Segundo Trabajo de Bases de Datos 2
Semestre 2014 – 1

Alexander Soto 1026150273
Gustavo Angarita 1037635327
Gustavo Preciado 1037635880

Explicación del desarrollo de los puntos

Punto 1

En la ventana principal de la interfaz se presentan dos JComboBox que permiten seleccionar una ciudad y un escenario (los nombres de las ciudades, sus ID's y el número de escenarios de cada ciudad se extraen de la base de datos). Al seleccionar un escenario de una ciudad y presionar el botón graficar, se muestran todos los edificios y huecos del escenario con sus respectivas imágenes de la tabla que contiene los íconos.

Al momento de graficar se cargan todos los íconos de la ciudad en un HashMap que contiene el nombre del elemento y su ícono (en una BufferedImage). Para obtener la URL y el nombre del elemento se realiza una consulta en la tabla icono utilizando la función EXTRACTVALUE sobre el XMLTYPE que corresponde a la ciudad.

Para obtener todos los edificios y huecos (en adelante llamados “elementos”) se utiliza el hecho de que los elementos de un escenario no se pueden superponer: se extraen de la tabla anidada las coordenadas X1 y Y1 del último elemento del escenario seleccionado y luego se recorre cada fila de la misma tabla, guardando las cuatro coordenadas de cada elemento hasta volver a encontrar las del último. A partir de esta información se generan objetos de la clase Elemento que contienen el rectángulo con la esquina superior izquierda en las coordenadas X e Y más cercanas al origen, y con ancho y alto iguales al valor absoluto de la diferencia entre X1 y X2, e Y1 e Y2, respectivamente.

Luego se procede a graficar las listas de edificios y huecos, y se agrega el ícono y nombre a cada edificio. Adicionalmente, se genera una matriz que representa el escenario, con posiciones que van desde el 0 hasta el 100. Los bordes del escenario y las coordenadas internas de los huecos se llenan con -1, los puntos al interior de los edificios se llenan con 1 y los demás puntos se rellenan con ceros (esta matriz es usada en los demás puntos del trabajo).

Se utilizó un ejemplo de transformaciones afines presentado en [1] para permitir efectos de zoom, y dragging del escenario (estas funcionalidades se encuentran en la clase Graficador).

Punto 2

Para ubicar los taxis en el escenario se utilizó la matriz descrita en el punto anterior. Se pide al usuario que ingrese la cantidad de taxis que desea y luego se cuenta la cantidad de posiciones disponibles en el escenario (puntos con 0 en la matriz). Si el número ingresado es mayor a la cantidad máxima de espacios disponibles, en lugar de generar la cantidad de taxis pedida por el usuario se genera la cantidad máxima. Si el número es menor a 0, no se grafican taxis. Para ubicar los taxis en el escenario se generan números aleatorios para las coordenadas X e Y y se comprueba que el punto esté libre (que tenga 0 en la matriz); si no lo

está se produce una nueva posición de forma aleatoria y se vuelve a comprobar la disponibilidad del espacio. Este proceso se repite hasta hallar un lugar en el que se pueda introducir el taxi, luego se añade un taxi con esas coordenadas a la lista que se guarda en un objeto de la clase Graficador, y se marca con un 2 esa posición de la matriz. Tras generar con el proceso anterior la cantidad pedida de taxis, se grafica toda la lista en el escenario.

Punto 3
En este punto se utilizó un Thread (Hilo) que corre en paralelo ya que la espera de 3 segundos que se pedía detendría el funcionamiento del programa si se realizara en el hilo principal.
Para mover los taxis se recorre la lista de taxis y se comprueba qué movimientos disponibles tiene cada uno (posiciones adyacentes horizontal o verticalmente cuyo valor absoluto en la matriz sea diferente de 1). Las posibilidades se guardan en una lista y luego se escoge una de manera aleatoria. Al cambiar la posición del taxi, se pone dentro de la matriz un 0 en su posición anterior y un 2 en su nueva posición.
Se agregó un botón “Parar” a la interfaz que sirve para detener y reanudar el movimiento.

Punto 4
Para realizar este punto se añadió un botón a la interfaz que al ser presionado pide al usuario las coordenadas de un punto en el escenario. Si el punto ingresado no está al interior de un hueco, se procede a buscar el taxi más cercano al punto recorriendo la lista de taxis y seleccionando al que esté a menor distancia en línea recta. Teniendo en cuenta que la ruta debe atravesar mínimo un edificio, se busca en la matriz el 1 que se encuentra a menor distancia del taxi y luego, mediante un algoritmo de búsqueda en anchura adaptado para tomar como raíz del grafo la posición ingresada, se halla una de las rutas óptimas desde la posición del taxi hasta la pedida por el usuario (que pase además por el interior del edificio mencionado y que no implique movimientos en diagonal). Esta ruta se almacena en una lista de puntos y luego se grafica en el escenario. El taxi recorre luego esta ruta, eliminando el último punto de la misma cada vez que avanza una posición. Al llegar al punto solicitado, se pide al usuario que ingrese el destino del taxi y se realiza un procedimiento similar al anterior para llegar a dicho punto. Para contar los edificios y huecos que se atravesaron, contamos una lista donde se guarda un HashSet de puntos por edificio, es decir, cada elemento con qué puntos cuenta para ver luego por qué puntos pasó la ruta (cada elemento se cuenta solo una vez). El movimiento del taxi se controla en el hilo mencionado en el punto 3.

Punto 5
Para realizar este punto se añadió otro botón a la interfaz que al ser presionado pide al usuario las coordenadas de un punto en el escenario. Si el punto ingresado no está al interior de un hueco, se procede a buscar el taxi más cercano de la misma forma que en el punto anterior. Luego, mediante un algoritmo de búsqueda en anchura adaptado para tomar como raíz del grafo la posición ingresada, se halla una de las rutas óptimas desde la posición del taxi hasta la pedida por el usuario, esta vez sin atravesar ningún edificio ni hueco y sin pasar por el borde del escenario (esto se logra al eliminar del “grafo” los puntos prohibidos). Esta ruta se almacena en una lista de puntos y luego se grafica en el escenario con un color diferente. El taxi recorre luego esta ruta, eliminando el último punto de la misma cada vez que avanza una posición. Al llegar al punto solicitado, se pide al usuario que ingrese el destino del taxi y se realiza un procedimiento similar al anterior para llegar a dicho punto. El movimiento del taxi se controla en el hilo mencionado en el punto 3.

Cibergrafía:

[1] B. T. Vander Zanden, “Geometric Transforms.” [Online]. Available: http://web.eecs.utk.edu/~bvz/gui/notes/transforms/transform.html.