Guessing game

Isak Skeie

16.01.2022

1 Introduction

The purpose of this assignment is to get familiar with the program LabView, and its components. This is done by walking through LabView tutorials made by Hans-Petter Halvorsen, and then create a unique program. The uniquely created program should deploy the most important functionality of LabView. This will repeat the information from the tutorials as well as confirm that the necessary skills are in place. To do this, a guessing game were created in LabView. A random number between 0 and 100 is created, the user is supposed to guess the right answer, with input from the program if the guessed number was too low or too high. When the right number is found a graph with all the tries, as well as the number of tries is presented in a cluster.

1.1 Problem Description

Figure 1 below shows the iterative process of the guessing game. The user comes with a guess and gets feedback if the guess is too high or too low. If its correct, a graph is populated, and number of tries are displayed.

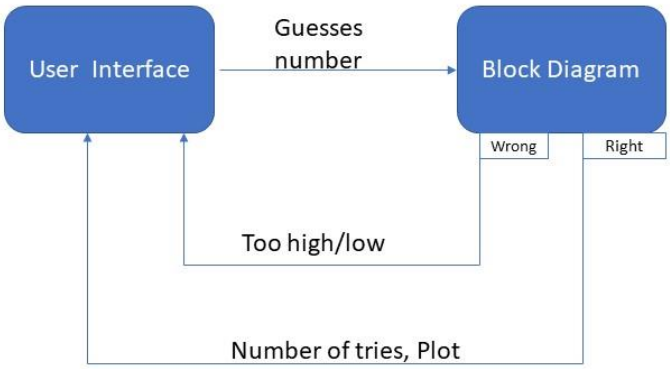

Figure 1: System overview

2 Material and Methods

The software used is LabView. The program consists of a User Interface, as well as a block diagram. In figure 2 the user interface is shown, it consists of a short description of how the

game is played. Under it a numeric input from the user, this is where the guess is entered. And under the numeric input a string output, here the user gets feedback from the guess. The cluster to the right contains a numeric indicator to show the number of tries, as well as a waveform graph.

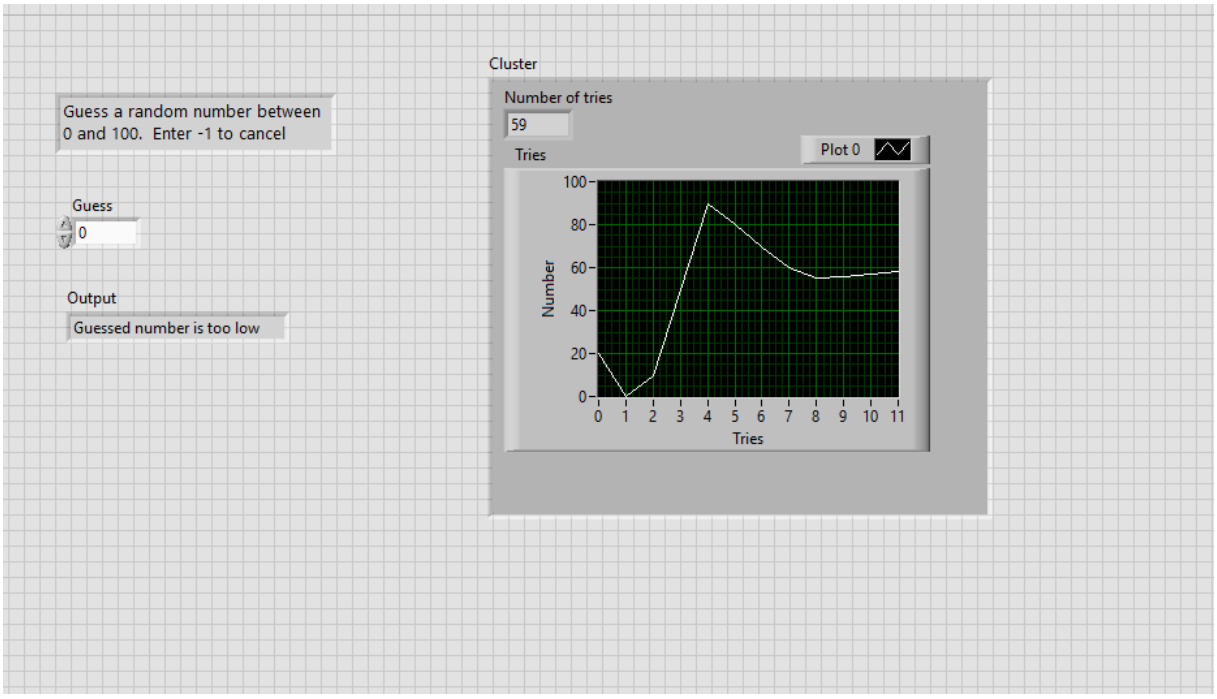

Figure 2: The user interface for the program

Figure 3 shows the main block diagram. Most of the code is situated inside a while loop with initializing on the left of the while-block, and the generated cluster on the right. Inside the loop, the guessed number is called, then sent to a Sub-VI. The Sub-VI outputs one of three number. Depending on if the guess is too high, too low, or correct. This number is then matched in a case structure which sends the result to the User Interface. At the top within the while-loop. An array of all the guesses is created, as well as the number of tries saved. This is made possible by shift registers in the while-loop.

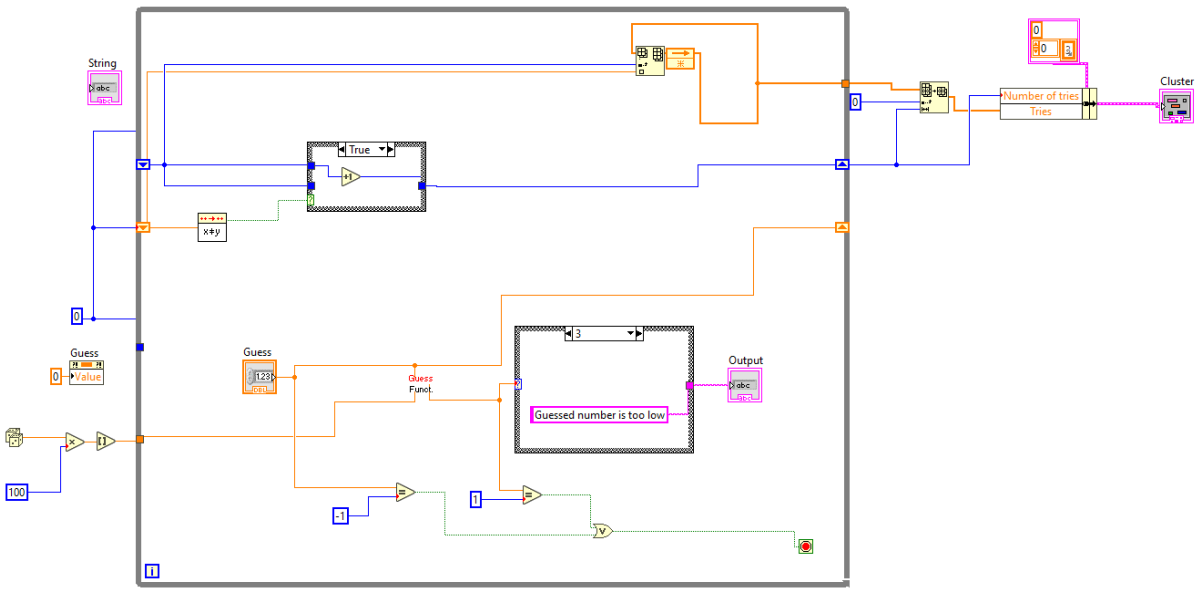

Figure 3: The main block diagram for the program

Figure 4 shows the Sub-VI which was created. It receives the random number and a guessed number, the sends the difference to one of the three case-structures. The output is either 1 if the guess was right, 2 if its too high, or 3 if it's too low.

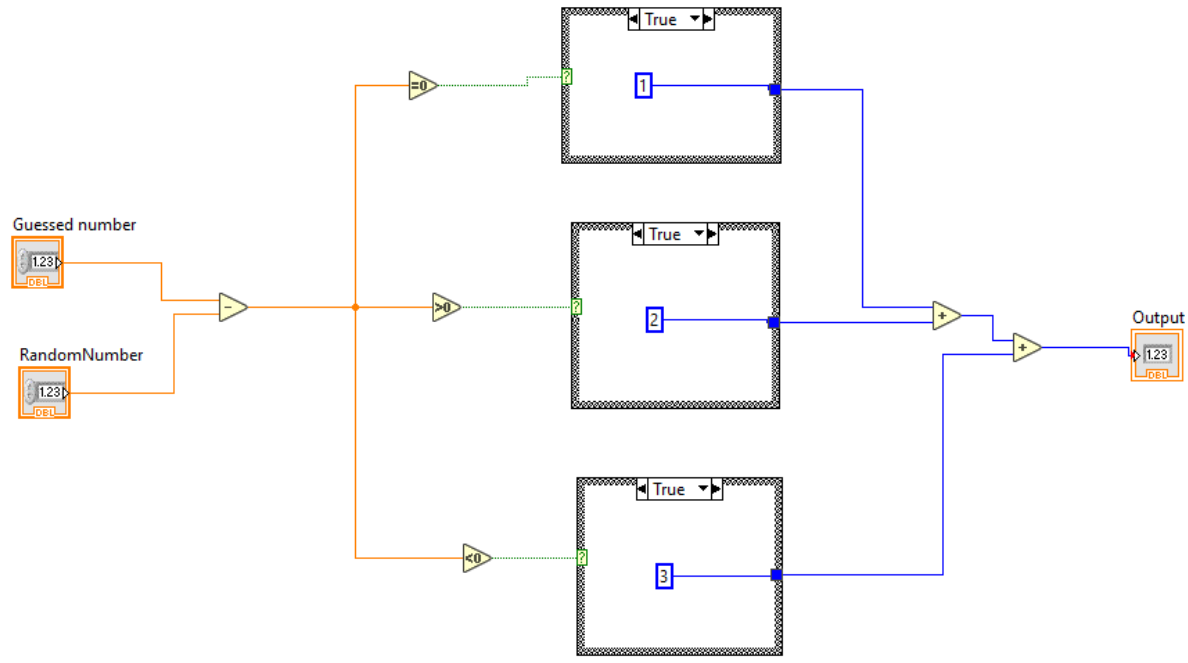

Figure 4: The Sub-Vi for the number comparison

Figure 5 shows the user interface for the Sub-VI. This was created for the development of the program. To test the Sub-VI and make sure it produced the correct output.

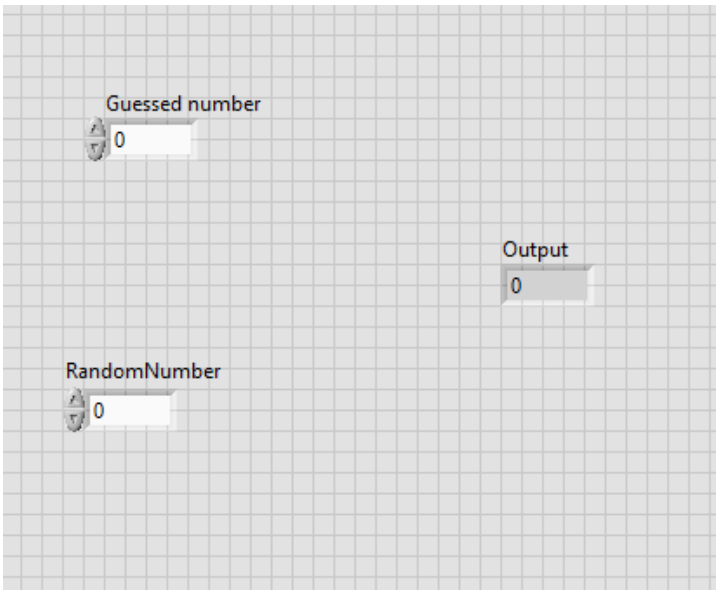

Figure 5: User Interface for Sub-VI

3 Results

When testing out the program it can be concluded that it works like intended. When running the program, the user has to reenter guessed values until it finds the right one or enter -1 to cancel. With the correct information being shown when the task is complete.

A small flaw with the program occurs when starting the program. The value which is initialized in the numeric control counts as one guess. Another flaw happens when the user guesses the correct number. When the correct is entered it exits from the loop. This logic prohibits the correct value from being saved in the array, which in turn is used to display all the tries in the waveform graph.

If the program were to be improved, it should be in the code. The three case-structures in the Sub-VI should most likely be combined into one. The code in the main block diagram should be cleaned up. This includes changing the logic behind the array being created. Theres probably a more robust and cleaner way to code the logic.