24 | Proyecto: El juego de la vida

Autor original: José Galaviz Casas
Manuel Alcántara Juárez
Verónica Esther Arriola Ríos

Prerrequisitos

- Arreglos
- Listas y genéricos
- Herencia

Meta

Que el alumno desarrolle la lógica de una aplicación aplicada a un tema real haciendo uso de orientación a objetos y arreglos.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Crear una animación en `JavaFX` gobernada por una simulación matemática.

Antecedentes

En términos generales un sistema dinámico es un modelo que describe el comportamiento, a lo largo del tiempo, de un sistema en función de sus estados previos. Un caso particular de un sistema dinámico son los sistemas dinámicos discretos, en los que el

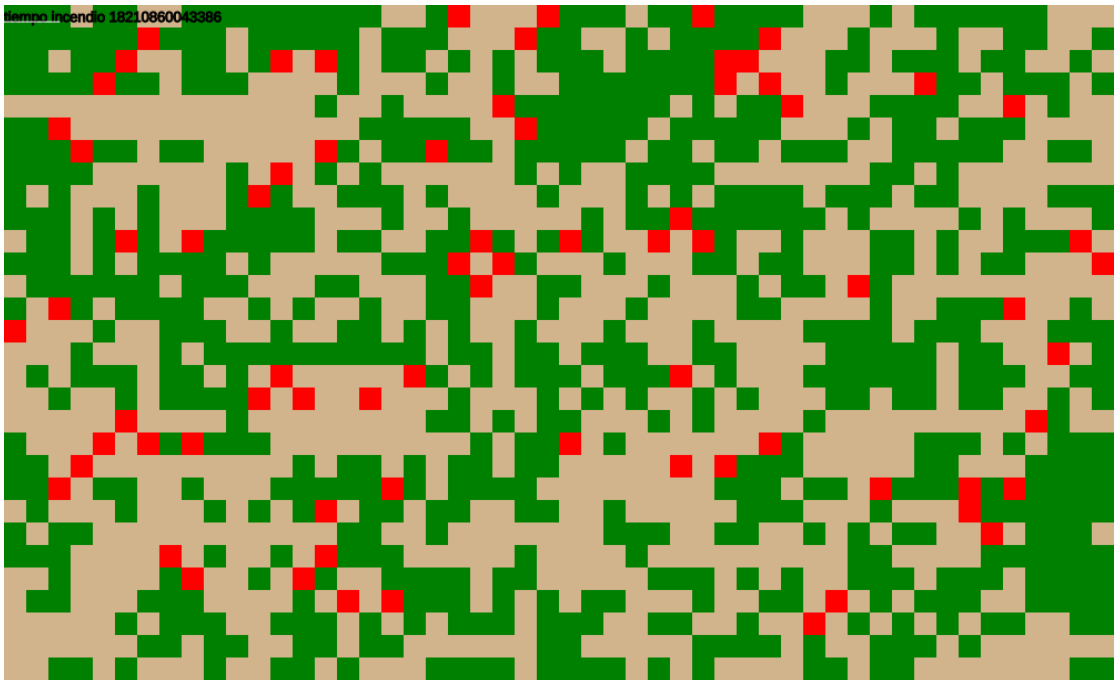

Figura 24.1 Autómata celular que simula propagación de incendios. El sistema está representado por una malla de células. Las células café representan tierra, las verdes, árboles y las rojas árboles incendiados.

tiempo no es una variable continua sino que avanza a "pasos". En general un sistema dinámico discreto está descrito completamente por la función $f : \mathbb{X} \to \mathbb{X}$ que permite determinar el estado actual del sistema x_t en función de su estado previo x_{t-1}, es decir $x_t = f(x_{t-1})$, donde x_{t-1} y x_t son dos estados de un conjunto de posibles estados del sistema $\mathcal{X}$, en notación: $x_{t-1}, x_t \in \mathcal{X}$.

Un autómata celular es un sistema dinámico discreto, en el que de hecho tanto el espacio como el tiempo y los posibles estados del sistema son discretos. El espacio se divide en *células* regulares de igual tamaño y forma, lo que constituye una *malla* Figura 24.1. En cualquier instante del tiempo t cada célula de la malla posee un valor determinado de un conjunto de posibles valores, lo que constituye un *estado*. Se comienza definiendo el estado inicial x_0 en forma determinista o aleatoria. Para determinar el estado de la célula en el siguiente paso temporal (en $t+1$), se aplica una *regla local*, una función que determina, con base en el valor de la célula misma y de algunas de sus células vecinas al tiempo t, el nuevo valor del estado de la célula.

Por supuesto, dado que el estado de una célula está en función de su estado previo y del estado previo de sus células vecinas conviene especificar quiénes son sus células vecinas. Hay una infinidad de opciones, pero las más usuales en los autómatas celulares bidimensionales son dos tipos de vecindades:

Von Neumann La vecindad de Von Neumann de una célula en la posición (i, j) está

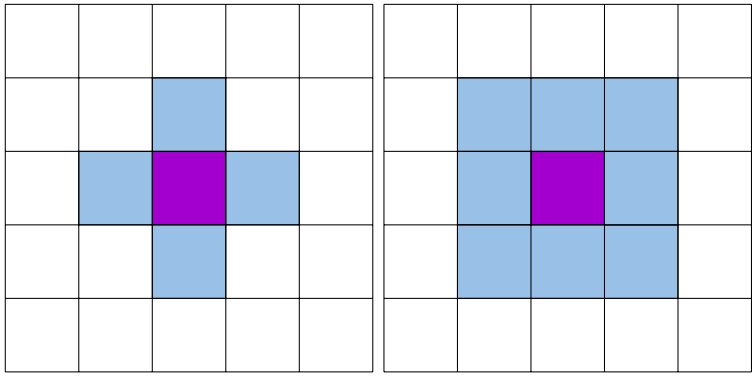

Figura 24.2 Izquierda: vecindad de Von Neumann. Derecha: vecindad de Moore. Los dos tipos de vecindades más usuales en autómatas celulares bidimensionales. El estado de la célula central es determinado, en el siguiente paso temporal, por su valor actual y el de las células coloreadas a su alrededor.

constituida por aquellas células que se encuentran arriba $(i, j-1)$, abajo $(i, j+1)$, a la derecha $(i+1, j)$ y a la izquierda $(i-1, j)$ de la célula en cuestión.

Moore La vecindad de Moore, definida por John Conway, considera además las células que están arriba a la derecha, arriba a la izquierda, abajo a la derecha y abajo a la izquierda.

En la Figura 25.1 se ilustran los dos tipos de vecindad. Una forma de definir la malla es de tal modo que **todas** las células tengan igual número de vecinos. Para conseguirlo, las células superiores tienen como vecinos en la parte de arriba a las células que se encuentran en la parte inferior y lo mismo para las células que se encuentran en el extremo izquierdo. Podría decirse que la malla está doblada en forma de dona.

Los autómatas celulares han recibido mucha atención porque han resultado ser modelos muy adecuados para describir el comportamiento de algunos fenómenos naturales muy complejos, con unas cuantas reglas locales muy simples. Desde el punto de vista teórico resultan ser también muy interesantes dado que son un ejemplo de sistemas con interacciones no lineales: el conocer cómo operan locamente no permite muchas veces conocer el comportamiento general del autómata a largo plazo: reglas simples que engendran comportamientos complejos, algo que suele calificarse como *propiedades emergentes*. Los autómatas celulares resultan estar muchas veces emparentados con los procesos caóticos o en la frontera del caos, con fenómenos de criticalidad auto-organizada y en general con lo que se denominan sistemas complejos.

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 un punto de vista teórico, es interesante porque es equivalente a una máquina universal de Turing, es decir, todo lo que se puede computar algorítmicamente se puede computar en el juego de la vida.

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.

El juego de la vida es en realidad un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El *tablero de juego* es una malla formada por cuadrados (*células*) que se extiende por el infinito en todas las direcciones. Cada célula tiene 8 células vecinas, que son las que están próximas a ella, incluso en las diagonale (vecindad de Moore). Las células tienen dos estados: están *vivas* o *muertas* (o *encendidas* y *apagadas*). El estado de la malla evoluciona a lo largo de unidades de tiempo discretas (se podría decir que por turnos). 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. Figura 24.3

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.

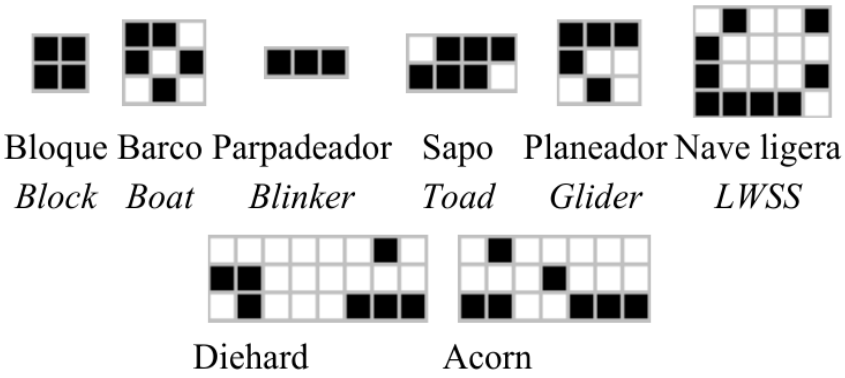

Figura 24.3 Patrones en el juego de la vida.

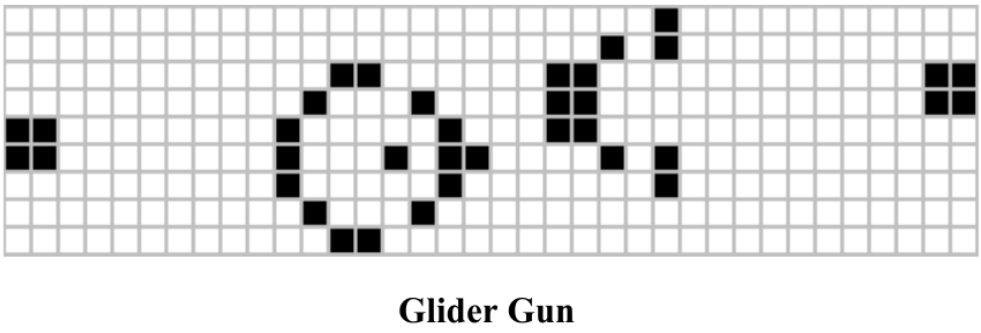

Figura 24.4 Patrones en el juego de la vida.

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* Figura 24.4

JavaFX

En este proyecto tendrás que realizar el programa completo. Para que tengas una idea de cómo puedes empezar, se te entrega un pequeño demo.

Actividad 24.1

Si tienes `Java` 10, puedes usar `ant` para ejecutarlo. Si tienes `Java` 11, usa los co-

mandos siguientes:

```
$ export PATH_TO_FX=<where_you_placed_it>/javafx-sdk-11.0.1/lib
$ javac --module-path $PATH_TO_FX --add-modules=javafx.controls
    ↪ -d ./classes --source-path ./src src/automatas/Demo.java
$ java --module-path $PATH_TO_FX --add-modules=javafx.controls -
    ↪ classpath classes automatas.Demo
```

Ojo, `ant clean` aún te servirá para entregar tu código sin archivos compilados.

Ejercicios

Hay varios modelos basados en autómatas celulares que podemos programar, todos requieren la elaboración previa de las mismas clases.

Opción 1: Actividad sísmica

Se utilizan vecindades de Von Neumann. Cada célula puede tomar valores en el conjunto $\mathbb{X} = \{0, ..., M\}$. A M se le llama *valor umbral*.

1. En $t = 0$ la malla se inicializa con valores aleatorios de entre los valores válidos inferiores a M.

2. En cada paso temporal t se elige aleatoriamente un sitio de la malla y su valor se incrementa en una unidad.

   Otra opción es incrementar los valores de todas las células, esta última opción evoluciona más rápido.

3. Si alguna célula alcanzó el valor umbral en $t - 1$, para t decrementa su valor en cuatro unidades.

4. Si un sitio tiene vecinos en valor umbral en $t - 1$, para t incrementa su propio valor en una unidad por cada vecino en valor umbral (sin rebasar el valor umbral).

Los sitios (células) de la malla con valor umbral modelan aquellos lugares de una falla tectónica que han acumulado mucha energía y ya no pueden más. Cuando un sitio de la falla alcanza el umbral lo único que le queda por hacer es romperse liberando energía que deben absorber, si pueden, sus sitios vecinos. Cuando muchos sitios se rompen al mismo tiempo se genera un sismo, la magnitud de éste está en función del número de células que se rompen.

Opción 2: Un modelo de propagación de epidemias

Se utilizan también vecindades de Von Neumann. Los valores de las células están en el conjunto $\mathbb{X} = \{0, ..., a + g\}$, dividido en tres subconjuntos: $A = \{0\}$, $B = \{1, ..., a\}$ y $C = \{a + 1, ..., a + g\}$, el conjunto A es *susceptible*, el B es *infeccioso* y el C es *inmune*.

1. Un individuo susceptible se vuelve infeccioso si al menos un vecino es infeccioso.
2. Un individuo inmune después de g pasos se vuelve susceptible.
3. Un individuo que ha sido infeccioso o inmune por menos de g pasos sólo incrementa su valor.
4. Un individuo infeccioso después de a pasos se vuelve inmune.
5. Un individuo susceptible sin vecinos infecciosos permanece susceptible.

Opción 3: Un modelo de incendios forestales

El modelo que se describe a continuación se suele llamar incendio forestal estocástico.

Los valores de las células son 0 (*vacío*), 1 (*árbol*) y 2 (*árbol incendiado*), se utilizan vecindades de Von Neumann. Este modelo es estocástico, esto es, se utilizan tres diferentes parámetros probabilísticos que controlan el proceso: p probabilidad de que cresta un árbol (transición $0 \to 1$), f probabilidad de que un árbol se incendie espontáneamente $(1 \to 2)$ y g probabilidad de que un árbol sea inmune al fuego. Las reglas son las siguientes:

1. Un sitio 0 se vuelve 1 con probabilidad p.
2. Un sitio 1 se vuelve 2 con probabilidad $(1 - g)$ si al menos un vecino está en estado 2.
3. Un sitio 1 se vuelve 2 con probabilidad $f * (1 - g)$ si no hay vecinos en estado 2.
4. Un sitio 2 se vuelve 0 al siguiente paso temporal.

Opción 4: Reglas 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** está dado por las siguientes transiciones entre estados:

Lo cual se interpreta de la siguiente manera:

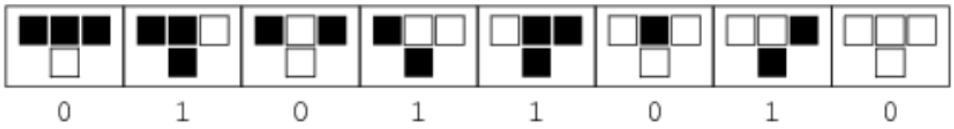

Figura 24.5 Regla 90 de Stephen Wolfram. El renglón superior muestra el estado de la célula i y sus dos vecinas al tiempo t. Abajo se muestra el valor de i en $t+1$.

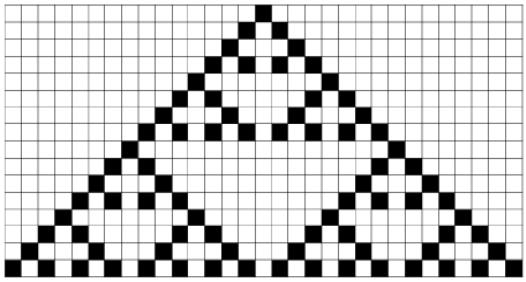

Figura 24.6 Representación bidimensional de la evolución de la regla 90. El eje vertical es el tiempo, de arriba hacia abajo.

- Para el primer 0 lo que nos indica es que si una celda está viva y sus dos vecinos también están vivos la celda pasa a morir.
- Para el primer 1: Si una celda está viva, su vecino de la izquierda también está vivo pero su vecino de la derecha está 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 gráfica 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 último 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 Figura 24.6.

Generalidades

El proyecto debe implementar, al menos, dos de los tres modelos planteados aquí, se dará máximo un punto extra a quien implemente correctamente los tres.

1. Hacer una clase abstracta de autómatas celulares que implemente las cosas que hay que hacer en todos los casos y que permita fijar la firma de algunos métodos; por ejemplo, uno que actualice el estado de una célula dado su estado actual y el de sus vecinos.

2. Debe presentarse gráficamente la evolución temporal del autómata. Las células en diferente estado deben distinguirse por su color. Los colores se dejan a elección del programador.

3. Cuando la simulación termine debe también mostrar una gráfica del número de células en estado crítico contra el número de pasos temporales (i.e. abscisas=paso temporal, ordenadas=número de células en estado crítico en ese paso temporal). Los estados criticos son:

   - Para el modelo de sismos: las células con energía en umbral.
   - Para el modelo de epidemias: las células infecciosas.
   - Para el modelo de incendio forestal: las células incendiadas.

4. Implementar un menú (puede ser en la consola), que pida al usuario qué modelo desea correr y con qué parámetros, antes de lanzar la simulación.

Entrada

- El número de pasos temporales que el usuario desea dejar evolucionar el sistema.
- El tamaño de la malla.
- Parámetros de control para el autómata (umbrales, valores de cambio de estado, etc.)

Salida

- El desplegado gráfico de la evolución del autómata.
- El desplegado de la gráfica de puntos críticos vs. tiempo.

Referencias

Bak, P. y K. Chen (1991). «Self-Organized Criticality». En: *Scientific American* 264. Una de las fuentes primigenias del tema de la criticalidad auto-organizada. En la hemeroteca de la Facultad está la colección de *Scientific American* desde hace unos 50 años a la fecha (nota del 2003)., págs. 46-53.

Gaylord, Richar J. y Kazume Nishidate (1996). «Cellular Automata Simulations with Mathematica». En: *Modelling Nature*. La colocación en la biblioteca es QA267.5,C45,G39. Incluye muchos ejemplos más de autómatas celulares como modelos de fenómenos naturales. TELOS-Springer Verlag.

Nakanishi, Hiizu (jun. de 1990). «Cellular-automaton model of earthquakes with deterministic dynamics». En: *Physical Review A* 41.12. Una generalización del modelo presentado aquí. Además se muestra que los resultados arrojados por éste son consistentes con la ley de Gutenberg-Richter, el modelo continuo más robusto en terremotos., págs. 7086-7089.

Schönfisch, Birgitt (1995). «Propagation of Fronts in Cellular Automata». En: *Physica D: Nonlinear Phenomena* 80. Incluye un par de modelos de propagación de epidemias., págs. 433-450.