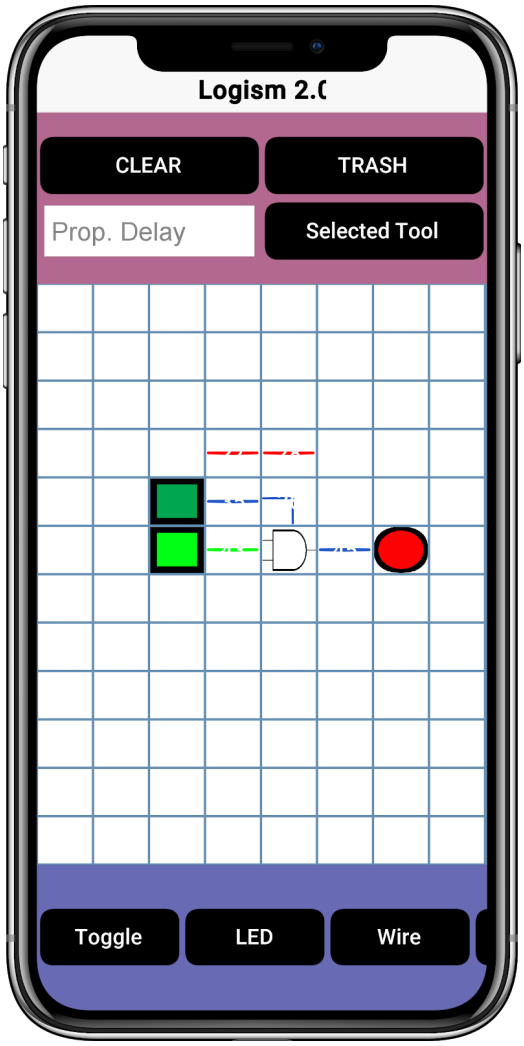
MOBILOGIC

CN1 Logic Simulator

BACKGROUND:

The MobiLogic Simulator is a mobile application that allows users to perform quick and simple logic calculations on the go. Although there are more fully fledged logic simulation software in the market, they are not intuitive and unnecessarily over complicate simple tasks. The product aims to cater towards younger audiences, students at the secondary and tertiary education levels who are just now being introduced to logic circuits. As a result, the design decisions were made to keep the product simple and engaging.

The main features include the ability to quickly set up an environment that allows creating logic circuits with basic gates, toggle switches, LEDSs, and wires. Along with this, color coded wires and toggles allows a user to easily visualize logic states and understand the basics of how these circuits work.

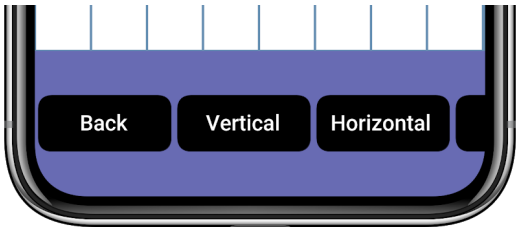


U/I:

A simple and clean grid layout is placed in the middle of the screen where users can place their components, make logic circuits, and view the output of their circuits. The user workspace is very large to allow as many components to fit comfortably on the screen as possible. In the image on the right, we see what a simple circuit looks like in the app, along with brightly colored wires for easy visualizations.

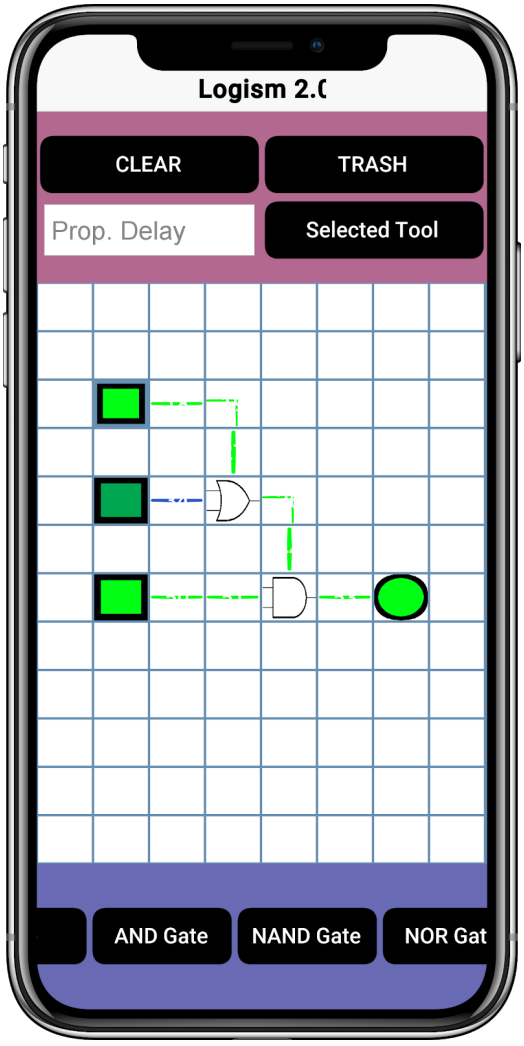
A horizontally scrollable toolbar with large buttons is placed at the bottom to allow access to all circuit components including gates, wires, LEDs and toggles. This decision was made so that the user can quickly click on components and place them into the main circuit. All components are represented as text buttons for accessibility.

Clicking on one of the buttons will select the circuit component. The currently selected component is shown in a box next to the propagation delay textbox. There are also buttons that, instead of causing a selection, will replace the scrollable bar with a cluster of components related to the button that was pressed. In the sample image shown on the right, we can see that the “Wire” button was pressed, now giving us options to select various different versions of the wire that the user needs to place.



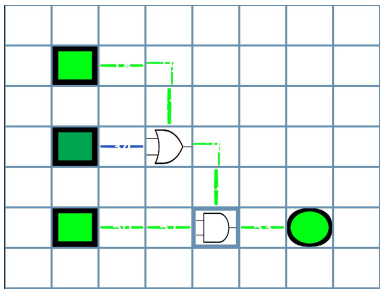
Operations Guide:

Upon opening the app, a user is directed to the main workspace. At the top, we see the name, along with 3 black buttons and 1 text box. The “Clear” button deletes everything that is currently in the workspace, while the “Trash” button will only delete the currently selected box, which will have a thicker, light blue border. The “Selected Tool” shows the current component that can be placed on the grid and the text box is for manually adding in a delay, in seconds, of when a selected gate will produce an output. Lastly, in the toolbar at the bottom, there are buttons for the toggle and LED components along with a menu button for wires, and an individual button for every basic gate. For this guide, we will create a simple circuit with 3 inputs, 1 output, an OR gate and 1 AND gate.

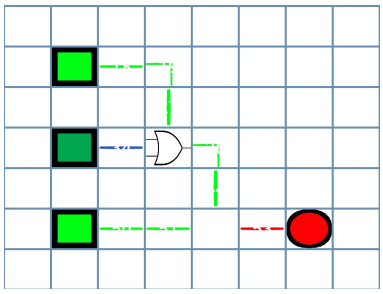


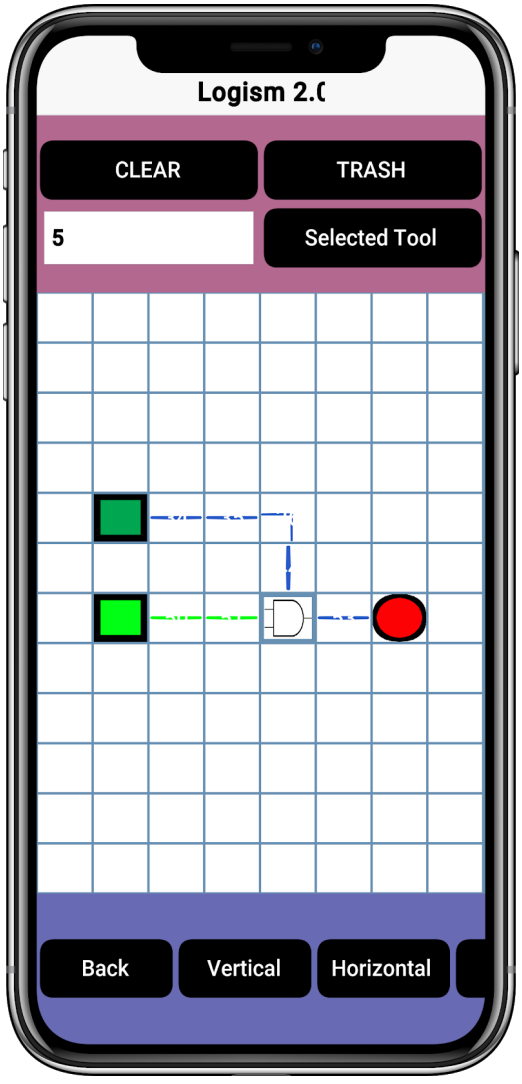
The first step would be to click on the toggle button in the toolbar and place 3 inputs, which, when placed on the grid, appear as colored boxes, either bright green for 1’s or dark green for 0’s . Next, we choose an OR gate and an AND gate and place them in the same boxes shown in the image on the right. Once we have the gates placed, we can put down the LED and have all of the main parts in the circuit ready to be connected. The last step would be to choose the appropriate orientation of wires, and place them in the same positions as shown in the image above.

Now that the circuit is fully complete, we can start to interact with the individual components. Clicking on the toggles will change its color based on the value that it is outputting. As the colors of the toggle change, the colors of the wires will also change, correlating to the value that they are currently holding. The LED will also be changing colors at the same time, red for 0's and green for 1's



With these simple instructions, we are able to quickly set up circuits that are visually appealing and easy to understand. If there are any errors that have been made, we can easily go back and fix it. For example, if we use the wrong gate, all we need to do is to select the gate that we need to replace and click on the trash button. This will delete the component at the selected box, in this case a gate, and update the colors of the rest of the components to reflect the changes. Now we are free to add in any type of relevant component into the slot that is freed.





Each gate inherently has a propagation delay of 0 when placed in the workspace. This can be modified by simply clicking on the gate, changing the value that is shown in the text box at the top bar and pressing enter. In the example, we used 5 seconds as the value, so the output of the gate will be updated exactly 5 seconds from the moment enter is pressed.

Functionality Constraints:

In order to provide the best experience for the user, we have purposely included some functionality constraints that users will need to observe while using the Mobilogic. The first is to make circuits flowing from left to right, with toggle switches, or the input, being on the left and LED's that display the output placed on the right side, as shown in the image. To properly incorporate the logic gates with the flow of the circuit, the inputs for the logic gates are limited to be from one box

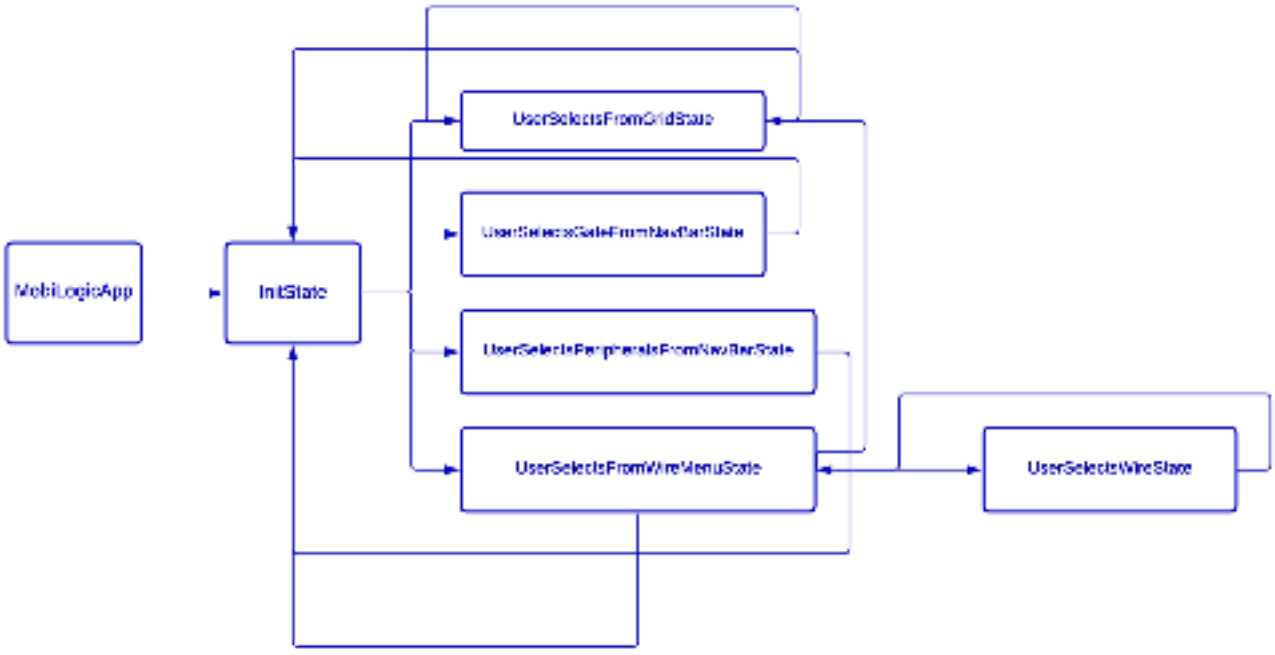
to the left and one box from above. This allows us to prevent overly complicated circuits with large, interweaving wires that are hard to keep track of.

Another constraint that has been built-in is an autosave, persistent storage feature, where the circuit is autosaved when changes are made. If a user makes many changes and closes the app, it will automatically look for the last change made, and save that circuit design. On reopening the app, the previous circuit will also be loaded back into the workspace for the user to continue working from where they left off.

Lastly, there is a constraint placed on when the propagation delay is saved. When a user enters a number into the propagation delay field, it is not saved unless the enter key is pressed on the phone's keyboard.

Design Pattern:

Implementation wise, this application was developed by following the state design pattern. We decided on this design pattern because we realized this application was heavily state based. If we had tried to implement this application imperatively (which we did at first), with many private primitive values and if-statements to keep track of application state, the program would have been cluttered, complicated, and illegible. Using a state design pattern allowed us to break apart the various logics for user-interaction into different classes that represent different states of the application.



UML Class Diagram:

The UML diagram below describes the connection between the various states of the application (contained in classes). Note that all state classes are connected by a context object of type MobiLogicContext, and all state classes derive from the interface MobiLogicState.