Elvin Tubillara
CS242 Final Project

A Musical Sudoku Solving App for Android

1 Abstract

1.1 Document Purpose: This document's content describes the development requirements and steps to build an efficient sudoku solver application for the android OS. This application will let the user enter numbers manually and music will play based on the numbers entered and solving algorithm.

1.2 Definitions:
Algo X: Algorithm X, by Donald Knuth, is a backtracking algorithm that finds solutions to the exact cover problem. More info can be found: http://en.wikipedia.org/wiki/Knuth's_Algorithm_X

PD: Pure Data, a graphical patching programming language. I will use this to build the synthesizer. http://puredata.info/

1.3 Background/Motivation:
People are sometimes lazy and want to surprise friends on how fast they can solve sudoku puzzles. Sometimes they also want to hear music. The app will let users efficiently enter numbers by hand and music will play based on the puzzle.

2 Technical Specifications

2.1 Platform:
Android OS

2.2 Programming Languages:
Java (Android API)
Pure Data

2.3 Coding Standard:
Android:
http://source.android.com/source/code-style.html
Java:
http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

2.4 SDK:
Android SDK

2.5 IDE:

Eclipse, Pure Data extended

2.6 Interface:

Android phones and tablets

2.7 Other Technical Details:

3 Functional Specifications

3.1 Affordances:

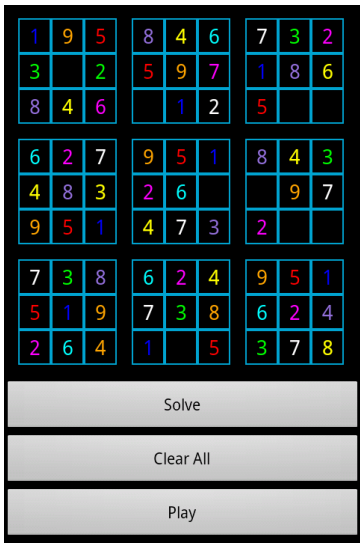
a. User will be able to enter numbers manually
b. User will be able to change settings to make entering numbers manually a better experience. For example a user can enable the option for hints for a certain sudoku square.
c. User will be able to listen to sound/music based on the puzzle.

3.2 Features:

a. The sudoku solving logic will be implemented using Algo X, which is an algorithm efficient enough for this project. It will be written in Java without any external libraries.
b. The music will be synthesized with Pure Data for Android

3.3 Scope:(none)

3.4 Current Look/Mockup:



Timeline

Week 1: Implement Sudoku solving logic (Model)

a. Implement efficient Data Structures
b. Implement Algo X
c. Test Data Structures and Algorithm

Week 2: Implement Basic Android Functionality

a. Improve model running time.
   1.)Use Heap for selecting column nodes
   2.)Use a non-recursive function to find solution
b. Implement a basic Android View
c. Implement a basic Sudoku Model
d. Implement listeners for view
e. Make sure basic functionality works on a android device
f. Create Manual Test Plans for View

Week 3:

a. Set up Pure Data for Android
b. Implement Pure Data modules (Sound Synthesis)
c. Create interface in Java/Android to work with pure data
d. Create snapshots in puzzle solver to send data to Pure Data to make music
e. Create Manual Test Plans

List of things I did for this week that compliment the bullet points above:

1. wrote a solver that solved step by step (part d)
2. make a music generator that parsed a file -it produced notes
3. based on which number was inserted by the solver and played a note
4. based on a map given by the file and played a chord progression from the file
5. reworked my pure data patches
6. wrote basic tests for the pure data patches
7. wrote basic test for music generator file parser
8. wrote basic test for stepping solver
9. wrote basic manual test plan for play/stop button

Things that took the most time:

Setting up Pure Data for Android
Figuring out miscellaneous Android things like Asnyctask
Debugging stupid my mistakes.
Refactoring like a noob.

Week 4: Improve/Add features and test on more devices

a. Improve model running time
    1.) Put less branches on search tree
        (do not put numbers already given in tree)
    2.) Use custom priority queue with update key when getting column
b. improve and test UI on Nexus 7 and smaller phones
c. add more color options
d. improve/add more chord progressions
e. put an option menu to change options like:
    i. color
    ii. chord progression
f. test application with (make sure they work fast):
    random sudoku puzzles
    non-real puzzles