Candy Crosh Soga

Paradigmas Avanzados de Programación

CURSO 2022/2023 - EVALUACIÓN CONTINUA

SEGUNDO CUATRIMESTRE

PECL1

APELLIDOS/NOMBRE ESTUDIANTES:

CANCIO FERNANDEZ, KEVIN ORLANDO

NAVARRO JUAN, MARCOS

Fecha: 29 de marzo de 2023

Contenido

Parte 1.

La práctica está dividida en diferentes partes:

- El Main, donde se llaman a todas las funciones auxiliares y todos los kernels.
- Las funciones auxiliares, que realizan diferentes acciones en la CPU.
- Los kernels, que implementan todos los movimientos del juego en la GPU. Dentro de los kernels se produce llamadas declaras en el propio device.

Main.

Lo primero que se realiza en el Main declara multitud de variables. Se declara la variable vidas y una serie de variables que en función de que si se ha completado las acciones del juego se van a poner a 0 o a 1 para indicar si se ha explotado una bomba, si se ha puesto un rompecabezas… La prioridad de orden de la acciones (comprobación de su ejecución) es: Explotar Rompecabezas > Poner Rompecabezas > Explotar TNT > Poner TNT > Explotar Bomba > Poner Bomba > Comprobar Pares > Perder Vida.

Después de las variables, se llaman a las funciones auxiliares, que se encargan de formar el tablero. También se pide al jugador de qué forma quiere jugar, sí de forma aleatoria (con un 1) o manual (con un 0). Si se juega de forma aleatoria se elegirá una posición en cada jugada sobre la que trabajar, si es de forma manual, el usuario elegirá esta posición

También se define el número de bloques e hilos por bloque, que esta parte de la práctica, sólo hacemos uso de un único bloque y los bloques por hilo va a ser el número de filas por el número de columnas.

Otra parte que considerar del Main, es que a través de un while se realizan todas las acciones del juego de forma permanente, hasta que la variable vida sea cero. Esta variable se inicializa a 5, si el programa no es capaz de realizar ninguna acción se decrementa en una unidad (se pierde una vida).

Las variables de control que me indican si ha realizado una acción se reinician al principio del bucle y se copian los datos en la memoria del device, para poder trabajar con ellos. También al principio se produce la llamada a la función imprimirMatriz(), para que en cada iteración muestre el tablero.

Después, se comprueba de qué forma ha elegido el usuario jugar, si es de forma manual, se pide al usuario que introduzca la posición, si es de forma aleatoria se genera una posición aleatoria.

A continuación, se producen las llamadas a los kernels. Es importante la manera en la que se ejecutan los kernels. Lo primero que se comprueba es que, si en la posición indicada por el usuario se puede explotar un rompecabezas, a continuación si se puede poner un rompecabezas, así consecutivamente hasta que el último kernel declarada es comprobar si hay un valor adyacente a la posición indicada.

En cada iteración del while solo se podrá ejecutar un kernel, para ello se utilizan las variables de control que se ponen a uno si ha ejecutado ese kernel evitando que ningún otro llegue a ejecutarse.

Si ningún kernel se ha ejecutado, las variables de control estarán a cero por lo que no se ha podido realizar ninguna acción en la posición indicada. Entonces se restará una vida.

A continuación de los kernels, cuando el juego haya terminado se devolverá al host la matriz final y se libera la memoria del device.

Funciones Auxiliares.

Se tienen una serie de funciones auxiliares, que son las siguientes:

- pedirFilas(): Pide por pantalla el número de filas para matriz.
- pedirColumnas(): Pide por pantalla el número de columnas para la matriz.
- validarEntero(): Que verifica que lo que se ha introducido por pantalla en se trata de un entero.
- pedirDificultad(): Pide por pantalla que se introduzca el nivel de dificultad, siendo 1 el nivel fácil, y el nivel 2 el difícil.
- generarMatriz(): Construye la matriz a partir de las dimensiones introducidas por pantalla. Se declara como puntero para poder realizar una matriz con dimensiones dinámicas.
- imprimirMatriz(): Muestra la matriz por pantalla, cambia los números por signos especiales para facilitar la compresión. Se declara la bomba como -1, el TNT como -2 y el rompecabezas como -3 y se cambian por las siguientes letras B, T,

Kernels

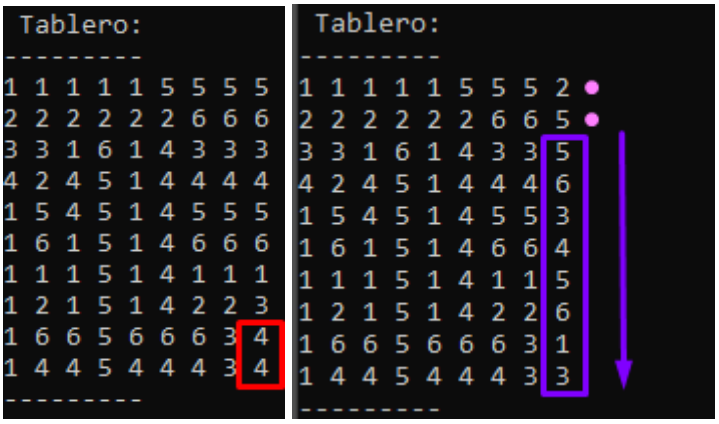
Se tienen los siguientes kernels:

- comprobarPares
- ponerBomba
- explotarBomba
- ponerTNT
- explotarTNT
- comprobarRompecabezas
- explotarRompecabezas

Comprobar Pares.

En el kernel comprobarPares se encarga de analizar si la posición consultada por el usuario, el bloque elegido tiene otro bloque igual adyacente, en caso afirmativo eliminará los dos bloques y el resto caerán por gravedad. La prioridad de análisis de adyacencia es arriba > abajo > izquierda > derecha, es decir, primero se examina si la adyacencia se cumple arriba, después abajo… y terminando las comprobaciones posteriores si se ha encontrado ya alguna adyacencia. Cabe recalcar a la hora del análisis del algoritmo, que si la adyacencia se produce en vertical, se necesitará generar 2 números aleatorios nuevos en la cima de la columna (para

rellenar la matriz), en el caso de que la adyacencia sea en horizontal sólo será necesario crear un nuevo número aleatorio en la la cima de cada columna del bloque indicado (para rellenar la matriz), estos números son aleatorios y determinados por la dificultad del juego. Si se ha detectado que se ha realizado una adyacencia, devolverá la variable de control a 1, para no perder ninguna vida.



Poner Bomba.

En el kerner ponerBomba, lo primero que se hace es comprobar si la posición en que indica en el usuario el corresponde el con el hilo que va a iterar esa posición. Después se comprueba que en esa ubicación existen cinco posiciones adyacentes con el mismo valor, lo comprueba tanto de forma vertical como horizontal y para cada una de las posiciones en las que estén repartidos los cincos valores, en esa misma comprobación, verifica que estos valores pertenecen o a la misma fila o a la misma columna.

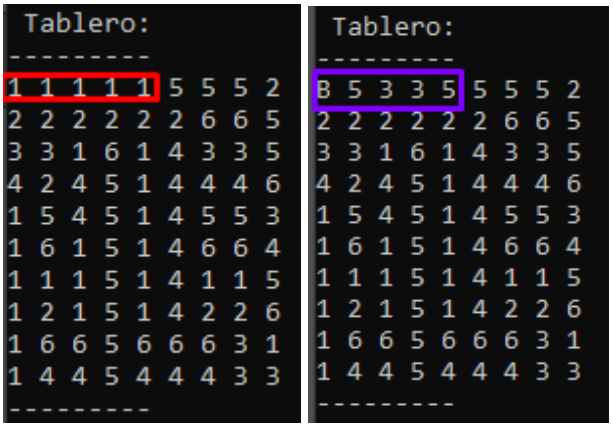
Si existen cinco ubicaciones adyacentes con el mismo valor y de forma horizontal, se llama a la función declarada en el device gravedad_horizontal_bomba y si es de forma vertical se llama a gravedad_vertical_bomba.

La función gravedad_horizontal_bomba se va a encargar de poner la bomba en la posición indicada por el usuario, y que se eliminen el resto de posiciones adyacentes que tenían el mismo valor y que se produzca la gravedad, que las posiciones de arriba ocupan las nuevas posiciones de abajo. Se comprueba si está en la fila cero, porque si es así, solo se tienen que generar los nuevos valores y no caer ninguna fila. Si no es así, las posiciones donde estaban los números adyacentes se eliminan menos donde se pone la bomba y las posiciones de arriba ocupan esas posiciones. Después como siempre va a caer una fila, se genera la primera fila de la matriz de forma aleatoria.

La función gravedad_vertical_bomba, se encarga de eliminar las posiciones adyacentes con el mismo valor y poner la bomba en la posición más abajo de esas posiciones ya que por la gravedad cae la bomba. Si ocurre el caso que arriba de esas posiciones adyacentes hay más valores en el tablero esos valores pasan a ocupar las posiciones que se han eliminado.

Lo primero que se hace es obtener la fila donde comienzan los números adyacentes, la última fila y la columna en la que sucede. Lo primero que comprobamos es que si hay alguno de estos valores que se encuentra en la primera fila, porque si es así, no caen los valores de arriba y solo se generan los nuevos aleatorios. Si tiene posiciones arriba, caen esos valores

consecutivamente empezando desde donde se pone la bomba y se generan los valores aleatorios en las posiciones que han caído.

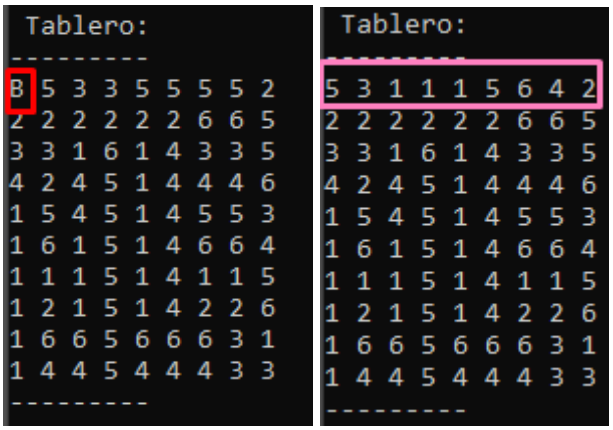


Explotar Bomba.

En el kernel explotarBomba se encarga de explotar una bomba, si está en la posición indicada por el usuario existe una bomba (el valor -1). Si es así, se elige aleatoriamente si se explota toda una fila o una columna.

Si se explota una columna, solo hace falta calcular en qué columna se encuentra la bomba y cambian todos los valores de la columna por valores nuevos generados aleatoriamente.

Si se explota una fila, se calcula cual es la posición del primer elemento de la fila y toda esa fila se cambia por la fila de arriba así consecutivamente hasta bajar todas las filas una posición. Y en la primera fila se generan nuevos valores aleatorios.

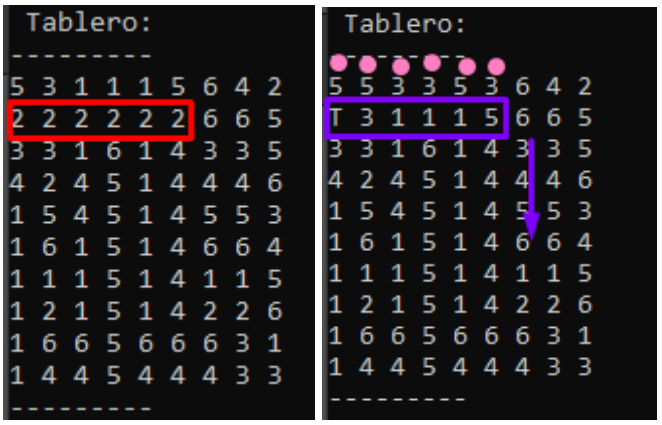


Poner TNT.

Este kernel tiene un procedimiento similar a ponerBomba. Se comprueba que en la posición indicada por el usuario existen seis posiciones adyacentes con el mismo valor, de forma vertical o horizontal. También se comprueba si esos valores pertenecen a la misma fila o columna. Al igual que en ponerBomba da igual que posición de las posiciones adyacentes indique, se coloca ahí el TNT comprobando cada una de las posiciones.

Si esos seis valores se encuentran de forma horizontal, se ejecuta la función gravedad_horizontal_TNT y si se encuentra de forma vertical la función gravedad_vertical_TNT, ambas declaradas en el device.

La función gravedad_horizontal_TNT funciona de la misma forma que gravedad_horizontal_bomba, pero trabajando para seis posiciones en vez de cinco. Y lo mismo ocurre para la función gravedad_vertical_TNT.

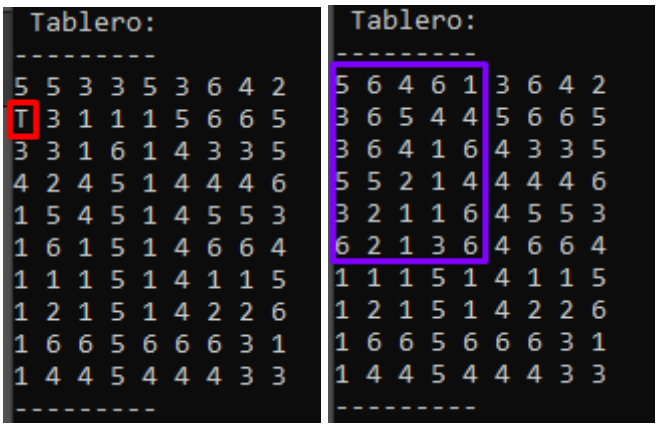


Explotar TNT.

Explotar TNT se encarga de que si en la posición indicada por el usuario, existe una bomba la explota. La explosión es de un radio de cuatro elementos a partir de la ubicación de la bomba, eliminando todos los valores que se encuentre dentro del radio y posteriormente haciendo que los valores de las posiciones de arriba caigan hasta las posiciones que se han eliminado. Por último, generando nuevos valores aleatorios en las posiciones que han caído. Para ello se realizan los siguientes pasos:

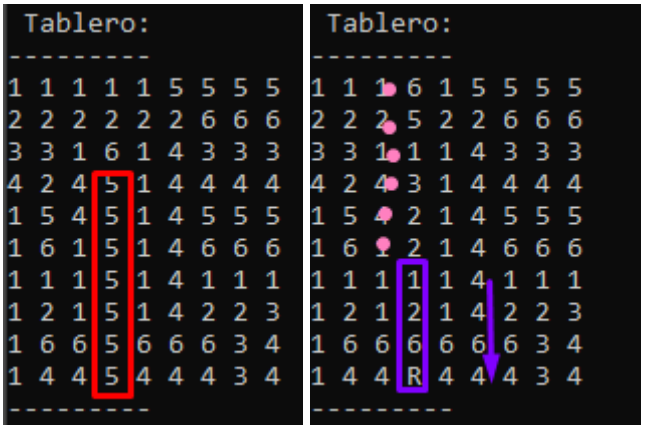
1. Se calcula cuál es la posición más a la izquierda posible dentro del radio de cuatro elementos. Se calcula el valor de la columna que llega al radio de la bomba sin salirse del tablero.
2. Se calcula cuál es la posición más a la derecha que puede llegar la explosión en esa misma fila.
3. Se calcula cual es la posición más abajo y la izquierda que puede llegar la explosión, por tanto a partir del valor calculado en el punto 1 se calcula hasta qué fila llega la explosión.
4. Se calcula cual es la fila por arriba que llegara la explosión, con estos primeros cuatro pasos se calcula las dimensiones que va a tener la explosión teniendo en cuenta la dimensiones del cuadrado y donde esté situado el TNT.
5. Se calcula cual es la diferencia entre la primera fila que puede llegar al explosión y la fila, se calcula para saber cuántas posiciones de arriba del radio de la explosión van a caer.
6. Se calcula cuál es la diferencia entre la columna más a la izquierda y la columna más a la derecha, se calcula para saber cuáles son las opciones por arriba que van a caer.
7. Se intercambian las posiciones que han explotado por las posiciones de arriba del radio de la explosión. Se empieza a cambiar las posiciones por el valor más abajo a la izquierda y sabiendo cuales son las filas que empiezan a caer y cual es hasta qué columna se ven afectadas, se van intercambiando las posiciones.

8. Se calculan cuántas posiciones se van a poner de forma aleatoria, debido a la caída de los antiguos valores. Se calcula tanto de forma horizontal como vertical.
9. Ya sabiendo cuántas posiciones nuevas se tienen que generar tanto de forma vertical como de forma horizontal, se cambia el valor de las posiciones por valores aleatorios.



Comprobar Rompecabezas.

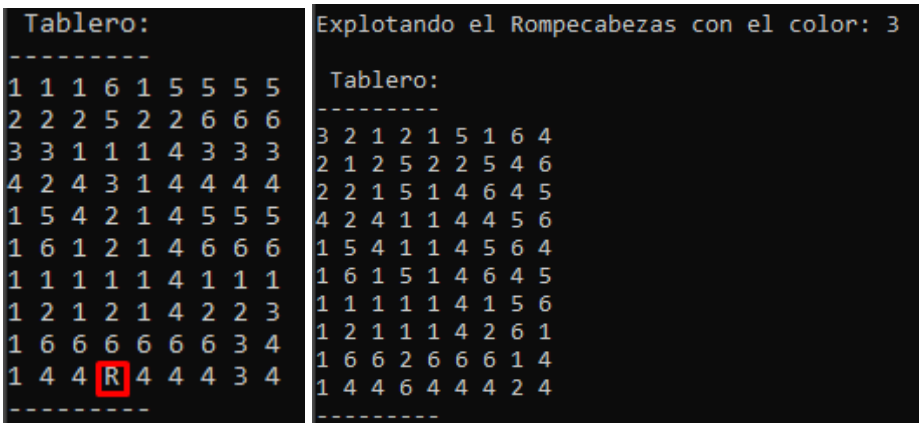
Para comprobar si hay un rompecabezas (7 bloques iguales juntos), el hilo que se encuentra en la posición indicada por el usuario se encarga de comprobar la adyacencias para saber si cuenta con 7 bloques iguales a sus lados en total, primero cuenta cuántos bloques tiene a la izquierda hasta que se encuentre uno diferente y luego cuenta cuántos bloques tiene a la derecha hasta que encuentre uno diferente o ya haya encontrado 7 bloques iguales. En caso de que se cumpla la condición de encontrar 7 bloques iguales, se devolverá la variable de estado a 1, para indicar que no se debe perder ninguna vida. Una vez identificado el rompecabezas, se eliminarán los 7 números de las respectivas posiciones y caerán por gravedad los bloques que tengan arriba, cabe recalcar al igual que al comprobar pares, que si la adyacencia es vertical, serán necesarios generar 7 números aleatorios nuevos en la cima de la columna para rellenarla, y en el caso de que la adyacencia sea horizontal, sólo será necesario generar 1 nuevo número aleatorio en la cima de cada columna.



Explotar Rompecabezas.

Cuando se detecta que la posición de la matriz examinada es un “-3”, es decir, un rompecabezas, que se mostrará como una letra “R”, se genera un número aleatorio determinado por la dificultad, para definir qué serie de bloques (números) se van a eliminar,

mostrándolo por pantalla previamente. En primera instancia, antes de poder ejecutar la explosión del rompecabezas, el hilo que está en la posición indicada por el jugador (el bloque rompecabezas), deberá “eliminar” el bloque rompecabezas y dejar caer todos los números que tiene arriba por gravedad y añadir uno nuevo en la cima de esa columna. Posteriormente se analizará columna a columna desde la posición inferior de la matriz hasta la superior, si ésta contiene un bloque que deba eliminarse, en su caso, lo elimina y deja caer el resto de bloques por gravedad, serán necesarios generar tantos números aleatorios en la cima de la columna como bloques se hayan eliminado. Si se ha detectado la explosión de un rompecabezas, se devolverá la variable de estado a 1 para indicar que no se debe perder ninguna vida.



Parte 2.

Esta parte corresponde al ejercicio (7), donde el tablero se supondrá finito y debe correrse un múltiples bloques.

Para cumplir con la parte de optimización pedida, el programa en primera instancia analizará e imprimirá cuáles son las características hardware de la tarjeta donde se va a realizar la ejecución, dependiendo de esas características el problema (bloques e hilos) se dimensiona de manera adecuada.

```
Obtener las caracteristicas basicas de CUDA de tu tarjeta grafica.

Nombre del Device: NVIDIA GeForce GT 720 (Numero: 0)
Numero maximo de hilos por bloque: 1024
Numero maximo de hilos en un SM: 2048
Dimensiones maximas para organizar los hilos en bloques (x,y,z): (1024,1024,64)
Dimensiones maximas para organizar los bloques en el grid (x,y,z): (2147483647,65535,65535)
```

El primer aspecto relevante es el número de bloques disponibles en las dimensiones (x, y, z), el programa elegirá el mínimo entre “x” e “y” para establecerlo como marco de referencia para los cálculos y no sobrepasar los bloques disponibles en una dimensión, ya que eso causaría un crash de la tarjeta gráfica y los controladores gráficos; en segundo lugar se tienen en cuenta el número de hilos disponibles en cada dimensión (x, y, z) de los bloques, el programa elegirá el mínimo entre “x” e “y” para establecerlo como marco de referencia para los cálculos y no sobrepasar los hilos disponibles en una dimensión del bloque.

Posteriormente se tendrán en cuenta el número de filas y columnas introducidas por el usuario para no sobrepasar en bloques las dimensiones de la matriz, el número bloques estará definido por un número aleatorio (mínimo 1) con límite en la mitad de lo que sea menor, o mínimo de bloques o mínimo de filas/columnas.

Los hilos estarán definidos por la misma lógica y posteriormente aplicando una fórmula que se encargue de comprobar que en función de los bloques establecidos, se pueda ocupar toda la matriz: (hilos_min + blockSize - 1) / blockSize

Una vez definidos los bloques e hilos se imprimirá por pantalla la elección automática:

```
Bloques Elegidos: 3, 3
Hilos Elegidos: 4, 4
dificultad 2
[Informacion] Desea jugar de forma manual(0) o automatico(1)
```

Parte 3.

Esta parte corresponde al ejercicio (8), donde el tablero se supondrá finito y la solución deberá permitir correr la matriz en múltiples bloques y utilizando la memoria compartida de ellos.

En este caso se ha cambiado la declaración de los bloques e hilos, aplicando otra fórmula diferente, el funcionamiento es el mismo de los límites de mínimos anterior, pero esta vez vamos a poner como límite superior para los bloques, la raíz cuadrada de la dimensión mínima (y mínimo 1), para que al hacer la multiplicación de bloques (X, Y, 1) no sean superados los límites: blockSize = (rand() % sqrt(minimo)) + 1;

Para asegurar que se cumplen el número de hilos necesarios según las dimensiones de la matriz y el número de hilos máximos de la GPU, se aplica esta fórmula que tiene en cuenta los bloques establecidos por la máquina: ceil((filas * columnas) / (blockSize * blockSize)).

Los resultados de bloques e hilos se imprimirá por pantalla:

```
Bloques Elegidos: 3, 3
Hilos Elegidos: 4, 4
dificultad 2
[Informacion] Desea jugar de forma manual(0) o automatico(1)
```

Para aplicar la memoria compartida se ha aplicado un cambio a los kernels:

Comprobar Pares.

Han sido creados 2 arrays de memoria compartida que contendrán información relevante para evitar tener que hacer tantos accesos a memoria global.

En primera instancia, el primer array de memoria compartida será escrito únicamente por el hilo que se encuentra en la posición del usuario, este array contendrá su valor numérico en la matriz y su posición para poder ser consultados por los bloques (números) adyacentes. El resto de hilos introducidos en la matriz comprobarán si su propia posición es adyacente superior o lateralmente y si coincide en valor, a los datos introducidos por memoria compartida del hilo que se encuentra en la posición del usuario, en caso afirmativo, guardará su posición en otro array de memoria compartida (junto con la posición especificada por el usuario). La comprobación de adyacencia vendrá dada por prioridad: Superior > Inferior > Lateral

Este último array de memoria compartida (posiciones) será usado para evitar más accesos a memoria global y saber en qué posiciones específicas es necesario eliminar los bloques y en consecuencia, añadir en la cima de su columna el número aleatorio para rellenar la matriz.

Comprobar Bomba.

Se utiliza la memoria compartida para poder detectar que existen cinco posiciones adyacentes con el mismo valor y a continuación aplicar la gravedad.

Para ello se utilizan dos arrays en memoria compartida, el primero array se define como valor[] (con tamaño 2) y el segundo como arr_adyacente[] (con tamaño 5).

- En el array valor[], se almacena tanto la posición definida por el usuario a la hora de determinar la coordenada de chequear, como el valor guardado en dicha posición.

- En el array arr_adyacente[] se almacenan, una vez encontrado las posiciones adyacentes del mismo valor, se almacena la posición de cada una.

Ya con todas las posiciones almacenadas, ese array de memoria compartida se pasa tanto a la funcion gravedad_horizontal_bomba como a la gravedad_vertical_bomba, donde se lleva a cabo la gravedad utilizando el array de memoria compartida.

Comprobar TNT.

Para comprobar TNT, tiene un funcionamiento similar a Poner Bomba, donde se usa la memoria compartida para detectar, en este caso, seis posiciones adyacentes con un mismo valor. Como en el procedimiento anterior se usan dos arrays en memoria compartida, el primero array se define como valor[] (con tamaño 2) y el segundo como arr_adyacente[] (con tamaño 6).

- En el array valor[], se almacena tanto la posición definida por el usuario a la hora de determinar la coordenada de chequear, como el valor guardado en dicha posición.
- En el array arr_adyacente[] se almacenan, una vez encontrado las posiciones adyacentes del mismo valor, se almacena la posición de cada una.

Ya, con las posiciones adyacentes almacenadas en el array se llaman a la funciones declaradas en el device, gravedad_horizontal_TNT y lla gravedad_vertical_TNT, que se encargan de poner en la ubicación exacta el TNT y en que se produzca la gravedad una vez que haya sido situado.

Comprobar Rompecabezas.

En el caso del rompecabezas, para aplicar la memoria compartida, se han declarado dos arrays que contendrán las posiciones de los valores adyacentes en columna o fila, de la posición indicada por el usuario para saber si está colindante con 7 números iguales en columna y fila, y aplicar en consecuencia el posicionamiento del rompecabezas “R”.

Estas dos listas de posiciones verticales y horizontales sirven para reducir los accesos a memoria global por parte de los hilos y saber en qué posiciones directas se encuentran estos 7 bloques (números) repetidos.

Una vez identificado el rompecabezas se debe eliminar los 7 números iguales adyacentes de prioridad fila > columna e introducir el bloque “R” (rompecabezas). Si los bloques han sido eliminados en columna, se deberá generar 7 números aleatorios para introducir al comienzo de esa columna de memoria compartida, si los bloques han sido eliminados en fila, se deberá introducir un número aleatorio al principio de cada columna de las posiciones de la memoria compartida, en ambos casos caerán los bloques (números) superiores por gravedad.