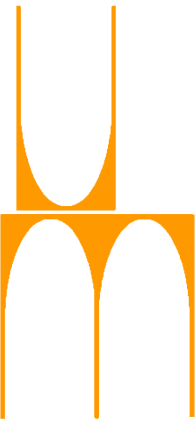
REPÚBLICA BOLIVARIANA DE VENEZUELA

UNIVERSIDAD METROPOLITANA

ASIGNATURA: ESTRUCTURAS DE DATOS



**PROYECTO #1:**

**GENERADOR DE LABERINTOS**

**(*MazeGenerator*)**

**Autores:**

Marco Suárez. 20171110890

Luis Torres. 20151110502

Daniel Marques. 20161120579

**Prof.:** Christian Guillén

28 de octubre de 2017

**Introducción**

*MazeGenerator* es un software generador de laberintos implementado con Java, que utiliza como principal recurso lógico el algoritmo *Aldous-Broder*. Este algoritmo de creación de laberintos se caracteriza por tomar decisiones de forma aleatoria, por lo cual, sus laberintos (dependiendo de sus dimensiones) pueden tener un tiempo de generación impredecible.

Además de generar laberintos a partir del anterior algoritmo (existente posterior a la implementación del programa), también se presenta de manera amigable al usuario a través de una interfaz gráfica. Los laberintos generados se pueden guardar en archivos de texto, que el usuario podrá consultar cada vez que lo desee, incluso si ya ha cerrado el programa. El usuario, al iniciar el programa tendrá para sí una ventana que le permite especificar el nombre del directorio en el que se guardarán los algoritmos que genere, junto a sus dimensiones y el tiempo que tardó en generarse. Los laberintos en sí también son impresos en los archivos de texto, a través de los caracteres “|” y “-“. La representación de los laberintos por medio de caracteres no se ve de la manera óptima con la mayoría de las fuentes, para resolver este inconveniente, en la sección de “Ayuda” del programa se le recomienda al usuario la fuente más idónea para la representación textual de los laberintos.

*MazeGenerator* de fácil manejo, y tiene una gran utilidad, ya que la creación de laberintos de forma manual puede ser muy engorrosa y toma mucho tiempo. Además, el programa es escalable; no está limitado a las dimensiones que tiene establecidas, puede mostrarse en prácticamente cualquier resolución, y su código sirve como instrumento didáctico.

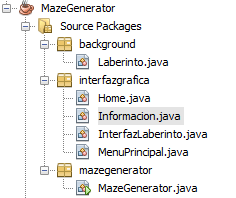
**Estructura básica para usuario**

*MazeGenerator* presenta se presenta al usuario con las siguientes ventanas:

1. Ventana principal: es la bienvenida del programa. Permite al usuario especificar el directorio en donde guardará los laberintos después de hacer clic en “Comenzar”. También trae dentro de sí una opción (botón “Ayuda”) que presenta al usuario una ventana de información e instrucciones de uso.
2. Ventana de Ayuda: expone al usuario de forma básica qué es lo que está usando y cómo hacerlo, acompañado de recomendaciones de uso.
3. Ventana de dimensiones: es donde el usuario especificará las dimensiones del laberinto a generar.
4. Ventana de laberinto: muestra gráficamente el laberinto generado a partir de las instrucciones especificadas.

**Estructura básica del programa**

*MazeGenerator* es un programa desarrollado con Java, así que está netamente implementado con orientación a objetos. Esta son las clases propias del programa y su organización:



* Paquete “background”
  + Clase “Laberinto”
    - Creación del laberinto: contiene el método “aldousBroder”.
    - Impresión del laberinto en archivo de texto.
    - Creación de instrucciones para la representación gráfica del laberinto: arreglos “starts” y “ends”.
    - Toma tiempo de generación del laberinto.
* Paquete “interfazgrafica”
  + Clase “Home”: implementación de la ventana principal del programa.
  + Clase “Informacion”: implementación de la ventana de información y ayuda del programa,
  + Clase “InterfazLaberinto”: genera de forma gráfica el laberinto, también implementa la ventana que lo muestra.
  + Clase “MenuPrincipal”: implementa la ventana donde el usuario selecciona las dimensiones del laberinto y es quien da luz verde para generarlo.
* Paquete “mazegenerator”
  + Clase “MazeGenerator”: contiene el método “main” y crea una instancia de la clase “Home” para iniciar el programa.

**Algoritmo Aldous-Broder**

El algoritmo Aldous-Broder está hecho exclusivamente para la generación de laberintos creado por David Aldous y Andrei Broder. El algoritmo se realiza dentro de una matriz o estructura dividida en celdas. Consiste en los siguientes pasos:

1. Seleccionar cualquier celda de forma aleatoria.
2. Elegir una celda adjunta a la seleccionada (no diagonal). Si la celda seleccionada no ha sido visitada, se marca como tal.
3. Repetir el paso 2 hasta que todas las columnas hayan sido visitadas. (J. Buck. *The Buckblog*).

La corrida en frío de Aldous-Broder se puede realizar en el sitio web de Jamis Buck (ver bibliografía).

**Estructuras de datos en MazeGenerator**

* Arreglos: es la estructura de datos con la que trabaja el algoritmo Aldous-Broder en MazeGenerator. El algoritmo se ubica y traslada dentro de una matriz, y esta se utiliza para trazar el laberinto. Las instrucciones para el trazado del laberinto también están representadas con arreglos (“starts” y “ends”), se hablará de ellos más adelante.
* Clases: Java es el lenguaje de programación orientada a objetos por excelencia. Todo dentro de MazeGenerator son clases, objetos, métodos, atributos, entre otros elementos.

**Arreglos “starts” y “ends”**

El algoritmo ha generado un laberinto y ya se almacenó en el archivo de texto, ahora, ¿cómo se dibujará? La clase “Laberinto” es la que ejecuta el algoritmo para generarlo, y la clase encargada de dibujarlo y mostrarlo es “InterfazLaberinto”. Estas dos clases no trabajan en paralelo, así que para que estas se comuniquen la clase “InterfazLaberinto” necesita instrucciones para poder generar el laberinto de manera gráfica.

Al mismo tiempo que genera el laberinto, “Laberinto” llena una serie de instrucciones que no son más que puntos de inicio y fin que representan la trayectoria que siguió el algoritmo en la generación del laberinto. Esos puntos son almacenados en dos vectores bidimensionales llamados “starts” y “ends”.

¿Por qué se necesitan puntos de inicio y fin? El trazado gráfico del laberinto se realiza con líneas rectas, y para crear una se necesitan dos puntos. Las líneas son generadas gracias a la clase “Graphics” del paquete “java.awt”. El trazado de líneas rectas en 2D se realiza a través del método “drawLine()” perteneciente a la clase “Graphics2D”, subclase de “Graphics”. El método recibe cuatro parámetros, que corresponden a las coordenadas en 2D dentro del “JPanel” para dibujar una línea recta. Supongamos que tenemos un objeto de tipo “Graphics2D” llamado “g2”, haríamos que dibuje una línea recta de la siguiente manera:

,

g2**.**drawLine**(**x1**,** y1**,** x2**,** y2**);**

donde x1 e y1 son el punto de inicio (*start*) y x2 e y2 son los puntos de fin (*end*). “Laberinto” establece nuevos puntos de inicio y fin solo cuando marca una celda como visitada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | starts | | ends | |
| filas | columnas | filas | columnas |
| \*marca como visitada” 🡪 | 1 | 3 | 1 | 4 |
| \*marca como visitada” 🡪 | 2 | 1 | 3 | 1 |
| \*marca como visitada” 🡪 | 4 | 5 | 5 | 5 |

Nótese que los puntos de inicio y fin distan únicamente entre celdas y columnas adyacentes.

Los arreglos “stars” y “ends” son de carácter *static*, ya que deben ser compartidos por distintas clases.

**Conclusiones**

La disciplina de la programación requiere de constante aprendizaje, y la mejor manera de aprender a programar es por medio de práctica constante y experimental, ya que existen disciplinas cuya mejor forma de dominarlas es haciéndolas, y la programación es una de ellas.

El desarrollo de MazeGenerator ha sido de gran utilidad para sus autores; es un programa que requiere de gran conocimiento del paradigma orientado a objetos, habilidad de resolución de problemas, investigación, y lo más importante, mucha imaginación. Es imposible crear un programa como MazeGenerator si no se usa la imaginación, una hoja de papel y un lápiz.

El código de MazeGenerator también sirve como instrumento didáctico, ya que utiliza el paradigma orientada a objetos para un proyecto real, cercano a lo que sería un desarrollo profesional. Demuestra lo necesaria que resulta la imaginación para crear software, y la aplicación de las matemáticas y la geometría en la programación.

La implementación de MazeGenerator otorgó un gran aprendizaje a sus autores y estos pueden asegurar que se llevaron algo de la asignatura correspondiente, además de algo para su currículum.

**Agradecimientos**

* Juan Gómez (España), por sus tutoriales en su canal de YouTube “pildorasinformaticas”.
* Jorge Bousaad, preparador de Estructuras de Datos de la UNIMET.

**Bibliografía**

* Gómez, J [pildorasinformaticas]. (2014, 5 de mayo). *Curso Java.* [vídeos de YouTube]. Disponible en: https://www.youtube.com/watch?v=coK4jM5wvko&list=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2lk
* Buck, J. (2011, 17 de enero). *Maze Generation: Aldous-Broder algorithm.* Disponible en: http://weblog.jamisbuck.org/2011/1/17/maze-generation-aldous-broder-algorithm.html
* API de Java (edición 7). *Java Platform, Standard Edition 7 API Specification.*Disponible en: https://docs.oracle.com/javase/7/docs/api/