PRÁCTICA 7: NOTAS AUXILIARES PARA SU CORRECTO DESARROLLO E IMPLEMENTACIÓN

La práctica tiene por objetivo implementar mediante un autómata celular bidimensional, un modelo de la reacción química oscilante de Belousov-Zhabotinsky, donde tres productos (o solutos) A, B y C, se combinan en proporciones que varían a lo largo del tiempo de reacción de acuerdo a las siguientes ecuaciones químicas (simplificadas):

- $A + B \rightarrow 2A$
- $B + C \rightarrow 2B$
- $C + A \rightarrow 2C$

¿Qué diferencia a un autómata celular 2-D de 1-D, además de la dimensión?

Pues dos cosas: la función de transición, y la vecindad; la primera depende de la segunda, y comenzamos por explicar esta última. Ahora, una célula dada no solo tiene vecinos a los lados, sino también arriba y abajo. Clásicamente, esto se tiene en cuenta mediante las vecindades de Moore y Von Neuman, ambas simétricas; pueden plantear alternativas asimétricas si se considera necesario, aunque la mayoría de fenómenos físicos, químicos o biológicos que se modelan con estos autómatas se adaptan mejor a una simulación con vecindad simétrica.

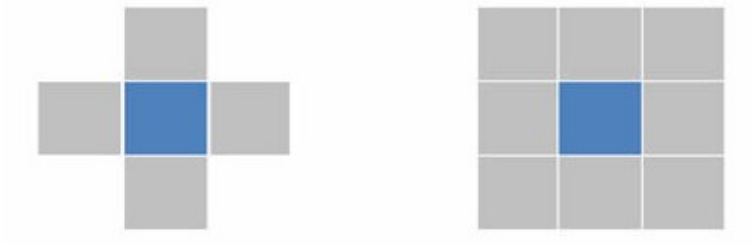

Figura 1: vecindades de Moore y Von Neumann

La función de transición ahora tiene en cuenta para calcular el nuevo estado de una célula $a_i^{(t+1)}$ bien a las cuatro vecinas situadas en los puntos cardinales (subimagen izquierda de la Figura 1) y a ella, bien a las ocho vecinas que la rodean (subimagen derecha de la Figura 1) y a ella misma.

¿Qué estructuras de datos necesito?

Cada punto discreto de la zona de reacción tiene concentraciones diferentes a, b y c de los solutos A, B y C. Para modelar esto, se necesitan tres retículas (matrices) distintas que almacenan respectivamente las concentraciones a, b y c. Además, será necesario utilizar la representación del estado de esas tres retículas en dos instantes de tiempo (actual y siguiente) por lo que en la práctica se necesitan un total de seis retículas, que se pueden organizar de muchas maneras distintas. En todo caso, se sugiere seguir el esquema del algoritmo disponible en el documento belousov-zhabotinsky.pd, que las modela como sigue:

```
a = new float [width][height][2];
b = new float [width][height][2];
c = new float [width][height][2];
```

Aquí, `a[i][j][0]` y `a[i][j][1]` representan respectivamente la cantidad del soluto A en en el punto `(i, j)` de la zona de reacción en el instante actual (tercer índice igual a 0) y siguiente (tercer índice igual a 1); lo mismo se aplica para los solutos B y C. **La condición de frontera debe ser toroidal.**

¿Qué función de transición necesito?

Se deriva directamente como resultado de modelar de forma discreta el conjunto de ecuaciones diferenciales que modelan de forma continua la reacción, y que indican cómo cambia la concentración de cada soluto en cada punto de la zona de reacción como una función de la concentración del propio soluto, y de las de los otros dos:

1. $a_{t+1} = a_t + a_t(b_t - c_t)$
2. $b_{t+1} = b_t + b_t(c_t - a_t)$
3. $c_{t+1} = c_t + c_t(a_t - b_t)$

En las expresiones anteriores, he prescindido de los aspectos ligados a la notación matricial, pero debería estar claro que durante la implementación esas ecuaciones deberían aplicarse a cada punto de la zona de reacción (retícula) para cada concentración de cada soluto. Las expresiones en pseudocódigo (siendo estrictos, se trata de código en el lenguaje "Processing"; si alguien tiene interés en entrar en esto, el sitio web de lenguaje es https://processing.org), teniendo en cuenta la operativa de matrices, pueden verse en el listado disponible en la página número 9 del documento belousov-zhabotinsky.pdf; concretamente, veáse el método `draw()`, y luego en el interior del doble bucle `for` más interno.

¿Cómo fijo la configuración inicial de mi retícula?

En este modelo esto es muy sencillo; las tres retículas que almacenan las concentraciones de los solutos A, B y C, se llenan con números aleatorios en el rango [0.0-1.0], que representan la concentración de cada soluto en cada punto, en tanto por uno. Esto da lugar a un estado inicial de cada retícula estadísticamente homologable a una nube de puntos sin estructura. Luego, la evolución en el tiempo da lugar a patrones que se pueden parecer al siguiente:

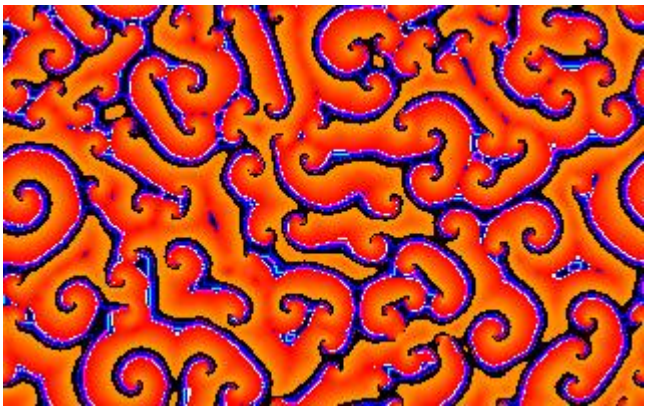

Figura 2: La reacción de Belousov-Zhabotinsky en un punto intermedio de simulación

En la figura se aprecia el habitual patrón de ondas que esta reacción produce a simple vista, y que el modelo simula adecuadamente. Este patrón se grafica utilizando el gradiente (cantidad) del soluto A presente (aunque también se podría utilizar el gradiente de los solutos B o C). Típicamente, el rango de valores que ese gradiente puede recorrer se utiliza para asignar color, por ejemplo, dividiendo ese gradiente en tantos subgradientes de igual tamaño como se desee (en la práctica 2, 3 ó 4 suelen bastar, pero está bien jugar con el concepto… a ver qué pasa); luego, se mira la concentración del soluto A para cada punto de la retícula, y en función del subgradiente en que se encuentre, se le asigna un color de una paleta predeterminada.

Ejemplo: supongamos que dividimos el gradiente normalizado (rango 0.0 a 1.0) de la concentración de soluto A en dos subrangos, a los que le asignamos una paleta de color:

Subgradiente	Color Asignado
0.0-0.4	BLANCO
0.5-1.0	NEGRO

Ahora, para pintar la retícula de concentraciones del soluto A, basta recorrerla, extraer el valor de cada punto, filtrar por la primera columna, asignar el color que la paleta determine, y graficar el punto en la interfaz de usuario. Distintas intensidades de un mismo color también pueden valer para montar la paleta. Por si os es de utilidad, el método de "Processing" que hace esto es color(v1, v2, v3). Consultada esta especificación, derivar un equivalente en Java debería ser inmediato.

¿Qúe curva debe graficar la aplicación?

Se deben generar curvas de concentración de solutos similares a las que ya habéis desarrollado para el Juego de la Vida.

Dado que es una reacción oscilante, la concentración de cada soluto como una función del número de generación debería aproximarse a un patrón que se repite cada cierto tiempo, con una estructura próxima a una sinusoide (aunque no tan perfecta, ni mucho menos; la literatura identifica hasta cuatro variaciones posibles de esta curva, que dependen de aspectos paramétricos que no citaré aquí, aunque con la naturaleza oscilante siempre presente), y con cierto decalaje entre las curvas de los solutos. La siguiente Figura da una idea del ouput de las curvas a obtener (que nadie se obsesione si no clava exactamente esto, ya que todo depende mucho, sobre todo al inicio, de las concentraciones iniciales, y en nuestro caso son aleatorias):

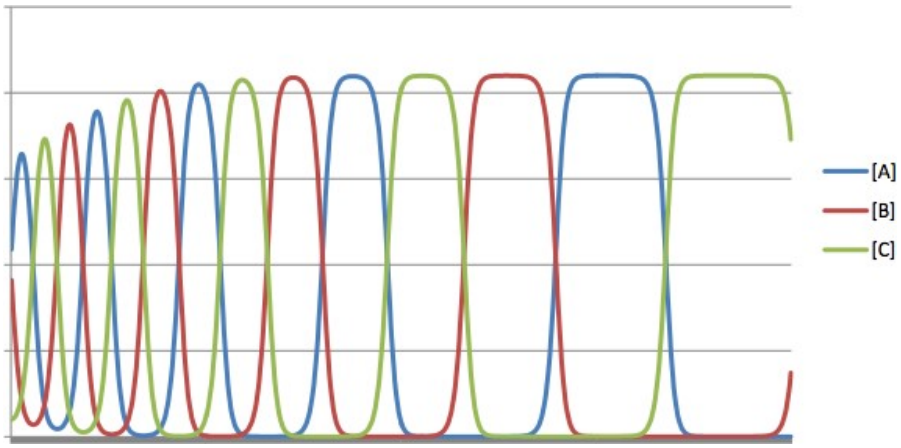

Figura 3: Cinética Temporal De Una Reacción Oscilante Típica

¿Cuál es el input de la simulación?

Trivial; el número de generaciones (tiempo discreto) durante el que se va a permitir que la reacción tenga lugar. Si os ponéis exquisitos, sería interesante también que el usuario pudiera elegir;

- El tamaño de la zona de reacción (alto y ancho de las retículas)
- La paleta de colores con que quiere visualizar la simulación.

¿Cuál es el output de la simulación?
La evolución gráfica espacio temporal de la reacción B-Z, junto con las curvas de concentración de solutos.

¿Qué elementos de control debería tener el GUI?

- Botones de Inicio y Reset.
- Si se desea, elección de la paleta de colores.