PROYECTO: Ciudad del caos.

Preparado por: Giovanni Fajardo Utria.

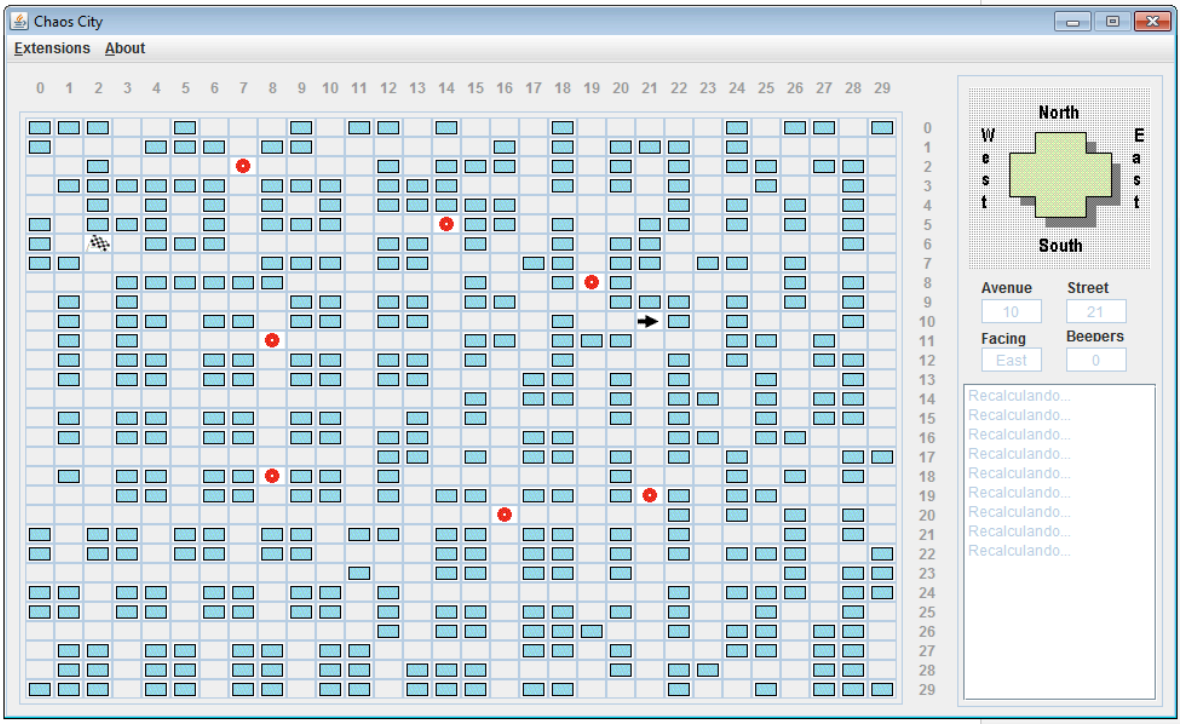
DESCRIPCION.

Ciudad del Caos es una ciudad representada en un mundo cuadriculado con avenidas y calles. Las avenidas aumentan de norte a sur y las calles de occidente a oriente. En este mundo virtual existen otros objetos tales como un automóvil que se representa por una flecha negra y las edificaciones se escenifican por cuadros azules, el lugar de destino o meta es simbolizado por una bandera a cuadros negros y blancos y los trancones que se reconocerán por círculos rojos.

El automóvil tiene ciertas capacidades que le permiten dirigirse de un lugar a otro. Estas capacidades se listan a continuación:

- Moverse hacia a delante.
- Girar a la izquierda o hacia la derecha.
- Identificar si el frente se encuentra despejado.
- Brújula que le permite conocer su orientación.

Interfaz gráfica del proyecto.



En la ciudad interna del programa, los edificios son representados por una "**B**", los espacios en blanco o vías de la ciudad "**.**", la meta de llegada por una "**H**" y los obstáculos o puntos rojos por una "**O**".

PROYECTO.

El objetivo del proyecto es desarrollar un algoritmo basado en búsqueda en anchura para encontrar la ruta más corta para ir de un lugar a otro. Como datos de entrada se debe suministrar un mundo previamente elaborado en donde se especifiquen las edificaciones, la ubicación del automóvil y el lugar de destino. Durante la ejecución del algoritmo el usuario debe poder agregar nuevos obstáculos "trancones" que bloqueen la ruta del automóvil y de esta forma simular el

comportamiento de un GPS recalculando la nueva ruta para llegar al destino desde la nueva posición. Cuando el automóvil llegue al destino trazado, el automóvil se detiene y termina la ejecución del algoritmo.

INTERFAZ DEL PROGRAMA.

La interfaz previamente diseñada permite realizar las siguientes operaciones:

- ✓ Guardar el diseño de una ciudad.
- ✓ Leer y cargar el diseño de una ciudad.
- ✓ Diseñar la ciudad del caos.
  - Clic derecho en una casilla vacía, coloca un edificio “cuadro azul”.
  - Clic derecho en una casilla con edificio, elimina el edificio.
  - Clic izquierdo en una casilla vacía, inicia la ubicación de un número.
  - Clic izquierdo en una casilla con número, incrementa este número en uno.
  - Clic derecho en una casilla con número, disminuye este número en uno.
- ✓ Ubicar la meta en la ciudad, Shif+botón derecho.
- ✓ Ubicar el vehículo en la ciudad. Shif+botón izquierdo.

- ✓ Los cuadros azules representan las edificaciones de la ciudad.
- ✓ Los números se interpretan de la siguiente forma:
  - 0: puede avanzar hacia cualquier dirección.
  - 1: Obligatorio girar a la derecha.
  - 2: Obligatorio girar a la izquierda.
  - 3: No es doble vía.

- ✓ Colocar obstáculos en la vía, “círculos rojos” Shift+botón izquierdo.
  Los círculos rojos u “obstáculos” se deben colocar en ejecución mientras el vehículo se encuentra en movimiento para simular el dinamismo de la ciudad, es decir, el acceso a las vías puede cambiar y con esto el vehículo debe recalcular la nueva ruta desde la nueva posición o determinar si no es posible llegar al destino en las condiciones actuales.

CONVENCIONES.

Para la ejecución del proyecto deben tener en cuenta las siguientes reglas:

- ✓ El estudiante debe realizar el algoritmo en el método findPath( ) de la clase Execution que se entrega junto con la interfaz.

- ✓ El intérprete del vehículo de la ciudad del caos atiende los siguientes comandos:
  - (0): Cero, lo interpreta como moverse hacia delante de acuerdo a su orientación.
  - (1): Uno, lo interpreta como girar a la izquierda de acuerdo a su orientación.
  - (2): Dos, lo interpreta como girar a la derecha de acuerdo a su orientación.

- ✓ "lst", es la lista de comandos que el algoritmo diseñado debe completar y que el vehículo seguirá automáticamente.

```
public void findPath()
{ lst.add("2"); // right
  lst.add("0"); lst.add("0"); lst.add("0"); lst.add("0");
  lst.add("0"); lst.add("0"); lst.add("0"); lst.add("0");
  lst.add("2"); // right
  lst.add("0"); lst.add("0"); lst.add("0"); lst.add("0");
  lst.add("1"); // left
  lst.add("0"); lst.add("0"); lst.add("0"); lst.add("0");
  lst.add("2"); // right
  lst.add("2"); // right
}
```

En el código anterior se puede apreciar una secuencia de ejemplo que sigue el vehículo automáticamente. Esta secuencia debe ser generada *"de forma inteligentemente"* de acuerdo a la situación que se presente durante la ejecución del algoritmo.

- ✓ Los obstáculos o "círculos rojos", son detectados por el vehículo y automáticamente se transfiere el control al método findPath( ) para recalcular la nueva ruta desde la ubicación actual.