MEMORIA GLOBAL EN UN BLOQUE

La práctica consiste en la creación del juego de la vida a través de una matriz, en la que los 1 representan células vivas y los 0 células muertas. Para esto usa 2 matrices, la inicial que se carga de 1 y 0 de forma que veremos después, y una auxiliar que se utiliza para cargar los resultados de cada turno de forma temporal

La aplicación empieza pidiendo por teclado información relativa a la ejecución de esta ( el modo de ejecución y el tamaño de la matriz), y construyendo la propia matriz, colocando inicialmente valores 1 (células vivas) en el tablero de forma aleatoria.

El proceso principal de la ejecución es un bucle que ejecuta cada turno a través del kernel y presenta el resultado tras los cambios.

Cada turno del juego se desarrolla a través de kernel, lanzando tantos hilos como posiciones tenga la matriz (filas \* columnas hilos), y cada hilo ejecuta la función llamadaCelula() pasando como parámetro las dos matrices, y las dos dimensiones de la matriz. LlamadaCelula() calcula la posición de hilo (idHilo) a través de su posición respecto a “y” y “x”, y llama a cambiarEstado(), que requiere como parámetros las 2 matrices, las dimensiones de la matriz y el idHilo. Tras llamar a la función, espera a que acaben todos los hilos a través de \_synchtreads().

La funcion del device cambiarEstado() utiliza el valor que devuelve comprobarVecinos() para comprobar el estado de la célula. Comprueba si la célula está viva y el valor es menos de 2 o más de 3 la célula muere, poniendo el valor 0 a la matriz auxiliar en la posición idHilo, y si estaba muerta y el valor devuelto es 3 la célula nace poniendo esta vez el valor 1 en la matriz nueva. Si no se cumple ninguna de estas condiciones, el valor de la matriz auxiliar será el mismo que el de la matriz anterior.

La última función de la aplicación es comprobarVecinos(), se ejecuta en el device, y recibe como parámetros la matriz original (no necesita la auxiliar) las dimensiones de la matriz y la posición del hilo actual. Esta función comprueba el estado de las células alrededor de la célula que la llama, sumando una por cada una viva, y devolviendo este valor para que lo use cambiaEstado(). Para eso mete todas las células en un array (el numero de celulas alrededor dependera de la posición) y las comprueba a través de un bucle.

En cada turno tras efectuar los cambios se iguala la matriz antigua por la nueva matriz auxiliar y se repite el proceso.

Una vez la ejecución ha acabado se libera el espacio de memoria utilizado por las matrices.