UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS
DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN
DEPARTAMENTO DE CIENCIAS COMPUTACIONALES
INGENIERÍA EN COMPUTACIÓN

INTELIGENCIA ARTIFICIAL I

SECCION: D01

CLAVE: I7038

Nombre: Elizade- Loera Felipe de Jesus

Codigo: 211715281

Actividad 1: Bresenham

Contenido

Link

El siguiente es un enlace para poder ver un video que muestra el funcionamiento del programa en donde generamos varios laberintos en donde vemos el mono link tratar de buscar la salida utilizando el algoritmo de bresenham: https://youtu.be/vFMfIkFxbFQ

Objetivo

En esta tarea se tendra que crea un programa que pueda crear un laberinto en donde puedamos poner un mono y una salida. Esto mono tendra que utilizar el algortimo de bresenham para llegar a esta salida. El mono se podra mover 8 direction en la siguientes directiones si es possible:

- North
- NorthWest
- West
- SouthWest
- South
- SouthEast
- East
- NorthEast

Esto program tendra que tener un interfaz para poder visualizar el laberinto y tendra un menu en donde vamos a poder declarar el tamaño del laberinto a generar y podramos declarar si vamos a querer generar muros que van a dificultar el movimiento del mono.

Bresenham

El algoritmo de bresenham es un metodo para el trazado de lineas en un dispositivo grafico. Una de sus caracteristicas principales es que este algoritmo realiza calculos con enteros. Este algoritmo tambien se puede adaptar para rasterizar circunferencias y curvas. Los ejes verticales muestran las posiciones de rastreo y los ejes horizontales identifican columnas de pixel.

Desarrollo

Interfaz

Para poder desarrollar esta tarea se utiliza el lenguaje de programacion Java con el IDE Eclipse. Para generar el interfaz se utilizo JButtons, Jsliders,Jlabels, Jpanels, Jframes y Gridbox Layouts. Podran ver en la siguiente figura un captura de la interfaz.

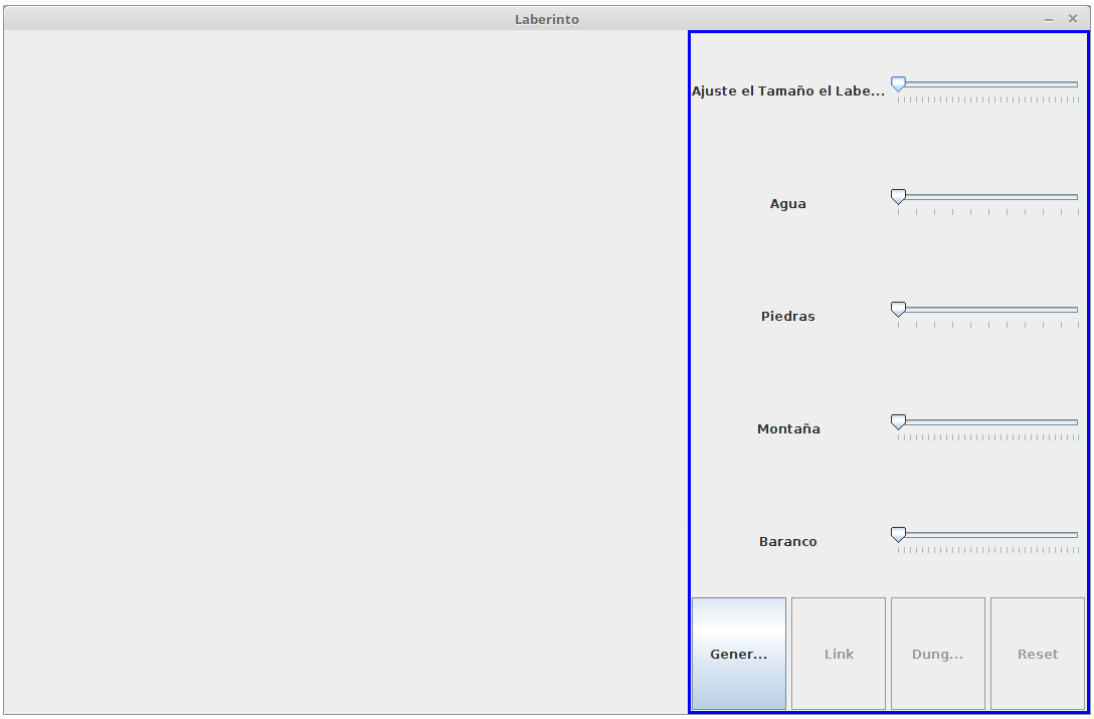


Figura 1: Interfaz Grafica

En esta inferfaz veran que tenemos varios sliders. Los uniocos sliders que vamos a utilizar con el "Ajuste el Tamaño del Laberinto" y Piedras. Todos los botones se van a utilizar. El que dice generar, generara el laberinto aleatorio. El boton ode Link dejara el usuario colocar el mono Link en la casillas que quiera excepto si es una piedra/muro. El boton "Dungeon" es la salida, cuando presionado el usuario podra colocar la salida. Y el boton "Reset" borra el laberinto por si deseas generar otro laberinto y probar el programa otra vez.

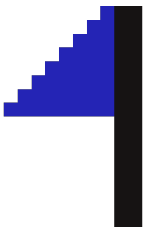
Tipo de Datos

Para poder resolver este problema se penso utilizar un arreglo de Jlabels para mostrar la imagen que esta en cada casilla. Se creo un vector de 3 capas. Esto significa que se genero una matriz en donde cada casilla pueda guardar varios valores. Los valores que guarda cada casilla son los siguientes:

- Tipo de Imagen: Puede ser piedra/muro o pasto.
- Nivel de visitado: Empieza en 0 pero puede llegar a 3

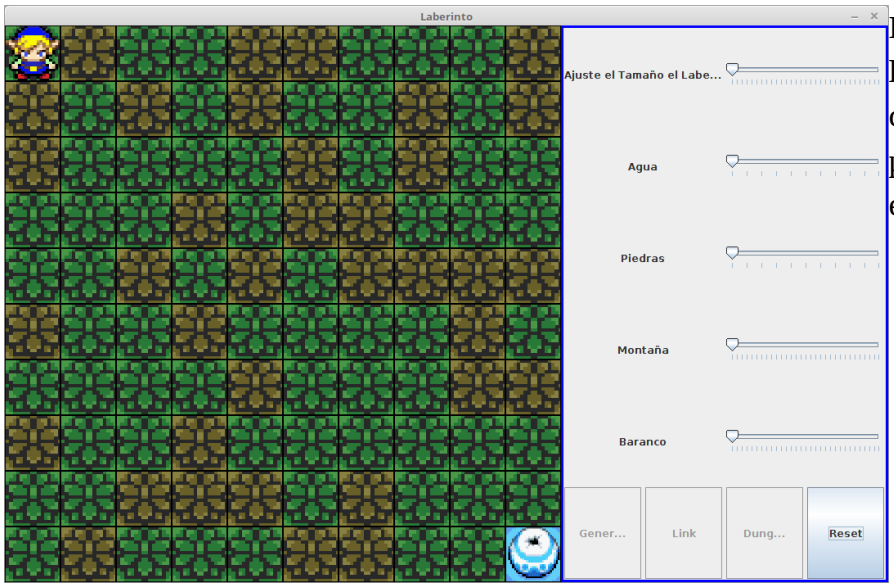
Solucion

Para poder resolver este problema se utiliza el algoritmo de bressinham para que el mono llegue a la salida pero se generan problemas cuando hay un muro en su camino. Cuando hay muros, utilizo un metodo en donde el mono escoge al azar un direccion si es valida va hacia ella si no si quita esa direccion y elige otra vez. Esto si reptie hasta que encuentra un direccion a la que puede ir. Si no encuentra se termina el recorrido por que no encontro la salido o por que se perdio en el laberinto. Cada vez el mono va a una direccion aumenta el nivel de visitado y cuando se va a otra casilla, la casilla anterior muestra el nivel de visitado con un banderin. Utilizamos los siugientes banderines:


- Amario: Nivel 1 de visitado
- Azul: Nivel 2 de visitado
- Rojo: Nivel 3 de visitado

Cuando una casilla llega a tener el banderin Rojo el mono ya no podra visitar esa casilla. Esto puede ocasionar que el mono no encuentra la salida y que se encierra como veran mas adelante. El programa tiene dos estados que utilza para en contrar la salida si es posible. El primero es usar bresinham cuando es posible, si no es posible escoge una casilla al azar, si no es posible irse a otra casilla termina el recorrido.



En la siguiente imagen veran un laberinto sin piedras/muros y como el programa utiliza el primer estado bressinham para encontrar la salida.

Figura 2: Antes de Comenzar

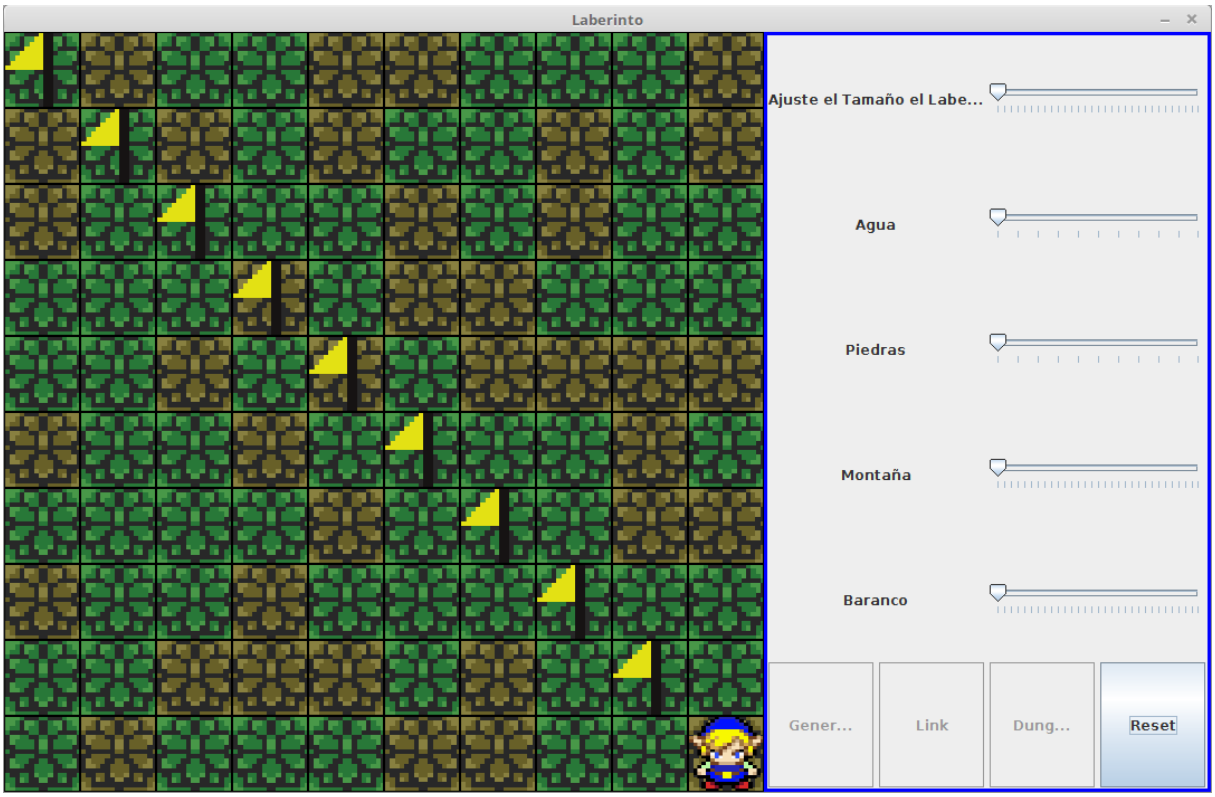


Figura 3: Encontrar salida

Como veran en la figura 2 y 3 se utilizo el estado 1, siendo bressinham para encontrar la salida nunca se utiliza el segundo estado porque no hubo piedras/muros en su camnio.

En las siguientes imganes veremos un laberinto mas grande pero con piedras/muros y como el mono utilizar los dos estados.

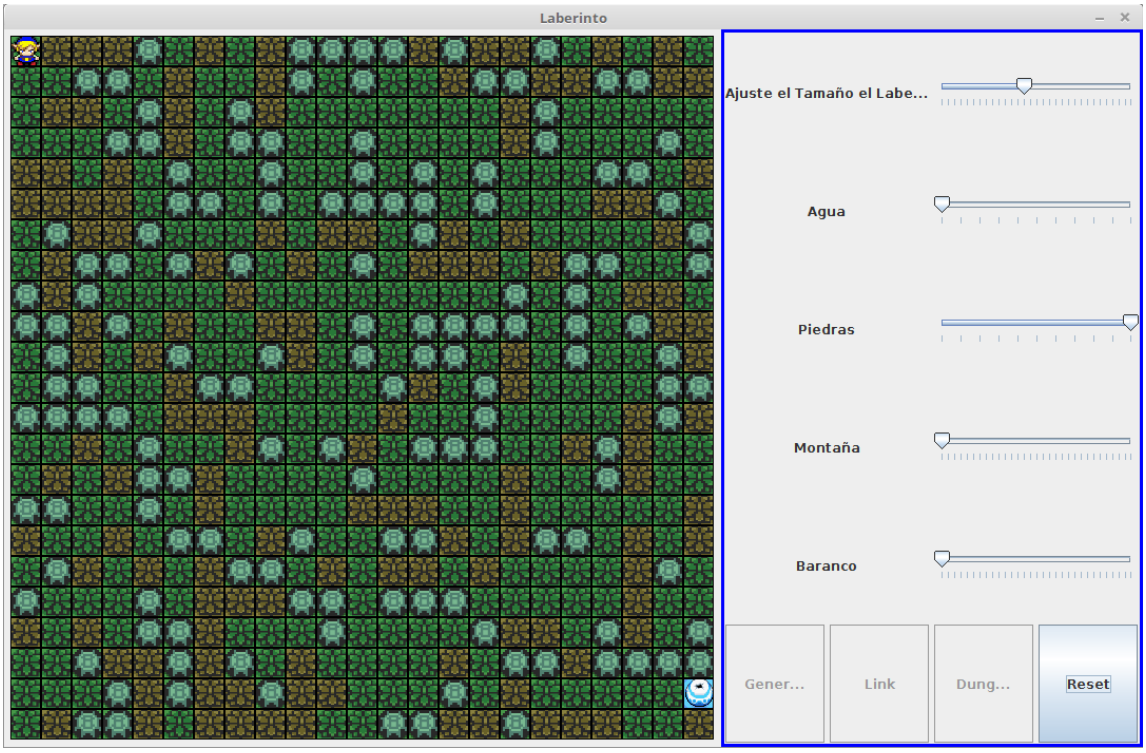


Figura 4

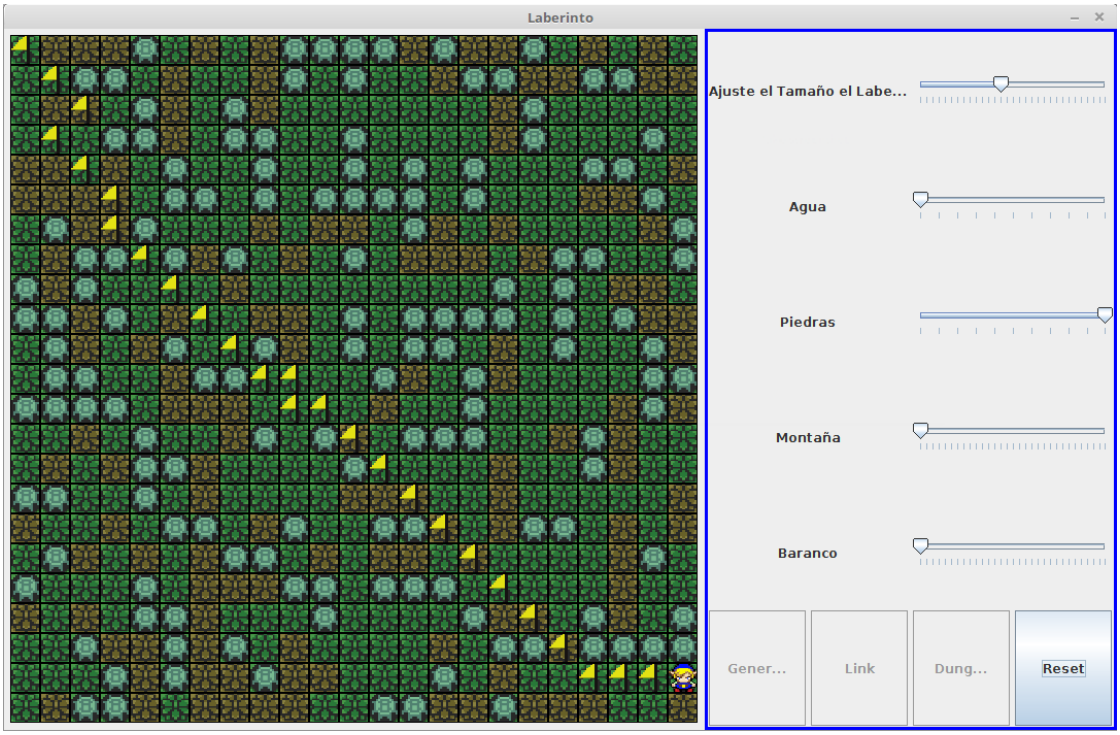


Figura 5

En la figura 4 y 5 podran ver que se utiliza los dos estados, aunque el segundo estado se utiliza no mas 3 veces. Como veran el mono nunca se visita la posicion mas de 1 una vez. En las siguientes figuras veran que el mono no llega a la salida.

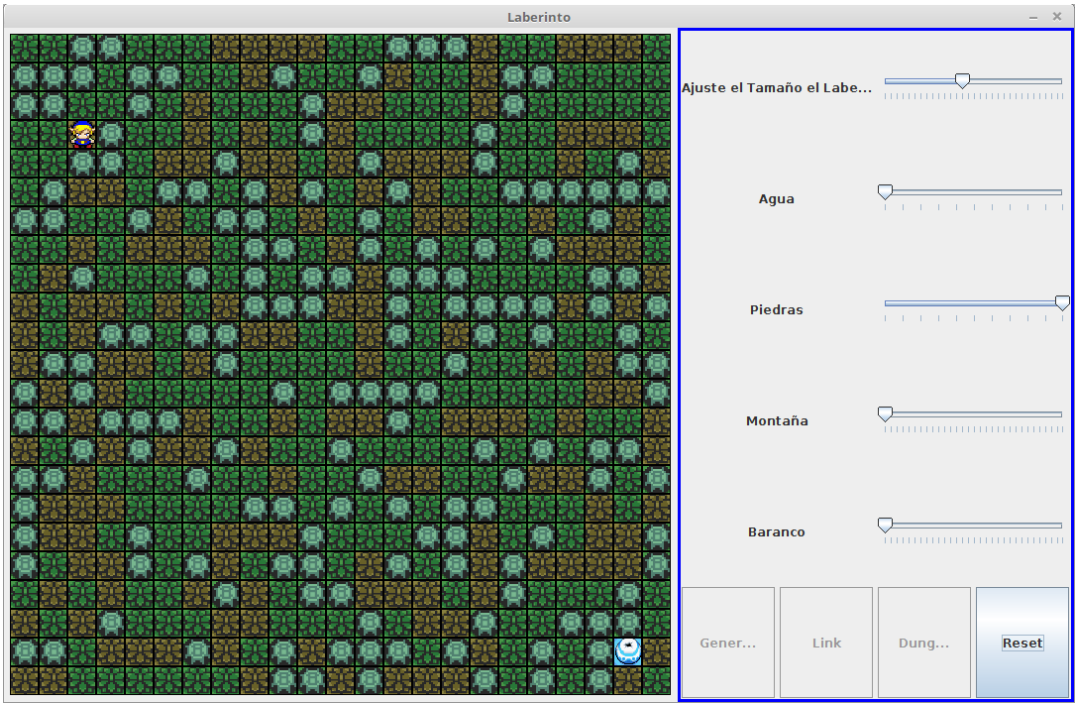


Figura 6

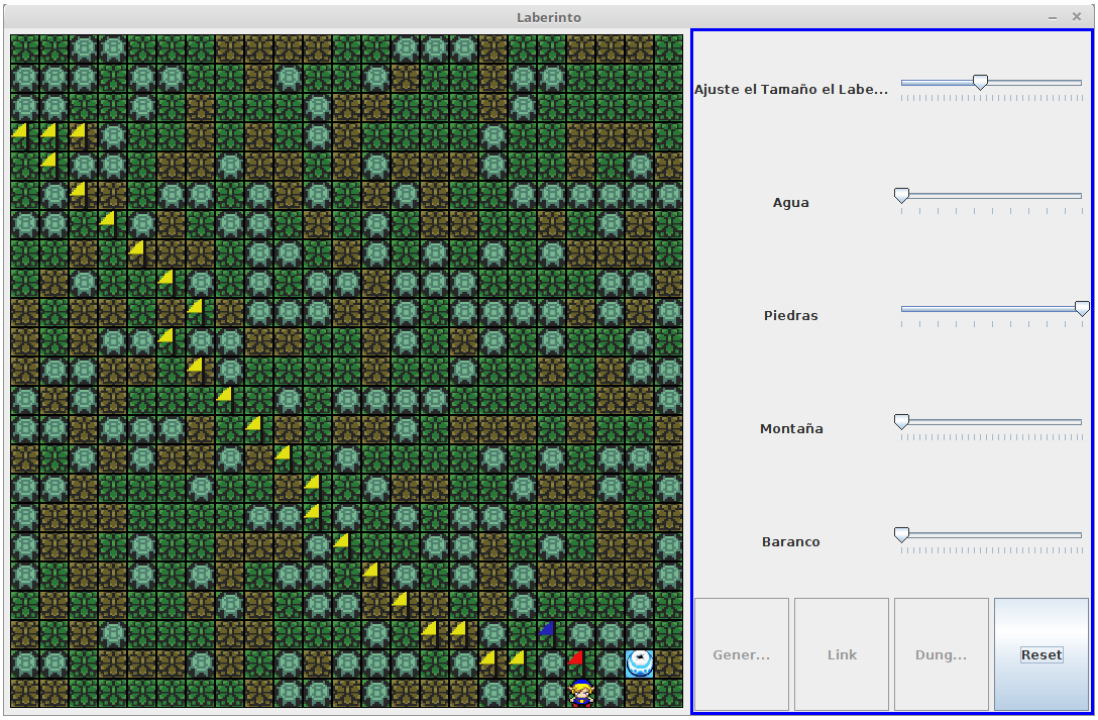


Figura 7

En las figuras 6 y 7 veran que el mono tiene que encontrar la salida en un bosque mas complicado. En este laberinto el mono tiene que utilizar el segundo estado multiples veces. Podemos ver que el mono se acerca a la salida pero llega a un estado en donde ya no se puede mover porque todas las posiciones a su alrededor no las puede visitar porque esas posiciones tienen un banderin rojo o son piedras/muros. En las siguientes imagenes veran un recorrido mas complicado.

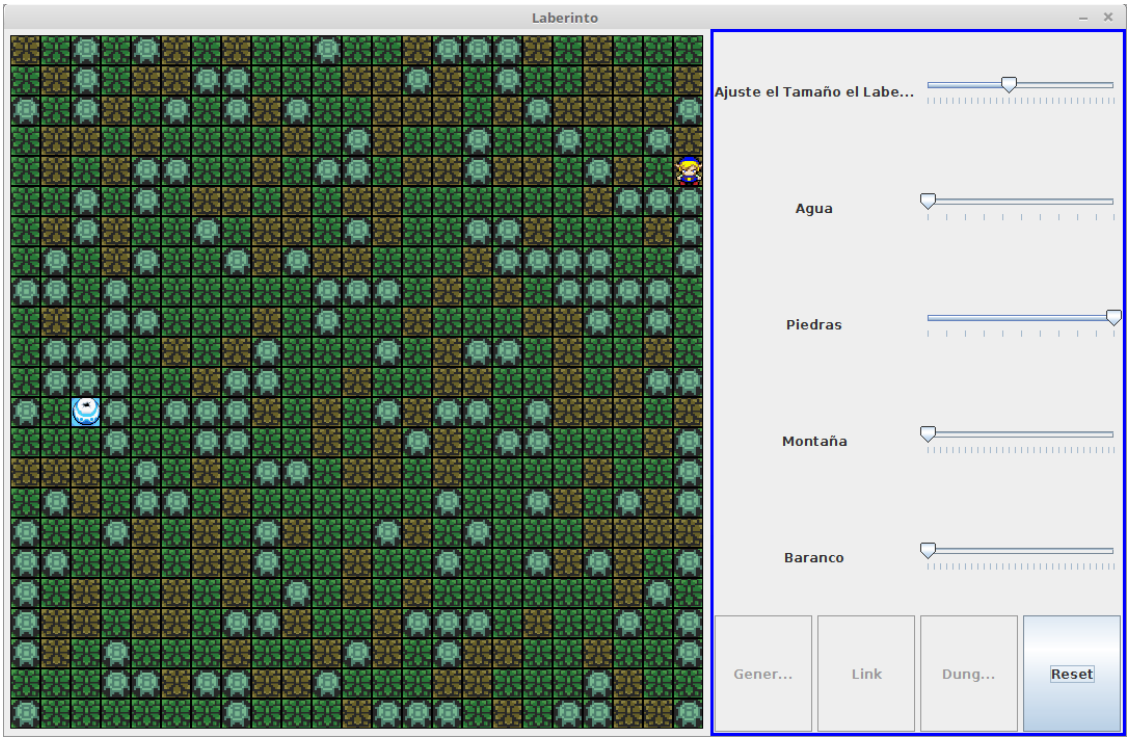


Figura 8

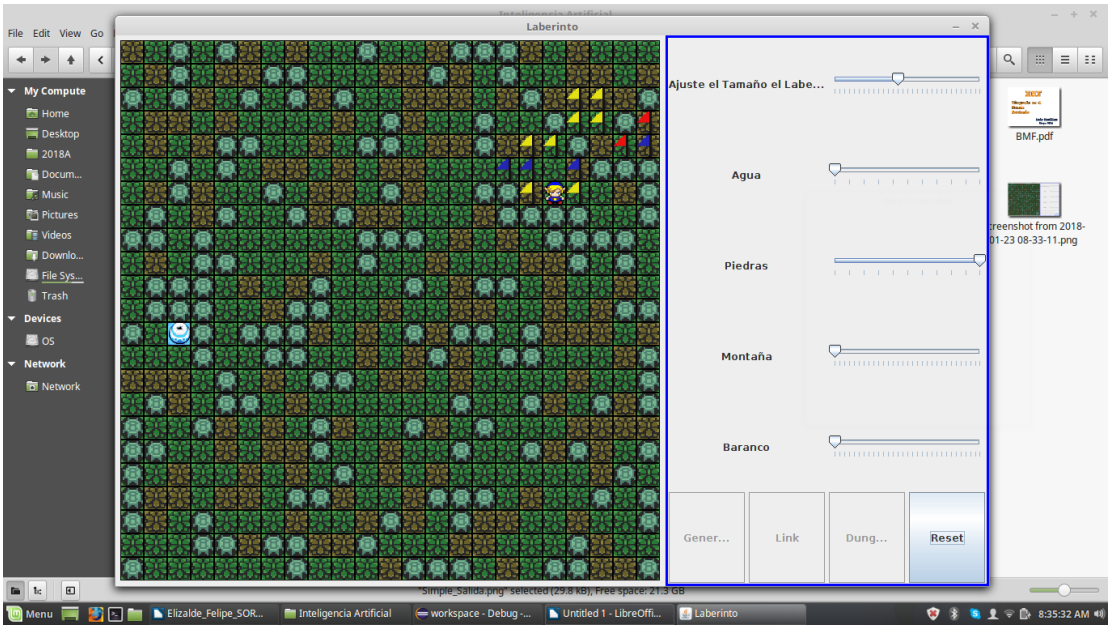


Figura 9

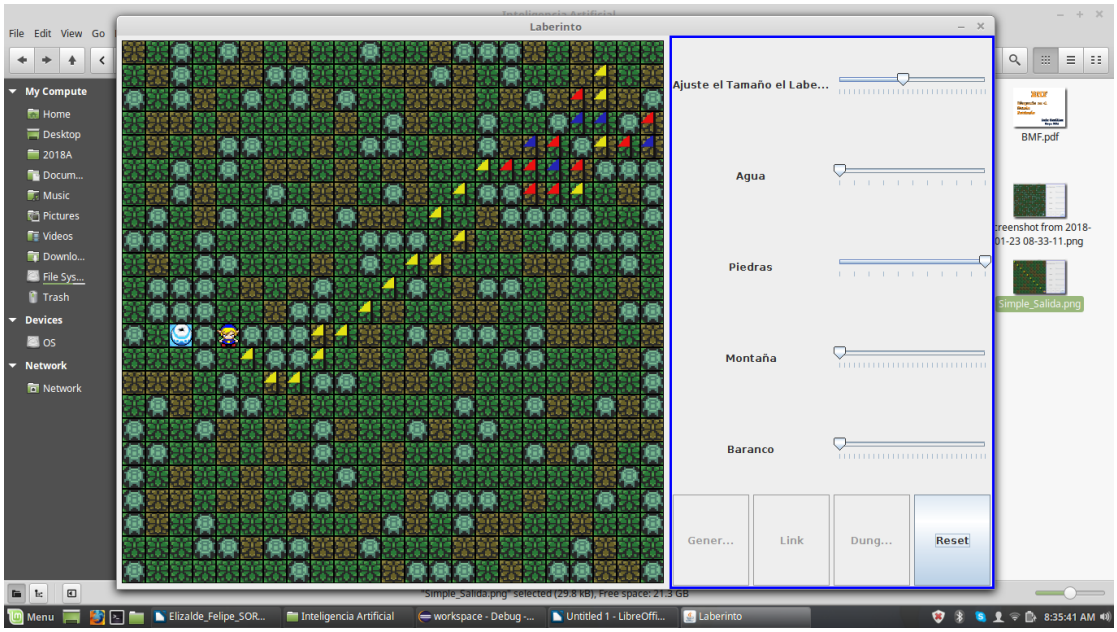


Figura 10

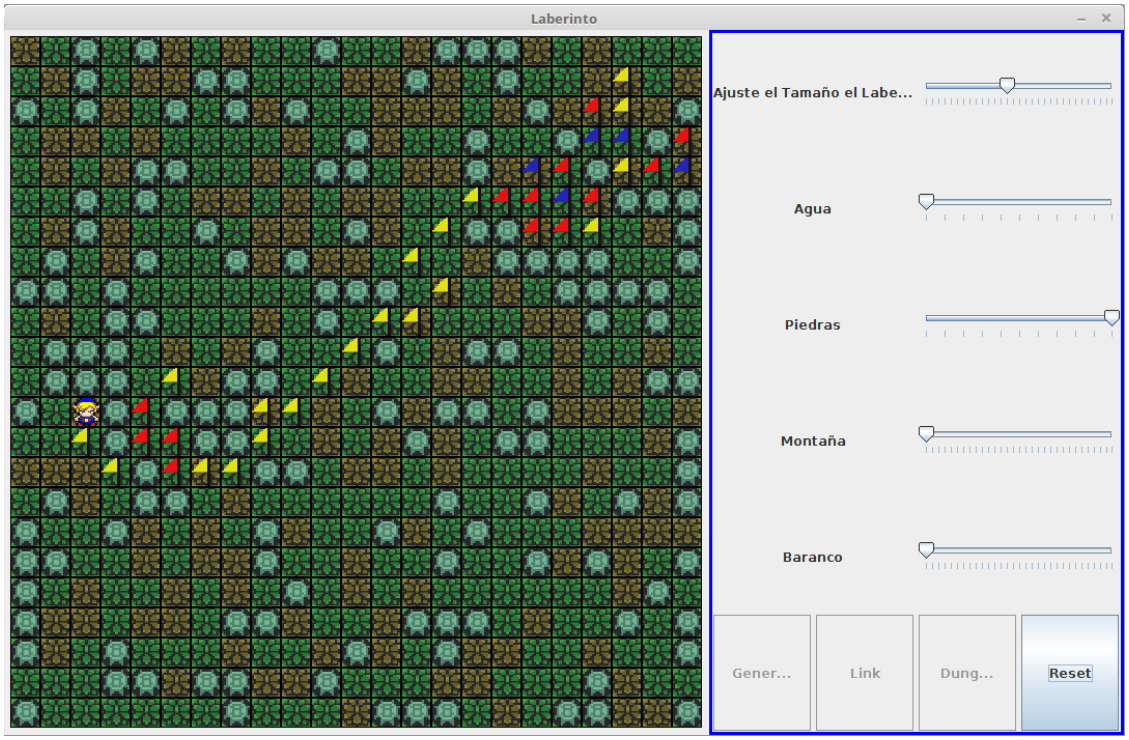
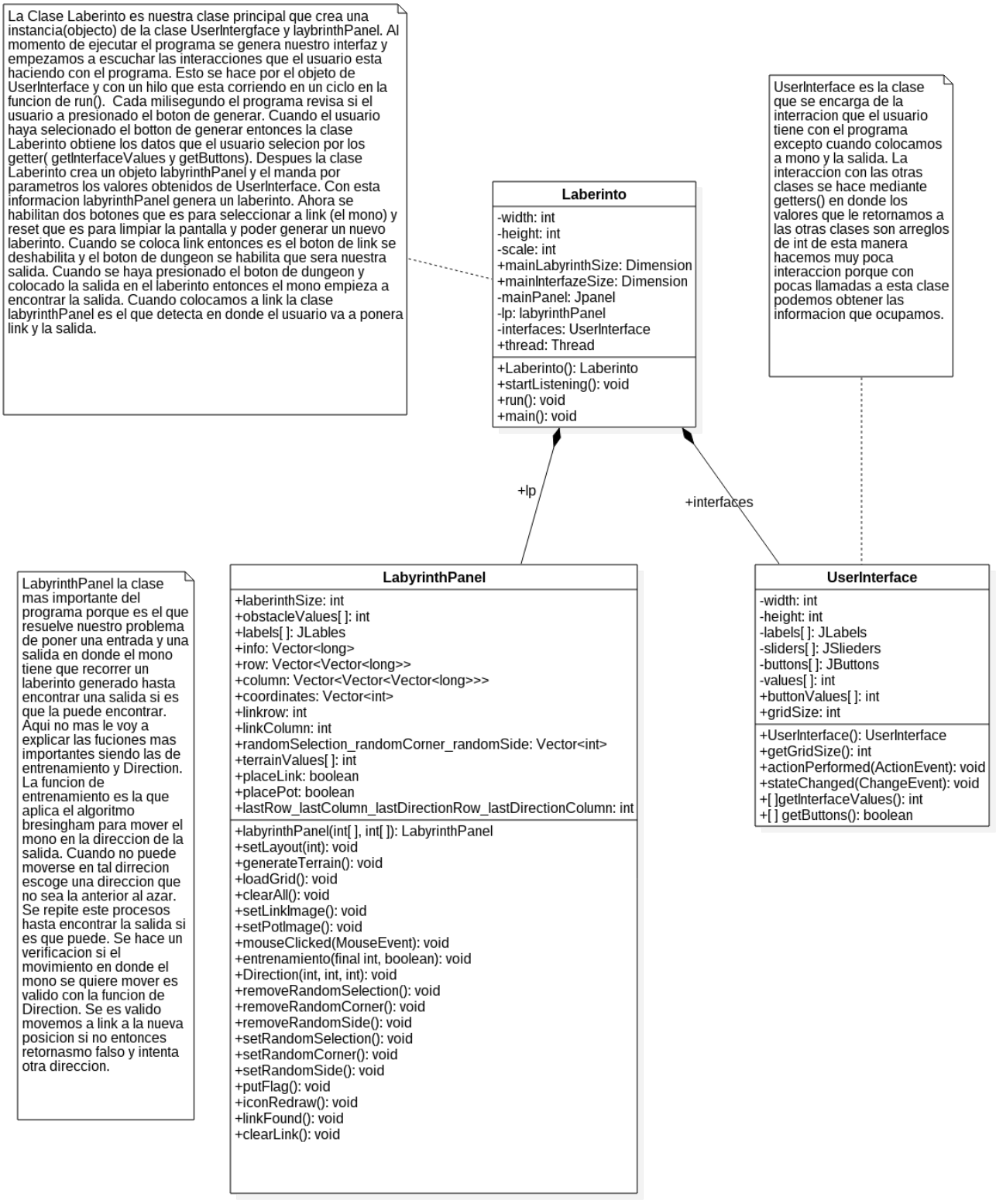


Figura 11

Podran ver en las figuras 8-11 que el mono entra al estado 2 multiples veces porque bressinham no funciona porque hay varias piedras/muros en su camino. El mono empieza a poner varios banderines rojos para emepzar a disminuyir los caminos posibles. Llega a un punto en donde puede utilizar bressinham otra vez aunque tenga que utilizar el segundo estado otra vez, pero logra llegar a la salida.

Diagrama de Clases



Conclusion

Se puede concluir que este problema me presento la oportunidad de aprender diferentes caracterisiticas del Lengauje de Programacion Java como la utilizacion de hilos para controlar tiempos en cuanto el programa tiene que ejecutar una funcion y de como cargar imagenes a Jlabels y como escalar imagenes al tamaño apropiado. Fue una actividad muy buena por estas razones y porque aprendi a optimizar el codigo y programarlo de un maner limpia en donde las variables y funciones estan declaradas de una manera en donde cualquier programador podra entender que esta pasando en el codigo.

Codigo

En el codigo podran ver que las partes mas importantes que son los movimientos del mono con el algoritmo de Bresenham. Esta parte no mas muestra cuando el mono tiene la opcion de mover en 8 direcciones. No incluye los movientos cuando el mono esta en una esquina o a los lados. Se puede decir que es el mismo codigo no mas quitando varios movientos. Tambien mostramos cuando es que el mono se borra de la pantalla y como lo pintamos otra vez en el laberinto, simulando sus movientos en el laberinto.

```
813
814     public void entrenamiento(final int monoSelect,boolean doBootStrap)
815     {
816         linkRow = linkLocation /obstacleValues[0];
817         linkColumn = linkLocation-(obstacleValues[0] * linkRow);
818         int potRow = potLocation /obstacleValues[0];
819         int potColumn = potLocation -(obstacleValues[0] * potRow);
820         boolean found = false;
821         coordinates = new Vector<Integer>();
822         coordinates.add(linkRow);
823         coordinates.add(linkColumn);
824
825         Random random = new Random();
826         int direction, cycle = 0;
827         setRandomSelection();
828         setRandomSide();
829         setRandomCorner();
830         while(!found)
831         {
832
833             try {
834                 TimeUnit.MILLISECONDS.sleep(looptime);
835             } catch (InterruptedException e) {
836                 // TODO Auto-generated catch block
837                 e.printStackTrace();
838             }
839
840             if(linkRow > 0 && linkColumn > 0 &&
841                     linkRow < obstacleValues[0]-1 && linkColumn < obstacleValues[0]-1)
842             {
843                 random = new Random();
844                 direction = -1;
845                 coordinates = new Vector<Integer>();
846                 boolean foundDirection = false;
847                 //System.out.println("RandomSelection Size:" + randomSelection.size());
848
849             while(!foundDirection)
850             {
851                 System.out.println("In firstLoop");
852                 if(linkRow < potRow &&
853                 linkColumn < potColumn)
854                 {
855                     System.out.println("SouthEast");
856                     if(Direction(1,1,monoSelect))
```

```
                    if(Direction(1,1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
            //South
                else if(linkRow < potRow && linkColumn  == potColumn)
                {
                    System.out.println("South");
                    if(Direction(1,0,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                //SouthWest
                else if(linkRow < potRow && linkColumn > potColumn)
                {
                    System.out.println("SouthWest");
                    if(Direction(1,-1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                //West
                else if(linkRow == potRow && linkColumn > potColumn)
                {
                    System.out.println("West");
                    if(Direction(0,-1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                //NorthWest
                else if(linkRow > potRow && linkColumn > potColumn)
                {
                    System.out.println("NorthWest");
                    if(Direction(-1,-1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                //North
                else if(linkRow > potRow && linkColumn == potColumn)
```

```
                else if(linkRow > potRow && linkColumn == potColumn)
                {
                    System.out.println("North");
                    if(Direction(-1,0,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                // NorthEast
                else if(linkRow > potRow && linkColumn < potColumn)
                {
                    System.out.println("NorthEast");
                    if(Direction(-1,1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }
                //East
                else if(linkRow == potRow && linkColumn < potColumn)
                {
                    System.out.println("East");
                    if(Direction(0,1,monoSelect))
                    {
                        foundDirection = true;
                    }
                }

                while(foundDirection == false)
                {
                    System.out.println(randomSelection.size());
                    System.out.println("In here");
                    System.out.println("Cycle: " + cycle);
                    if(randomSelection.size() != 0)
                     direction = randomSelection.get(random.nextInt(randomSelection.size()));
                    else direction = -1;
                    switch(direction)
                    {
                    case 0:
                        //NorthWest
                        if(Direction(-1,-1,monoSelect))
                        {setRandomSelection(); foundDirection = true;}
                        else removeRandomSelection(0);
                        break;
                    case 1:
```

```
941                     case 1:
942                         //North
943                         if(Direction(-1,0,monoSelect))
944                             {setRandomSelection(); foundDirection = true;}
945                         else removeRandomSelection(1);
946                         break;
947                     case 2:
948                         //NorthEast
949                         if(Direction(-1,1,monoSelect)){
950                             {setRandomSelection(); foundDirection = true;}
951                         }
952                         else
953                             removeRandomSelection(2);
954                         break;
955                     case 3:
956                         //East
957                         if(Direction(0,1,monoSelect))
958                             {setRandomSelection(); foundDirection = true;}
959                         else
960                             removeRandomSelection(3);
961                         break;
962                     case 4:
963                         //SouthEast
964                         if(Direction(1,1,monoSelect)){
965                             {setRandomSelection(); foundDirection = true;}
966                         }
967                         else
968                             removeRandomSelection(4);
969                         break;
970                     case 5:
971                         //South
972                         if(Direction(1,0,monoSelect)){
973                             {setRandomSelection(); foundDirection = true;}
974                         }
975                         else
976                             removeRandomSelection(5);
977                         break;
978                     case 6:
979                         //SouthWest
980                         if(Direction(1,-1,monoSelect)){
981                             {setRandomSelection(); foundDirection = true;}
982                         }
983                         else
984                             removeRandomSelection(6);
```

```
984                             removeRandomSelection(6);
985                         break;
986                     case 7:
987                         //West
988                         if(Direction(0,-1,monoSelect)){
989                             {setRandomSelection(); foundDirection = true;}
990                         }
991                         else
992                             removeRandomSelection(7);
993                         break;
994                         default:
995                             if(!Direction(lastDirectionRow,lastDirectionColumn,monoSelect))
996                             {
997                                 setRandomSelection();
998                                 foundDirection = true;
999                                 found = true;
1000                                break;
1001                            }
1002                            System.out.println("Went to last Position");
1003                            setRandomSelection();
1004                            break;
1005                    }
1006                }
1007            }
1008        }
```

```
1813    public boolean Direction(int rowDirection, int columnDirection,int mono)
1814    {
1815
1816        System.out.println("New Link Row: " + (linkRow+rowDirection));
1817        System.out.println("New Link Column: " + (linkColumn+columnDirection));
1818        System.out.println("Old Link Row: " + lastRow);
1819        System.out.println("Old Link Column: " + lastColumn);
1820        if(column.get(linkRow+rowDirection).get(linkColumn+columnDirection).get(0) != ids.rock.getValue()
1821                && column.get(linkRow+rowDirection).get(linkColumn+columnDirection).get(ids.visited.getValue()) < 3
1822                && ((linkRow + rowDirection) != lastRow || (linkColumn + columnDirection) != lastColumn))
1823                {
1824                    coordinates.add(linkRow+rowDirection);
1825                    coordinates.add(linkColumn+columnDirection);
1826                    iconRedraw(linkColumn,linkRow,linkColumn+columnDirection,linkRow+rowDirection,mono);
1827                    long visitedValue = column.get(linkRow).get(linkColumn).get(ids.visited.getValue());
1828                    System.out.println("VisitedValue: " + visitedValue);
1829                    column.get(linkRow).get(linkColumn).set(ids.visited.getValue(),++visitedValue);
1830                    lastColumn = linkColumn;
1831                    lastRow = linkRow;
1832                    linkColumn += columnDirection;
1833                    linkRow += rowDirection;
1834                    lastDirectionRow = rowDirection*-1;
1835                    lastDirectionColumn = columnDirection*-1;
1836                    System.out.println("Current Link Row: " + linkRow);
1837                    System.out.println("Current Link Column: " + linkColumn);
1838                    return true;
1839                }
1840
1841        return false;
1842    }
```