Introducción a las Ciencias de la Computación
Tarea 8
16 de Noviembre de 2016

1 Objetivo

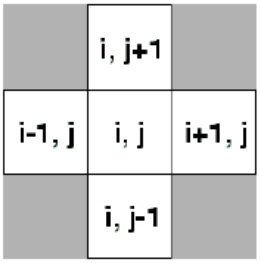
Que el alumno desarrolle una aplicación utilizando los conocimientos adquiridos en las prácticas anteriores y así pueda reafirmar su entendimiento de estos.

2 Desarrollo del proyecto

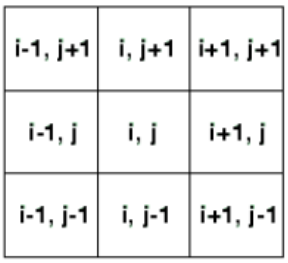
Un autómata celular (AC) es un objeto matemático, compuesto de células dispuestas en una rejilla de una forma determinada. Cada celda en la rejilla se modifica en pasos discretos de tiempo, siguiendo un conjunto de reglas basadas en el estado de las celdas vecinas. Las reglas del autómata son aplicadas iterativamente a un número deseado de unidades de tiempo. El estado inicial de las celdas en la rejilla puede ser definido de manera determinada o de manera aleatoria.

Existen dos tipos de vecindades para calcular el estado próximo de una célula.

- En el modelo de Von Neumann cada estado de una célula del arreglo localizada en una posición (i,j) (en donde i es la columna y j es el renglón) en un tiempo t estará determinado por los valores de los estados en que se encuentran las células localizadas en las posiciones (i-1,j), (i+1,j), (i,j+1), (i,j-1).
- John Conway, descubrió un tipo de vecindad parecida a la de von Neumann, con la variante de que se toma en cuenta a los vecinos ubicados en las esquinas. A este tipo de vecindad se le conoce como vecindad Moore.



Vecindad de Von Neumann



Vecindad de Moore

3 El juego de la vida

El juego de la vida es el mejor ejemplo de un autómata celular, diseñado por el matemático británico John Horton Conway en 1970.

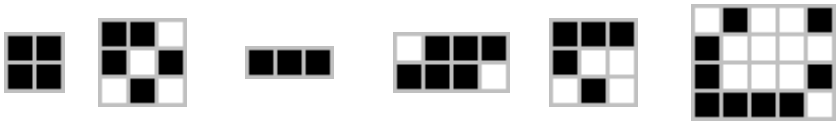
Desde su publicación, ha atraído mucho interés debido a la gran variabilidad de la evolución de los patrones. Se considera que la vida es un buen ejemplo de emergencia y auto organización. Es interesante para los científicos, matemáticos, economistas y otros observar cómo patrones complejos pueden provenir de la implementación de reglas muy sencillas. Para muchos aficionados, el juego de la vida sólo era un desafío de programación y una manera divertida de usar ciclos de la CPU. Para otros, sin embargo, el juego adquirió más connotaciones filosóficas. Desarrolló un seguimiento casi fanático a lo largo de los años 1970 hasta mediados de los 80.

En el juego de la vida el "tablero de juego" es una malla formada por cuadrados ("células") que se extiende por el infinito en todas las direcciones. Cada célula tiene su propia vecindad, que son las células que están próximas a ella, incluso en las diagonales. Las células tienen dos estados: están "vivas"(Color azul) o "muertas"(Color blanco) (o "encendidas" y "apagadas"). El estado de todas las células se tiene en cuenta para calcular el estado de las mismas al turno siguiente. Todas las células se actualizan simultáneamente.

Las transiciones dependen del número de células vecinas vivas:

- Una célula muerta con exactamente 3 células vecinas vivas "nace" (al turno siguiente estará viva).
- Una célula viva con 2 ó 3 células vecinas vivas sigue viva, en otro caso muere o permanece muerta (por "soledad" o "superpoblación").

Existen numerosos tipos de patrones que pueden tener lugar en el juego de la vida, como patrones estáticos "vidas estáticas", patrones recurrentes "osciladores" y patrones que se trasladan por el tablero "naves espaciales". Los ejemplos más simples de estas tres clases de patrones se muestran abajo. Las células vivas se muestran en negro y las muertas en blanco. Los nombres son más conocidos en inglés, por lo que también se muestra el nombre de estas estructuras en dicho idioma.

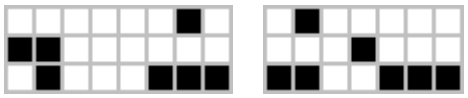


Bloque Barco Parpadeador Sapo Planeador Nave ligera

Block *Boat* *Blinker* *Toad* *Glider* *LWSS*

El bloque y el barco son vidas estáticas, el parpadeador y el sapo son osciladores y el planeador y la nave espacial ligera son naves espaciales que recorren el tablero a lo largo del tiempo.

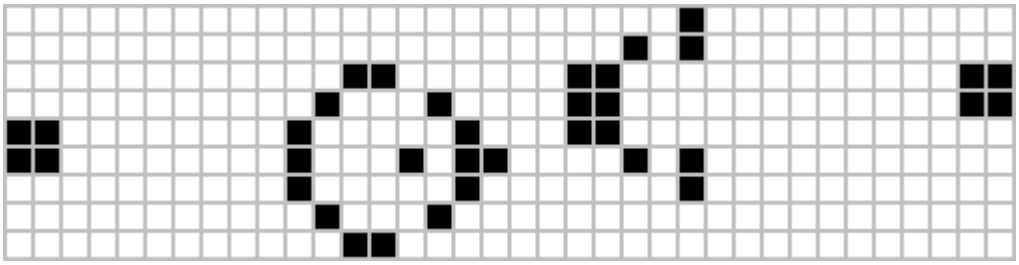
Los patrones llamados "Matusalenes" pueden evolucionar a lo largo de muchos turnos, o generaciones, antes de estabilizarse. El patrón "*Diehard*" desaparece después de 130 turnos, mientras que "*Acorn*" tarda 5206 turnos en estabilizarse en forma de muchos osciladores, y en ese tiempo genera 13 planeadores.



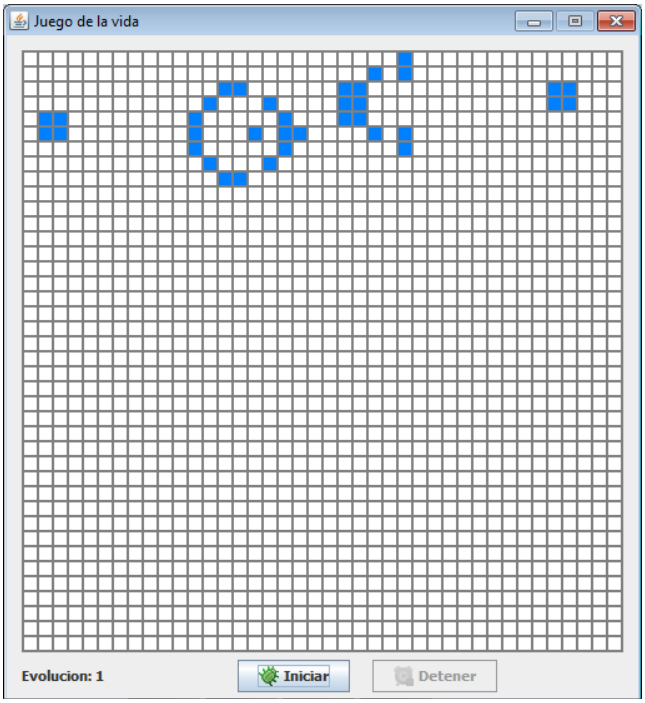
Diehard Acorn

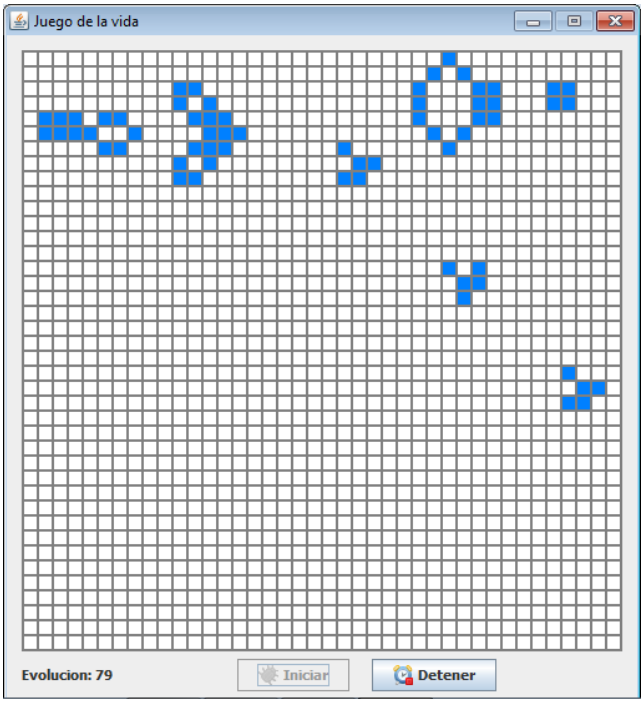
En la aparición original del juego en la revista, Conway ofreció un premio de 50 dólares por el descubrimiento de patrones que crecieran indefinidamente. El primero fue descubierto por Bill Gosper en noviembre de 1970. Entre los patrones que crecen indefinidamente se encuentran las "pistolas", que son estructuras fijas en el espacio que generan planeadores u otras naves espaciales; "locomotoras", que se mueven y dejan un rastro de basura y "rastrillos", que se mueven y emiten naves espaciales.

El primer planeador que se ha descubierto sigue siendo el más pequeño que se conoce:



Glider Gun





4 El bosque en llamas

El modelo de bosque en llamas es un AC probabilístico. Para el modelado se van a seguir las reglas definidas por Drossel y Schwabl las cuales se citan a continuación.

- Una celda con un árbol ardiendo pasa a ser una celda vacía.
- Una celda con un árbol pasa a arder si al menos un vecino cercano esta ardiendo.
- En una celda vacía, con la probabilidad p, se establece un nuevo árbol.
- En una celda con un árbol, con una probabilidad f, este pasa a ser un árbol ardiendo.

Los estados posibles de cada celda de AC son:

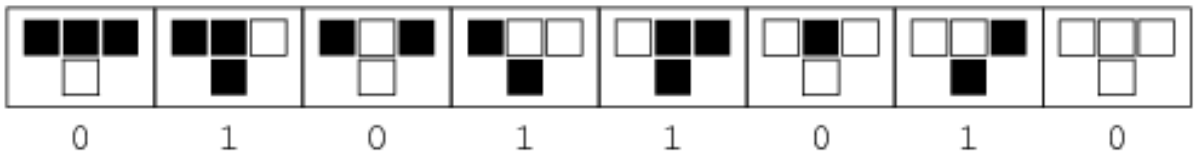
1. Celda con árbol ardiendo (Color Rojo)
2. Celda con árbol normal (Color Verde)

3. Celda vacía, sin árbol (Color Blanco)

Los valores de probabilidad del AC se van a fijar en p = 0.001 y f = 0.00001.

5 Regla 90 de Stephen Wolfram

Woldfram introduce una serie de reglas para autómatas unidimensionales, en ellos solo se tiene un arreglo de celdas, y los posibles vecinos de una celda solo son el que se encuentra a su izquierda y a su derecha. Cada celda solo puede tener dos posibles estados “1”(célula viva), “0”(célula muerta). Su regla 90 esta dado por las siguientes transiciones entre estados:



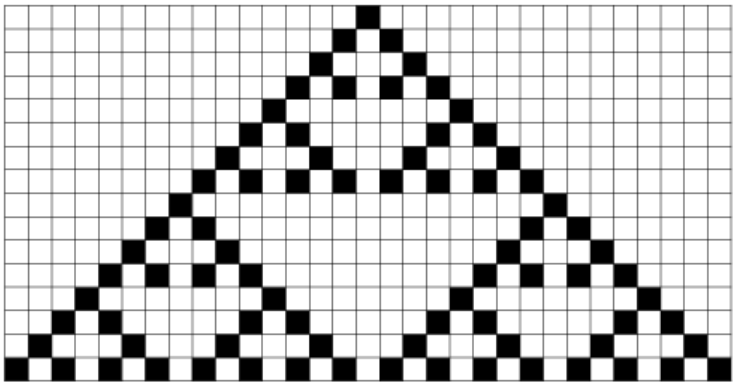
Lo cual se interpreta de la siguiente manera:

- Para el primer 0 lo que nos indica es que si una celda esta viva y sus dos vecinos también están vivos la celda pasa a morir.
- Para el primer 1: Si una celda esta viva, su vecino de la izquierda también esta vivo pero su vecino de la derecha esta muerto, entonces la celda permanece viva.
- Y así sucesivamente se construyen las 6 reglas que faltan.
-

Debido a que este es un autómata unidimensional y nuestra representación grafica es para autómatas bidimensionales, lo que haremos es lo siguiente:

- Pintaremos solo una celda viva en la mitad del primer arreglo.
- Cuando evolucione dejaremos el primer renglón tal cual y pasaremos su evolución al segundo renglón, y así sucesivamente.
- Cuando la evolución llegue al ultimo renglón disponible de la malla, ocuparemos el renglón 0 para poner la evolución del autómata, y se podrá volver a empezar.

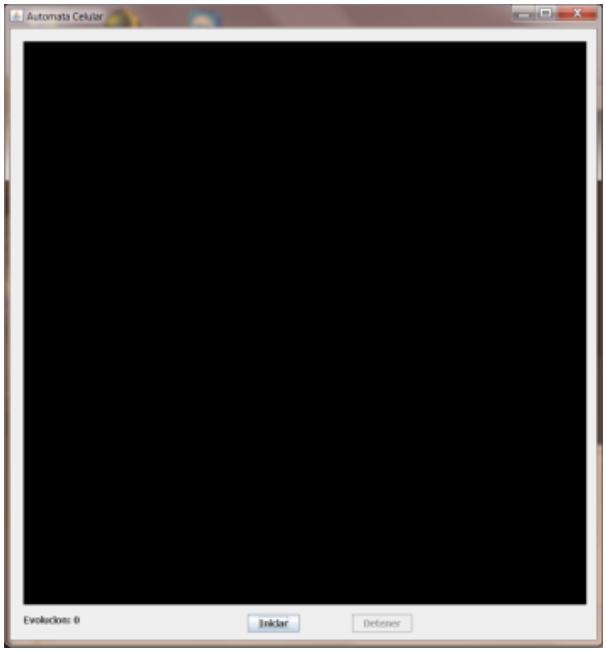
En la iteración 15 tu autómata se tendrá que ver como la siguiente figura.



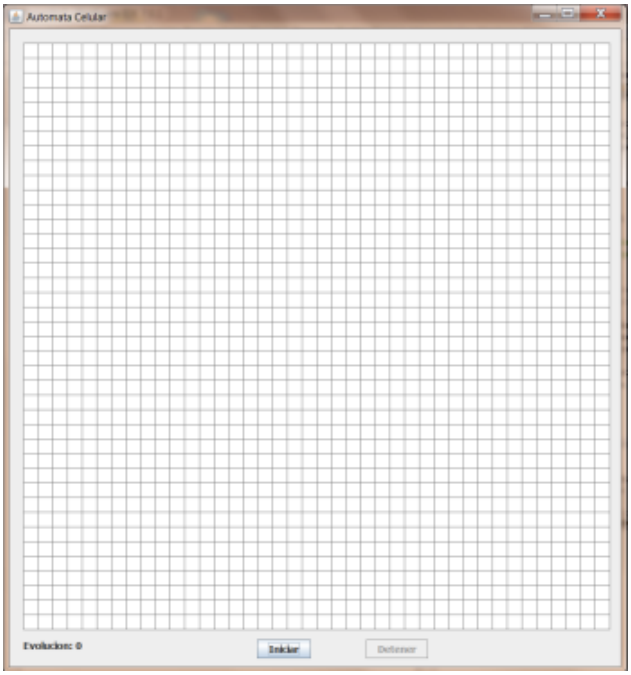
Agente: Tu misión si decides aceptarla en este proyecto es crear un programa que simule el juego de la vida con el patrón el gosper glum, que simule un incendio forestal con estado inicial aleatorio y la regla 90 de Wolfram. Debemos aclarar que solo para los dos últimos casos en la malla “todas” las celda tienen igual número de vecinos. Es decir las celdas superiores tienen como vecinos en la parte de arriba a las celdas que se encuentran en la parte inferior, y lo mismo para para las celdas que se encuentran en el extremo izquierdo. Podría decirse que la malla esta doblada en forma de dona.

6 Misión Top Secret

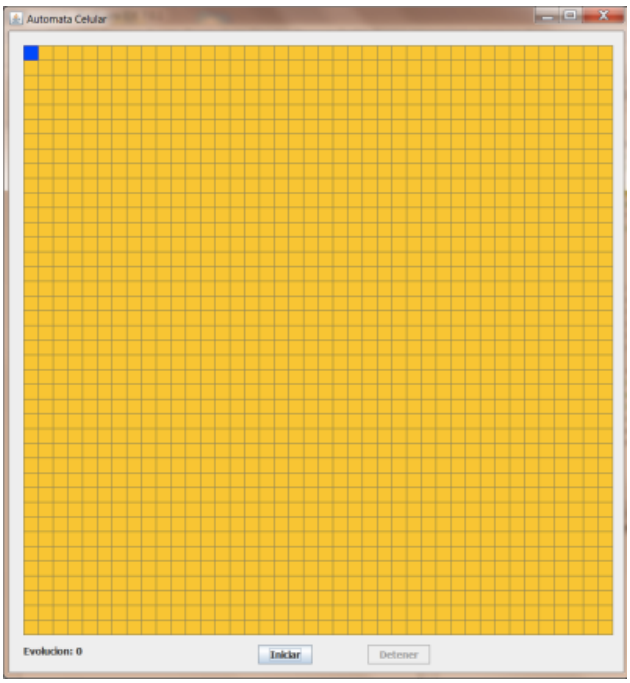
1. Junto con este archivo recibirá una carpeta llamada src, que contiene 4 clases, ábralas todas y dele un vistazo a cada una de ellas.
2. Solo compile la clase VentanaPrincipal y con ello se compilaran todos los demás archivos. Ejecute la clase y tendrá que ver algo parecido a esto:



3. Lo primero que se ve es un cuadro negro ahí es donde tendrá que derrotar al primer gran villano. Para derrotarlo tendrá que implementar los métodos *creategrid* y *paint* que se encuentran en la clase Imagen. Podríamos decir que esta clase es el marco donde pintaremos nuestro autómata, y tiene un ancho y alto declarado con la variable tam. Entonces lo primero que se necesita es un pinzel para poder pintar en el cuadro esto se hace mediante la instrucción ***Graphics2D gc = imagen.createGraphics();,*** este objeto tiene varios métodos que nos ayudan a pintar, los mas importantes que debes de considerar para esta practica son: *setColor, drawLine, fillRect.* Asi que dirigete al API para buscar información acerca de como trabajan estos métodos.
4. Pinta todo el marco de blanco y luego realiza un entramado de color gris. El punto (0,0) de la imagen se encuentra en la esquina superior. Si todo salió bien al ejecutarlo deberás de ver algo como esto:



5. Ahora implementa el método paint, aquí te pasan la matriz y la serie de colores que quieren que les pongas a cada cuadro, realiza un poco de geometría para pintar los cuadros de su color respectivo. Si todo salió bien al ejecutarlo nuevamente tendrás que ver:



6. Después de que hayas derrotado a tu primer villano lo siguiente es implementar los autómatas. Para ello primero crea una clase abstracta que contenga las características más importantes que tu creas que se utilizaran para todos los Autómatas. (No te olvides de implementar la interface Automata Celular).

7. Crea tres clases para cada uno de los autómatas a realizar.
8. En la clase VentanaPrincipal se encuentra la siguiente línea de código: ***AutomataCelular a = new PruebaPintar();*** simplemente cambia la clase PruebaPintar por la clase que contenga el autómata que quieras probar.
9. Si logra terminar todos los puntos regresara como un Héroe. Le deseo lo mejor en su misión agente y que "Los libres errores de compilación lo acompañen"