Sharikistones

A partir de los últimos conocimientos aprendidos en el equipo de desarrollo, se decidió plantear de cero el desarrollo de Sharikistones, descartando todo lo previamente realizado. Repasemos el enunciado y veamos cómo ha cambiado.

El Shariki (las esferas, en ruso) es un videojuego de tipo puzzle desarrollado originalmente para DOS por Eugene Almezhin. El objetivo del juego era intercambiar una esfera por otra lindante, de forma tal de obtener líneas horizontales o verticales que contuvieran tres o más esferas del mismo color. Al lograrlo, esas esferas explotaban, otorgando puntos al jugador, y apareciendo en su lugar nuevas esferas de colores aleatorios. El juego se terminaba cuando no había más movimientos válidos para realizar. Shariki se volvió muy popular y su mecánica vió múltiples reencarnaciones bajo distintos nombres, como Bejeweled, Jewel Quest, Candy Crush Saga y Pokemon Shuffle.

En esta actividad vamos a trabajar con una adaptación propia del juego, implementada en Gobstones, a la que llamaremos Shariikistones. Primero explicaremos en detalle las reglas del juego, y luego su representación en Gobstones.

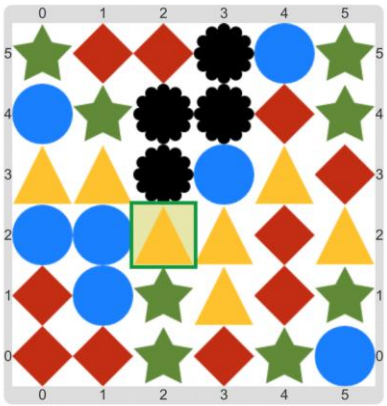
El juego se desarrolla en una grilla rectangular de tamaño variable, en donde en cada ubicación puede haber, o bien una ficha, o bien un obstáculo, o bien estar vacía. Las fichas pueden ser de 4 tipos distintos: estrellas, círculos, rombos o triángulos. El juego consiste en intercambiar una ficha por otra, mediante un movimiento válido. Un movimiento es válido sí:

- Las fichas que se intercambian son colindantes.
- Al intercambiar las fichas se forma una una línea de 3 o más fichas iguales ya sea de forma horizontal o vertical, en cualquier fila o columna de la grilla.

Al realizar un movimiento válido, se “queman” las fichas que forman parte de la línea de 3 o más fichas iguales, quedando por un breve período de tiempo vacías. Luego, esos espacios vacíos son rellenados por nuevas fichas aleatorias que aparecen en la misma posición.

Los obstáculos consisten en elementos que se ubican en los mismos lugares que podrían ocupar fichas, pero no pueden moverse, limitando la cantidad de movimientos que se pueden realizar, y aumentando el nivel de dificultad del juego. El nivel del juego está determinado por la cantidad de obstáculos presentes en la grilla. Representaremos la grilla del juego con el tablero de gobstones.

A continuación hay un tablero de ejemplo que muestra un posible tablero inicial, con vestimentas:



En este ejemplo se muestra el cabezal sobre 2,2, una ficha “triángulo”. Sin embargo no se puede mover ese triangulo al Norte para intercambiar, porque la mancha negra al norte es un obstáculo, y no puede moverse. Hay dos movimientos válidos posibles, o bien moviendo el triangulo en 4,3 hacia el Sur, o bien moviendo el rombo en 5,3 hacia el Oeste. En ambos casos se formarán líneas.

El equipo de desarrollo planteó los siguientes tipos para poder trabajar con el juego.

```
type TipoDeElemento is variant {
  // PROPÓSITO: Modela el tipo de un elemento posible en una ubicación.
  case Libre     {}
  case Obstáculo {}
  case Ficha     {}
}
```

```
type FormaDeFicha is variant {
   // PROPÓSITO: Modela el tipo de una ficha
   case Círculo   {}
   case Triángulo {}
   case Rombo     {}
   case Estrella  {}
}
```

Además se plantearon las siguientes primitivas.

elementoAcá
PROPÓSITO: Describe el elemento en la ubicación actual.
TIPO: TipoDeElemento
PRECONDICIONES: Ninguna

fichaAcá
PROPÓSITO: Describe la forma de la ficha en la ubicación actual.
TIPO: FormaDeFicha
PRECONDICIONES:
* El elemento en la ubicación actual debe ser una Ficha.

SacarObstáculo
PROPÓSITO: Saca el obstáculo de la ubicación actual.
PRECONDICIONES:
* Hay un obstáculo en la ubicación actual.

QuemarFicha
PROPÓSITO: Quema la ficha en la ubicación actual.
PRECONDICIONES:
* Hay una ficha en la ubicación actual.

PonerFicha_
PROPÓSITO: Pone la ficha con la forma dada en la ubicación actual.
PARÁMETROS:
* formaAPoner: FormaDeFicha - La forma de ficha a poner
PRECONDICIONES:
* La ubicación actual está libre.

Modelo 1

Ejercicio 1)

Realice la función **fichaAleatoria** que describe la forma de ficha con la cual reponer un espacio vacío según un proceso pseudo-aleatorio, basado en las fichas alrededor (en direcciones ortogonales, es decir, línea recta) con las reglas que siguen:

- Si la ubicación a reponer tiene al menos 2 fichas estrella alrededor y está sobre un borde, se rellena con un círculo.
- Si la ubicación a reponer tiene 3 fichas iguales alrededor, se rellena con una ficha rombo.
- Si la ubicación a reponer tiene un obstáculo alrededor, se rellena con una estrella.
- Cualquier otro caso se rellena con un triángulo.

Ejercicio 2)

Realice la función **cantidadDe_EnGrilla** que dado tipo de elemento describe la cantidad total de elementos del tipo dado en la grilla.

Ejercicio 3)

Realice el procedimiento **ReponerEspaciosLibresEnGrilla** que repone mediante una ficha aleatoria todos los espacios libres que haya en la grilla. Para resolver el problema puede hacer uso de la función total **fichaAleatoria**, que describe una forma de ficha aleatoria.

Ejercicio 4)

Se desea poder recuperar el estado del juego como un dato que se pueda manipular y analizar. Recordar que el nivel está dado por la cantidad de obstáculos. Para esto, los desarrolladores definieron el siguiente tipo:

```
type EstadoDelJuego is record {
   // PROPÓSITO: Modela el estado del juego de Sharikistones.
   // INVARIANTE DE REPRESENTACIÓN: El nivel es >= 0.
   field nivel                  // Número
   field hayMásTriángulosQueCírculos // Booleano
   field quedanRombos           // Booleano
   field fichaDeLaCualHayMás    // FormaDeFicha
}
```

Se pide programe la función **estadoActualDelJuego** que, asumiendo que hay una forma de ficha de la cual hay más cantidad que las otras, describe el estado actual del juego.

Modelo 2

Ejercicio 1)

Realice la función **fichaAleatoria** que describe la forma de ficha con la cual reponer un espacio vacío según un proceso pseudo-aleatorio, basado en las fichas alrededor (en direcciones ortogonales, es decir, línea recta) con las reglas que siguen:

- Si la ubicación a reponer tiene 2 fichas círculo, pero no un triángulo alrededor y está sobre un borde, se rellena con un triángulo.
- Si la ubicación a reponer tiene 3 fichas iguales alrededor, se rellena con una ficha estrella.
- Si la ubicación a reponer tiene al menos dos obstáculos alrededor, se rellena con un rombo.
- Cualquier otro caso se rellena con un círculo.

Ejercicio 2)

Realice la función **hayFichasConForma_EnGrilla** que dada una forma de ficha indica si hay alguna ficha con la forma dada en la grilla.

Ejercicio 3)

Sí se queman una hilera de 5 o más fichas de la misma forma, el juego aplica un bonus especial, quemando absolutamente todas las fichas con la misma forma en la grilla.

Realice el procedimiento **QuemarTodasLasFichasDe_DeLaGrilla** que dada una forma de ficha quema todas las fichas con la forma dada de la grilla, dejando la grilla libre. Notar que no se espera compruebe sí se quemaron 5 o más fichas, sino que simplemente realice el efecto posterior, dado por el bonus.

Ejercicio 4)

Se desea poder recuperar el estado del juego como un dato que se pueda manipular y analizar. Para esto, los desarrolladores definieron el siguiente tipo:

```
type EstadoDelJuego is record {
   // PROPÓSITO: Modela el estado del juego de Sharikistones.
```

```
   // INVARIANTE DE REPRESENTACIÓN: Las fichas restantes son >= 0.
   field fichasRestantes            // Número
   field hayMásEstrellasQueTriángulos // Booleano
   field quedanCírculos             // Booleano
   field fichaDeLaCualHayMenos      // FormaDeFicha
}
```

Se pide programe la función **estadoActualDelJuego** que, asumiendo que hay una forma de ficha de la cual hay menos cantidad que las otras, describe el estado actual del juego.

Modelo 3

Ejercicio 1)

Realice la función **fichaAleatoria** que describe la forma de ficha con la cual reponer un espacio vacío según un proceso pseudo-aleatorio, basado en las fichas alrededor (en direcciones ortogonales, es decir, línea recta) con las reglas que siguen:

- Si la ubicación a reponer tiene 2 fichas triángulo pero no círculo alrededor y no está sobre un borde, se rellena con un rombo.
- Si la ubicación a reponer tiene 3 fichas iguales alrededor, se rellena con una ficha triángulo.
- Si la ubicación a reponer tiene a lo sumo dos obstáculos alrededor, se rellena con un círculo.
- Cualquier otro caso se rellena con una estrella.

Ejercicio 2)

Realice la función **cantidadDeFichasConForma_EnGrilla** que dada una forma de ficha describe la cantidad de fichas con la forma dada en la grilla.

Ejercicio 3)

Sí se queman una hilera de 6 o más fichas de la misma forma, el juego aplica un bonus especial, quemando la totalidad de los obstáculos del tablero.

Realice el procedimiento **QuitarLosObstáculosDeLaGrilla** que quema la totalidad de obstáculos en el tablero. Notar que no se espera compruebe sí se quemaron 6 o más fichas, sino que simplemente realice el efecto posterior, dado por el bonus

Ejercicio 4)

Se desea poder recuperar el estado del juego como un dato que se pueda manipular y analizar. Para esto, los desarrolladores definieron el siguiente tipo:

```
type EstadoDelJuego is record {
   // PROPÓSITO: Modela el estado del juego de Sharikistones.
   // INVARIANTE DE REPRESENTACIÓN: La cantidad de triangulos es >= 0.
   field cantidadDeEspaciosLibres      // Número
   field hayMásRombosQueCírculos       // Booleano
   field quedanEstrellas               // Booleano
   field fichaDeLaCualHayMenos         // FormaDeFicha
}
```

Se pide programe la función **estadoActualDelJuego** que, asumiendo que hay una forma de ficha de la cual hay menos cantidad que las otras, describe el estado actual del juego.