Práctica 4

Hilos

Programación Orientada a Objetos

Humberto Alejandro Ortega Alcocer (2016630495)

Opción: 2 “Pégale al topo (continuación)”

3 de diciembre de 2020

Grupo: 2CM1

Introducción

El juego de: “pégale al topo” es un clásico de los juegos de azar y las tan famosas “maquinitas”. El juego consiste en un tablero, a modo de matriz, en donde se tienen definidos una serie de “agujeros” por los que pueden salir los topos, el usuario deberá golpear a los mismos antes de que se agote el tiempo.

Para fines prácticos, el juego realmente no representa una gran complejidad, sin embargo ofrece una serie de problemáticas a solucionar en función de la dificultad con la que se desee trabajar. En principio, tenemos que los topos deberán mostrarse en el tablero durante una cantidad de tiempo homogénea y razonable, de ser muy corta el juego se volverá muy complicado, y de ser muy extensa, muy sencillo.

En algunos juegos de “pégale al topo” (conocido en inglés como “Whack-A-Mole”), las “partidas” tienen una duración predefinida, en dónde el usuario golpea sin cesar topos durante tres, cinco, diez minutos o más, y el marcador registrará el número de topos que fueron golpeados, sin embargo, es más fascinante jugar sin un tiempo límite de ejecución ya que, al ser un juego de mera destreza visual, se puede volver divertido jugar únicamente con el objetivo de tratar de alcanzar la mayor puntuación posible, sin límites de tiempo.

Aunque el juego original presenta un esquema concurrente en donde, conforme se van golpeando los topos éstos van desapareciendo, en nuestra versión mostraremos los topos disponibles durante 3 segundos, posterior a este tiempo, ocultaremos los topos que se encuentren en el tablero, elegiremos posiciones aleatorias para los mismos y, finalmente, los volveremos a mostrar.

Esta lógica de programación representa un reto ya que no podremos hacer uso de programación estructurada u orientada a objetos en dónde la ejecución sea

procedural, y debemos adaptar nuestro código a un modelo de ejecución concurrente.

La definición de la práctica, en palabras del profesor Roberto Tecla, es:

2.-**Pégale al topo** (continuación)
Si hizo el tablero de pégale al topo en una practica anterior agregue un hilo para sustituir (después de una cierta cantidad de segundos) el tablero actual por un tablero con una nueva distribución aleatoria de topos.

Desarrollo

Para el desarrollo del juego de “pégale al topo”, realmente no es necesario separar la lógica de ejecución de la interfaz gráfica puesto que, tanto la lógica como la interfaz, son elementos extraordinariamente simples si logramos abstraer las complejidades del juego apropiadamente.

Sin embargo, el diagrama representativo de la clase principal, en formato UML se describe a continuación:

Si bien, existe otra clase dentro del modelado que propongo, me parece irrelevante presentarla junto a un diagrama UML ya que dicha clase es únicamente para instancias un hilo y me gustaría explorarla más desde una perspectiva

funcional ya que, en dicha clase, es dónde se realiza la redistribución de los “topos” en el tablero con el fin de volver aleatorio el comportamiento del programa.

Para comenzar, me gustaría dar un vistazo al constructor de la clase topo:

```
public topo() {
  super("topo");
  botones = new JButton[20];
  m = new int[20];
  punt = new JLabel("Puntuación: 0");
  setLayout(new GridLayout(5, 5));
  for (int i = 0; i < botones.length; i++) {
    r = Math.random();
    if (r < 0.5) {
      m[i] = 1;
      add(botones[i] = new JButton(new ImageIcon("tapa.jpg")));
      botones[i].addActionListener(this);
      botones[i].setEnabled(true);
    }
    if (r ≥ 0.5) {
      m[i] = 0;
      add(botones[i] = new JButton(new ImageIcon("topo.jpg")));
      botones[i].addActionListener(this);
      botones[i].setEnabled(true);
    }
  }
  add(punt);
  hilo TOPO = new hilo();
  TOPO.start();
  setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
  setSize(600, 600);
  setVisible(true);
}
```

Como se puede observar, es un constructor muy sencillo, los pasos que realiza el constructor son simples:

- Inicializa la interfaz gráfica
- Crea los botones (donde vivirán nuestros topos)
- Asigna las posiciones iniciales aleatorias de nuestros topos
- Inicializa el hilo dónde se redistribuirá nuestro tablero

Es importante notar que, para esta práctica, se han predefinido las longitudes de los arreglos y las dimensiones de la ventana, esto ha sido en gran parte para simplificar el proceso de creación de la misma, así como tener un marco de referencia sólido sobre el cuál trabajar para determinar las posiciones y las acciones realizadas por el usuario durante el juego.
Hacia el final del constructor, se iniciativa el hilo, este hilo transcurre de forma permanente durante la ejecución del juego, el código que lo describe se muestra a en la siguiente página.

```java
public class hilo extends Thread {
  public void run() {
    while (true) {
      try {
        Thread.sleep(3000);
        for (int i = 0; i < botones.length; i++) {
          r = Math.random();
          if (r < 0.5) {
            m[i] = 1;
            botones[i].setIcon(new ImageIcon("tapa.jpg"));
            botones[i].setEnabled(true);
          }
          if (r ⩾ 0.5) {
            m[i] = 0;
            botones[i].setIcon(new ImageIcon("topo.jpg"));
            botones[i].setEnabled(true);
          }
        }
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
```

En este código, es muy simple ver su modelo de ejecución.

Todo el código se encuentra contenido dentro de un ciclo infinito, el cual se ejecutará durante todo el tiempo que nuestro programa siga corriendo. Dentro del ciclo, pausamos la ejecución de nuestro hilo durante 3000ms, es decir, 3 segundos. Posteriormente procedemos a re-distribuir el tablero empleando exactamente el mismo algoritmo utilizado en el constructor.

Todo el código del hilo se encuentra dentro de un bloque `try-catch`, el cual sirve para “atrapar” errores en tiempo de ejecución y reaccionar ante ellos. Dado que nuestro programa es uno muy simple, realmente no esperamos complicaciones con la ejecución del hilo por lo que simplemente imprimimos en la terminal su pila de ejecución para que, en caso de presentarse un error, podamos determinar de forma rápida dónde se originó.

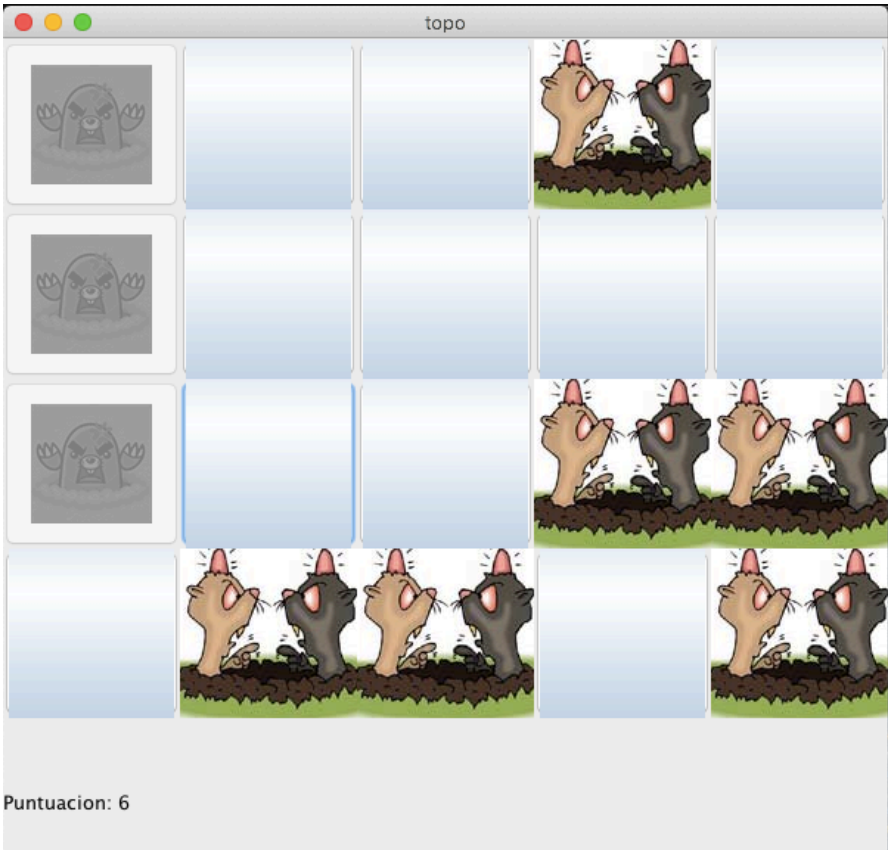
La última parte a revisar de nuestro juego de “pégale al topo”, es la interacción con el usuario, para esto, nos referimos al siguiente método:

```
public void actionPerformed(ActionEvent e) {
  JButton btn = (JButton) e.getSource();
  for (int j = 0; j < botones.length; j++) {
    if (btn == botones[j]) {
      if (m[j] == 0) {
        botones[j].setIcon(new ImageIcon("pegatopo.jpg"));
        botones[j].setEnabled(false);
        cta++;
        punt.setText("Puntuación: " + cta);
      }
    }
  }
}
```

En este método, es dónde realizamos la acción de “golpear” al topo. Para ello, al recibir un evento (un click), determinamos el origen, es decir, cuál fue el botón que se presionó.

Con la información del botón que ha sido presionado, procedemos a iterar dentro de nuestros botones en el tablero, y, en cuanto encontremos el botón en cuestión

que fue presionado, verificamos si se trató de un topo, una tapa o un “topo golpeado”. Si fuera una tapa o un topo golpeado, dejamos el botón sin hacer nada, pero en caso de que el botón corresponda a un topo, procedemos a colocar la imagen para el topo golpeado, desactivamos el botón, aumentamos nuestro contador de puntuación, y actualizamos el texto mostrado con la puntuación. Finalmente nuestra interfaz de usuario para el juego se muestra de la siguiente manera:



Conclusión

El juego de “pégale al topo” siempre ha sido uno de mis favoritos en los centros de entretenimiento como Recórcholis, puedo pasar horas y horas golpeando a los (pobrecitos) topos sin parar. Para el desarrollo de esta práctica intenté replicar la emoción y adrenalina que uno puede llegar a sentir cuando se enfrenta al juego original.

Si bien existen muchas similitudes entre el juego original y mi práctica, quedan pendientes aún varias cosas por implementar que le darían al juego una sensación mucho más alta de credibilidad, sin embargo éstos pendientes se refieren a detalles que, siendo muy honestos, me parecen que difieren de los objetivos de la presente práctica.

Trabajar con hilos es algo que, como programadores, incrementa nuestro panorama de alcances y posibilidades dentro del ramo, es importante siempre tener claro cómo, cuándo, dónde y por qué emplear cierta tecnología dentro de nuestros programas, y el juego de “pégale al topo” no puede ser sino la opción más acertada en visibilidad de la necesidad de utilizar un modelo concurrente de ejecución.

En un futuro, será interesante comenzar a combinar los conocimientos de las prácticas en algún juego mucho más completo en dónde, sin lugar a dudas, el esquema concurrente será de vital importancia puesto que, así como en el mundo real, en los videojuegos constantemente están sucediendo múltiples cosas al mismo tiempo.