Simulator Requirement Document

York University | York University

EECS 2311 Project

Kevin Banh, Zaeem Israr, Amir-Hossein Khademi

2017

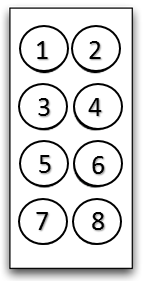
**Introduction**

This document describes the required features of the Braille Simulator program. The main purpose of this software is to simulate the Braille hardware in order to help kids read Braille. The hardware has the capability of outputting Braille characters and accepting user inputs via a button and thus, the simulator program should be able to do that.

**Definitions**

For the purpose of this document, a “**cell**” is basically a Braille character. They are a rectangular box possibly containing raised bumps on them. These bumps are called “**raised pins**” and based on the number of them and their positioning, represents a character. In the English language, one Braille cell represents one English character. An English character in Braille typically requires six raised pins in a Braille cell. The pins are located in three rows, with two possible locations per row aligned with the other rows. The location where raised pins may occurred will be referred to as a “**pin**”. The hardware has eight pins, four rows with two pins each.

For the purpose of every document and the software, Cell 1 is the leftmost cell, Cell 2 is the one to the right of Cell 1, Cell 3 is right of Cell 2 and so on. In a cell, there are eight pins. Pin 1 is the top left pin, Pin 2 is the top right pin, Pin 3 is below the top left pin and et cetera, denoted by the image below.



**Customer Requirements**

1. Program must provide a GUI.
2. The GUI must have the capability to display cells and their pin configurations.
3. Each of the cells should have 8 pins, similar to the hardware.
4. The GUI should be able to receive button inputs.
5. Users should be able to tell the program initially how many buttons and cells to simulate, up to an acceptable amount. These numbers should not be allowed to change throughout the execution of the program and the GUI should respect/reflect these values.
6. The program should provide support for English language translation into Braille, for an appropriate set of English characters. (Appropriate is defined as at least the alphabet and space characters.) Support for English language translation into Braille means, for example, users can input a string of English characters and assuming the characters are all supported, have the program translate it to Braille characters and display the Braille on the GUI.
7. The program should allow users to raise and clear a particular pin of a particular cell and reflect that change on the GUI to allow support for other languages.

Other requirements not required from the customer, but should exists include when users press a button that is “activated” (which is based on how many numbers of button the user tells the simulator to simulate), it should tell the user in the GUI that that certain button has been pressed.

If the person tries to translate an invalid sentence or the number of characters in a sentence exceeds the number of activated cells, display an error message in the GUI and do not translate the message.

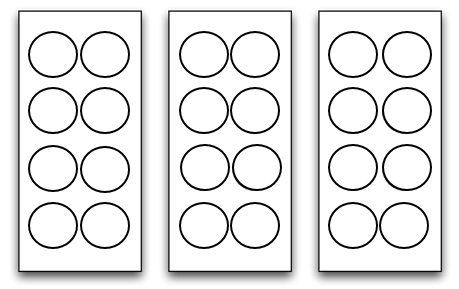
Similarly, for 7., if invalid pins (or cells) are selected and users attempt to set them to a certain state, do not do anything.

For 6. of customer requirements, this program should support all letters of the alphabet regardless of case, all the numbers and the spacebar.

There should be a way for the user to reset the Braille translation and the GUI displaying that translation, thus clearing all the raised pins and leaving the cells only.

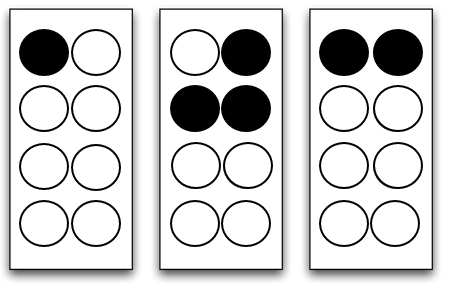
**Acceptance Test Case**

For the acceptance test case, assume that the user has defined the simulator to simulate 3 cells and 3 buttons. The GUI at this point should appear with three rectangles representing the 3 cells like the image below. **NOTE:** Assume a pin is raised if it is coloured in. Graphics below are a representation of what the GUI will produce, but were not taken directly from the final product and may vary.

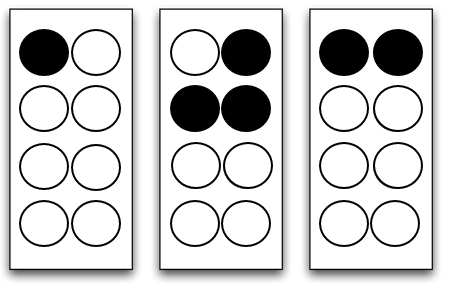


Since 3 buttons are activated, in the GUI, three keys on the keyboard would correspond to Button 1, 2 and 3. If the user presses a key on the keyboard that corresponds to Button 2, a message will appear on the GUI stating Button 2 has been pressed. If the user presses a key that does not map to a button, nothing will happen. If the user presses a key that maps to a higher number button whose number exceeds the number of buttons defined on the simulator by the user (like Button 4 in this case), it will not do anything.

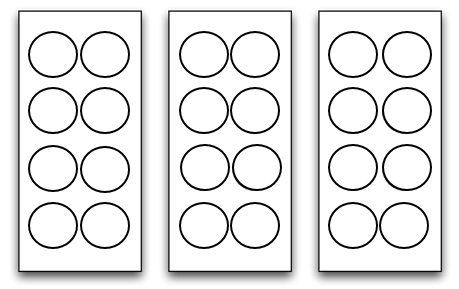
If the user asks to translate the sentence “1jC”, the GUI will display the following image below.



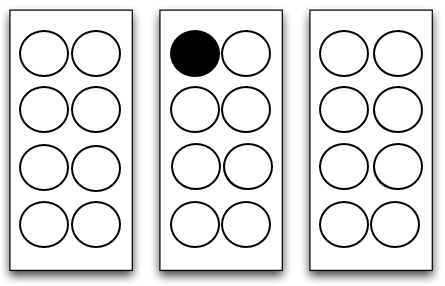
If the user then asks to translate “%$^” or “abcs”, the program will not translate the string of the characters. The former string has invalid symbols (as they are not numbers, letters or space) and the latter has four characters, more than the three cells provided for this instance of the program. (Remember user can set the amount of cells they want at the beginning of the program and since this is a hardware simulator, the number of cells should not change during the program.) The GUI will display an error message and the cells will have the same configuration as before (see below).



If the user then asks to clear the braille simulator, all the raised pins will be cleared resulting the GUI looking like the image below.



From there, if the user chooses to translate “ a”, it will result in the below image.



From that point, if the user tries to set a pin to be raised in a non-existing cell or a non-existing pin, nothing will happen and the GUI will not change. Similar effects will occur if the user tries to raise a pin that is already raised or clear a pin that is not raised. If the user raises Pin 1 of Cell 3 (please see **Definitions** for explanation on pin and cell numbering), the image below will be the result.

