XML PARSER

Edinson Sanchez
Kevin Filella
Adrian Aguilar

18 de enero de 2014

Índice

1. Introducción

Buscaminas es un popular video juego, el cual lo podemos descargar y jugar en practicamente todas las plataformas que existen. El objetivo del juego es limpiar un campo abstracto sin detonar las minas. Normalmente el campo se representa como una matriz cuadrada, aunque debido a la popularidad del juego se han desarrollado muchisimas variantes en el diseño original del juego. Imagenes tomadas de wikipedia diseño en cubo tridimensional y diseño con muhcas minas por posicion

1.1. Objetivo

Nuestro objetivo en este proyecto es realizar una fiel implementacion del popular juego Buscaminas para sistemas Operativos Android. Sabemos tambien que la competencia en el mercado es amplia por lo cual debemos , ademas de implementar fielmente las funcionalidades del buscaminas de windows, darle un toque personal y diferente la presentacion del juego

2. Desarrollo

2.1. Desarrollo inicial

Inicialmente debido a la naturaleza del proyecto y a las caracteristicas del curso presente tuvimos que aprender a instalar y manejar las herraminetas de desarrollo adecuadas. Ademas de aprender desde 0 a programar en Android. Con la ventaja de que su entorno de desarrollo default ofrece una IDE mastante amigable al usuario y su lenguaje de programacion es parecido al Java.

3. PROBLEMAS

Problemas en el aprendizaje e implementación del lenguaje Haskell Al momento de iniciar el curso de lenguajes de programación nosotros como estudiantes debido al programa de estudios hemos sido iniciados en algunos lenguajes de programación. Lenguajes como .c, java, matlab , Android etc.. Estos lenguajes son todos de alguna manera u otra muy similares entre sí. Haskell es el primer lenguaje funcional en nuestro programa de estudio y por consiguiente la dificultad de aprender el funcionamiento y la sintaxis del mismo fue muy elevada. Entre las dificultades del lenguajes las más prominentes fueron la sintaxis y la utilización de contadores.

3.1. primer problema

Uno de los problemas más notorios que tuvimos se dio a cabo a la hora de la asignación de los datos a la estructura. Inicialmente, aparecían errores de “match” entre tipos de datos ([String] vs [Char], IO String vs [String], entre otros). Esto lo solucionamos modificando extensivamente las funciones que manipulaban los Strings y los arreglos de

Strings para que retornen los tipos de datos correctos. Más adelante, nos encontramos con otro problema muy tedioso, que fue el de la manipulación de listas. Aunque probamos todas nuestras funciones en consola, y funcionaban perfectamente, por alguna razón (aún desconocida para nosotros) estas funciones simplemente no funcionaban a la hora de correr nuestro programa (no se creaban las listas de Devices, Groups, Capabilities, etc). Esto lo solucionamos simplemente modificando la técnica de asignación de datos en la estructura. Creamos el tipo de dato Grupo, que contiene Device, Group y Capability; y nuestra estructura se guarda en una lista [Grupo]. Posterior a nuestra incómoda experiencia con el uso de listas, nos dimos cuenta que nuestra estructura (que se intentaba guardar en [Grupo]) no estaba siendo guardada correctamente. Esto fue más notorio a la hora de tratar de realizar las consultas, ya que simplemente nos lanzaba un error ("Non-exhaustive procedure", o algo parecido). La solución de este problema fue uno de los últimos cambios que hicimos en nuestro proyecto, antes de llegar a su etapa final.

Vista del Menu Principal

3.2. segundo problema

La colocacion de minas y de numeros en la matriz del juego fue bastante trivial. Sin embargo empezarona surgir problemas en los bordes de la matriz, especialmente en lo que respecta al conteo de las minas. Al parecer nuestro algoritmo de conteo no tenia la suficiente validacion como para operar en las celdas limite de la matriz. El contador simplemente se detenia cada vez que el amgoritmo se topaba con una mina del borde, Y por las caracteristicas recursivas del mismo era bastante poco eficiente poner condiciones limites para los cuatro bordes de la matriz. Al final el problema fue solucionado enviando la funcion recursiva a cada una de las 8 celdas adyacentes individualmente a travez de bloques IF, los cuales salian cuando la celda estaba fuera de limites.

Vista del la matriz del juego con todos sus elementos.

3.3. tercer problema

Al colocar las minas trivial y aleatoriamente sobre la matriz del juego todo parecia funcionar muy bien, no fue sino luego de extensa experimentacion que hallamos un problema. Las minas siempre se colocaban de igual o menor numero del que nosotros le enviabamos como parametro. Al principio pensamos que la causa de este problema era logica mal implementada en el contador. Luego de verificar que todo estaba aparentemente bien decidimos que la validacion no estaba funcionando y que varias minas podrian asignarse en una misma posicion, lo cual realmente era el caso ya que cuando cambiamos el metodo de validacion el problema se soluciono

3.4. cuarto problema

La implementacion del algoritmo recursivo para descubrir minas adyacentes vacias o numeros parecia a primera instancia facil de implementar. Pero realmente no fue asi.

El algoritmo fallaba en los bordes, pero solo en el borde derecho y en el borde inferior. Ademas debido a la maturaleza expansiva y recursiva del mismo este se volvia perceptiblemente mas lento a medida que aumentaba el tamaño del tablero del juego. Esto debido a que la cantidad de veces que este debia ejecutarse aumenta exponencialmente a medida que aumentan las dimensiones de la matriz. Ademas de eso el algoritmo tambien tenia problemas cuando este llegaba al borde de la matriz de juego. Poco a poco se fueron solucionando los problemas. Uno por uno. Primero introducimos nuevas validaciones para evitar los bordes, Luego optimizamos un poco el codigo para que se ejecute mas rapido el algoritmo.

Un aspecto persistente de este problema fueron los bordes derecho e inferior. Lo solucionamos temporalmente aplicando una estricta validacion en la ejecucion del algoritmo. Problemas con los bordes derecho e inferior del tablero se presentaron en practicamente todo el proceso de desarrollo de la aplicacion. Al final nos dimos cuenta que los limites estaban mal establecidos en la creacion de la matriz y ademas los ejes estaban intercambiados. Aunque igual pudimos lograr que el juego se ejecute normalmente a pesar de tener todos estos problemas, nos vimos obligados a reescribir codigo para arreglar el inconveniente luego del cual acabaron nuestros problemas con los bordes antes mencionados.

Luego de esto el juego podia ejecutarse con un 100

Algoritmo descubrir celdas

3.5. quinto problema

Luego de solucionar bastantes problemas evidentes y que el juego se pudiese implementar de manera confiable en los niveles Facil, Intemedio y Dificil. Solo quedaba implementar el modo personalizado, en el cual el usuario decide las dimensiones de la matriz y el numero de minas que esta posee. Aqui nos dimos cuenta de algunos problemas que habian escapado nuestra atencion, como por ejemplo la incapacidad de nuestro codigo para trabajar con matrices no cuadradas. O la gigantesca cantidad de memoria que ocupaba la ejecucion del programa cuando se seleccionaba un tablero relativamente grande con pocas minas, ya que las minas y los bordes son lo unico que paran la ejecucion de la funcion recursiva.

Estos problemas los solucionando revisando de nuevo el codigo e introduciendo nuevos elementos de validacion en algunas funciones. En el caso del problema de la matriz cuadrada se soluciono satisfactoriamente. El caso de la recursividad ocupando exesiva memoria solo lo pudimos solucionar parcialemnte. Nos vimos obligados a implementar limites en cuanto al numero minimo de minas que el usuario puede implementar, de acuerdo a las dimensiones elegidas por el usuario.

interface Personalizado. Incluye actualizacion en tiempo real del limite de minas

3.6. sexto problema DRAG AND DROP

Como especificacion de este proyecto se encuentra la caracteristica de poder agregar banderas a las celdas mediante el uso de la funcionalidad Drag and Drop. Luego de varios intentos decidimos implementar las banderas usando el metodo LongClick. el cual es compatible con anteriores versiones de Android y es mas facil de implementar.

4. Alcance

Aunque pudimos solucionar el problema propuesto, que fue el de crear un XML parser para un archivo con una estructura específica, nuestro XML parser solo funciona para ese documento, con el mismo número de niveles y nombres de atributos. Para poder utilizar este programa como un XML parser general, se le tienen que hacer extensivas modificaciones, no solo en la estructura, sino también en la lógica de asignación de los datos y la manipulación de los Strings y las listas. En cuanto a las consultas, nuestro programa realiza las búsquedas recorriendo la lista de Grupos y comparando los Strings. Además, incorpora un contador para los Devices que cumplan con la condición de búsqueda.

Ingresa el nombre del ganador

5. Conclusion

Kevin Filella: Ha decir verdad, no encuentro nada atractivo el uso del lenguaje de programación Haskell. Quizás esto se debe a que vengo de un background de lenguajes imperativos como Java, BASIC, C, C++ y próximamente, Python. La mayor dificultad que tuve a la hora de programar en Haskell, fue entender y acostumbrarme a la recursión, seguido casi de la mano con aprender la sintaxis del lenguaje.

Edinson Sánchez

El uso del lenguaje de programación Haskell tiene bastantes ventajas y desventajas en cuanto a la capacidad de resolver problemas tal como lo hemos experimentado en este proyecto. Fue muy entretenido aprender el nuevo lenguaje. Sus características funcionales hicieron que el aprendizaje fuese muy rápido y directo. Entre las desventajas esta la poca información que arrojan los errores en tiempo de ejecución.

Adrián Aguilar. El lenguaje Haskell es bastante distinto a los demás lenguajes que hemos estudiado. Me gusto el uso extensivo de la herramienta de la recursión. La recursión le da simplicidad y elegancia al código.

Scoreboard