Práctica 2: Nuevas Células en el Mundo

Puri Arenas
DSIC-UCM
Curso 2015/2015

Célula Compleja

Comprobamos que la célula compleja no muere (es decir que todavía no ha comido suficiente para explotar). Si muere no se hace nada. En otro caso:

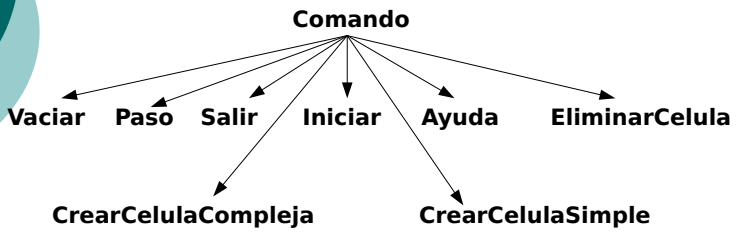
Generamos posición aleatoria en todo el tablero (no en posiciones vecinas):

a) *Si la posición está libre*, la célula compleja se mueve a esa Posición.

b) *Si la posición contiene una célula simple*, se mueve y se come a la célula simple. Se incrementa el número de células comidas.

c) *Si la posición contiene una célula compleja*, entonces no se mueve.

Jerarquía de los comandos



Clase abstracta Comando

```
public abstract class Comando {

  public abstract void ejecuta(Mundo mundo);
  public abstract Comando parsea(String[] cadenaComando);
  Public abstract String textoAyuda();

}
```

Comando Iniciar

```
public class Iniciar extends Comando {

 @Override
 public void ejecuta(Mundo mundo) {
     mundo.inicializaMundo();
 }
 @Override
 public Comando parsea(String[] palabras) {
   if (palabras.length==1){
     if (palabras[0].equals("INICIAR")) return new Iniciar();
     else return null;
   }
   else return null;
 }
 @Override
 public String textoAyuda() {
  return "  INICIAR: Inicia la población de forma aleatoria";
 }
}
```


Clase ParserComandos

```
public class ParserComandos {

 private static final Comando[] comandos =
  { new Ayuda(), new Iniciar(), new CrearCelulaSimple(),
    new CrearCelulaCompleja(),
    new EliminarCelula(), new Paso(), new Vaciar(), new Salir()
  };
 static public String AyudaComandos(){
  // Crea la ayuda del juego
 }
 static public Comando parseaComandos(String[] cadenas){
   int i=0; boolean seguir = true;
   Comando comando = null;
   while (i < ParserComandos.comandos.length && seguir){
       comando = comandos[i].parsea(cadenas);
       if (comando!=null) seguir = false;
       else i++;
   }
   return comando;
 }
}
```


Clase Controlador

```
public class Controlador {

  // atributos iguales a la Práctica 1

  // constructora igual a la Práctica 1public Controlador(Mundo

  public void realizaSimulacion(){
    while (!mundo.esSimulacionTerminada()){
        // muestra el mundo y dibuta el prompt
        // lee una linea
        // genera el array de palabras (palabras)
        Comando comando = ParserComandos.parseaComandos(palabras);
        if (comando!=null) comando.ejecuta(this.mundo);
        else System.out.println("Comando incorrecto");
    }
  }
}
```


Clase Mundo

- Similar a la Práctica 1, pero ahora contiene un nuevo atributo:
  - **private boolean simulacionTerminada**: que indica cuando termina la simulación del juego debido a que se ejecuta el comando **salir**.
  - En el método **evoluciona()**:
    - Debe controlarse que una célula no se mueva dos veces.
    - Debe recorrerse la superficie, y por cada posición **(f,c)** no vacía debe invocarse a:

      Casilla casilla = this.superficie.ejecutaMovimiento(f,c)

      // devuelve la casilla a la que se ha movido la célula,

      // o null si la célula no se ha movido.

Clase Superficie

```
public class Superficie {
  // atributos similares a la Práctica 1
  // constructora similar  a la Práctica 1

  // métodos similares a la Práctica 1, pero aparecen los
  // nuevos métodos:

  // solicita a la célula que se muera sobre ella
  public Casilla ejecutaMovimiento(int f, int c) {
     return this.superficie[f][c].ejecutaMovimiento(f,c, this);
  }
  // pregunta a la célula (fila,columna) si es comestible
  public boolean esComestible(int fila, int columna) {
    return this.superficie[fila][columna].esComestible();
  }
}
```


Clase Célula

```
public abstract class Celula {


  abstract public Casilla ejecutaMovimiento(int f, int c,
                                            Superficie superficie);
  abstract public boolean esComestible();
}
```


Clase CelulaSimple (células de la Práctica 1)

```
public class CelulaSimple extends Celula{


   public  Casilla ejecutaMovimiento(int f, int c,
                                     Superficie superficie){
     // similar a la Práctica 1, pero ahora
     // la información de la Célula está en esta clase
   }

   @Override
   public boolean esComestible() {
     return true;
   }

}
```