

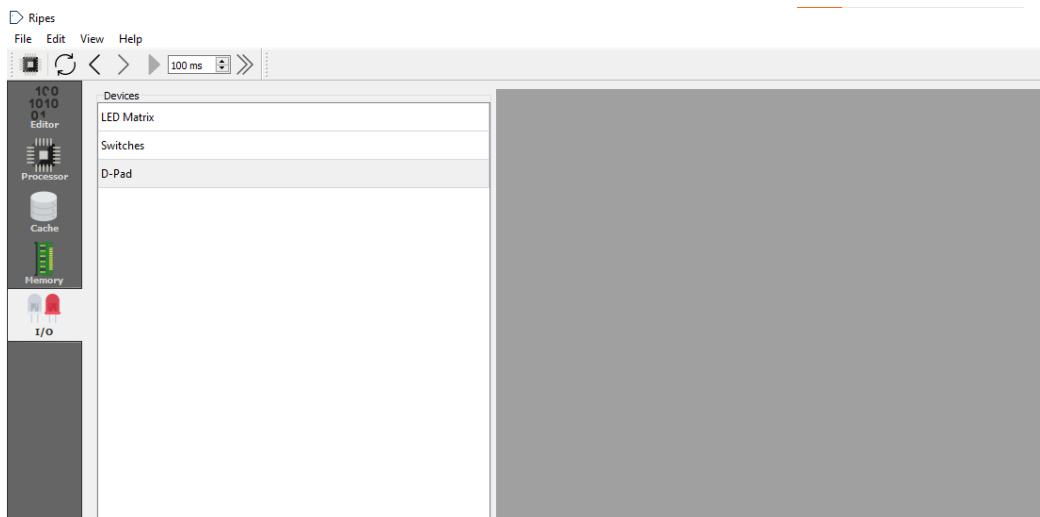
# Simon Documentation

## Summary:

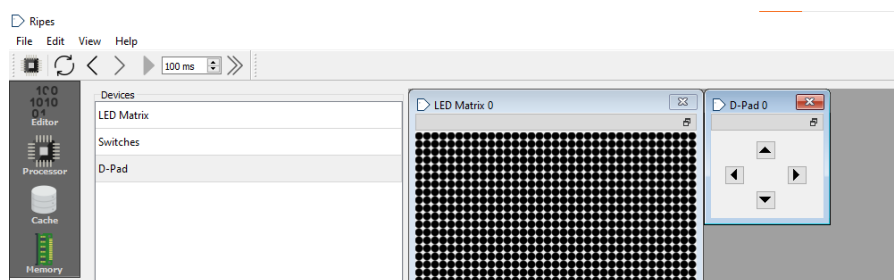
Simon is a memorization game in which the computer flashes a sequence of lights on the screen which the player must recall in order. The objective of the game is to correctly repeat the sequence back to the computer by pressing the corresponding buttons. The game becomes more challenging with every correctly inputted sequence. The sequences become longer, and flash on the screen for less time, making it more difficult for the user to memorize and repeat back to the computer.

## How To Open Simon in Ripes:

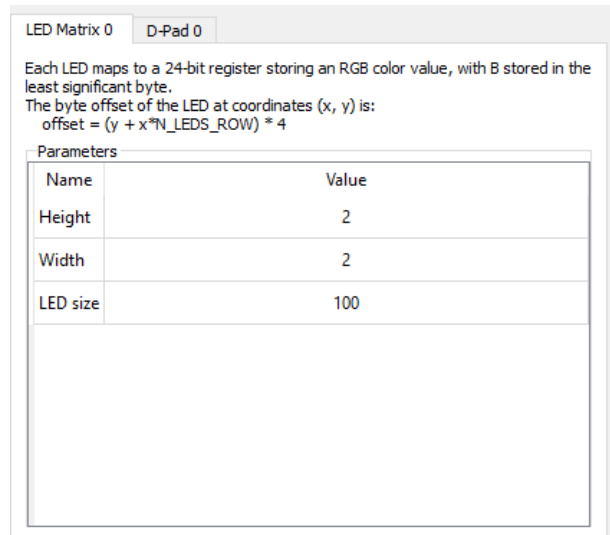
In order to play Simon in Ripes, the user must open the file *Simon.s* as a source file. This can be done by going to File, and then selecting the Load Program option from the dropdown menu. From there, a menu to open the file will pop up. The user must select “Source File” for the file type, and then find the file location on their computer. Once it has been found, select “Ok”, and the file will open. The user can then cycle over to the “I/O” tab on the left hand side of the page. A screen that looks similar to this should pop up.



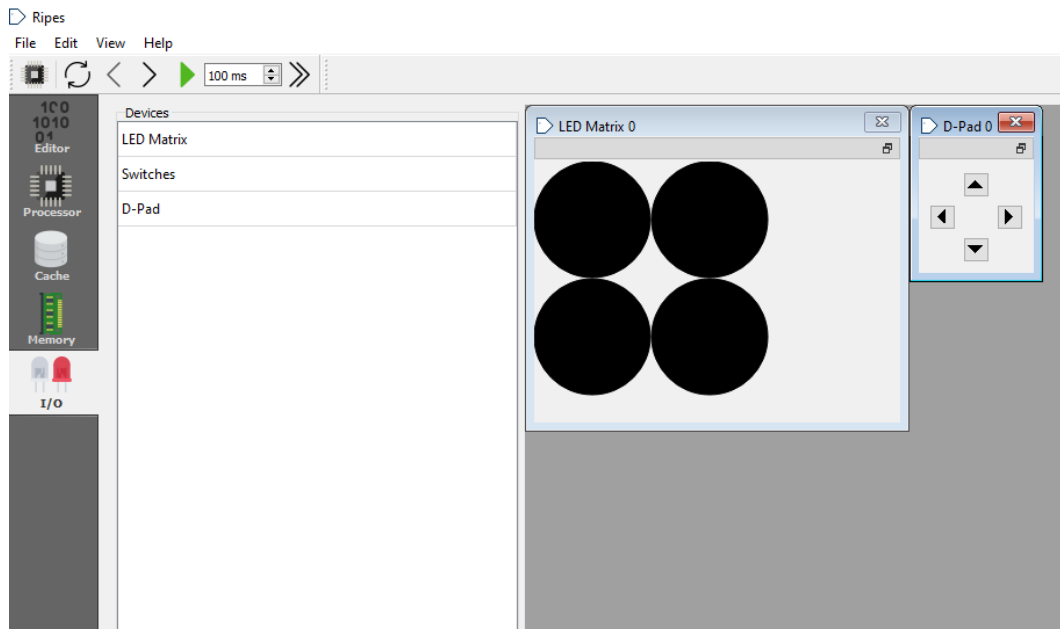
Under the devices tab, double click on “LED Matrix”, and then double click on “D-Pad”. Your screen should now look like this:



On the right hand side of the screen, select the “LED Matrix 0” tab. There should be a table with the parameters “Height”, “Width”, and “LED size”. Double click on the Height value tab and type in 2, and do the same thing for the width. The user can set the LED size to whatever they prefer (Recommended size is 100).

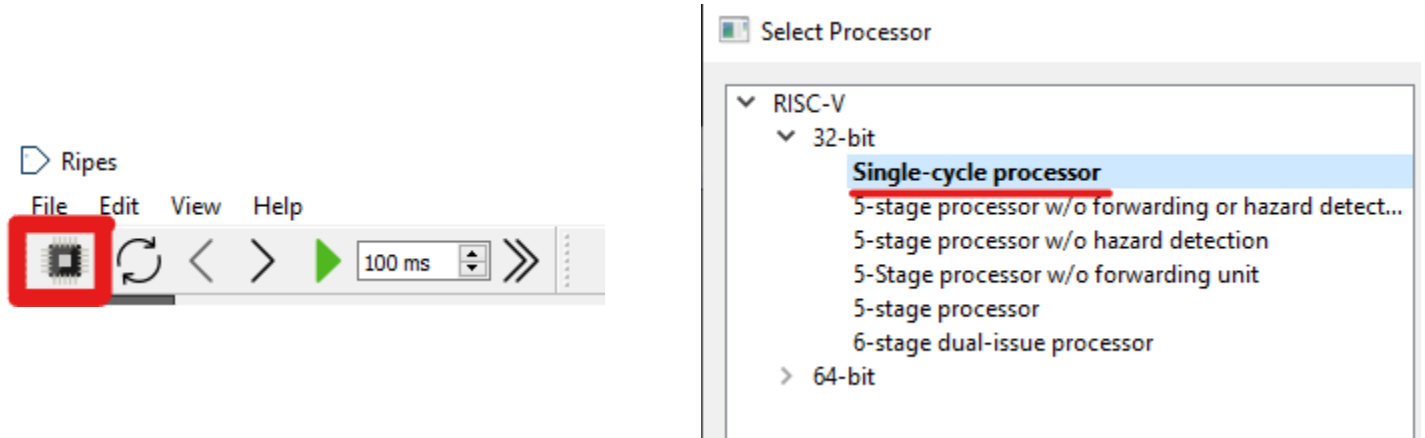


The window should now look something like this:



This is where the user will play the game. The inner window with the four black circles are the LEDs, and the inner window with the arrow buttons is the D-Pad. These will be explained more in the “How to Play Simon” section of this document.

Before starting the game, the user should also verify that they are using the 32-bit single-cycle processor option. To do this, select the button that has the image of a processor in the top-left corner of Ripes. A menu that looks like this will pop up.



From here, select Single-cycle processor, and then press “Ok”. The user should now be set up to play the game.

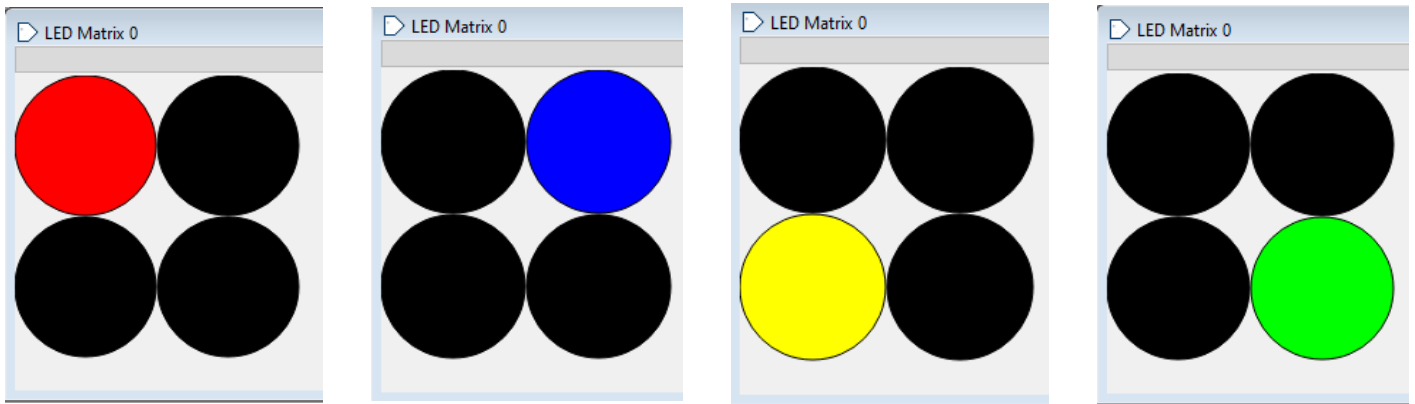
## How to Play Simon:

This part of the document will explain how Simon is played, including which buttons correspond to which lights, and the rules for the game. The objective of the game is to correctly input the sequence of LEDs that is flashed to the user.

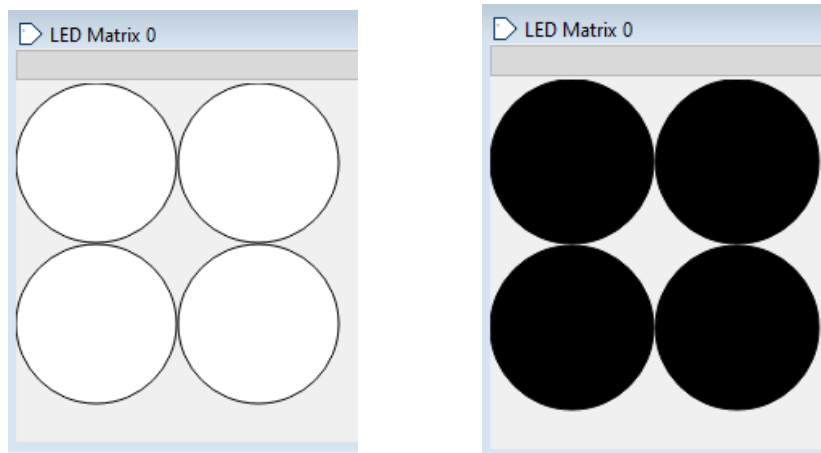
To start the game, the user must select the fast execute option. This can be done by selecting the button with the two black arrows at the top-left of the screen.




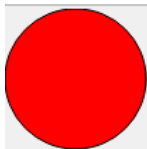
This will start the game, causing the first random sequence of lights to flash on the LEDs. The first sequence has a size of 4 LEDs that flash. Each LED corresponds to a specific colour. The top left LED corresponds to the colour red, the top right LED corresponds to the colour blue, the bottom left one corresponds to yellow, and the bottom right one corresponds to green. This is summarized in the images below:



The sequence will flash a combination of these lights, which the user has to memorize as they light up. After the sequence has fully been displayed to the user, every LED will flash with a white light, before quickly returning to all black. An example of this can be seen below:



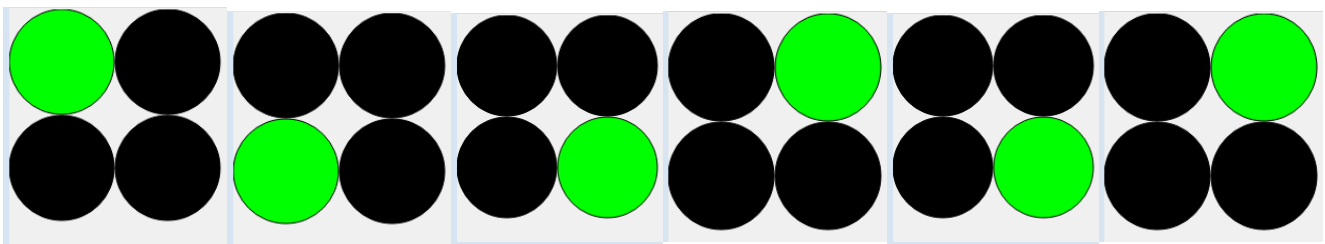
This is the signal to the user that they can now input the sequence they remember. They can do so by clicking on the D-Pad buttons. Each LED/colour corresponds to a button on the D-Pad. This is summarized in the table below:

Button on D-Pad	LED Location	LED Colour
Up Button 	Top Left	Red 

Down Button 	Bottom Right	Green 
Left Button 	Bottom Left	Yellow 
Right Button 	Top Right	Blue 

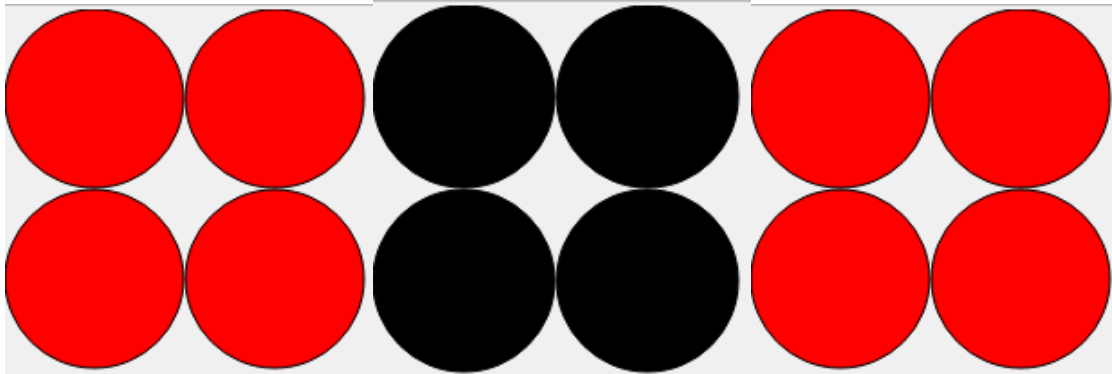
To show the user which button they inputted, the light corresponding to that button will flash its colour, before changing back to black. This will continue until the user has entered the number of colours in the sequence regardless of whether they are correct or not. After which, one of two lights combinations will be displayed for whether the users inputted sequence is right or wrong.

If the pattern that the user has inputted is correct, then the sequence below will light up on the lights:



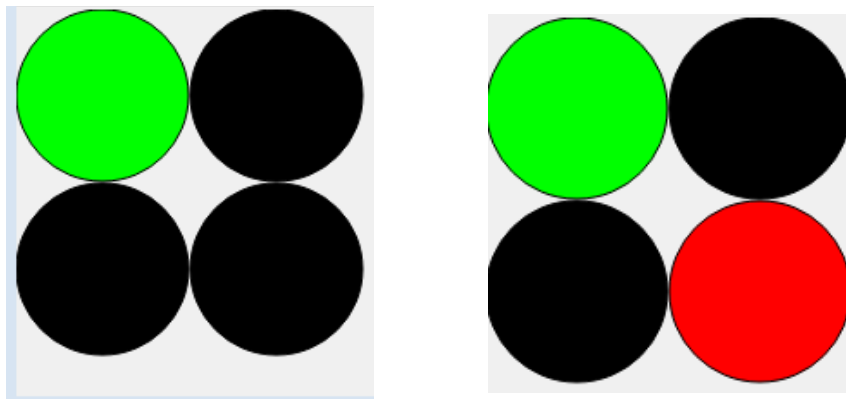
This sequence of lights is inspired by the “charge” sound effect.

If the pattern that the user inputted is incorrect, then the entire board will flash red two times quickly as shown below:



The LEDs are all flashed black in between to give this effect.

After either sequence is flashed, the computer will prompt the player if they would like to play again. This option will be showed to the player by the following light sequence:



For this, the upwards arrow key on the D-Pad corresponds to “Play Again”, while the downwards arrow corresponds to “Exit” (The other two arrow keys have no effect). If the user chooses to exit, then the program will exit. This is the best way to exit the program. In order to rerun the program, it has to be reset. The user can do this by pressing the circle with the two arrows in the top left corner of the screen, and then run the game as usual (By pressing the two fast execution button).



If the user chooses to play again, then the game will get ready to replay. If the user got the last sequence correct, then two enhancements occur to make the game more challenging.

### Enhancement A:

First, the sequence size dynamically increases by one extra light. The entire sequence is generated again every round, so the original lights in the sequence may not be the same as they were in the previous round. This makes the game more challenging as it is harder to memorize the longer sequence. There is no definite max number of lights that a sequence could have.

### Enhancement B:

Second, the game decreases the time between each light flashing in sequence. The time that the colour flashes on screen and the time between each light flash is decreased by 50 ms for each correct input. At minimum, the time that a colour flashes is 50 ms, and the time of delay between each colour is 550 ms. This may seem like it makes the game less challenging, but these times make it quite difficult to remember a sequence that grows longer indefinitely.

This concludes how to play this version of Simon, and concludes the user documentation of the game. Hope you enjoy the experience!