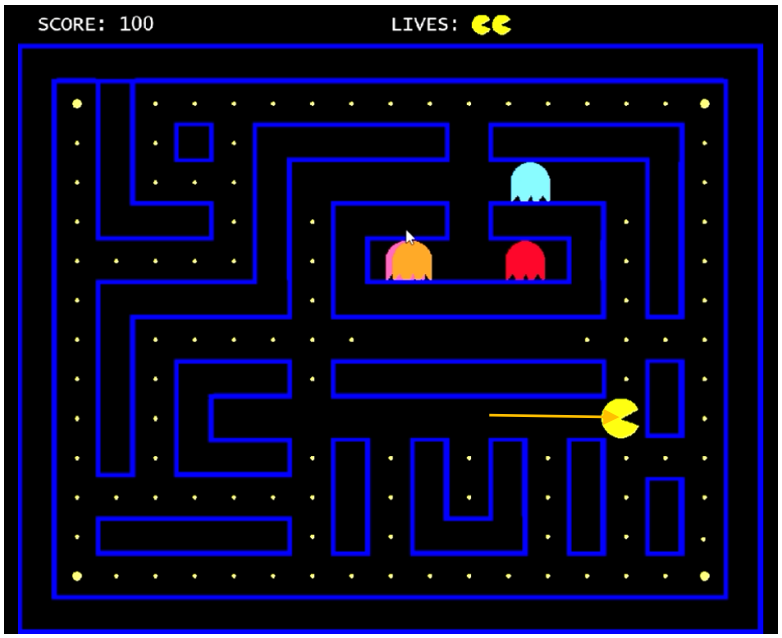


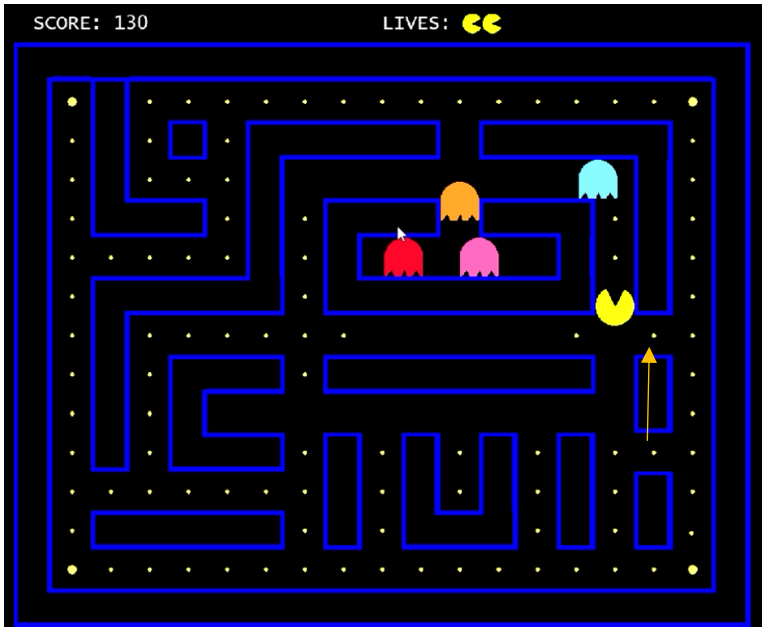
Ilustración 1Diagrama de transición



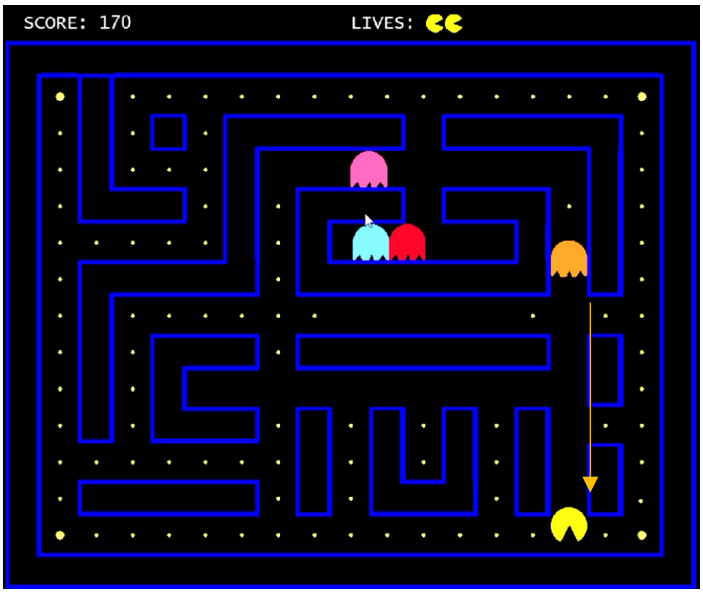
En esta imagen se puede apreciar que PacMan se encuentra en el estado inicial / $q_0(Inicio\ de\ juego)$.



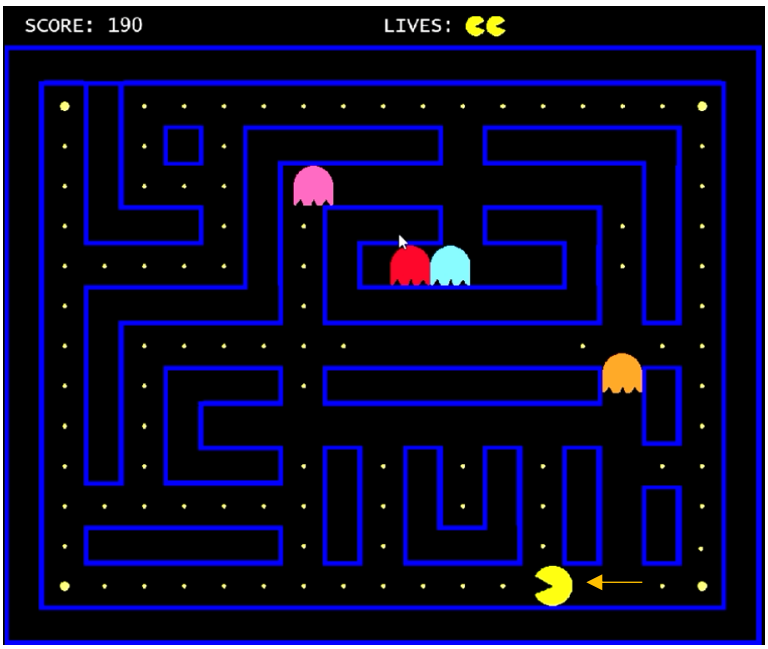
Se recibe como entrada la tecla de flecha derecha (R) *y se actualiza el estado a* $q_1(Animación\ de\ PacMan\ hacia\ la\ derecha)$.



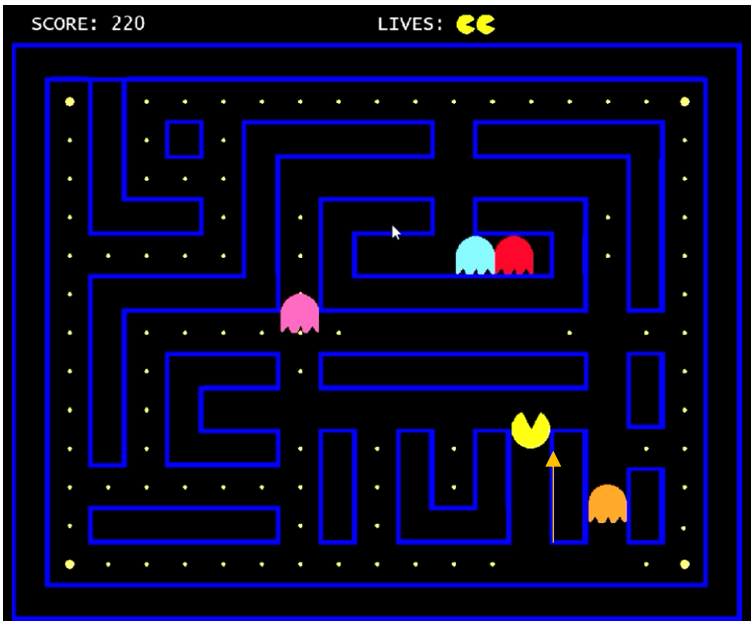
Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



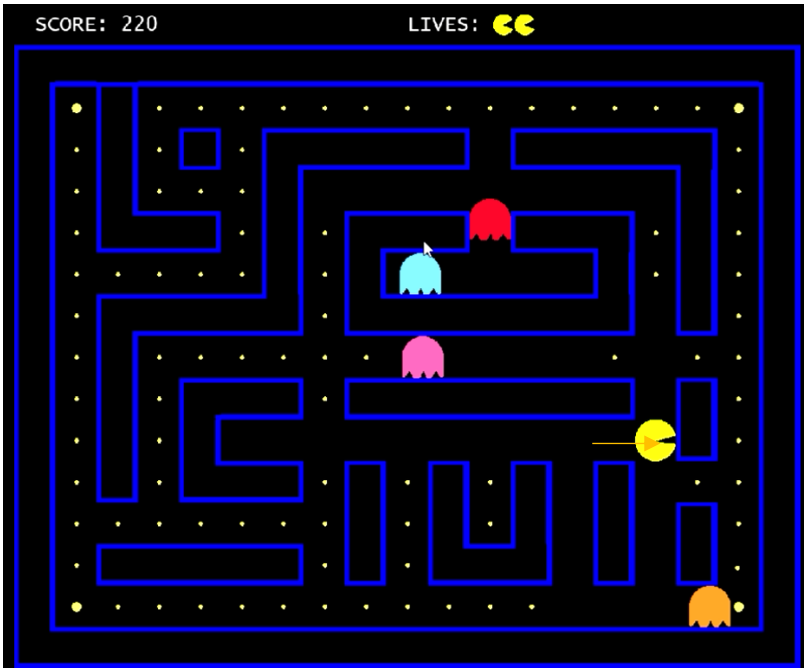
Se recibe como entrada la tecla de flecha hacia abajo (D) y cambia de estado a $q_4(Animación\ de\ PacMan\ hacia\ abajo)$.



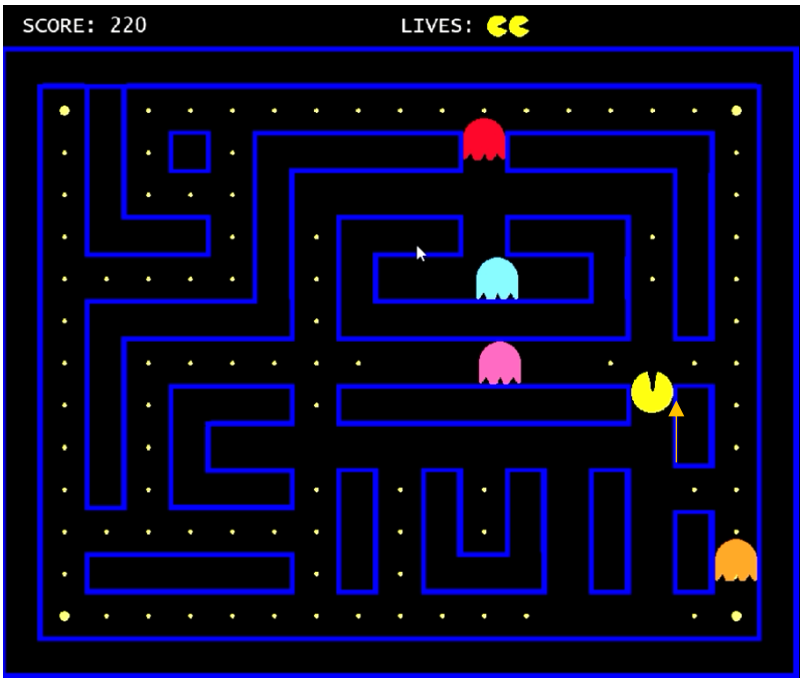
Se recibe como entrada la tecla de flecha hacia la izquierda (L) y cambia de estado a $q_2(Animación\ de\ PacMan\ hacia\ la\ izquierda)$.



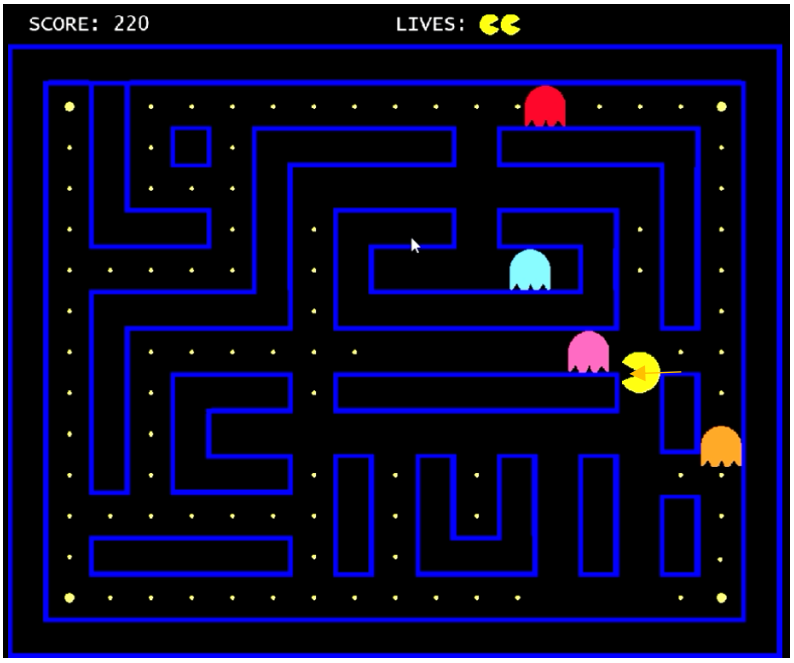
Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



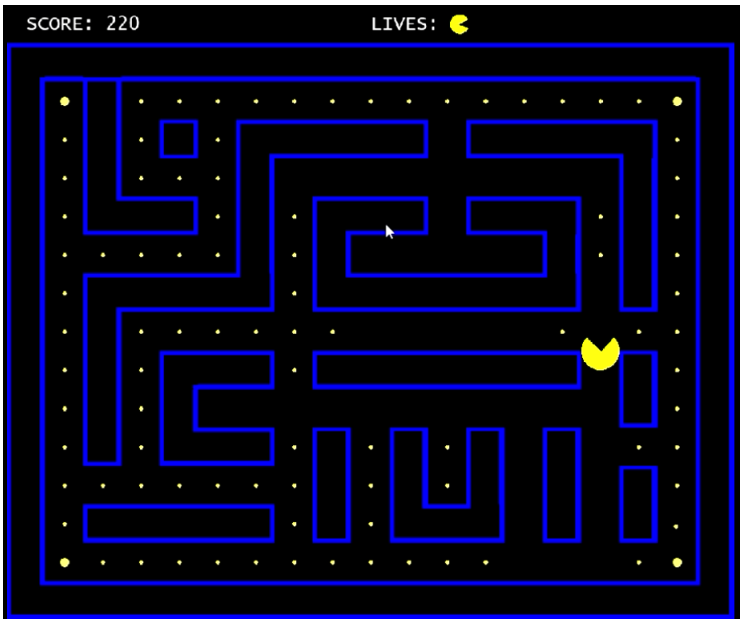
Se recibe como entrada la tecla de flecha hacia la derecha (R) y cambia de estado a $q_1(Animación\ de\ PacMan\ hacia\ la\ derecha)$.



Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



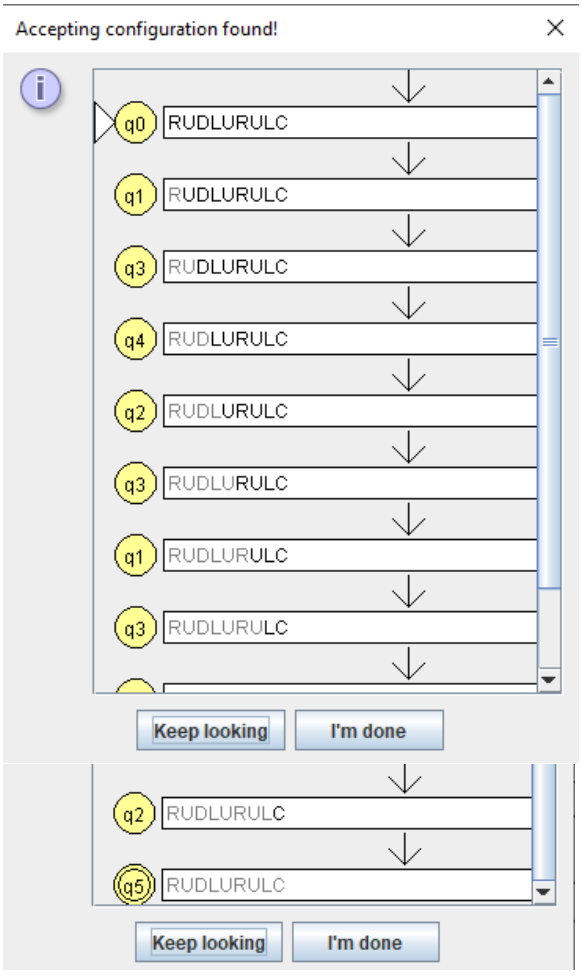
Se recibe como entrada la tecla de flecha hacia la izquierda (L) y cambia de estado a $q_2(Animación\ de\ PacMan\ hacia\ la\ izquierda)$.



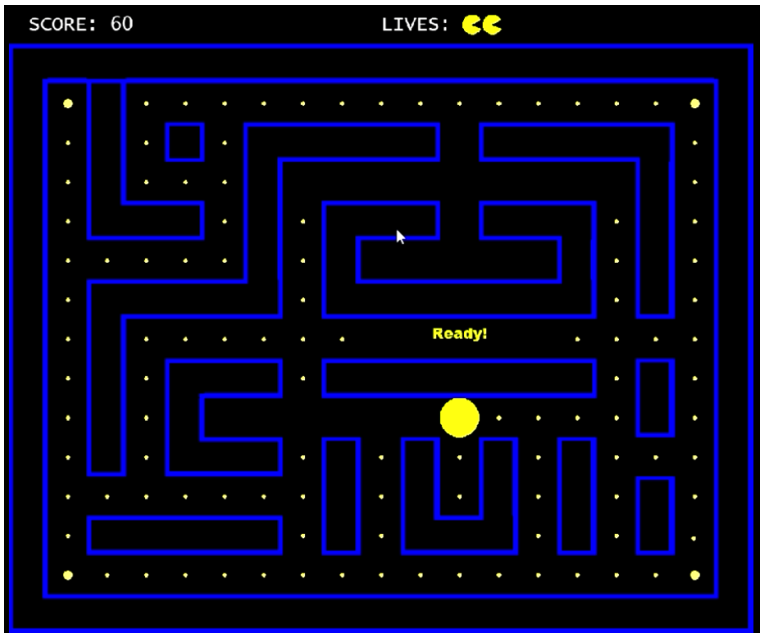
Aunque no se aprecie muy bien, se recibe como entrada un choque contra un fantasma (C) y cambia de estado a $q_5(Animación\ de\ PacMan\ muriendo)$.

Cadena generada: RUDLURULC

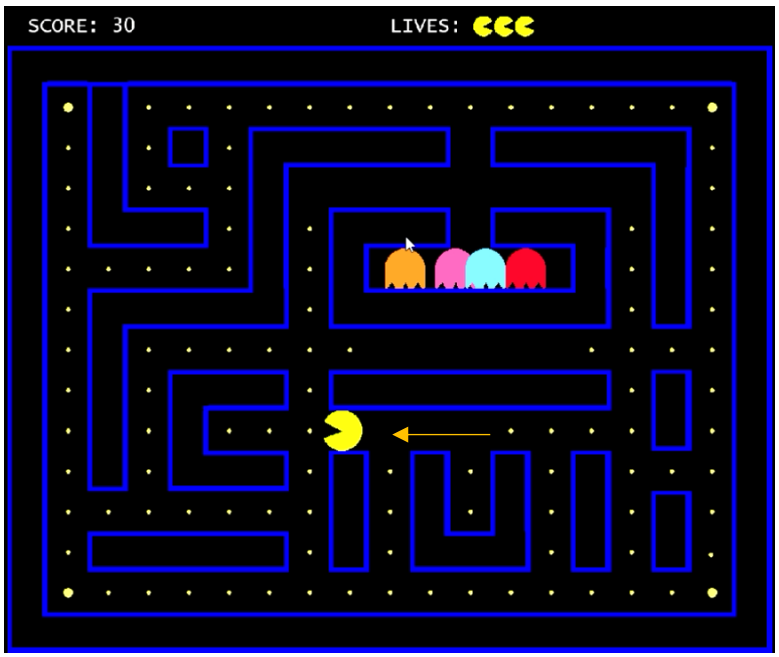
Probando la aceptación de la cadena con JFLAP:



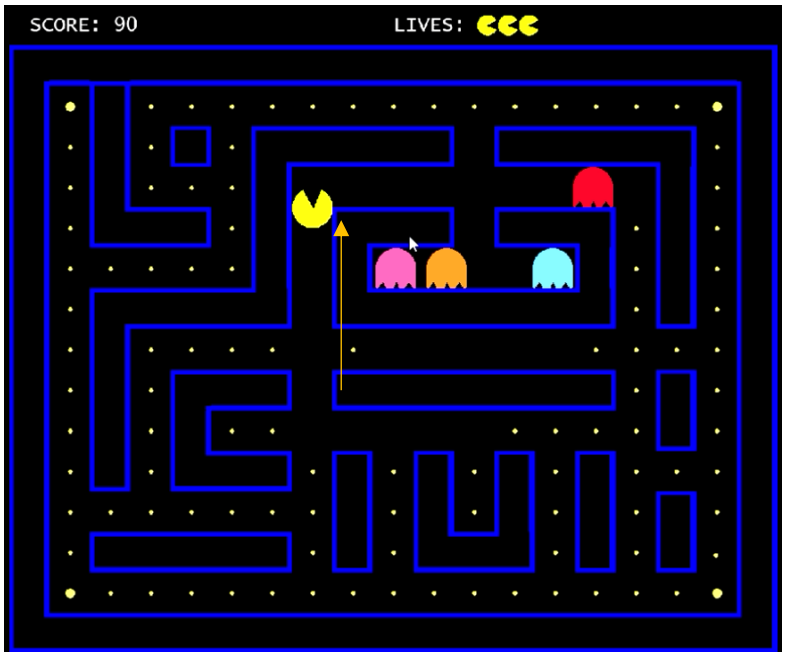
Input	Result
RUDLURULC	Accept



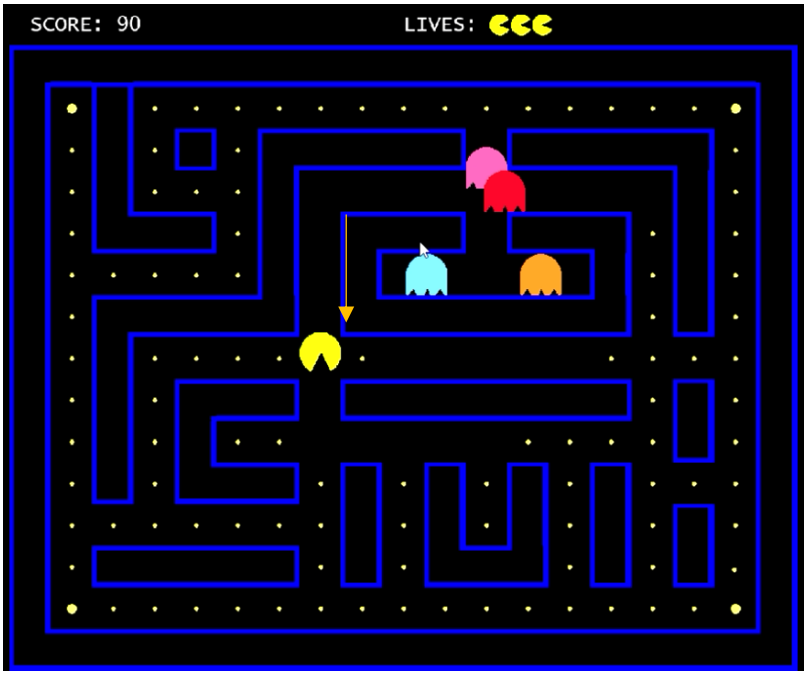
En esta imagen se puede apreciar que PacMan se encuentra en el estado inicial / $q_0(Inicio\ de\ juego)$.



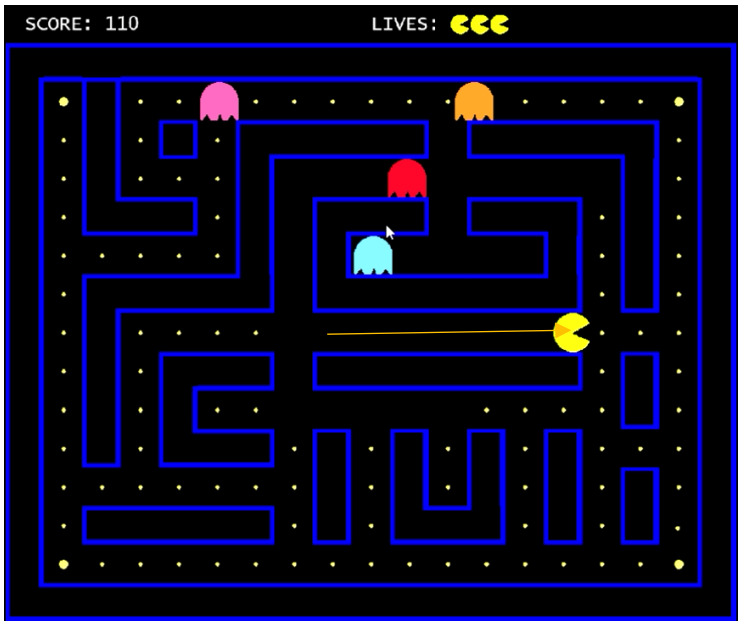
Se recibe como entrada la tecla de flecha hacia la izquierda (L) y cambia de estado a $q_2(Animación\ de\ PacMan\ hacia\ la\ izquierda)$.



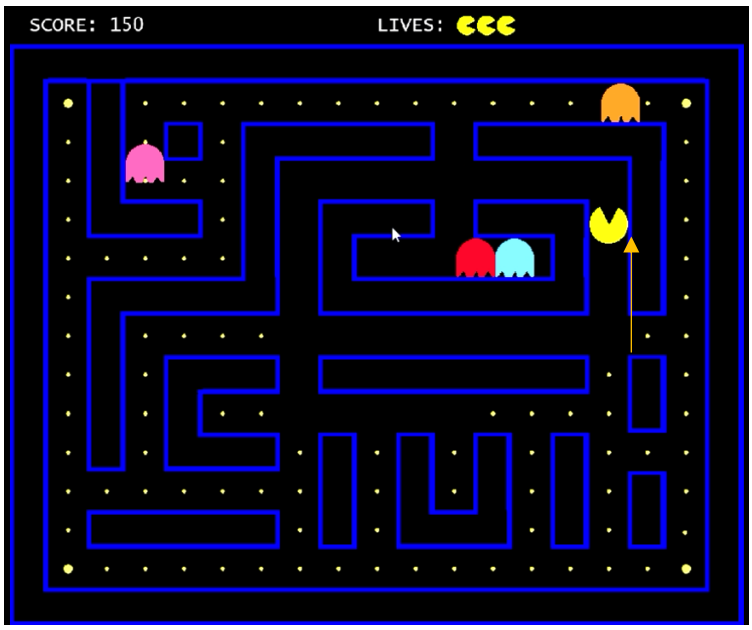
Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



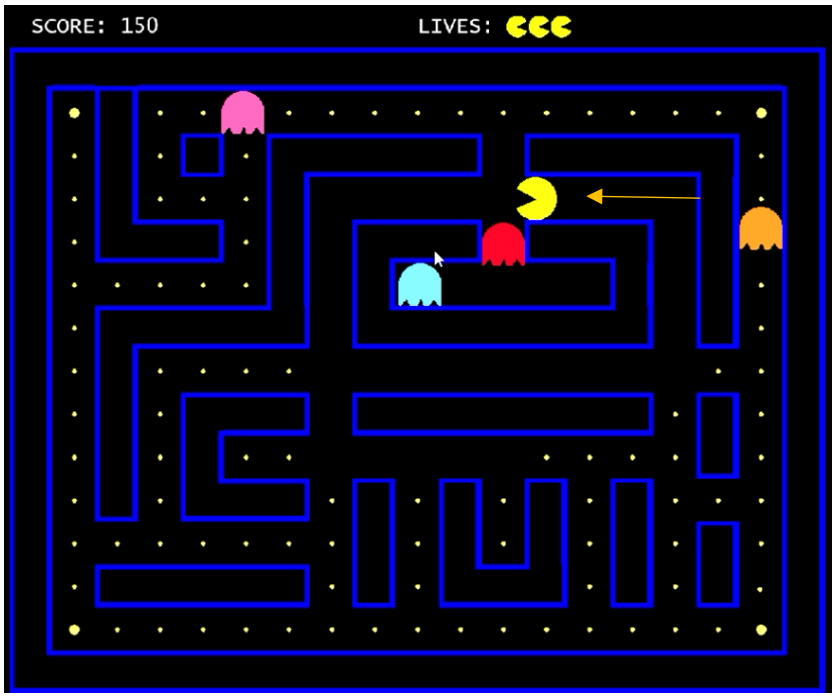
Se recibe como entrada la tecla de flecha hacia abajo (D) y cambia de estado a $q_4(Animación\ de\ PacMan\ hacia\ abajo)$.



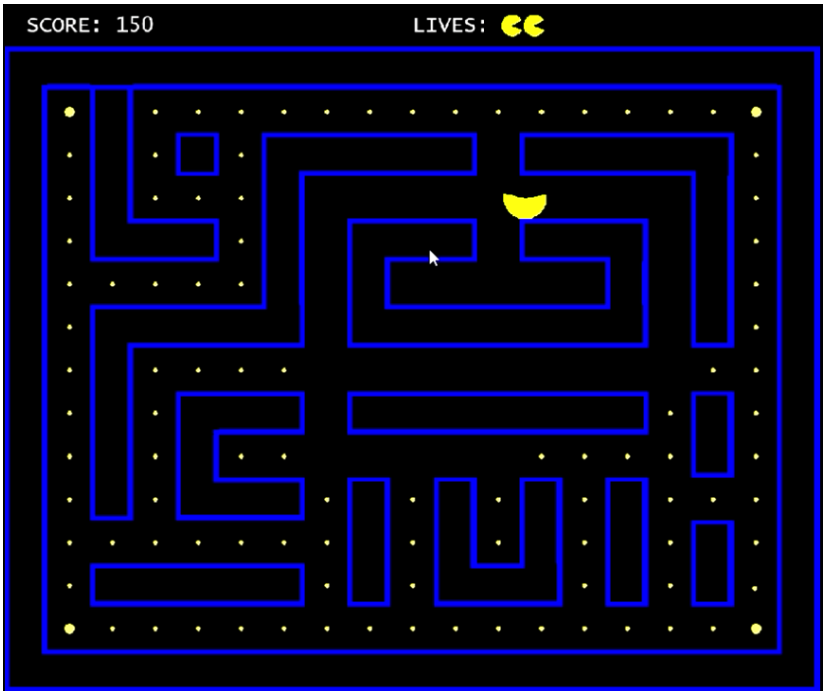
Se recibe como entrada la tecla de flecha hacia la derecha (R) y cambia de estado a $q_1(Animación\ de\ PacMan\ hacia\ la\ derecha)$.



Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



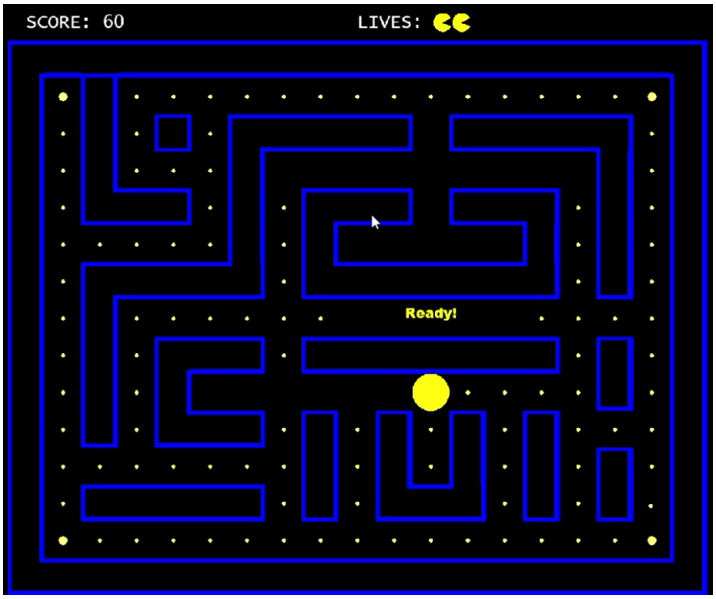
Se recibe como entrada la tecla de flecha hacia la izquierda (L) y cambia de estado a $q_2(Animación\ de\ PacMan\ hacia\ la\ izquierda)$.

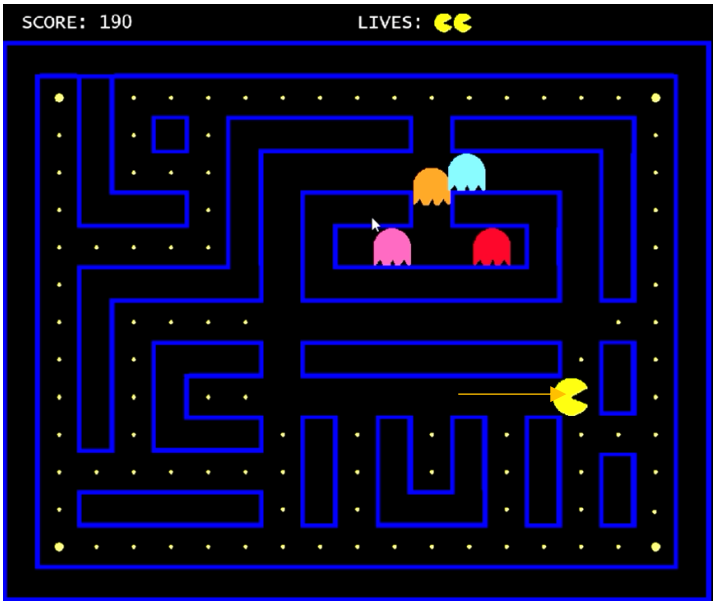


Se recibe como entrada un choque contra un fantasma (C) y cambia de estado a $q_5(Animación\ de\ PacMan\ muriendo)$.

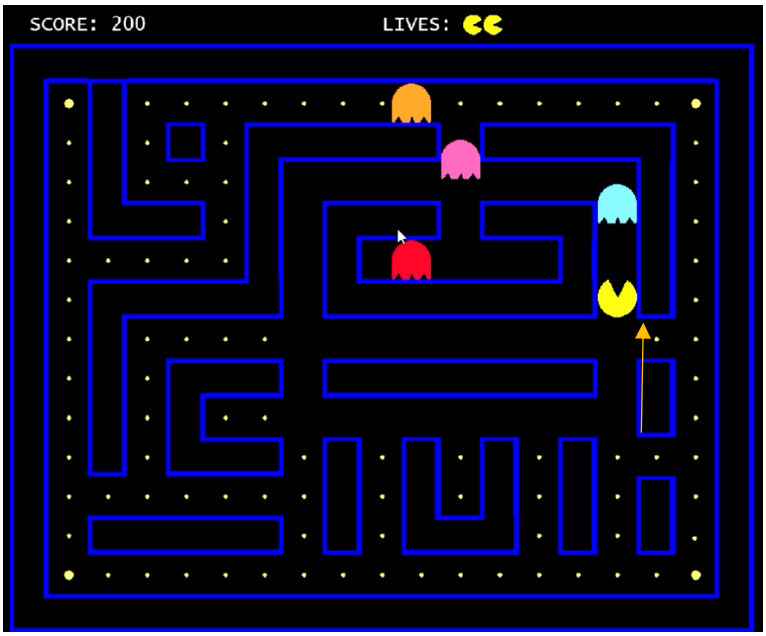
Ahora, supongamos que hemos recorrido la cadena anterior (LUDRULC) y decidimos seguir con el juego. Nuestra cadena

actual sería: LUDRULCA, donde la entrada A sería continuar con el juego o reaparición, esta entrada nos llevará al estado $q_0(Inicio\ de\ juego)$.

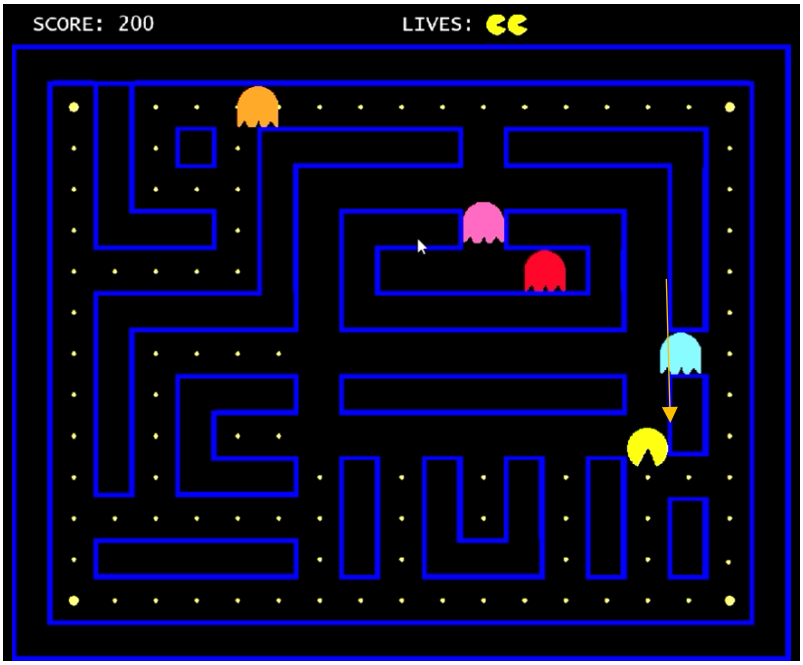




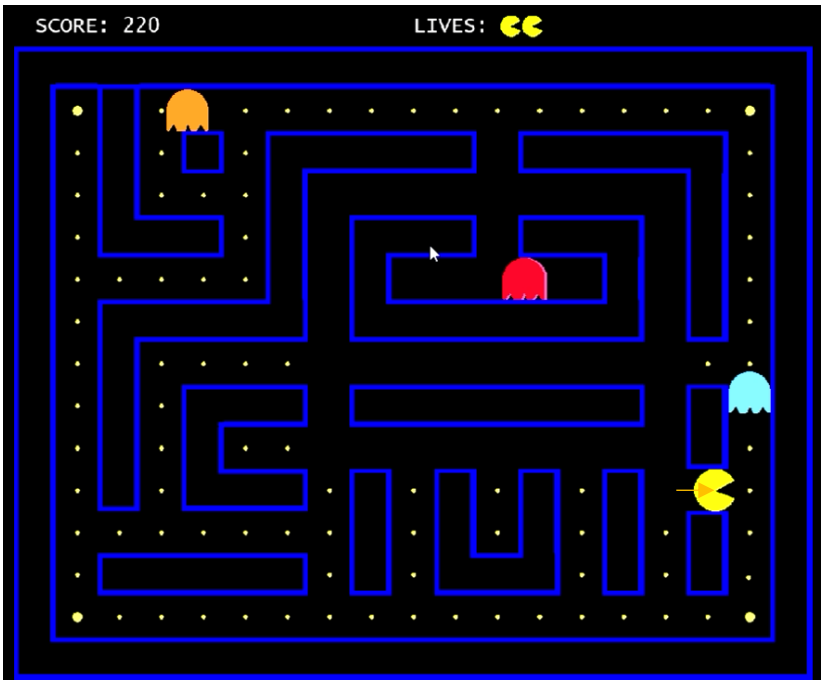
Se recibe como entrada la tecla de flecha hacia la derecha (R) y cambia de estado a $q_1(Animación\ de\ PacMan\ hacia\ la\ derecha)$.



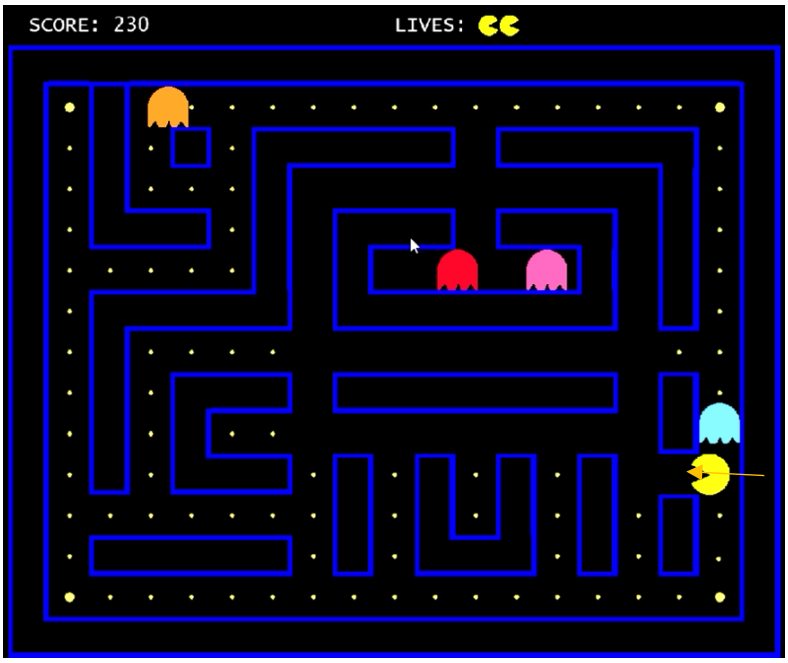
Se recibe como entrada la tecla de flecha hacia arriba (U) y cambia de estado a $q_3(Animación\ de\ PacMan\ hacia\ arriba)$.



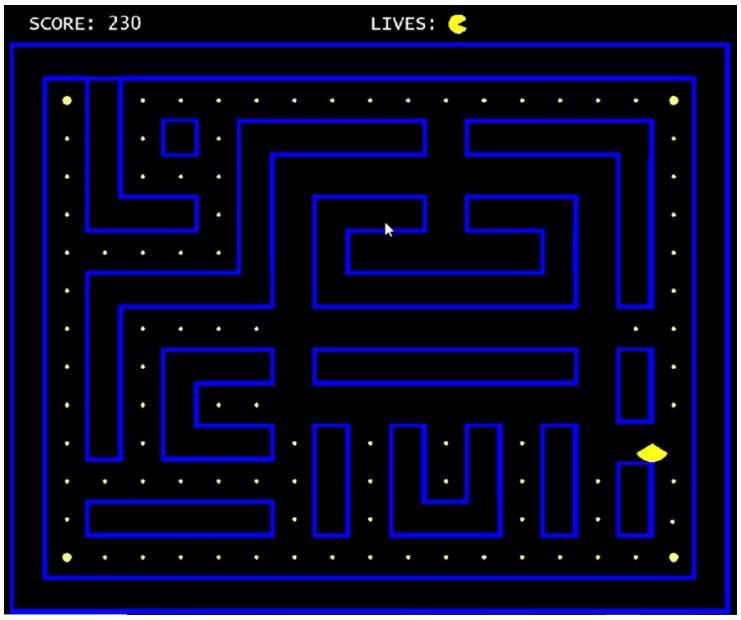
Se recibe como entrada la tecla de flecha hacia abajo (D) y cambia de estado a $q_4(Animación\ de\ PacMan\ hacia\ abajo)$.



Se recibe como entrada la tecla de flecha hacia la derecha (R) y cambia de estado a
$q_1(Animación\ de\ PacMan\ hacia\ la\ derecha)$.



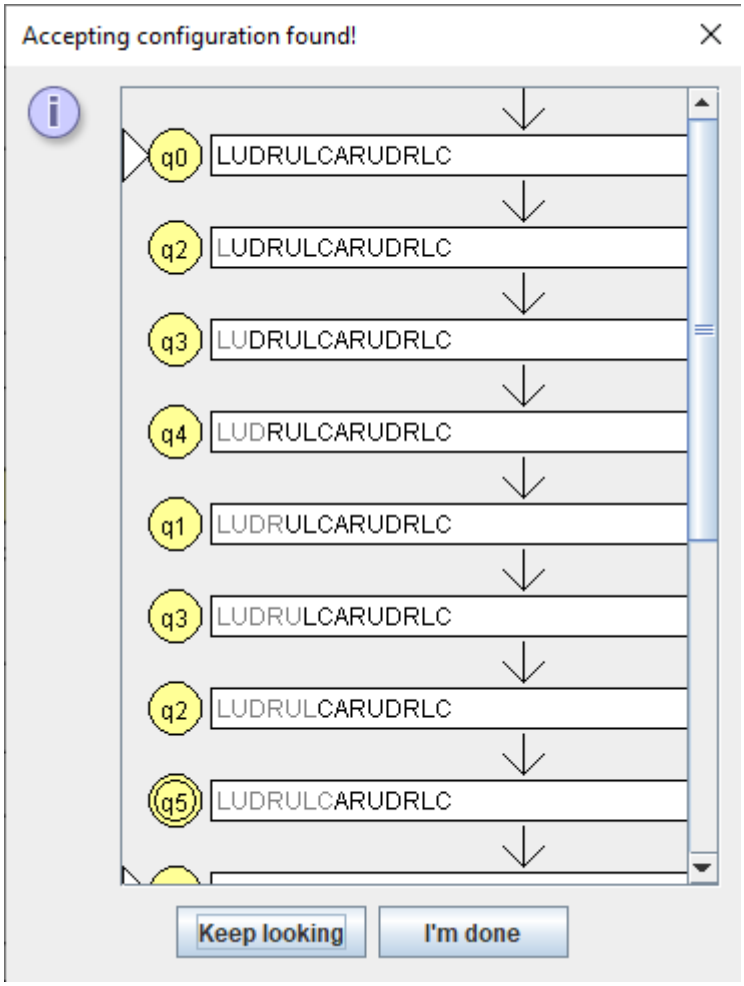
Se recibe como entrada la tecla de flecha hacia la izquierda (L) y cambia de estado a
$q_2(Animación\ de\ PacMan\ hacia\ la\ izquierda)$.



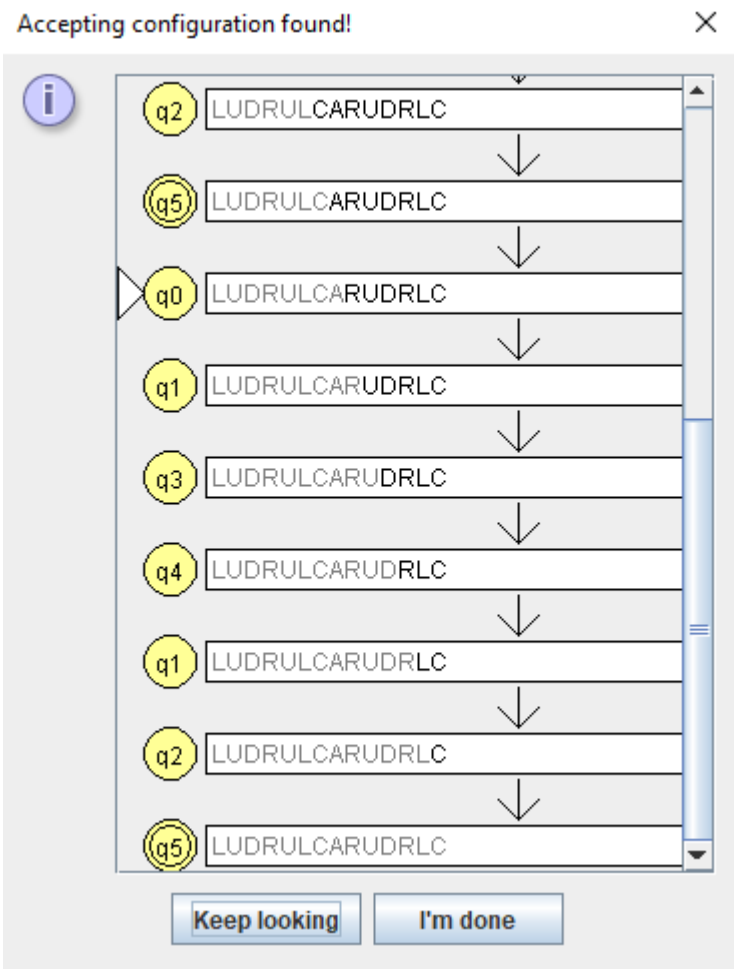
Se recibe como entrada un choque contra un fantasma (C) y cambia de estado a $q_5(Animación\ de\ PacMan\ muriendo)$.

Podemos considerar este recorrido de caracteres como finalizado, habiendo llegado al estado terminal $q_5(Animación\ de\ PacMan\ muriendo)$.

Probemos que la cadena LUDRULCARUDRLC sea aceptada por el autómata usando JFLAP:



Podemos observar que la cadena LUDRULC, mostrada en el segundo ejemplo, fue aceptada al llegar al estado q_5.



Después de recorrer esa cadena, decidimos continuar con el juego y así llegamos al estado q_0.

Finalmente llegamos al estado q_5 aceptando la cadena LUDRULCARUDRLC.

Conclusión:

Al probar el autómata haciendo un recorrido real por el juego, es muy sencillo probar que, al tener un registro de las entradas, podemos obtener una cadena que es aceptada por el autómata. Sin embargo, si la entrada “fin de puntos” no puede ser dada hasta que en realidad se cumpla que no existan más puntos por comer, y es algo que el autómata no puede detectar, pero el programa sí.

Finalmente, decidimos que esta entrada sólo se cumplirá cuando anteriormente se hayan recorrido casi todo el mapa del juego y se detecte que no haya más puntos en éste.

También pudimos haber considerado las tres vidas en el juego, pero esto complicaría más el diagrama y haría a lo sumo dos copias del diagrama de transiciones que utilizamos (ya que el autómata no tiene la capacidad de detectar cuántas vidas se han perdido).

El autómata y el programa fueron modificados varias veces para que cumpliera con la definición y coincidieran entre sí. Podríamos considerar que el resultado final es una buena propuesta de autómata finito no determinista.

Referencias bibliográficas:

- Chakraborty, Samarjit (17 de marzo de 2003). «Formal Languages and Automata Theory. Regular Expressions and Finite Automata». *Computer Engineering and Networks Laboratory. Swiss Federal Institute of Technology (ETH) Zürich* (en inglés): Consultado el 20 de mayo de 2020.